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

May 12, 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

In this chapter, I want to talk about/draw the attention to:

- The differentiation of boolean operators and intuitionistic logic operators
- The equivalence relation between SITPN and VHDL.
- The correspondence between combinational signal value and there assignment expression deduced from the code. Explain that this is where the $\mathcal{H}$-VHDL semantics plays its part in the proof; although we are not detailling how assignment expressions are deduced from running the semantics of the $\mathcal{H}$-VHDL code. Give some examples of correspondence between combinational signal value and assignment expressions.
- The particularity of the similarity relation for time counters.

In this chapter, we present our semantic preservation theorem along with its proof. The written proof is about a hundred-page long after compilation of the LaTeX files. Therefore, we will only present here the "high-level" theorems and lemmas used in the proof, and some hints regarding the proof strategy. The full proof is available to the reader in Appendix C. The theorems and lemmas presented in this chapter will be refering to the lemmas of Appendix C. The structure of this chapter is the following one: in Section 1.1, we present our review of the literature pertaining to the proof of semantic preservation theorems for transformation functions; in Section , we detail our state similarity relation, i.e, the semantic bound between an SITPN and its $\mathcal{H}$-VHDL translation; in Section, we draw out our semantic preservation theorem; in Section, we detail a particularly tricky point of the proof related to the computation of fired transitions, and we show how it has led to a bug detection in HILECOP's code; in Section, we present some points of the mechanization of the proof verification with the Coq proof assistant.

1.1 Semantic preserving transformations in the literature

In this section, we present the review of the literature pertaining to the verification of transformation functions. A transformation function is understood here as any kind of mapping from a source representation to a target representation, where the source and target representations possess a behavior of their own (i.e, they are executable). Here, we will focus on verification techniques based on the proof of semantic preservation theorems. We are interested in how to prove that transformation functions are semantic preserving. Especially, we are interested in the expression of semantic preservation theorems, i.e, what does one mean by semantic preservation, and in seeking usual proof strategies.

The goal is to draw our inspiration from the literature, and to see how far the correspondence holds between our specific case of transformation, and other cases of transformations. The material we used for the literature review is divided in three categories. Each category covers a specific case of transformation function; the three categories are:

- Compilers for generic programming languages
- Compilers for hardware description languages
- Model-to-model and model-to-text transformations

1.1.1 Transformations and proofs of semantic preservation

In the introduction of his article about CompCert [12], X.Leroy presents the two points of major importance to express semantic preservation theorems for GPL compilers, and more generally to get the meaning of semantic preservation.

The first point is to clearly state how things are compared between the source and the target programs. It is to describe the runtime state of the source and the target, and to draw a correspondence between two. This is expressed through a state comparison relation.

The second point is to relate the execution of the source program to the execution of the target program through a simulation, or bisimulation, diagram. Figure shows the different kind of simulation diagrams possibly relating two programs. Choosing an adequate simulation diagram to express a semantic preservation theorem depends on the kind of possible behaviors that can exhibit a given program. In the case of GPL programs, X.Leroy lists three kinds of possible behaviors: either the program execution succeeds and returns a value, or the program execution fails and returns an error, or the program execution diverges.

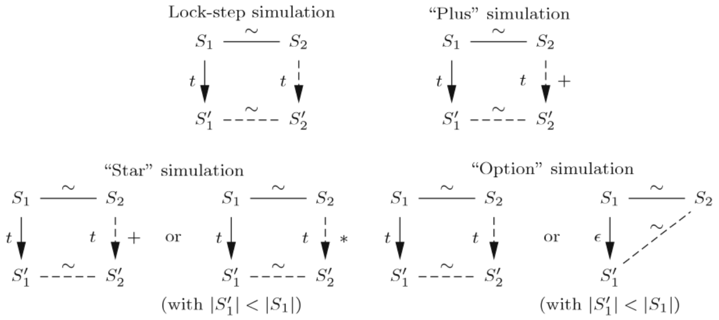

FIGURE 1.1: Simulation diagrams between source and target programs

Anyway, in the case where the source program execution succeeds, the theorem of semantic preservation takes this general form:

Consider a source program P_1 compiled into a target program P_2, a starting state S_1 for program P_1 and a starting state S_2 for program P_2 such that S_1 and S_2 are similar states w.r.t. the exhibited state comparison relation. If the execution of P_1 leads from state S_1 to state S_1', then there exists a state S_2' resulting of the execution of program P_2 from state S_2 such that S_1' and S_2' are similar w.r.t. the exhibited state comparison relation.

Compiler verification tasks aims at proving the kind of theorem stated above. The other kind of task that can be applied to certify a compiler is to perform compiler validation. Compiler validation is interested in generating a proof of behavior preservation (or a counter-example showing that behaviors diverge) for a given input program alongside the compilation process. Thus, for a given input program, the compiler yields a target program and the proof that the input and target have the same behavior. Exhibiting a theorem of semantic preservation is stronger than building a proof of semantic preservation for each input program. Therefore, compiler verification is stronger than compiler validation. The aim of the thesis is to perform compiler *verification* over the HILECOP methodology. Some of the works, cited afterwards, are more interested in compiler or transformation validation techniques than in verification. They are presented here for the sake of coverage.

Now that we have clarified the meaning of semantic preservation for GPL compilers, we state that this definition of semantic preservation holds also for more general case of transformation from a source representation to a target representation. The only condition to be able to verify that a transformation is semantic preserving is that the source and target representation must have an execution semantics (i.e, the instances of the source and target representations must be executable).

For each article used in the literature review and presenting a specific case of transformation, the following questions have been asked:

- What are the similarities/differences between source and target representations?
- How are described the runtime state for the source and target representations?
- How is expressed the state comparison relation?
- How is stated the semantic preservation theorem?
- What is the employed proof strategy?

Compilers for generic programming languages

Taking the CompCert compiler as an example, the compilation pass from Clight programs to Cminor programs is described in [2, 12]. Clight is a subset of the C language, and Cminor is a low-level imperative language. The two languages are endowed with a big-step operational semantics. Here, the execution state of the source and target languages are memory models (of course, we are dealing with programming languages). The memory model is the same for all intermediate language involved in the CompCert compiler. The memory model consists in block references; each block has a lower and an upper bound. To access a data, one has to specify the block reference along with the size of the accessed data (i.e, the data type) and the offset from the start of the block reference (i.e, where to begin the data reading). About the proof of semantic preservation, the most difficult point is to relate the memory state sof the source program to the memory states of the target program. To do so, the authors define a *memory injection* relation that binds the values of source and target together. They also establish a relation to compare execution environments, i.e, the environments holding the declaration of functions, global variables…The proof of semantic preservation is built incrementally: the authors prove a simulation lemma for the Clight expressions, then for the Clight statements, and finally for the entire Clight program. The proof strategy is to reason by induction over the evaluation relation of the Clight programs, and to perform case analysis on the translation function.

The pattern to compiler verification for GPLs is more or less the same as presented above. May it be compilers for imperative languages [12, 15], or compilers for functional languages [7, 17], compiler verfication proceeds as follows:

1. establish a relation between the memory models of the source and target languages, and between the global execution environments
2. prove simulation lemmas starting from simple constructs, and building up incrementally to consider entire programs
3. reason by induction over the evaluation relation of the source language, and the translation function

Relating memory models is more difficult when the gap between the source and target languages is important (for instance, the translation of Cminor programs into RTL programs in [12]). As a consequence, the complexity of the relation for memory model comparison increases.

Compilers for hardware description languages

In the case of HDL compilers, proving semantic preservation is very similar to the case of GPL compilers. Of course, the difference lies in the semantics of HDL languages, and in the description of execution states. The semantics of HDLs is intrinsically related to the notion of execution over time, or over multiple clock cycles; indeed, we are dealing with reactive systems. Therefore, the semantic preservation theorems are formulated w.r.t. the synchronous or time-related semantics of the considered languages.

In [3, 5], the source languages are a subset of the BlueSpec specification language for hardware synthesis , and the target is an RTL representation of the circuit. The execution states of the source and target are based on registers. In [3], the execution state also hold a log of the read and write operations of the input program, and this log is compared to the log of the RTL representation. The semantic preservation theorem states that the registers hold the same values after the execution of source program and the resulting RTL circuit after one clock cycle.

In [4], the source language is a subset of Lustre and the target language is imperative language called Obc. A Lustre program is composed of nodes; each node treats a set of input streams and publishes output streams after the computation of its statement body. In its statement body, a Lustre node possibly refer to instances of other nodes. In the compilation process, each Lustre node is translated into an Obc class. An Obc class hold a vector of variables composing its internal memory and a vector of other Obc class instances. The authors define a data flow semantics for the Lustre language; judgments of the semantics describe how output streams are computed based on input streams. Also, as we are dealing with hardwares, the judgments treat synchronous statements and combinational ones. On the side of the Obc language, the semantics define a function *step* that computes the execution the Obc classes over one clock cycle. To prove the semantic preservation theorem, the state comparison relation binds the values of input and output streams on one side to the values of variables and Obc class instances on the other side. The semantic preservation theorem is as follows: if a Lustre node yields output streams o from input streams i, then the iterative execution of the *step* function for the corresponding Obc class builds every step of output streams o given the values of input streams i. The proof is done by induction over the clock step count, and by induction over the evaluation derivation of the nody instruction body.

In [13], the HDL compiler translates Verilog modules into netlists. The execution state of Verilog module holds the value of the variables declared in the module. The execution state of a netlist circuit holds the value of the registers declared in the circuit. Therefore, the state comparison relation used to state the semantic preservation theorem binds the values of variables on one side to the values of registers on the other side. The semantics of Verilog resembles the one of VHDL; the set of processes composing a module are executed w.r.t. the simulation semantics of the language, i.e, composed of synchronous and combinational execution steps. The semantics of netlists is set as a big-step operational semantics by means of an interpreter that runs a netlist list over n clock cycles. The semantic preservation theorem is as follows: Assuming that a module is transformed into a circuit, and that some well-formation hypotheses hold on the module, if the module executes without error, and yields a final state *venv*, then there exists a final state *cenv* yielded by the execution of the circuit over n clock cycles s.t. *venv* and *cenv* are similar according to the relation *verilog_netlist_rel*. Here, the *verilog_netlist_rel* is the state comparison relation.

In [19], the compiler transforms programs of the synchronous language SIGNAL into Synchronous Clock Guarded Actions programs (S-CGA programs). A SIGNAL program describes a set of processes; each process holds a set of equations describing the relation between signals. The equations can be synchronous equations (refering to a clock) or combinational ones. An S-CGA program defines a set of actions to be applied to some variables when some conditions (the guards) are met. The SIGNAL (resp. the S-CGA) language has been endowed with a trace semantics describing the computation of signal values (resp. variable values) over time. The authors describe a function to translate the traces of SIGNAL and S-CGA programs into a common trace model. Thus, the semantic preservation theorem is stated by comparing two traces of execution defined through the same model. The proof of the semantic preservation theorem is built incrementally. For each statement of a SIGNAL process, the authors exhibit a lemma proving that the trace resulting from the execution of the statement is equivalent to the trace resulting of the execution of the corresponding guarded actions (obtained through the compilation). The proof is fully mechanized within the Coq proof assistant.

In [11], the authors verify a methodology to design hardware models with SystemC models. SystemC models describe hardware with modules; a module is a C++ class with ports, data members and methods. The methodology describes a transformation from SystemC into Abstract State Machine (ASM) thus enabling to model-check the hardware models. ASMs are described in the language AsmL; in AsmL, an ASM is implemented by a class with data members and methods. A denotational (fixpoint) semantics for SystemC modules is defined along with a denotational semantics for AsmL. The semantics is another variant of simulation cycle, similar to all other synchronous languages. There are two phases: evaluate and update and the gap between the two is called a delta-delay. The execution state of a SystemC module is divided into a signal store, mapping signal to value, and a variable store, mapping variable to value. The execution state of an AsmL class is only composed of a variable store. The theorem of semantic preservation states that, after translation, a SystemC model has the same *observational* behavior than its corresponding AsmL class. What is compared between a SystemC model and its corresponding AsmL class through their observational behavior is the activity of the processes of the first one and the activity of the methods of the second one. Processes and methods must be active at the same delta cycles. Therefore, what is compared here are not the values that the execution states hold, but rather the activity of the source and target programs.

Model transformations

Regarding model transformations, a lot of works consider semantic preservation as the preservation of structural properties in the transformed model [1, 6, 14].

Still, there are many cases where the source model and th target one have both an execution semantics. In these cases, the authors are interested in proving that the transformation is semantic preserving by showing that the computation of the source model and the target model follow a simulation relation (see Figure 1.1).

In [8] and [18], the authors are interested in giving a translational semantics to a given model having itself a reference execution semantics. In [8], the source models are called xSpem models; they describe a set of activities exchanging resources and an holding an internal state. The target models are PNs. Both xSpem models and PNs have a state transition semantics. The state comparison is performed by checking the correspondence between each current status of the activities describe in an xSpem model and the marking of the PN. Then, the authors prove a bisimulation theorem, illustrated in Figure 1.2.

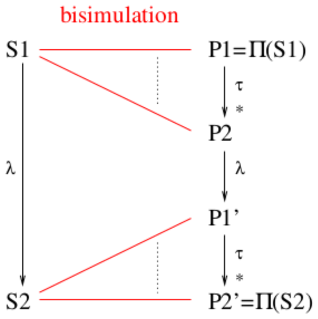

FIGURE 1.2: Bisimulation diagram relating an xSpem model execution and a Petri net execution

In Figure 1.2, one the right side of the diagram, i.e, the Petri net side, one can see that a Petri net possibly performs many internal actions (represented the arrow $\xrightarrow{\tau}^{*}$) before and after executing the computation step that is interest for the proof (i.e, action λ). Referring to the diagrams of Figure 1.1, this is a case of "star" simulation. The proof is performed by reasoning by induction on the structure of the xSpem model, and then by reasoning of the state transition semantics of xSpem models and PNs.

In [18], the authors describe a transformation from a model of the AADL formalism (Architecture Analysis and Design Language) to a particular kind of Abstract State Machine (ASM) called Timed Abstract State Machines (TASM). To verify that the transformation is semantic preserving, the authors define the semantics of AADL models and TASMs through Timed Transition Systems (TTSs). Thus, the execution state of an AADL model is the execution state of the corresponding TTS, and the same holds for a TASM. Comparing the state of two TTSs is easier than comparing the state of two different models. Then, the authors prove a strong bisimulation theorem to verify that the transformation is semantic preserving. The whole proof is mechanized within the Coq proof assistant.

In [10], the authors describe a transformation from LLVM-labelled Petri nets to LLVM programs, where LLVM is low-level assembly language. Precisely, the generated LLVM program implements the state space of the source Petri net (i.e, the graph of reachable markings). The authors want to verify if an LLVM program trully implements the PN state space, i.e if each marking present in the PN state space can be reached by running a specific $fire_t$ function on the generated LLVM program. The state of an LLVM program is defined by a memory model composed of a heap and a stack. The marking of an LLVM-labelled PN is defined in such a manner that the correspondence with the LLVM program memory model is straight-forward. The PN model has a classical firing semantics, and LLVM programs follow a small-step operational semantics. The semantic preservation theorem states that for all transition t being fired, leading from marking M to marking M', then applying running the $fire_t$ function over the generated LLVM program at state LM (such that LM implements marking M) leads to a new state LM', such that LM' implements marking M'. To prove this theorem, the authors proceed by induction on the number of places of the Petri net.

Discussions on transformations and proof strategies

In this thesis, we are interested in the verification of a semantic preservation property for a given transformation by proving a bisimulation theorem. To achieve this kind of proof task, the proceedings are quite similar, at least in the three cases of transformation presented above (i.e, GPLs compilation, HDLs compilation and model transformations). Even though the source and target languages or models are different from one case of transformation to the other, however, bisimulation theorems carry the same structure. The state comparison relation and the choice of the bisimulation diagram are the two angular stones of the process.

One can notice that when verifying the transformation of HDL programs, the bisimulation theorems are expressed around a time-related computational step. It can either be a clock cycle, or another kind of time step. The state equivalence checking is made at the end this time-related computational step. This differs from the expression of bisimulation theorems for GPLs, where a computational step is not related to time, but rather expresses the one-time computation of programs.

Concerning proof strategies, in the case of programming languages, proving the bisimulation theorems are systematically done by induction over the semantics relation of the source languages. The semantics relation are themselves defined by following the inductive structure of the language ASTs. In the case of model transformations, when the source model permits it, the proofs are performed similarly by applying inductive reasoning over the structure of the input model. This enable compositional reasoning, i.e: to split the difficulty of proving the bisimulation theorem into simpler lemmas about the execution of simpler programs or simple model structures.

1.2 The state similarity relation

Before stating the behavior preservation theorem, we must clarify the meaning of semantic preservation between an SITPN and a $\mathcal{H}$-VHDL design. To do so, we must define:

1. what does semantical matching means between an SITPN state and an $\mathcal{H}$-VHDL state?

2. when, in the course of the execution of an SITPN and an $\mathcal{H}$-VHDL design, does this semantical matching must hold?

We must relate the elements that constitute the execution state of an SITPN to the elements that constitute the execution state of an $\mathcal{H}$-VHDL design. An SITPN state is an abstract structure relating the places, transitions, actions, functions and conditions of a given SITPN to the values of certain domains. A $\mathcal{H}$-VHDL design state is composed a signal store mapping signals to values, and of a component store mapping component instances to their own internal states. Thanks to the binder function γ generated alongside the transformation from an SITPN to a $\mathcal{H}$-VHDL design, we are able to relate the elements of the SITPN structure to the component instances and signals on the $\mathcal{H}$-VHDLside. Thus, the state similarity relation expressing a semantical match between an SITPN state and an $\mathcal{H}$-VHDL design is defined as follows:

Definition 1 (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. $\gamma(p) = id_p,\ s.M(p) = \sigma(id_p)(\text{"}s\_marking\text{"})$.
2. $\forall t \in T_i, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$,
$\big(upper(I_s(t)) = \infty \wedge s.I(t) \leq lower(I_s(t)) \Rightarrow s.I(t) = \sigma(id_t)(\text{"}s\_time\_counter\text{"})\big)$
$\wedge\big(upper(I_s(t)) = \infty \wedge s.I(t) > lower(I_s(t)) \Rightarrow \sigma(id_t)(\text{"}s\_time\_counter\text{"}) = lower(I_s(t))\big)$
$\wedge\big(upper(I_s(t)) \neq \infty \wedge s.I(t) > upper(I_s(t)) \Rightarrow \sigma(id_t)(\text{"}s\_time\_counter\text{"}) = upper(I_s(t))\big)$
$\wedge\big(upper(I_s(t)) \neq \infty \wedge s.I(t) \leq upper(I_s(t)) \Rightarrow s.I(t) = \sigma(id_t)(\text{"}s\_time\_counter\text{"})\big)$.
3. $\forall t \in T_i, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t, s.reset_t(t) = \sigma(id_t)(\text{"}s\_reinit\_time\_counter\text{"})$.
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. $\gamma(f) = id_f,\ s.ex(f) = \sigma(id_f)$.

In Item 1, based on the γ binder, we relate the marking value of a place p to the value of the *s_marking* signal inside the internal state of the place component instance id_p. Items 2 and 3 similarly relate the value of time counters (and reset orders) of transitions to the value of the signals *s_time_counter* (resp. *s_reinit_time_counter*) in the internal state of the corresponding transition component instances.

Definition 2 (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. $\gamma(p) = id_p,\ s.M(p) = \sigma(id_p)(\text{"}s\_marking\text{"})$.
2. $\forall t \in T_i, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$,
$\big(upper(I_s(t)) = \infty \wedge s.I(t) \leq lower(I_s(t)) \Rightarrow s.I(t) = \sigma(id_t)(\text{"}s\_time\_counter\text{"})\big)$
$\wedge\big(upper(I_s(t)) = \infty \wedge s.I(t) > lower(I_s(t)) \Rightarrow \sigma(id_t)(\text{"}s\_time\_counter\text{"}) = lower(I_s(t))\big)$
$\wedge\big(upper(I_s(t)) \neq \infty \wedge s.I(t) > upper(I_s(t)) \Rightarrow \sigma(id_t)(\text{"}s\_time\_counter\text{"}) = upper(I_s(t))\big)$
$\wedge\big(upper(I_s(t)) \neq \infty \wedge s.I(t) \leq upper(I_s(t)) \Rightarrow s.I(t) = \sigma(id_t)(\text{"}s\_time\_counter\text{"})\big)$.
3. $\forall t \in T_i, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t, s.reset_t(t) = \sigma(id_t)(\text{"}s\_reinit\_time\_counter\text{"})$.
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.* $\gamma(t) = id_t, t \notin Sens(s.M) \Leftrightarrow \sigma(id_t)(''s\_enabled'') = \texttt{false}$.

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

Definition 3 (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. 1, general state similarity) and*

1. $\forall t \in T, id_t \in Comps(\Delta)$ *s.t.* $\gamma(t) = id_t, t \in Firable(s) \Leftrightarrow \sigma(id_t)(''s\_firable'') = \texttt{true}$.

2. $\forall t \in T, id_t \in Comps(\Delta)$ *s.t.* $\gamma(t) = id_t, t \notin Firable(s) \Leftrightarrow \sigma(id_t)(''s\_firable'') = \texttt{false}$.

3. $\forall t \in T, id_t \in Comps(\Delta)$ *s.t.* $\gamma(t) = id_t, t \in Fired(s) \Leftrightarrow \sigma(id_t)(''fired'') = \texttt{true}$.

4. $\forall t \in T, id_t \in Comps(\Delta)$ *s.t.* $\gamma(t) = id_t, t \notin Fired(s) \Leftrightarrow \sigma(id_t)(''fired'') = \texttt{false}$.

5. $\forall p \in P, id_p \in Comps(\Delta)$ *s.t.* $\gamma(p) = id_p$, $\sum_{t \in Fired(s)} pre(p,t) = \sigma(id_p)(''s\_output\_token\_sum'')$.

6. $\forall p \in P, id_p \in Comps(\Delta)$ *s.t.* $\gamma(p) = id_p$, $\sum_{t \in Fired(s)} post(t,p) = \sigma(id_p)(''s\_input\_token\_sum'')$.

Definition 4 (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:*

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

1.3 Behavior Preservation Theorem

Definition 5 (Similar Environments). *For a given sitpn* $\in$ *SITPN, a* $\mathcal{H}$*-VHDL design* $d \in$ *design, a design store* $\mathcal{D} \in$ *entity-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* $E_p \in (\mathbb{N} \times \{\uparrow, \downarrow\}) \rightarrow Ins(\Delta) \rightarrow value$*, 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.* $\gamma(c) = id_c$*,* $E_p(\tau, clk)(id_c) = E_c(\tau)(c)$*.*

1.3.1 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.

Make a list of all signals and constants of the T and P components, and their related aliases.

Constants and signals reference			
Full name	*Alias*	*Category*	*Type*
$''input\_conditions''$	$''ic''$	input port (T)	$\mathbb{B}$
$''input\_conditions''$	$''ic''$	input port (T)	$\mathbb{B}$
$''reinit\_time''$	$''rt''$	input port (T)	$\mathbb{B}$
$''input\_arcs\_valid''$	$''iav''$	input port (T)	$\mathbb{B}$
$''fired''$	$''f''$	output port (T)	$\mathbb{B}$
$''s\_condition\_combination''$	$''scc''$	internal signal (T)	$\mathbb{B}$
$''s\_reinit\_time\_counter''$	$''srtc''$	internal signal (T)	$\mathbb{B}$
$''s\_priority\_combination''$	$''spc''$	internal signal (T)	$\mathbb{B}$
$''s\_fired''$	$''sf''$	internal signal (T)	$\mathbb{B}$
$''s\_firable''$	$''sfa''$	internal signal (T)	$\mathbb{B}$
$''s\_enabled''$	$''se''$	internal signal (T)	$\mathbb{B}$
$''input\_arcs\_number''$	$''ian''$	generic constant (T)	$\mathbb{N}$
$''transition\_type''$	$''tt''$	generic constant (T)	$\{\texttt{NOT\_TEMP}, \texttt{TEMP\_A\_B}, \texttt{TEMP\_A\_A}, \texttt{TEMP\_A\_INF}\}$
$''conditions\_number''$	$''cn''$	generic constant (T)	$\mathbb{N}$
$''maximal\_time\_counter''$	$''mtc''$	generic constant (T)	$\mathbb{N}$
$''s\_marking''$	$''sm''$	internal signal (P)	$\mathbb{N}$
$''s\_output\_token\_sum''$	$''sots''$	internal signal (P)	$\mathbb{N}$
$''s\_input\_token\_sum''$	$''sits''$	internal signal (P)	$\mathbb{N}$
$''reinit\_transition\_time''$	$''rtt''$	output port (P)	$\mathbb{B}$
$''output\_arcs\_types''$	$''oat''$	input port (P)	$\{\texttt{BASIC}, \texttt{TEST}, \texttt{INHIB}\}$
$''output\_arcs\_weights''$	$''oaw''$	input port (P)	$\mathbb{N}$
$''output\_transition\_fired''$	$''otf''$	input port (P)	$\mathbb{B}$
$''input\_arcs\_weights''$	$''iaw''$	input port (P)	$\mathbb{N}$
$''input\_transition\_fired''$	$''itf''$	input port (P)	$\mathbb{B}$

1.3.2 Behavior Preservation Theorem and Proof

Theorem 1 (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 $E_p \in (\mathbb{N} \times \{\uparrow, \downarrow\}) \rightarrow Ins(\Delta) \rightarrow value$ 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 $\Delta, \theta_\sigma, \sigma_e$ 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 2 (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 3 (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 4 (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:* $\mathcal{D}_{\mathcal{H}}, E_p, \Delta, \tau, \sigma_0 \vdash \mathrm{d.cs} \rightarrow \theta_\sigma$

1.3.3 Bisimulation Theorem and Proof

Theorem 5 (Full Bisimulation)**.** *For all* $sitpn \in SITPN$*,* $d \in design$*,* $\gamma \in WM(sitpn, d)$*,* $\tau \in \mathbb{N}$*,* $E_c \in \mathbb{N} \to \mathcal{C} \to \mathbb{B}$*,* $\theta_s \in \texttt{list}(S(sitpn))$*,* $\Delta \in ElDesign(d, \mathcal{D}_{\mathcal{H}})$*,* $E_p \in (\mathbb{N} \times \{\uparrow, \downarrow\}) \to Ins(\Delta) \to value$*,* $\theta_\sigma \in \texttt{list}(\Sigma(\Delta))$ *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 $sitpn \in SITPN, d \in design, \gamma \in WM(sitpn, d), s \in S(sitpn), \Delta \in ElDesign(d, \mathcal{D}_\mathcal{H})$, $\sigma_e, \sigma_0, \sigma \in \Sigma(\Delta)$, $E_c \in \mathbb{N} \rightarrow \mathcal{C} \rightarrow \mathbb{B}$, $E_p \in (\mathbb{N} \times \{\uparrow, \downarrow\}) \rightarrow Ins(\Delta) \rightarrow value$, 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 $E_c, \tau - 1 \vdash sitpn, s \rightarrow \theta_s$.
  - $E_p, \Delta, \tau, \sigma \vdash \mathrm{d.cs} \xrightarrow{\uparrow,\downarrow} \sigma'$ 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 $\Delta, \sigma_{injr} \vdash \mathrm{d.cs} \xrightarrow{\uparrow} \sigma_r$ and $\Delta, \sigma_r \vdash \mathrm{d.cs} \xrightarrow{\theta} \sigma'$
- $\texttt{Inject}_\downarrow(\sigma', E_p, \tau, \sigma_{injf})$ and $\Delta, \sigma_{injf} \vdash \mathrm{d.cs} \xrightarrow{\downarrow} \sigma_f$ and $\Delta, \sigma_f \vdash \mathrm{d.cs} \xrightarrow{\theta'} \sigma''$

Solved by applying Rising Edge and then "Falling Edge" lemmas. □

1.4 A detailled proof: equivalence of fired transitions

Definition 6 (Falling Edge Hypotheses). *Given an* $sitpn \in SITPN$*,* $d \in design$*,* $\gamma \in WM(sitpn, d)$*,* $E_c \in \mathbb{N} \to \mathcal{C} \to \mathbb{B}$*,* $\Delta \in ElDesign(d, \mathcal{D}_{\mathcal{H}})$*,* $E_p \in (\mathbb{N} \times \{\uparrow, \downarrow\}) \to Ins(\Delta) \to value$*,* $\tau \in \mathbb{N}$*,* $s, s' \in S(sitpn)$*,* $\sigma_e, \sigma, \sigma_i, \sigma_{\downarrow}, \sigma' \in \Sigma(\Delta)$*, assume that:*

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

Lemma 4 (Falling Edge Equal Fired). *For all* $sitpn$*,* d*,* γ*,* Δ*,* σ_e*,* E_c*,* E_p*,* τ*,* s*,* s'*,* σ*,* σ_i*,* $\sigma_{\downarrow}$*,* σ' *that verify the hypotheses of Def. 6, then* $\forall t \in T, id_t \in Comps(\Delta)$ *s.t.* $\gamma(t) = id_t$*,* $t \in Fired(s') \Leftrightarrow \sigma'(id_t)(''fired'') = \texttt{true}$*.*

Proof. Given a $t \in T$ and an id_t s.t. $\gamma(t) = id_t$, let us show $\boxed{t \in Fired(s') \Leftrightarrow \sigma'(id_t)(''fired'') = \texttt{true}.}$ The proof is in two parts:

1. Assuming that $t \in Fired(s')$, let us show $\boxed{\sigma'(id_t)(''fired'') = \texttt{true}.}$

   By definition of $t \in Fired(s')$, there exists $fset \subseteq T$ s.t. $IsFiredSet(s', fset) \wedge t \in fset$.

   Let us take such an $fset$, and apply Lemma Falling Edge Equal Fired Set to solve the goal.

2. Assuming that $\sigma'(id_t)(''fired'') = \texttt{true}$, let us show $\boxed{t \in Fired(s').}$

   By definition of $t \in Fired(s')$, let us show that $\boxed{\exists fset \subseteq T \text{ s.t. } IsFiredSet(s', fset) \wedge t \in fset}$

   Assuming that $sitpn$ is a well-defined $SITPN$ (see Section), we can always find an $fset \subseteq T$ such that $\forall s \in S(sitpn)$, $IsFiredSet(s, fset)$ is derivable. Let us take an $fset \subseteq T$ s.t. $IsFiredSet(s', fset)$, and use it to prove the goal by applying Lemma Falling Edge Equal Fired Set.

   Add ref.
   defined S

□

Lemma 5 (Falling Edge Equal Not Fired). *For all* $sitpn$*,* d*,* γ*,* Δ*,* σ_e*,* E_c*,* E_p*,* τ*,* s*,* s'*,* σ*,* σ_i*,* $\sigma_{\downarrow}$*,* σ' *that verify the hypotheses of Def. 6, then* $\forall t, id_t$ *s.t.* $\gamma(t) = id_t$*,* $t \notin Fired(s') \Leftrightarrow \sigma'_t(''fired'') = \texttt{false}$*.*

Proof. Proving the above lemma is trivial by appealing to Lemma Falling Edge Equal Fired and by reasoning on contrapositives. □

Lemma 6 (Falling Edge Equal Fired Set). *For all* $sitpn$*,* d*,* γ*,* Δ*,* σ_e*,* E_c*,* E_p*,* τ*,* s*,* s'*,* σ*,* σ_i*,* $\sigma_{\downarrow}$*,* σ' *that verify the hypotheses of Def. 6, then* $\forall t \in T$*,* $id_t \in Comps(\Delta)$ *s.t.* $\gamma(t) = id_t$*,* $\forall fset \subseteq T$*, s.t.* $IsFiredSet(s', fset)$*,* $t \in fset \Leftrightarrow \sigma'(id_t)(''fired'') = true$*.*

Proof. Given a $t \in T$, and $id_t \in Comps(\Delta)$, and a $fset \subseteq T$ s.t. $IsFiredSet(s', fset)$, let us show $\boxed{t \in fset \Leftrightarrow \sigma'(id_t)(''fired'') = true.}$

By definition of $IsFiredSet(s', fset)$, we have $IsFiredSetAux(s', \emptyset, T, fset)$.
Then, we can appeal to Lemma Falling Edge Equal Fired Set Aux to solve the goal, but first we must prove the following *extra hypothesis* (i.e, one of the premise of Lemma Falling Edge Equal Fired Set Aux):

$$\boxed{\begin{array}{l}\forall t' \in T, id_{t'} \in Comps(\Delta) \text{ s.t. } \gamma(t') = id_{t'},\\ (t' \in \emptyset \Rightarrow \sigma'(id_{t'})(''fired'') = \texttt{true}) \wedge (\sigma'(id_{t'})(''fired'') = \texttt{true} \Rightarrow t' \in \emptyset \ \vee \ t' \in T).\end{array}}$$

Given a $t' \in T$ and an $id_{t'} \in Comps(\Delta)$ s.t. $\gamma(t') = id_{t'}$, there are two points to prove:

1. $\boxed{t' \in \emptyset \Rightarrow \sigma'(id_{t'})(''fired'') = \texttt{true}}$

2. $\boxed{\sigma'(id_{t'})(''fired'') = \texttt{true} \Rightarrow t' \in \emptyset \ \vee \ t' \in T}$

Let us show these two points:

1. Assuming $t' \in \emptyset$, let us show $\boxed{\sigma'(id_{t'})(''fired'') = \texttt{true}.}$

   $t' \in \emptyset$ is a contradiction.

2. Assuming $\sigma'(id_{t'})(''fired'') = \texttt{true}$, let us show $\boxed{t' \in \emptyset \ \vee \ t' \in T.}$

   By definition, $t' \in T$.

□

Lemma 7 (Falling Edge Equal Fired Set Aux). *For all $sitpn$, d, γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_\downarrow$, σ' that verify the hypotheses of Def. 6, then $\forall t \in T, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$, $\forall fired \subseteq T$, $T_s \subseteq T$, $fset \subseteq T$, assume that:*

- *$IsFiredSetAux(s', fired, T_s, fset)$*

- *EH (Extra. Hypothesis):*
  $\forall t' \in T$, $id_{t'} \in Comps(\Delta)$ s.t. $\gamma(t') = id_{t'}$,
  $(t' \in fired \Rightarrow \sigma'(id_{t'})(''fired'') = \texttt{true}) \wedge (\sigma'(id_{t'})(''fired'') = \texttt{true} \Rightarrow t' \in fired \ \vee \ t' \in T_s)$.

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

Proof. Given a $t \in T$, an $id_t \in Comps(\Delta)$, a $fired, T_s, fset \subseteq T$, and assuming $IsFiredSetAux(s', fired, T_s, fset)$ and EH, let us show $\boxed{t \in fset \Leftrightarrow \sigma'(id_t)(''fired'') = \texttt{true}.}$
Let us reason by induction on $IsFiredSetAux(s', fired, T_s, fset)$.

- **BASE CASE**: $\boxed{t \in fired \Leftrightarrow \sigma'(id_t)(''fired'') = \texttt{true}.}$

  In that case, $fired = fset$ and $T_s = \emptyset$, EH looks like this:

  $\forall t' \in T$, $id_{t'} \in Comps(\Delta)$ s.t. $\gamma(t') = id_{t'}$,
  $(t' \in fired \Rightarrow \sigma'(id_{t'})(''fired'') = \texttt{true}) \wedge (\sigma'(id_{t'})(''fired'') = \texttt{true} \Rightarrow t' \in fired \ \vee \ t' \in \emptyset)$.

  From EH, we can deduce $t \in fired \Leftrightarrow \sigma'(id_t)(''fired'') = \texttt{true}.$

- **INDUCTION CASE**: $\boxed{t \in fset \Leftrightarrow \sigma'(id_t)(\text{"}fired\text{"}) = \texttt{true}.}$

  In that case, we have:

  – $IsTopPrioritySet(T_s, tp)$

  – $ElectFired(s', fired, tp, fired')$

  – $FiredAux(s', fired', T_s \setminus tp, fset)$

  $(\forall t' \in T, id_{t'} \in Comps(\Delta)$ s.t. $\gamma(t') = id_{t'}$,
  $(t' \in fired' \Rightarrow \sigma'(id_{t'})(\text{"}fired\text{"}) = \texttt{true}) \wedge (\sigma'(id_{t'})(\text{"}fired\text{"}) = \texttt{true} \Rightarrow t' \in fired' \vee t' \in T_s \setminus tp)) \Rightarrow$
  $t \in fset \Leftrightarrow \sigma'_t(\text{"}fired\text{"}) = true.$

  Applying the induction hypothesis, then, the new goal is:

  $\forall t' \in T, id_{t'} \in Comps(\Delta)$ s.t. $\gamma(t') = id_{t'}$,
  $(t' \in fired' \Rightarrow \sigma'(id_{t'})(\text{"}fired\text{"}) = \texttt{true})$
  $\wedge(\sigma'(id_{t'})(\text{"}fired\text{"}) = \texttt{true} \Rightarrow t' \in fired' \vee t' \in T_s \setminus tp)$

  Apply Lemma Elect Fired Equal Fired to solve the goal.

□

Lemma 8 (Elect Fired Equal Fired). *For all sitpn, d, γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_\downarrow$, σ' that verify the hypotheses of Def. 6, then $\forall fired, fired', T_s, tp,\ fset \subseteq T$, assume that:*

- $IsTopPrioritySet(T_s, tp)$

- $ElectFired(s', fired, tp, fired')$

- $FiredAux(s', fired', T_s \setminus tp, fset)$

- *EH (Extra. Hypothesis):*
  $\forall t' \in T, id_{t'} \in Comps(\Delta)$ *s.t.* $\gamma(t') = id_{t'}$,
  $(t' \in fired \Rightarrow \sigma'(id_{t'})(\text{"}fired\text{"}) = \texttt{true}) \wedge (\sigma'(id_{t'})(\text{"}fired\text{"}) = \texttt{true} \Rightarrow t' \in fired \vee t' \in T_s)$

then $\forall t \in T, id_t \in Comps(\Delta)$ *s.t.* $\gamma(t) = id_t$,
$(t \in fired' \Rightarrow \sigma'(id_t)(\text{"}fired\text{"}) = \texttt{true}) \wedge (\sigma'(id_t)(\text{"}fired\text{"}) = \texttt{true} \Rightarrow t \in fired' \vee t \in T_s \setminus tp).$

Proof. Given a $t \in T$ and an $id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$, let us show
$\boxed{(t \in fired' \Rightarrow \sigma'(id_t)(\text{"}fired\text{"}) = \texttt{true}) \wedge (\sigma'(id_t)(\text{"}fired\text{"}) = \texttt{true} \Rightarrow t \in fired' \vee t \in T_s \setminus tp).}$

Let us reason by induction on $ElectFired(s', fired, tp, fired')$; there are three cases:

1. **BASE CASE**: $tp = \emptyset$ and $fired = fired'$.

2. **INDUCTIVE CASE**: $tp = \{t_0\} \cup tp_0$ and t_0 is elected to be fired.

3. **INDUCTIVE CASE**: $tp = \{t_0\} \cup tp_0$ and t_0 is not elected to be fired.

Let us prove the goal in these three contexts:

1. **BASE CASE**:

   $\boxed{\big(t \in fired \Rightarrow \sigma'(id_t)(''fired'') = \mathtt{true}\big) \wedge \big(\sigma'(id_t)(''fired'') = \mathtt{true} \Rightarrow t \in fired \vee t \in T_s\big).}$

   Apply EH to solve the goal.

2. **INDUCTIVE CASE**: $tp = \{t_0\} \cup tp_0$ and t_0 is elected to be fired.

   In that case, we have:

   - $IsTopPrioritySet(T_s, \{t_0\} \cup tp_0)$
   - $ElectFired(s', fired \cup \{t_0\}, tp_0, fired')$
   - $IsFiredSetAux(s', fired', T_s \setminus \{t_0\} \cup tp_0, fset)$
   - $t_0 \in Firable(s')$
   - $t_0 \in Sens(s'.M - \sum\limits_{t_i \in Pr(t,fired)} pre(t_i))$ where $Pr(t, fired) = \{t' \mid t' \succ t \wedge t' \in fired\}$
   - EH: $\forall t' \in T,\ id_{t'} \in Comps(\Delta)$ s.t. $\gamma(t') = id_{t'}$,
     $(t' \in fired \Rightarrow \sigma'(id_{t'})(''f'') = \mathtt{true}) \wedge (\sigma'(id_{t'})(''f'') = \mathtt{true} \Rightarrow t' \in fired\ \vee\ t' \in T_s)$

   $\forall T_s' \subseteq T,$
   $IsTopPrioritySet(T_s', tp_0) \Rightarrow$
   $IsFiredSetAux(s', fired', T_s' \setminus tp_0, fset) \Rightarrow$
   $\big(\forall t' \in T, id_{t'} \in Comps(\Delta)$ s.t. $\gamma(t') = id_{t'},$
   $(t' \in fired \cup \{t_0\} \Rightarrow \sigma'_{t'}(''f'') = \mathtt{true}) \wedge (\sigma'(id_{t'})(''f'') = \mathtt{true} \Rightarrow t' \in fired \cup \{t_0\}\ \vee\ t' \in T_s')\big) \Rightarrow$
   $\forall t \in T,\ id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t,$
   $\big(t \in fired' \Rightarrow \sigma'(id_t)(''f'') = \mathtt{true}\big) \wedge \big(\sigma'(id_t)(''f'') = \mathtt{true} \Rightarrow t \in fired' \vee t \in T_s' \setminus tp_0\big)$

   $\forall t \in T,\ id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t,$
   $\big(t \in fired' \Rightarrow \sigma'_t(''f'') = \mathtt{true}\big) \wedge \big(\sigma'_t(''f'') = \mathtt{true} \Rightarrow t \in fired' \vee t \in T_s \setminus \{t_0\} \cup tp_0\big)$

   To solve the goal, we can apply the induction hypothesis with $T_s' = T_s \setminus \{t_0\}$; then, there are three points to prove:

   (a) $\boxed{IsTopPrioritySet(T_s \setminus \{t_0\}, tp_0)}$

   (b) $\boxed{IsFiredSetAux(s', fired', (T_s \setminus \{t_0\}) \setminus tp_0, fset)}$

   (c) $\forall t' \in T, id_{t'} \in Comps(\Delta)$ s.t. $\gamma(t') = id_{t'},$
   $(t' \in fired \cup \{t_0\} \Rightarrow \sigma'_{t'}(''f'') = \mathtt{true}) \wedge (\sigma'(id_{t'})(''f'') = \mathtt{true} \Rightarrow t' \in fired \cup \{t_0\}\ \vee\ t' \in T_s \setminus \{t_0\})$

   Let us prove these three points:

   (a) $\boxed{IsTopPrioritySet(T_s \setminus \{t_0\}, tp_0)}$

Not provable yet.

(b) $\boxed{IsFiredSetAux(s', fired', (T_s \setminus \{t_0\}) \setminus tp_0, fset)}$.
We know that $(T_s \setminus \{t_0\}) \setminus tp_0 = T_s \setminus (\{t_0\} \cup tp_0)$, and thus $IsFiredSetAux(s', fired', T_s \setminus (\{t_0\} \cup tp_0), fset)$ is an assumption.

(c) $\boxed{\forall t' \in T, id_{t'} \in Comps(\Delta) \text{ s.t. } \gamma(t') = id_{t'}, (t' \in fired \cup \{t_0\} \Rightarrow \sigma'(id_{t'})(''f'') = \mathtt{true}) \land (\sigma'(id_{t'})(''f'') = \mathtt{true} \Rightarrow t' \in fired \cup \{t_0\} \ \lor \ t' \in T_s \setminus \{t_0\})}$

Given a $t' \in T$ and an $id_{t'} \in Comps(\Delta)$ s.t. $\gamma(t') = id_{t'}$, let us show
$\boxed{(t' \in fired \cup \{t_0\} \Rightarrow \sigma'(id_{t'})(''f'') = \mathtt{true}) \land (\sigma'(id_{t'})(''f'') = \mathtt{true} \Rightarrow t' \in fired \cup \{t_0\} \ \lor \ t' \in T_s \setminus \{t_0\}).}$
The proof is in two parts.

i. Assuming that $t' \in fired \cup \{t_0\}$, let us show $\boxed{\sigma'(id_{t'})(''f'') = \mathtt{true}.}$
Case analysis on $t' \in fired \cup \{t_0\}$; there are two cases:

- $t' \in fired$
- $t' = t_0$

Let us prove the goal in these two contexts.

- **CASE** $t' \in fired$: Thanks to EH, we can deduce $\sigma'_{t'}(''f'') = \mathtt{true}.$
- **CASE** $t' = t_0$:
By definition of $id_{t'}$, there exist a $gm_{t'}$, $ipm_{t'}$, $opm_{t'}$ s.t. $\mathtt{comp}(id_{t'}, ''transition'', gm_{t'}, ipm_{t'}, opm_{t'}) \in d.cs$.
By property of the stabilize relation and $\mathtt{comp}(id_{t'}, ''transition'', gm_{t'}, ipm_{t'}, opm_{t'}) \in d.cs$:
$$\sigma(id_{t'})(''f'') = \sigma(id_{t'})(''sfa'') \, . \, \sigma(id_{t'})(''spc'') \tag{1.1}$$
Rewriting the goal with (1.1): $\boxed{\sigma(id_{t'})(''sfa'') \, . \, \sigma(id_{t'})(''spc'') = \mathtt{true}.}$
Then, we can show that:
  - $\sigma(id_{t'})(''sfa'') = \mathtt{true}$ by applying Lemma Falling Edge Equal Firable
  - $\sigma(id_{t'})(''spc'') = \mathtt{true}$ by applying Lemma Stabilize Compute Priority Combination After Falling Edge.

ii. Assuming that $\sigma'(id_{t'})(''f'') = \mathtt{true}$, let us show $\boxed{t' \in fired \cup \{t_0\} \ \lor \ t' \in T_s \setminus \{t_0\}.}$
From $\sigma'(id_{t'})(''f'') = \mathtt{true}$ and EH, we can deduce that $t' \in fired \lor t' \in T_s$.
Case analysis on $t' \in fired \lor t' \in T_s$.

- **CASE** $t' \in fired$: then, it is trivial to show $t' \in fired \cup \{t_0\}.$
- **CASE** $t' \in T_s$: We know that $t_0 \in T_s$. Therefore, either $t' \in T_s \setminus \{t_0\}$, or $t' = t_0$, and then, $t' \in fired \cup \{t_0\}.$

3. **INDUCTIVE CASE**: $tp = \{t_0\} \cup tp_0$ and t_0 is not elected to be fired.

- $IsTopPrioritySet(T_s, \{t_0\} \cup tp_0)$
- $ElectFired(s', fired, tp_0, fired')$

- $IsFiredSetAux(s', fired', T_s \setminus \{t_0\} \cup tp_0, fset)$
- $\neg\big(t_0 \in Firable(s') \wedge t_0 \in Sens(s'.M - \sum_{t_i \in Pr(t, fired)} pre(t_i))\big)$
- EH:
  $\forall t' \in T, id_{t'} \in Comps(\Delta)$ s.t. $\gamma(t') = id_{t'}$,
  $(t' \in fired \Rightarrow \sigma'(id_{t'})(''f'') = \texttt{true}) \wedge (\sigma'(id_{t'})(''f'') = \texttt{true} \Rightarrow t' \in fired \ \vee \ t' \in T_s)$

$\forall T'_s \subseteq T$,
$IsTopPrioritySet(T'_s, tp_0) \Rightarrow$
$IsFiredSetAux(s', fired', T'_s \setminus tp_0, fset) \Rightarrow$
$\big(\forall t' \in T, id_{t'} \in Comps(\Delta)$ s.t. $\gamma(t') = id_{t'}$,
$(t' \in fired \Rightarrow \sigma'(id_{t'})(''f'') = \texttt{true}) \wedge (\sigma'(id_{t'})(''f'') = \texttt{true} \Rightarrow t' \in fired \ \vee \ t' \in T'_s)\big) \Rightarrow$
$\forall t \in T, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$,
$\big(t \in fired' \Rightarrow \sigma'(id_t)(''f'') = \texttt{true}\big) \wedge \big(\sigma'(id_t)(''f'') = \texttt{true} \Rightarrow t \in fired' \vee t \in T'_s \setminus tp_0\big)$

$\forall t \in T, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$,
$\big(t \in fired' \Rightarrow \sigma'(id_t)(''f'') = \texttt{true}\big) \wedge \big(\sigma'(id_t)(''f'') = \texttt{true} \Rightarrow t \in fired' \vee t \in T_s \setminus \{t_0\} \cup tp_0\big)$.

Then, we can apply the induction hypothesis with $T'_s = T_s \setminus \{t_0\}$, then, there are three points to prove:

(a) $\boxed{IsTopPrioritySet(T_s \setminus \{t_0\}, tp_0)}$

(b) $\boxed{IsFiredSetAux(s', fired', (T_s \setminus \{t_0\}) \setminus tp_0, fset)}$

(c) $\forall t' \in T, id_{t'} \in Comps(\Delta)$ s.t. $\gamma(t') = id_{t'}$,
$(t' \in fired \Rightarrow \sigma'(id_{t'})(''f'') = \texttt{true}) \wedge (\sigma'(id_{t'})(''f'') = \texttt{true} \Rightarrow t' \in fired \ \vee \ t' \in T_s \setminus \{t_0\})$

Let us prove these three points:

(a) $\boxed{IsTopPrioritySet(T_s \setminus \{t_0\}, tp_0)}$

Not provable yet.

(b) $\boxed{IsFiredSetAux(s', fired', (T_s \setminus \{t_0\}) \setminus tp_0, fset)}$
We know that $(T_s \setminus \{t_0\}) \setminus tp_0 = T_s \setminus (\{t_0\} \cup tp_0)$, and thus
$IsFiredSetAux(s', fired', T_s \setminus (\{t_0\} \cup tp_0), fset)$ is an assumption.

(c) $\forall t' \in T, id_{t'} \in Comps(\Delta)$ s.t. $\gamma(t') = id_{t'}$,
$(t' \in fired \Rightarrow \sigma'(id_{t'})(''f'') = \texttt{true}) \wedge (\sigma'(id_{t'})(''f'') = \texttt{true} \Rightarrow t' \in fired \ \vee \ t' \in T_s \setminus \{t_0\})$
Given a $t' \in T$ and an $id_{t'} \in Comps(\Delta)$ s.t. $\gamma(t') = id_{t'}$, let us show

$(t' \in fired \Rightarrow \sigma'(id_{t'})(''f'') = \texttt{true}) \wedge (\sigma'(id_{t'})(''f'') = \texttt{true} \Rightarrow t' \in fired \ \vee \ t' \in T_s \setminus \{t_0\})$

The proof is in two parts:

i. Assuming that $t' \in fired$, let us show $\boxed{\sigma'(id_{t'})("f") = \texttt{true}.}$
From $t' \in fired$ and EH, $\sigma'(id_{t'})("f") = \texttt{true}.$

ii. Assuming that $\sigma'(id_{t'})("f") = \texttt{true}$, let us show $\boxed{t' \in fired \ \vee \ t' \in T_s \setminus \{t_0\}.}$
Thanks to $\sigma'(id_{t'})("f") = \texttt{true}$ and EH, we know that: $t' \in fired \vee t' \in T_s$.
Case analysis on $t' \in fired \vee t' \in T_s$; there are two cases:

- **CASE** $t' \in fired$.
- **CASE** $t' \in T_s$:
  From $IsTopPrioritySet(T_s, \{t_0\} \cup tp_0)$, we can deduce that $t_0 \in T_s$. Therefore, either $t' \in T_s \setminus \{t_0\}$ or $t' = t_0$.
  In the case where $t' = t_0$, we need to show a contradiction by proving $t' \in Firable(s')$ and $t' \in Sens(s'.M - \sum\limits_{t_i \in Pr(t,fired)} pre(t_i))$ based on $\sigma'(id_{t'})("f") = \texttt{true}$.
  By definition of $id_{t'}$, there exist a $gm_{t'}$, $ipm_{t'}$, $opm_{t'}$ s.t. $\texttt{comp}(id_{t'}, "transition", gm_{t'}, ipm_{t'}, opm_{t'}) \in d.cs$.
  By property of the stabilize relation and $\texttt{comp}(id_{t'}, "transition", gm_{t'}, ipm_{t'}, opm_{t'}) \in d.cs$:
  $$\sigma(id_{t'})("f") = \sigma(id_{t'})("sfa") \,.\, \sigma(id_{t'})("spc") = \texttt{true} \tag{1.2}$$
  From $\sigma(id_{t'})("sfa") = \texttt{true}$, and appealing to Lemma Falling Edge Equal Firable, we can deduce $t' \in Firable(s')$.
  From $\sigma(id_{t'})("spc") = \texttt{true}$, and appealing to Lemma Stabilize Compute Priority Combination After Falling Edge, we can deduce $t' \in Sens(s'.M - \sum\limits_{t_i \in Pr(t,fired)} pre(t_i))$.
  Then, as $t' = t_0$, $\neg(t_0 \in Firable(s') \wedge t_0 \in Sens(s'.M - \sum\limits_{t_i \in Pr(t,fired)} pre(t_i)))$ is a contradiction.

□

Lemma 9 (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, $\sigma_\downarrow$, σ' that verify the hypotheses of Def. 6, then $\forall t \in T, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$, $\forall fired, fired'$, T_s, tp, fset $\subseteq T$ assume that:*

- $IsTopPrioritySet(T_s, \{t\} \cup tp)$
- $ElectFired(s', fired, tp, fired')$
- $FiredAux(s', fired', T_s \setminus \{t\} \cup tp, fset)$
- *EH:* $\forall t' \in T, id_{t'} \in Comps(\Delta)$ *s.t.* $\gamma(t') = id_{t'}$,
  $(t' \in fired \Rightarrow \sigma'(id_{t'})("f") = \texttt{true}) \wedge (\sigma'(id_{t'})("f") = \texttt{true} \Rightarrow t' \in fired \ \vee \ t' \in T_s)$.
- $t \in Firable(s')$

then $t \in Sens(s'.M - \sum\limits_{t_i \in Pr(t,fired)} pre(t_i)) \Leftrightarrow \sigma'(id_t)("spc") = \texttt{true}$

Proof. Given a $t \in T$ and an $id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$, a $fired$, $fired'$, T_s, tp, $fset \subseteq T$ and assuming all the above hypotheses, let us show $\boxed{t \in Sens(s'.M - \sum_{t_i \in Pr(t,fired)} pre(t_i)) \Leftrightarrow \sigma'(id_t)(''spc'') = \texttt{true}.}$

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 stabilize relation and $\texttt{comp}(id_t, ''transition'', gm_t, ipm_t, opm_t) \in d.cs$:

$$\sigma'(id_t)(''spc'') = \prod_{i=0}^{\Delta(id_t)(''ian'')-1} \sigma'(id_t)(''pauths'')[i] \tag{1.3}$$

Rewriting the goal with (1.3):

$$\boxed{t \in Sens(s'.M - \sum_{t_i \in Pr(t,fired)} pre(t_i)) \Leftrightarrow \prod_{i=0}^{\Delta(id_t)(''ian'')-1} \sigma'(id_t)(''pauths'')[i] = \texttt{true}.}$$

Then, the proof is in two parts:

1. $t \in Sens(s'.M - \sum_{t_i \in Pr(t,fired)} pre(t_i)) \Rightarrow \prod_{i=0}^{\Delta(id_t)(''ian'')-1} \sigma'(id_t)(''pauths'')[i] = \texttt{true}$

2. $\prod_{i=0}^{\Delta(id_t)(''ian'')-1} \sigma'(id_t)(''pauths'')[i] = \texttt{true} \Rightarrow t \in Sens(s'.M - \sum_{t_i \in Pr(t,fired)} pre(t_i))$

Let us prove both sides of the equivalence:

1. Assuming that $t \in Sens(s'.M - \sum_{t_i \in Pr(t,fired)} pre(t_i))$, let us show

   $$\boxed{\prod_{i=0}^{\Delta(id_t)(''ian'')-1} \sigma'(id_t)(''pauths'')[i] = \texttt{true}.}$$

   Let us perform case analysis on $input(t)$; there are 2 cases:

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

     By construction, <`input_arcs_number` ⇒1> $\in gm_t$ and <`priority_authorizations(0)` ⇒ `true`> $\in ipm_t$.

     By property of the elaboration relation, we have $\Delta(id_t)(''ian'') = 1$, and by property of the stabilize relation, we have $\sigma'(id_t)(''pauths'')[0] = \texttt{true}$.

     Rewriting the goal with $\Delta(id_t)(''ian'') = 1$ and $\sigma'(id_t)(''pauths'')[0] = \texttt{true}$, and simplifying the goal: tautology.

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

     Then, let us show an equivalent goal:

     $\boxed{\forall i \in [0, \Delta(id_t)(''ian'') - 1], \sigma'(id_t)(''pauths'')[i] = \texttt{true}.}$

     Given an $i \in [0, \Delta(id_t)(''ian'') - 1]$, let us show $\boxed{\sigma'(id_t)(''pauths'')[i] = \texttt{true}.}$

     By construction, <`input_arcs_number` ⇒$|input(t)|$> $\in gm_t$.

     By property of the elaboration relation, we have $\Delta(id_t)(''ian'') = |input(t)|$. Then, we can deduce $i \in [0, |input(t)| - 1]$.

By construction, for all $i \in [0, |input(t)| - 1]$, there exist a $p \in input(t)$ and an $id_p \in Comps(\Delta)$ s.t. $\gamma(p) = id_p$, there exist a gm_p, ipm_p, opm_p s.t. $\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$, and there exist a $j \in [0, |output(p)|]$ and an $id_{ji} \in Sigs(\Delta)$ s.t. $<\texttt{input\_arcs\_valid(i)} \Rightarrow \texttt{id}_{\texttt{ji}}> \in ipm_t$ and $<\texttt{output\_arcs\_valid(j)} \Rightarrow \texttt{id}_{\texttt{ji}}> \in opm_t$. Let us take such a $p \in input(t)$, $id_p \in Comps(\Delta)$, gm_p, ipm_p, opm_p, $j \in [0, |output(p)|]$ and $id_{ji} \in Sigs(\Delta)$.

Now, let us perform case analysis on the nature of the arc connecting p and t; there are 2 cases:

- **CASE** $pre(p,t) = (\omega, \texttt{test})$ or $pre(p,t) = (\omega, \texttt{inhib})$:
  By construction, $<\texttt{priority\_authorizations(i)} \Rightarrow \texttt{true}> \in ipm_t$, and by property of the stabilize relation: $\sigma'(id_t)("pauths")[i] = \texttt{true}$.
- **CASE** $pre(p,t) = (\omega, \texttt{basic})$:
  Let us define $output_c(p) = \{t \in T \mid \exists \omega,\ pre(p,t) = (\omega, \texttt{basic})\}$, the set of output transitions of p that are in conflict. Then, there are two cases, one for each way to solve the conflicts between the output transitions of p:
  * **CASE** For all pair of transitions in $output_c(p)$, all conflicts are solved by mutual exclusion:
    By construction, $<\texttt{priority\_authorizations(i)} \Rightarrow \texttt{true}> \in ipm_t$, and by property of the stabilize relation: $\sigma'(id_t)("pauths")[i] = \texttt{true}$.
  * **CASE** The priority relation is a strict total order over the set $output_c(p)$:
    By construction, there exists an $id'_{ji} \in Sigs(\Delta)$ s.t.
    $<\texttt{priority\_authorizations(i)} \Rightarrow \texttt{id}'_{\texttt{ji}}> \in ipm_t$ and
    $<\texttt{priority\_authorizations(j)} \Rightarrow \texttt{id}'_{\texttt{ji}}> \in opm_p$.
    By property of the stabilize relation, $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$ and $\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$:

$$\sigma'(id_t)("pauths")[i] = \sigma'(id'_{ji}) = \sigma'(id_p)("pauths")[j] \tag{1.4}$$

Rewriting the goal with (1.4): $\boxed{\sigma'(id_p)("pauths")[j] = \texttt{true}.}$
By property of the stabilize relation and $\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$:

$$\sigma'(id_p)("pauths")[j] = (\sigma'(id_p)("sm") \geq \texttt{rsum} + \sigma'(id_p)("oaw")[j]) \tag{1.5}$$

Let us define the $\texttt{rsum}$ term as follows:

$$\texttt{rsum} = \sum_{i=0}^{j-1} \begin{cases} \sigma'(id_p)("oaw")[i] \ \texttt{if}\ \sigma'(id_p)("otf")[i]. \\ \qquad\qquad\qquad\qquad \sigma'(id_p)("oat")[i] = \texttt{basic} \\ 0 \ otherwise \end{cases} \tag{1.6}$$

Rewriting the goal with (1.5): $\boxed{\sigma'(id_p)("sm") \geq \texttt{rsum} + \sigma'(id_p)("oaw")[j]}$
By definition of $t \in Sens(s'.M - \sum\limits_{t_i \in Pr(t,fired)} pre(t_i))$, we have $s'.M(p) \geq \sum\limits_{t_i \in Pr(t,fired)} pre(p, t_i) + \omega$.
Then, there are three points to prove:

(a) $\boxed{s'.M(p) = \sigma'(id_p)("sm")}$

(b) $\boxed{\omega = \sigma'(id_p)("oaw")[j]}$

(c) $\boxed{\sum\limits_{t_i \in Pr(t,fired)} pre(p,t_i) = \texttt{rsum}}$

Let us prove these three points:

(a) $\boxed{s'.M(p) = \sigma'(id_p)("sm")}$

Appealing to Lemma Falling Edge Equal Marking: $s'.M(p) = \sigma'(id_p)("sm")$.

(b) $\boxed{\omega = \sigma'(id_p)("oaw")[j]}$

By construction, and as $pre(p,t) = (\omega, \texttt{basic})$, we have $<$`output_arcs_weights(j)` $\Rightarrow \omega > \in ipm_p$.

By property of the stabilize relation and $\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$: $\omega = \sigma'(id_p)("oaw")[j]$.

(c) $\boxed{\sum\limits_{t_i \in Pr(t,fired)} pre(p,t_i) = \texttt{rsum}}$

Let us replace the left and right term of the equality by their full definition:

$$\boxed{\begin{array}{c}\sum\limits_{t_i \in Pr(t,fired)} \begin{cases} \omega \texttt{ if } pre(p,t_i) = (\omega, \texttt{basic}) \\ 0 \textit{ otherwise} \end{cases} \\ = \\ \sum\limits_{i=0}^{j-1} \begin{cases} \sigma'(id_p)("oaw")[i] \texttt{ if } \sigma'(id_p)("otf")[i]. \\ \qquad\qquad \sigma'(id_p)("oat")[i] = \texttt{basic} \\ 0 \textit{ otherwise} \end{cases}\end{array}}$$

Let us define $f(t_i) = \begin{cases} \omega \texttt{ if } pre(p,t_i) = (\omega, \texttt{basic}) \\ 0 \textit{ otherwise} \end{cases}$ and

$$g(i) = \begin{cases} \sigma'(id_p)("oaw")[i] \texttt{ if } \sigma'(id_p)("otf")[i]. \\ \qquad\qquad \sigma'(id_p)("oat")[i] = \texttt{basic} \\ 0 \textit{ otherwise} \end{cases}$$

Let us reason by induction on the right term of the goal.

BASE CASE: then, we have $i > j - 1$, and then $j = 0$.

$$\boxed{\sum\limits_{t_i \in Pr(t,fired)} \begin{cases} \omega \texttt{ if } pre(p,t_i) = (\omega, \texttt{basic}) \\ 0 \textit{ otherwise} \end{cases} = 0}$$

We know that the priority relation is a strict total order over the transitions of set $output_c(p)$. This ordering is reflected in the ordering of the indexes of output port `priority_authorizations` of place component instances. Thus, in the `priority_authorizations` output port of a place component instance, the element of index 0 is connected to the transition of $output_c(t)$ with the highest firing priority. We know that component id_t is connected to `priority_authorizations(0)` in the output port map of component id_p. By construction, transition t is the transition of $output_c(p)$ with the highest firing priority, i.e, $\nexists t' \in output_c(p)$ s.t. $t' \succ t$.

The following part of the proof is the result of induction over term $\sum_{t_i \in Pr(t,fired)} f(t_i)$. Induction is not detailled here.

For all transition $t_i \in Pr(t, fired)$, either t_i is not in $output_c(p)$, and thus t_i has no effect in the value of the sum term $\sum_{t_i \in Pr(t,fired)} f(t_i)$; or, $t_i \in output_c(p)$. Then, by definition of $t_i \in Pr(t, fired)$, $t_i \succ t$, which is contradiction with $\nexists t' \in output_c(p)$ s.t. $t' \succ t$.

INDUCTIVE CASE: then, $0 \leq j - 1$, and thus $j > 0$.

For all $Pr' \subseteq T$, $g(0) + \sum_{t_i \in Pr'} f(t_i) = g(0) + \sum_{i=1}^{j-1} g(i)$

$$\boxed{\sum_{t_i \in Pr(t,fired)} f(t_i) = g(0) + \sum_{i=1}^{j-1} g(i).}$$

By definition of $g(0)$:

$$\boxed{\sum_{t_i \in Pr(t,fired)} f(t_i) = \begin{cases} \sigma'(id_p)(''oaw'')[0] \; \texttt{if} \; \sigma'(id_p)(''otf'')[0]. \\ \qquad\qquad\qquad \sigma'(id_p)(''oat'')[0] = \texttt{basic} \\ 0 \; otherwise \end{cases} + \sum_{i=1}^{j-1} g(i).}$$

Case analysis on the value of $\sigma'(id_p)(''otf'')[0] \; . \; \sigma'(id_p)(''oat'')[0] = \texttt{basic}$:

In the case where $(\sigma'(id_p)(''otf'')[0] \; . \; \sigma'(id_p)(''oat'')[0] = \texttt{basic}) = \texttt{false}$, then $g(0) = 0$, and we can use the induction hypothesis with $Pr' = Pr(t, fired)$ to prove the goal.

In the case where $(\sigma'(id_p)(''otf'')[0] \; . \; \sigma'(id_p)(''oat'')[0] = \texttt{basic}) = \texttt{true}$, then $g(0) = \sigma'(id_p)(''oaw'')[0]$:

$$\boxed{\sum_{t_i \in Pr(t,fired)} f(t_i) = \sigma'(id_p)(''oaw'')[0] + \sum_{i=1}^{j-1} g(i).}$$

By construction, and knowing that $j > 0$ and that the priority relation is a strict total order over the set $output_c(p)$, there exist a $t_0 \in output_c(p)$ s.t. $t_0 \succ t$. Moreover, there exist an $id_{t_0} \in Comps(\Delta)$ s.t. $\gamma(t_0) = id_{t_0}$, and by definition of id_{t_0}, there exist gm_{t_0}, ipm_{t_0} and opm_{t_0} s.t. $\texttt{comp}(id_{t_0}, ''transition'', gm_{t_0}, ipm_{t_0}, opm_{t_0}) \in d.cs$. Finally, there exist an $id_{ft_0} \in Sigs(\Delta)$ s.t. $<\texttt{fired} \Rightarrow \texttt{id}_{\texttt{ft}_0}> \in opm_{t_0}$ and $<\texttt{output\_transitions\_fired}(0) \Rightarrow \texttt{id}_{\texttt{ft}_0}> \in ipm_p$.

By property of the stabilize relation, $\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$ and $\texttt{comp}(id_{t_0}, "transition", gm_{t_0}, ipm_{t_0}, opm_{t_0}) \in d.cs$:

$$\sigma'(id_{t_0})("f") = \sigma'(id_{ft_0}) = \sigma'(id_p)("otf")[0] = \texttt{true} \tag{1.7}$$

From EH and $\sigma'(id_{t_0})("f") = \texttt{true}$, we have either $t_0 \in fired$ or $t_0 \in T_s$.

❑ In the case where $t_0 \in fired$, then, by definition of $\sum$:

$$\boxed{f(t_0) + \sum_{t_i \in Pr(t,fired) \setminus \{t_0\}} f(t_i) = \sigma'(id_p)("oaw")[0] + \sum_{i=1}^{j-1} g(i).}$$

By definition of $t_0 \in output_c(p)$, there exists $\omega \in \mathbb{N}^*$ s.t. $pre(p, t_0) = (\omega, \texttt{basic})$. Thus, we have $f(t_0) = \omega$
By construction, $<\texttt{output\_arcs\_weights}(0) \Rightarrow \omega>$, and by property of the stabilize relation, we have $\sigma'(id_p)("oaw")[0] = \omega$. Thus, we can deduce that $g(0) = \omega$, and then we can rewrite the goal in order to apply the induction hypothesis with $Pr' = Pr(t, fired) \setminus \{t_0\}$.

❑ In the case where $t_0 \in T_s$:
As t is a top-priority transition in set T_s, there exists no transition $t' \in T_s$ s.t. $t' \succ t$. Contradicts $t_0 \succ t$.

2. Assuming that $\prod_{i=0}^{\Delta(id_t)("ian")-1} \sigma'(id_t)("pauths")[i] = \texttt{true}$, let us show

$$\boxed{t \in Sens(s'.M - \sum_{t_i \in Pr(t,fired)} pre(t_i)).}$$

By definition of $t \in Sens(s'.M - \sum_{t_i \in Pr(t,fired)} pre(t_i))$:

$$\boxed{\begin{array}{l} \forall p \in P, \omega \in \mathbb{N}^*, \\ \big((pre(p,t) = (\omega, \texttt{basic}) \vee pre(p,t) = (\omega, \texttt{test})) \Rightarrow s'.M(p) - \sum_{t_i \in Pr(t,fired)} pre(p,t_i) \geq \omega\big) \\ \wedge \big(pre(p,t) = (\omega, \texttt{inhib}) \Rightarrow s'.M(p) - \sum_{t_i \in Pr(t,fired)} pre(p,t_i) < \omega\big) \end{array}}$$

Given a $p \in P$ and an $\omega \in \mathbb{N}^*$, let us show

$$\boxed{\begin{array}{l} \big((pre(p,t) = (\omega, \texttt{basic}) \vee pre(p,t) = (\omega, \texttt{test})) \Rightarrow s'.M(p) - \sum_{t_i \in Pr(t,fired)} pre(p,t_i) \geq \omega\big) \\ \wedge \big(pre(p,t) = (\omega, \texttt{inhib}) \Rightarrow s'.M(p) - \sum_{t_i \in Pr(t,fired)} pre(p,t_i) < \omega\big) \end{array}}$$

By construction, there exists an $id_p \in Comps(\Delta)$ s.t. $\gamma(p) = id_p$. By definition of id_p, there exist gm_p, ipm_p, opm_p s.t. $\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$.

There are three different cases:

(a) Assuming that $pre(p,t) = (\omega, \texttt{test})$, let us show $\boxed{s'.M(p) - \sum_{t_i \in Pr(t,fired)} pre(p,t_i) \geq \omega.}$

Then, 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 $\sum_{t_i \in Pr(t,fired)} pre(p,t_i) = 0$.
Then, the new goal is $s'.M(p) \geq \omega$.
Knowing that $t \in Firable(s')$, thus, $t \in Sens(s'.M)$, thus, we have $s'.M(p) \geq \omega$.

(b) Assuming that $pre(p,t) = (\omega, \texttt{inhib})$, let us show $\boxed{s'.M(p) - \sum_{t_i \in Pr(t,fired)} pre(p,t_i) < \omega.}$

Use the same strategy as above.

(c) Assuming that $pre(p,t) = (\omega, \texttt{basic})$, let us show $\boxed{s'.M(p) - \sum_{t_i \in Pr(t,fired)} pre(p,t_i) \geq \omega.}$

Then, there are two 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 $\sum_{t_i \in Pr(t,fired)} pre(p,t_i) = 0$.
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 $output_c(p)$.
By construction, there exists $id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$. 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 construction, there exist $j \in [0, |input(t)| - 1]$, $k \in [0, |output(t)| - 1]$, and $id_{kj} \in Sigs(\Delta)$ s.t. $<\texttt{priority\_authorizations(j)} \Rightarrow \texttt{id}_{\texttt{kj}}> \in ipm_t$ and $<\texttt{priority\_authorizations(k)} \Rightarrow \texttt{id}_{\texttt{kj}}> \in opm_p$. Let us take such an j, k and id_{kj}.
From $\prod_{i=0}^{\Delta(id_t)("ian")-1} \sigma'(id_t)("pauths")[i] = \texttt{true}$, we can deduce that for all $i \in [0, \Delta(id_t)("ian") - 1]$, $\sigma'(id_t)("pauths")[i] = \texttt{true}$.
By construction, $<\texttt{input\_arcs\_number} \Rightarrow |input(t)|> \in gm_t$, and by property of the elaboration relation, we have $\Delta(id_t)("ian") = |input(t)|$. Then, from $j \in [0, |input(t)| - 1]$, we can deduce $j \in [0, \Delta(id_t)("ian") - 1]$. And, from $\forall i \in [0, \Delta(id_t)("ian") - 1]$, $\sigma'(id_t)("pauths")[i] = \texttt{true}$, we can deduce $\sigma'(id_t)("pauths")[j] = \texttt{true}$.
By property of the stabilize relation, $\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$ and $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$:

$$\sigma'(id_p)("pauths")[k] = \sigma'(id_{kj})\sigma'(id_t)("pauths")[j] = \texttt{true} \tag{1.8}$$

By property of the stabilize relation and $\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$:

$$\sigma'(id_p)("pauths")[k] = (\sigma'(id_p)("sm") \geq \texttt{rsum} + \sigma'(id_p)("oaw")[k]) \tag{1.9}$$

Let us define the rsum term as follows:

$$\texttt{rsum} = \sum_{i=0}^{k-1} \begin{cases} \sigma'(id_p)(''oaw'')[i] \texttt{ if } \sigma'(id_p)(''otf'')[i]. \\ \qquad\qquad\qquad\qquad \sigma'(id_p)(''oat'')[i] = \texttt{basic} \\ 0 \; otherwise \end{cases} \tag{1.10}$$

From (1.8) and (1.9), we can deduce that $\sigma'(id_p)(''sm'') \geq \texttt{rsum} + \sigma'(id_p)(''oaw'')[k]$. Then, there are three points to prove:

A. $\boxed{s'.M(p) = \sigma'(id_p)(''sm'')}$

B. $\boxed{\omega = \sigma'(id_p)(''oaw'')[k]}$

C. $\boxed{\sum_{t_i \in Pr(t,fired)} pre(p, t_i) = \texttt{rsum}}$

See 1 for the remainder of the 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
  - ...

Appendix C

Semantic preservation proof

C.1 Initial States

Definition 7 (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:*

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

- σ_0 *is the initial state of* Δ*:* $\Delta, \sigma_e \vdash d.cs \xrightarrow{init} \sigma_0$

Lemma 10 (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. 7, then* $\gamma \vdash s_0 \sim \sigma_0$*.*

Proof. By definition of **??**, 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.

□

C.1.1 Initial states and marking

Lemma 11 (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. 7, 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 $\sigma_0(id_p) = \sigma_p^0$, $s_0.M(p) = \sigma_p^0(''s\_marking'')$.*

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

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

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

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

By definition of s_0, rewriting $s_0.M(p)$ as $M_0(p)$, $\sigma_p^0(''initial\_marking'') = s_0.M(p)$.

□

C.1.2 Initial states and time counters

Lemma 12 (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. 7, 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 $\sigma_0(id_t) = \sigma_t^0$,*
$upper(I_s(t)) = \infty \land s_0.I(t) \leq lower(I_s(t)) \Rightarrow s_0.I(t) = \sigma_t^0(''s\_tc'') \land$
$upper(I_s(t)) = \infty \land s_0.I(t) > lower(I_s(t)) \Rightarrow \sigma_t^0(''s\_tc'') = lower(I_s(t)) \land$
$upper(I_s(t)) \neq \infty \land s_0.I(t) > upper(I_s(t)) \Rightarrow \sigma_t^0(''s\_tc'') = upper(I_s(t)) \land$
$upper(I_s(t)) \neq \infty \land s_0.I(t) \leq upper(I_s(t)) \Rightarrow s_0.I(t) = \sigma_t^0(''s\_tc'')$.

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. $\boxed{upper(I_s(t)) = \infty \land s_0.I(t) \leq lower(I_s(t)) \Rightarrow s_0.I(t) = \sigma_t^0(''s\_tc'')}$

2. $\boxed{upper(I_s(t)) = \infty \land s_0.I(t) > lower(I_s(t)) \Rightarrow \sigma_t^0(''s\_tc'') = lower(I_s(t))}$

3. $\boxed{upper(I_s(t)) \neq \infty \wedge s_0.I(t) > upper(I_s(t)) \Rightarrow \sigma_t^0(''s\_tc'') = upper(I_s(t))}$

4. $\boxed{upper(I_s(t)) \neq \infty \wedge s_0.I(t) \leq upper(I_s(t)) \Rightarrow s_0.I(t) = \sigma_t^0(''s\_tc'')}$

By definition of id_t, there exist gm_t, ipm_t, opm_t s.t. $\texttt{comp}(id_t, ''transition'', gm_t, ipm_t, opm_t) \in d.cs$.

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 $\sigma_t^0(''s\_tc'') = 0.$

□

C.1.3 Initial states and reset orders

Lemma 13 (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. 7, 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. $\mathtt{comp}(id_t, ''transition'', gm_t, ipm_t, opm_t) \in d.cs$, 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 $\mathtt{comp}(id_t, ''transition'', gm_t, ipm_t, opm_t) \in d.cs$, then $<\mathtt{id_t.in\_arcs\_nb} \Rightarrow |input(t)|> \in gm_t$.

  By property of the $\mathcal{H}$-VHDL elaboration relation and $<\mathtt{id_t.in\_arcs\_nb} \Rightarrow |input(t)|> \in gm_t$, 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 $<\mathtt{id_t.rt(i)} \Rightarrow id_{ji}> \in ipm_t$, 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 $\mathtt{comp}(id_p, "place", gm_p, ipm_p, opm_p)$ $d.cs$, 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 $< \mathtt{id_p.rtt(j)} \Rightarrow id_{ji} >\in opm_p$, 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_time_evaluation`), and $\mathtt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$, 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 $\mathtt{comp}(id_p, "transition", gm_p, ipm_p, opm_p) \in d.cs$, then $<\mathtt{id_p.out\_arcs\_nb} \Rightarrow |output(p)|> \in gm_p$.

By property of the $\mathcal{H}$-VHDL elaboration relation and $<\mathtt{id_p.out\_arcs\_nb} \Rightarrow |output(p)|> \in gm_p$, 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 $\sigma_p^0("rtt")(j) = false$.

□

C.1.4 Initial states and condition values

Lemma 14 (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. 7, 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 $false$, $\sigma_0(id_c) = false.$

□

C.1.5 Initial states and action executions

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

Lemma 15 (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. 7, 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 $false$, $\sigma_0(id_a) = false.$

□

C.1.6 Initial states and function executions

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

Lemma 16 (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. 7, 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 $false$, $\sigma_0(id_f) = false$.

□

C.2 First Rising Edge

Definition 8 (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 17 (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. 8, 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 $\sigma(id_p) = \sigma_p$, $s_0.M(p) = \sigma_p(\text{"}s\_marking\text{"})$.
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 $\sigma(id_t) = \sigma_t$,
$upper(I_s(t)) = \infty \wedge s_0.I(t) \leq lower(I_s(t)) \Rightarrow s_0.I(t) = \sigma_t(\text{"}s\_tc\text{"}) \wedge$
$upper(I_s(t)) = \infty \wedge s_0.I(t) > lower(I_s(t)) \Rightarrow \sigma_t(\text{"}s\_tc\text{"}) = lower(I_s(t)) \wedge$
$upper(I_s(t)) \neq \infty \wedge s_0.I(t) > upper(I_s(t)) \Rightarrow \sigma_t(\text{"}s\_tc\text{"}) = upper(I_s(t)) \wedge$
$upper(I_s(t)) \neq \infty \wedge s_0.I(t) \leq upper(I_s(t)) \Rightarrow s_0.I(t) = \sigma_t(\text{"}s\_tc\text{"})$.
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 $\sigma(id_t) = \sigma_t$,
$s_0.reset_t(t) = \sigma_t(\text{"}s\_reinit\_time\_counter\text{"})$.
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. $\forall t \in T_i, id_t \in Comps(\Delta)$ $s.t.$ $\gamma(t) = id_t$,
$t \in Sens(s.M) \Leftrightarrow \sigma(id_t)(\text{"}s\_enabled\text{"}) = \texttt{true}$.

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 $conds(t) = \{c \in \mathcal{C} \mid \mathbb{C}(t,c) = 1 \vee \mathbb{C}(t,c) = -1\}$.

– 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.

□

C.2.1 First rising edge and marking

Lemma 18 (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. 8, then $\forall p \in P, id_p \in Comps(\Delta), \sigma_p \in \Sigma(\Delta(id_p))\ s.t.\ \gamma(p) = id_p$ and $\sigma(id_p) = \sigma_p$, $s_0.M(p) = \sigma_p(\text{"}s\_marking\text{"})$.*

Proof. Given a p, id_p, σ_p s.t. $\gamma(p) = id_p$ and $\sigma(id_p) = \sigma_p$, let us show that $\boxed{s_0.M(p) = \sigma_p(\text{"}s\_marking\text{"}).}$ 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 elaboration relation, the $\mathcal{H}$-VHDL initialization relation, the $\texttt{Inject}_\uparrow$ relation, the $\mathcal{H}$-VHDL rising edge relation and $\texttt{comp}(id_p, \text{"}place\text{"}, gm_p, ipm_p, opm_p) \in d.cs$, 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 $\texttt{comp}(id_p, \text{"}place\text{"}, gm_p, ipm_p, opm_p) \in d.cs$, then
$\sigma_p^r(\text{"}s\_marking\text{"}) = \sigma_p^{injr}(\text{"}s\_marking\text{"}) + \sigma_p^{injr}(\text{"}s\_input\_token\_sum\text{"}) - \sigma_p^{injr}(\text{"}s\_output\_token\_sum\text{"})$.

> 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 $\texttt{comp}(id_p, \text{"}place\text{"}, gm_p, ipm_p, opm_p) \in d.cs$, then $\sigma_p^r(\text{"}s\_marking\text{"}) = \sigma_p(\text{"}s\_marking\text{"})$.

> 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, $s_0.M(p) = \sigma_p^0(''s\_marking'')$. □

C.2.2 First rising edge and time counters

Lemma 19 (First Rising Edge Equal Time Counters). *For all $sitpn, d, \gamma, \Delta, \sigma_e, \sigma_0, \sigma_i, \sigma_\uparrow, \sigma, E_c, E_p, \tau$ that verify the hypotheses of Def. 8, 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, $s_0.I(t) = \sigma_t^0(''s\_tc'').$

□

C.2.3 First rising edge and reset orders

Lemma 20 (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. 8, then*
$\forall t \in T, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$,
$s_0.reset_t(t) = \sigma(id_t)(''s\_reinit\_time\_counter'')$.

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 $\sigma(id_t)(''srtc'') = \sum\limits_{i=0}^{\Delta(id_t)(''input\_arcs\_number'')-1} \sigma(id_t)(''reinit\_time'')[i]$.

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

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

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

  By construction, $<\texttt{input\_arcs\_number} \Rightarrow \texttt{1}> \in gm_t$, 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, $s_0.reset_t(t) = \sum\limits_{i=0}^{\Delta(''ian'')-1} \sigma(id_t$

- **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. $<\texttt{reinit\_transition\_time(j)} \Rightarrow \texttt{id}_{\texttt{ji}}> \in opm_p$ and $<\texttt{reinit\_time(i)} \Rightarrow \texttt{id}_{\texttt{ji}}> \in ipm_t$.

  By property of the $\mathcal{H}$-VHDL stabilize relation, $<\texttt{reinit\_transition\_time(j)} \Rightarrow \texttt{id}_{\texttt{ji}}> \in opm_p$ and $<\texttt{reinit\_time(i)} \Rightarrow \texttt{id}_{\texttt{ji}}> \in ipm_t$, 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, $false = false$

□

C.2.4 First rising edge and action executions

Lemma 21 (First Rising Edge Equal Action Executions). *For all $sitpn, d, \gamma, \Delta, \sigma_e, \sigma_0, \sigma_i, \sigma_{\uparrow}, \sigma, E_c, E_p, \tau$ that verify the hypotheses of Def. 8, then*
$\forall a \in \mathcal{A}, id_a \in Outs(\Delta)$ s.t. $\gamma(a) = id_a$, $s_0.ex(a) = \sigma(id_a)$.

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 $false$, $false = false.$

□

C.2.5 First rising edge and function executions

Lemma 22 (First Rising Edge Equal Function Executions). *For all $sitpn, d, \gamma, \Delta, \sigma_e, \sigma_0, \sigma_i, \sigma_{\uparrow}, \sigma, E_c, E_p, \tau$ that verify the hypotheses of Def. 8, then*
$\forall f \in \mathcal{F}, id_f \in Outs(\Delta)$ s.t. $\gamma(f) = id_f$, $s_0.ex(f) = \sigma(id_f)$.

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 $\mathtt{ps}(''function'', \emptyset, sl, ss) \in d.cs$.

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, $\mathtt{id_f} \Leftarrow \mathtt{false} \in ss_{\uparrow}$ 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, $\mathtt{id_f} \Leftarrow \mathtt{id_{ft_0}} + \cdots + \mathtt{id_{ft_n}} \in ss_{\uparrow}$ 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 $\mathtt{Inject}_{\uparrow}$, 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. $\mathtt{comp}(id_{t_i}, ''transition'', gm_{t_i}, ipm_{t_i}, opm_{t_i}) \in d.cs$.

  By construction, $<\mathtt{fired} \Rightarrow \mathtt{id_{ft_i}}> \in opm_{t_i}$, 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
  $false = false.$

□

C.3 Rising Edge

Definition 9 (Rising Edge Hypotheses). *Given an $sitpn \in SITPN$, $d \in design$, $\gamma \in WM(sitpn, d)$, $E_c \in \mathbb{N} \to \mathcal{C} \to \mathbb{B}$, $\Delta \in ElDesign(d, \mathcal{D}_{\mathcal{H}})$, $E_p \in (\mathbb{N} \times \{\uparrow, \downarrow\}) \to Ins(\Delta) \to value$, $\tau \in \mathbb{N}$, $s, s' \in S(sitpn)$, $\sigma_e, \sigma, \sigma_i, \sigma_{\uparrow}, \sigma' \in \Sigma(\Delta)$, 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* $\mathcal{D}_{\mathcal{H}}, \Delta, \sigma_i \vdash \mathrm{d.cs} \xrightarrow{\uparrow} \sigma_{\uparrow}$ *and* $\mathcal{D}_{\mathcal{H}}, \Delta, \sigma_{\uparrow} \vdash \mathrm{d.cs} \xrightarrow{\rightsquigarrow} \sigma'$
- *State* σ *is a stable design state:* $\mathcal{D}_{\mathcal{H}}, \Delta, \sigma \vdash \mathrm{d.cs} \xrightarrow{comb} \sigma$

Lemma 23 (Rising Edge). *For all* $sitpn$, d, γ, E_c, E_p, τ, Δ, σ_e, s, s', σ, σ_i, $\sigma_{\uparrow}$, σ' *that verify the hypotheses of Def. 9, 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. $\forall t \in T_i, id_t \in Comps(\Delta)\ s.t.\ \gamma(t) = id_t,$
   $\big(upper(I_s(t)) = \infty \wedge s'.I(t) \leq lower(I_s(t)) \Rightarrow s'.I(t) = \sigma'(id_t)(''s\_time\_counter'')\big)$
   $\wedge \big(upper(I_s(t)) = \infty \wedge s'.I(t) > lower(I_s(t)) \Rightarrow \sigma'(id_t)(''s\_time\_counter'') = lower(I_s(t))\big)$
   $\wedge \big(upper(I_s(t)) \neq \infty \wedge s'.I(t) > upper(I_s(t)) \Rightarrow \sigma'(id_t)(''s\_time\_counter'') = upper(I_s(t))\big)$
   $\wedge \big(upper(I_s(t)) \neq \infty \wedge s'.I(t) \leq upper(I_s(t)) \Rightarrow s'.I(t) = \sigma'(id_t)(''s\_time\_counter'')\big)$.
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. $\forall t \in T, id_t \in Comps(\Delta)\ s.t.\ \gamma(t) = id_t, t \in Sens(s'.M) \Leftrightarrow \sigma'(id_t)(''s\_enabled'') = \mathtt{true}$.
7. $\forall t \in T, id_t \in Comps(\Delta)\ s.t.\ \gamma(t) = id_t, t \notin Sens(s'.M) \Leftrightarrow \sigma'(id_t)(''s\_enabled'') = \mathtt{false}$.
8. $\forall t \in T, id_t \in Comps(\Delta)\ s.t.\ \gamma(t) = id_t,$
   $$\sigma'(id_t)(''s\_condition\_combination'') = \prod_{c \in conds(t)} \begin{cases} E_c(\tau, c) & if\ \mathbb{C}(t,c) = 1 \\ \mathtt{not}(E_c(\tau, c)) & if\ \mathbb{C}(t,c) = -1 \end{cases}$$
   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 Lemma Rising Edge Equal Time Counters lemma to solve 2.
- Apply Lemma Rising Edge Equal Reset Orders to solve 3.
- Apply Lemma Rising Edge Equal Action Executions to solve 4.
- Apply Lemma Rising Edge Equal Function Executions to solve 5.

- Apply Lemma Rising Edge Equal Sensitized to solve 6.
- Apply Lemma Rising Edge Equal Not Sensitized to solve 7.
- Apply Lemma Rising Edge Equal Condition Combination to solve 8.

□

C.3.1 Rising Edge and Marking

Lemma 24 (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' that verify the hypotheses of Def. 9, then $\forall p, id_p$ s.t. $\gamma(p) = id_p$ and $\sigma'(id_p) = \sigma'_p$, $s'.M(p) = \sigma'_p(''s\_marking'')$.*

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:

$$s'.M(p) = s.M(p) - \sum_{t \in Fired(s)} pre(p,t) + \sum_{t \in Fired(s)} post(t,p) \tag{C.1}$$

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

$$\begin{aligned} \sigma'(id_p)(''sm'') = \sigma(id_p)(''sm'') - \sigma(id_p)(''s\_output\_token\_sum'') \\ +\sigma(id_p)(''s\_input\_token\_sum'') \end{aligned} \tag{C.2}$$

By the definition of Post Falling Edge State Similarity relation:

$$s.M(p) = \sigma(id_p)(''sm'') \tag{C.3}$$

$$\sum_{t \in Fired(s)} pre(p,t) = \sigma(id_p)(''sots'') \tag{C.4}$$

$$\sum_{t \in Fired(s)} post(t,p) = \sigma(id_p)(''sits'') \tag{C.5}$$

Rewriting the goal with C.1, C.2, C.3, C.4 and C.5, tautology .

□

C.3.2 Rising edge and condition combination

Lemma 25 (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' that verify the hypotheses of Def. 9, then*
$\forall t \in T, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$,

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

where $conds(t) = \{c \in \mathcal{C} \mid \mathbb{C}(t,c) = 1 \vee \mathbb{C}(t,c) = -1\}$.

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

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

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

$$\sigma'(id_t)(''scc'') = \prod_{i=0}^{\Delta(id_t)(''conditions\_number'')-1} \sigma'(id_t)(''input\_conditions'')[i] \tag{C.6}$$

Rewriting the goal with C.6,

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

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

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

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

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

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

  $$\Delta(id_t)(''cn'') = 1 \tag{C.7}$$
  $$\sigma'(id_t)(''ic'')[0] = \texttt{true} \tag{C.8}$$

  Rewriting the goal with C.7 and C.8, 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:

  $$\Delta(id_t)(''cn'') = |conds(t)| \tag{C.9}$$

  Rewriting the goal with (C.9),

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

  Applying Theorem **??**, there are two points to prove:

  1. $|conds(t)| = |conds(t)|$
  2. $\exists$ an injection $\iota \in [0, |conds(t)| - 1] \rightarrow conds(t)$ s.t.

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

  By construction, there exists a bijection $\beta \in [0, |conds(t)| - 1] \rightarrow conds(t)$ such that for all $i \in [0, |conds(t)| - 1]$, there exists an $id_c \in Ins(\Delta)$ and:

- $\gamma(\beta(i)) = id_c$
- $\mathbb{C}(t, \beta(i)) = 1$ implies $<\texttt{input\_conditions(i)} \Rightarrow \texttt{id}_\texttt{c}> \in ipm_t$
- $\mathbb{C}(t, \beta(i)) = -1$ implies $<\texttt{input\_conditions(i)} \Rightarrow \texttt{not id}_\texttt{c}> \in ipm_t$

Let us take such a bijection β to prove the goal. Then, given an $i \in [0, |conds(t)| - 1]$, let us show

$$\boxed{\sigma'(id_t)(''ic'')[i] = \begin{cases} E_c(\tau, \beta(i)) & \textit{if } \mathbb{C}(t, \beta(i)) = 1 \\ \texttt{not}(E_c(\tau, \beta(i))) & \textit{if } \mathbb{C}(t, \beta(i)) = -1 \end{cases}}$$

By definition of $\beta(i) \in conds(t)$:

$$\mathbb{C}(t, \beta(i)) = 1 \vee \mathbb{C}(t, \beta(i)) = -1 \tag{C.10}$$

Case analysis on (C.10):

- **CASE** $\mathbb{C}(t, \beta(i)) = 1$: $\boxed{\sigma'(id_t)(''ic'')[i] = E_c(\tau, \beta(i))}$

  By property of β, there exists $id_c \in Ins(\Delta)$ s.t. $\gamma(\beta(i)) = id_c$ and $<\texttt{input\_conditions(i)} \Rightarrow \texttt{id}_\texttt{c}> \in ipm_t$.
  By property of the stabilize relation and $<\texttt{input\_conditions(i)} \Rightarrow \texttt{id}_\texttt{c}> \in ipm_t$:

  $$\sigma'(id_t)(''ic'')[i] = \sigma'(id_c) \tag{C.11}$$

  By property of the $\mathcal{H}$-VHDL rising edge and stabilize relations, and $id_c \in Ins(\Delta)$:

  $$\sigma'(id_c) = \sigma_i(id_c) \tag{C.12}$$

  By property of the $\texttt{Inject}_\uparrow$ relation and $id_c \in Ins(\Delta)$:

  $$\sigma_i(id_c) = E_p(\tau, \uparrow)(id_c) \tag{C.13}$$

  By property of $\gamma \vdash E_p \overset{env}{=} E_c$:

  $$E_p(\tau, \uparrow)(id_c) = E_c(\tau, c) \tag{C.14}$$

  Rewriting the goal with (C.11), (C.12), (C.13), (C.14), tautology.

- **CASE** $\mathbb{C}(t, c) = -1$: $\boxed{\sigma'(id_t)(''ic'')[i] = \texttt{not } E_c(\tau, \beta(i))}$
  By property of β, there exists $id_c \in Ins(\Delta)$ s.t. $\gamma(\beta(i)) = id_c$ and $<\texttt{input\_conditions(i)} \Rightarrow \texttt{not id}_\texttt{c}> \in ipm_t$.
  By property of the stabilize relation and $<\texttt{input\_conditions(i)} \Rightarrow \texttt{not id}_\texttt{c}> \in ipm_t$:

  $$\sigma'(id_t)(''ic'')[i] = \texttt{not } \sigma'(id_c) \tag{C.15}$$

  Then, equations (C.12), (C.13) and (C.14) also hold this case.
  Rewriting the goal with (C.15), (C.12), (C.13) and (C.14), tautology.

□

C.3.3 Rising edge and time counters

Lemma 26 (Rising Edge Equal Time Counters). *For all $sitpn$, d, γ, E_c, E_p, τ, Δ, σ_e, s, s', σ, σ_i, $\sigma_\uparrow$, σ' that verify the hypotheses of Def. 9, then*
$\forall t \in T_i, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$,
$(upper(I_s(t)) = \infty \land s'.I(t) \leq lower(I_s(t)) \Rightarrow s'.I(t) = \sigma'(id_t)(''s\_time\_counter''))$
$\land(upper(I_s(t)) = \infty \land s'.I(t) > lower(I_s(t)) \Rightarrow \sigma'(id_t)(''s\_time\_counter'') = lower(I_s(t)))$
$\land(upper(I_s(t)) \neq \infty \land s'.I(t) > upper(I_s(t)) \Rightarrow \sigma'(id_t)(''s\_time\_counter'') = upper(I_s(t)))$
$\land(upper(I_s(t)) \neq \infty \land s'.I(t) \leq upper(I_s(t)) \Rightarrow s'.I(t) = \sigma'(id_t)(''s\_time\_counter'')).$

Proof. Given a $t \in T_i$ and an $id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$, let us show

$$\boxed{\begin{array}{l}(upper(I_s(t)) = \infty \land s'.I(t) \leq lower(I_s(t)) \Rightarrow s'.I(t) = \sigma'(id_t)(''s\_time\_counter''))\\ \land(upper(I_s(t)) = \infty \land s'.I(t) > lower(I_s(t)) \Rightarrow \sigma'(id_t)(''s\_time\_counter'') = lower(I_s(t)))\\ \land(upper(I_s(t)) \neq \infty \land s'.I(t) > upper(I_s(t)) \Rightarrow \sigma'(id_t)(''s\_time\_counter'') = upper(I_s(t)))\\ \land(upper(I_s(t)) \neq \infty \land s'.I(t) \leq upper(I_s(t)) \Rightarrow s'.I(t) = \sigma'(id_t)(''s\_time\_counter''))\end{array}}$$

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, there are 4 points to show:

1. $\boxed{upper(I_s(t)) = \infty \land s'.I(t) \leq lower(I_s(t)) \Rightarrow s'.I(t) = \sigma'(id_t)(''s\_time\_counter'')}$

   Assuming $upper(I_s(t)) = \infty$ and $s'.I(t) \leq lower(I_s(t))$, let us show $\boxed{s'.I(t) = \sigma'(id_t)(''s\_time\_counter'').}$

   By property of the $\texttt{Inject}_\uparrow$, $\mathcal{H}$-VHDL rising edge and stabilize relations, and $\texttt{comp}(id_t, ''transition'', gm_t, ipm_t, opm_t) \in d.cs$:

   $$\sigma'(id_t)(''s\_time\_counter'') = \sigma(id_t)(''s\_time\_counter'') \tag{C.16}$$

   By property of $\gamma \vdash s \overset{\downarrow}{\sim} \sigma$:

   $$s.I(t) = \sigma(id_t)(''s\_time\_counter'') \tag{C.17}$$

   Rewriting the goal with (C.16) and (C.17), tautology.

2. $\boxed{upper(I_s(t)) = \infty \land s'.I(t) > lower(I_s(t)) \Rightarrow \sigma'(id_t)(''s\_time\_counter'') = lower(I_s(t).}$

   Proved in the same fashion as 1.

3. $\boxed{upper(I_s(t)) \neq \infty \land s'.I(t) > upper(I_s(t)) \Rightarrow \sigma'(id_t)(''s\_time\_counter'') = upper(I_s(t).}$

   Proved in the same fashion as 1.

4. $\boxed{upper(I_s(t)) \neq \infty \land s'.I(t) \leq upper(I_s(t)) \Rightarrow s'.I(t) = \sigma'(id_t)(''s\_time\_counter'')}$

   Proved in the same fashion as 1.

□

C.3.4 Rising edge and reset orders

Lemma 27 (Rising Edge Equal Reset Orders). *For all* $sitpn$, d, γ, E_c, E_p, τ, Δ, σ_e, s, s', σ, σ_i, $\sigma_\uparrow$, σ' *that verify the hypotheses of Def. 9, then*
$\forall t \in T_i, id_t \in Comps(\Delta)\ s.t.\ \gamma(t) = id_t, s'.reset_t(t) = \sigma'(id_t)("s\_reinit\_time\_counter")$

Proof. Given a $t \in T_i$ and an $id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$, let us show
$\boxed{s'.reset_t(t) = \sigma'(id_t)("s\_reinit\_time\_counter").}$
By definition of id_t, there exist gmt_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 $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$:

$$\sigma'(id_t)("srtc") = \sum_{i=0}^{\Delta(id_t)("input\_arcs\_number")-1} \sigma'(id_t)("reinit\_time")[i] \tag{C.18}$$

Rewriting the goal with (C.18), $\boxed{s'.reset_t(t) = \sum_{i=0}^{\Delta(id_t)("ian")-1} \sigma'(id_t)("rt")[i].}$

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

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

  By construction, $<\texttt{input\_arcs\_number} \Rightarrow \texttt{1}> \in gm_t$, and by property of the elaboration relation:
  $$\Delta(id_t)("ian") = 1 \tag{C.19}$$

  By construction, there exists an $id_{ft} \in Sigs(\Delta)$ s.t. $<\texttt{reinit\_time(0)} \Rightarrow \texttt{id}_{\texttt{ft}}> \in ipm_t$ and $<\texttt{fired} \Rightarrow \texttt{id}_{\texttt{ft}}> \in opm_t$, and by property of the $\mathcal{H}$-VHDL stabilize relation and
  $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$:

  $$\begin{aligned}
  \sigma'(id_t)("rt")[0] &= \sigma'(id_{ft}) && \text{(C.20)}\\
  \sigma'(id_{ft}) &= \sigma'(id_t)("fired") && \text{(C.21)}\\
  \sigma'(id_t)("fired") &= \sigma'(id_t)("s\_fired") && \text{(C.22)}\\
  \sigma'(id_t)("s\_fired") &= \sigma'(id_t)("s\_firable").\sigma'(id_t)("s\_priority\_combination") && \text{(C.23)}
  \end{aligned}$$

  Rewriting the goal with (C.20), (C.35), (C.22) and (C.23),
  $\boxed{s'.reset_t(t) = \sigma'(id_t)("s\_firable").\sigma'(id_t)("s\_priority\_combination").}$
  By property of the stabilize relation, and $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$:

  $$\sigma'(id_t)("spc") = \prod_{i=0}^{\Delta(id_t)("ian")-1} \sigma'(id_t)("priority\_authorizations")[i] \tag{C.24}$$

  By construction, $<\texttt{priority\_authorizations(0)} \Rightarrow \texttt{true}> \in ipm_t$, and by property of the stabilize relation and $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$:

  $$\sigma'(id_t)("priority\_authorizations")[0] = true \tag{C.25}$$

Rewriting the goal with (C.19), (C.24) and (C.25), and simplifying the equation, $\boxed{s'.reset_t(t) = \sigma'(id_t)(''s\_firable'').}$

Case analysis on $t \in Fired(s)$ or $t \notin Fired(s)$:

- **CASE** $t \in Fired(s)$:

  By property of $E_c, \tau \vdash s \xrightarrow{\uparrow} s'$:

  $$s'.reset_t(t) = \texttt{true} \tag{C.26}$$

  Rewriting the goal with (C.26), $\boxed{\sigma'(id_t)(''s\_firable'') = \texttt{true}.}$

  By property of the stabilize, the $\mathcal{H}$-VHDL rising edge and the $\texttt{Inject}_{\uparrow}$ relations, and $\texttt{comp}(id_t, ''transition'', gm_t, ipm_t, opm_t) \in d.cs$:

  $$\sigma(id_t)(''s\_firable'') = \sigma'(id_t)(''s\_firable'') \tag{C.27}$$

  Rewriting the goal with (C.27), $\boxed{\sigma(id_t)(''s\_firable'') = \texttt{true}.}$

  By property of $\gamma \vdash s \overset{\downarrow}{\sim} \sigma$:

  $$t \in Firable(s) \Leftrightarrow \sigma(id_t)(''sfa'') = \texttt{true} \tag{C.28}$$

  Rewriting the goal with (C.28), $\boxed{t \in Firable(s).}$

  By property of $t \in Fired(s)$, $t \in Firable(s).$

- **CASE** $t \notin Fired(s)$:

  By property of $input(t) = \emptyset$, there does not exist any input place connected to t by a `basic` or `test` arc. Thus, by property of $E_c, \tau \vdash s \xrightarrow{\uparrow} s'$:

  $$s'.reset_t(t) = \texttt{false} \tag{C.29}$$

  Rewriting the goal with (C.29), $\boxed{\sigma'(id_t)(''s\_firable'') = \texttt{false}.}$

  By property of the stabilize, the $\mathcal{H}$-VHDL rising edge and the $\texttt{Inject}_{\uparrow}$ relations, and $\texttt{comp}(id_t, ''transition'', gm_t, ipm_t, opm_t) \in d.cs$, equation (C.27) holds.

  Rewriting the goal with (C.27), $\boxed{\sigma(id_t)(''s\_firable'') = false.}$

  By property of $\gamma \vdash s \overset{\downarrow}{\sim} \sigma$:

  $$t \notin Firable(s) \Leftrightarrow \sigma(id_t)(''sfa'') = \texttt{false} \tag{C.30}$$

  By property of $t \notin Fired(s)$ and $input(t) = \emptyset$, $t \notin Firable(s)$.

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

By construction, $<$`input_arcs_number` $\Rightarrow |input(t)|> \in gm_t$, and by property of the $\mathcal{H}$-VHDL elaboration relation:

$$\Delta(id_t)(''ian'') = |input(t)| \tag{C.31}$$

Rewriting the goal with (C.31), $\boxed{s'.reset_t(t) = \sum_{i=0}^{|input(t)|-1} \sigma'(id_t)(''rt'')[i].}$

Case analysis on $t \in Fired(s)$ or $t \notin Fired(s)$:

- **CASE** $t \in Fired(s)$:

  By property of $E_c, \tau \vdash s \xrightarrow{\uparrow} s'$, equation (C.26) holds.

  Rewriting the goal with (C.26), $\boxed{\sum_{i=0}^{|input(t)|-1} \sigma'(id_t)(''rt'')[i] = \mathtt{true}.}$

  To prove the goal, let us show $\boxed{\exists i \in [0, |input(t)| - 1] \text{ s.t. } \sigma'(id_t)(''rt'')[i] = \mathtt{true}.}$ 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. $\mathtt{comp}(id_p, ''place'', gm_p, ipm_p, opm_p) \in d.cs$. By construction, there exist an $i \in [0, |input(t)| - 1]$, a $j \in [0, |output(p)| - 1]$ and $id_{ji} \in Sigs(\Delta)$ s.t. $<\mathtt{reinit\_transition\_time(j)} \Rightarrow \mathtt{id_{ji}}> \in opm_p$ and $<\mathtt{reinit\_time(i)} \Rightarrow \mathtt{id_{ji}}> \in ipm_t$. Let us take such an i, j and id_{ji}, and let us use i to prove the goal: $\boxed{\sigma'(id_t)(''rt'')[i] = \mathtt{true}.}$
  By property of the stabilize relation, $<\mathtt{reinit\_transition\_time(j)} \Rightarrow \mathtt{id_{ji}}> \in opm_p$ and $<\mathtt{reinit\_time(i)} \Rightarrow \mathtt{id_{ji}}> \in ipm_t$:

  $$\sigma'(id_t)(''rt'')[i] = \sigma'(id_{ji}) = \sigma'(id_p)(''rtt'')[j] \tag{C.32}$$

  Rewriting the goal with (C.32), $\boxed{\sigma'(id_p)(''rtt'')[j] = \mathtt{true}.}$
  By property of the $\mathtt{Inject}_{\uparrow}$, the $\mathcal{H}$-VHDL rising edge and the stabilize relations:

  $$\begin{aligned}\sigma'(id_p)(''rtt'')[j] =&((\sigma(id_p)(''oat'')[j] = \mathtt{BASIC} + \sigma(id_p)(''oat'')[j] = \mathtt{TEST})\\ &.(\sigma(id_p)(''sm'') - \sigma(id_p)(''sots'') < \sigma(id_p)(''oaw'')[j])\\ &.(\sigma(id_p)(''sots'') > 0))\\ &+ \sigma(id_p)(''otf'')[j]\end{aligned} \tag{C.33}$$

  Rewriting the goal with (C.33),

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

  By construction, there exists $id_{ft} \in Sigs(\Delta)$ s.t. $<\mathtt{output\_transitions\_fired(j)} \Rightarrow \mathtt{id_{ft}}> \in ipm_p$ and $<\mathtt{fired} \Rightarrow \mathtt{id_{ft}}> \in opm_t$. By property of state σ as being a stable state:

  $$\sigma(id_t)(''fired'') = \sigma(id_{ft}) = \sigma(id_p)(''otf'')[j] \tag{C.34}$$

Rewriting the goal with (C.34),

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

By property of $\gamma \vdash s \overset{\downarrow}{\sim} \sigma$:

$$t \in Fired(s) \Leftrightarrow \sigma(id_t)(''fired'') = \texttt{true} \tag{C.35}$$

Knowing that $t \in Fired(s)$, we can rewrite the goal with the right side of (C.35) and simplify the goal (i.e, $\forall b \in \mathbb{B},\ b + \texttt{true} = \texttt{true}$), then tautology .

- **CASE** $t \notin Fired(s)$: Then, there are two cases that will determine the value of $s'.reset_t(t)$. Either there exists a place p with an output token sum greater than zero, that is connected to t by an `basic` or `test` arc, and such that the transient marking of p disables t; or such a place does not exist (the predicate is decidable).
  - **CASE** there exists such a place p as described above:

    Then, let us take such a place p and $\omega \in \mathbb{N}^*$ s.t.:
    1. $\sum\limits_{t_i \in Fired(s)} pre(p, t_i) > 0$
    2. $pre(p,t) = (\omega, \texttt{basic}) \vee pre(p,t) = (\omega, \texttt{test})$
    3. $s.M(p) - \sum\limits_{t_i \in Fired(s)} pre(p, t_i) < \omega$

    We will only consider the case where $pre(p,t) = (\omega, \texttt{basic})$; the proof is the similar when $pre(p,t) = (\omega, \texttt{test})$.

    Assuming that p exists, and by property of $\gamma \vdash s \overset{\downarrow}{\sim} \sigma$:

    $$s'.reset_t(t) = \texttt{true} \tag{C.36}$$

    Rewriting the goal with (C.36), $\boxed{\sum\limits_{i=0}^{|input(t)|-1} \sigma'(id_t)(''rt'')[i] = \texttt{true}.}$

    To prove the goal, let us show $\boxed{\exists i \in [0, |input(t)| - 1] \text{ s.t. } \sigma'(id_t)(''rt'')[i] = \texttt{true}.}$

    By construction, there exists $id_p \in Comps(\Delta)$ 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, there exist an $i \in [0, |input(t)| - 1]$, a $j \in [0, |output(p)| - 1]$ and $id_{ji} \in Sigs(\Delta)$ s.t. $<\texttt{reinit\_transition\_time(j)} \Rightarrow \texttt{id}_{\texttt{ji}}> \in opm_p$ and $<\texttt{reinit\_time(i)} \Rightarrow \texttt{id}_{\texttt{ji}}> \in ipm_t$. Let us take such an i, j and id_{ji}, and let us use i to prove the goal: $\boxed{\sigma'(id_t)(''rt'')[i] = \texttt{true}.}$

    By property of the stabilize relation, $<\texttt{reinit\_transition\_time(j)} \Rightarrow \texttt{id}_{\texttt{ji}}> \in opm_p$ and $<\texttt{reinit\_time(i)} \Rightarrow \texttt{id}_{\texttt{ji}}> \in ipm_t$:

    $$\sigma'(id_t)(''rt'')[i] = \sigma'(id_{ji}) = \sigma'(id_p)(''rtt'')[j] \tag{C.37}$$

Rewriting the goal with (C.37), $\boxed{\sigma'(id_p)(''rtt'')[j] = \texttt{true}.}$
By property of the $\texttt{Inject}_{\uparrow}$, the $\mathcal{H}$-VHDL rising edge and the stabilize relations:

$$\begin{aligned}\sigma'(id_p)(''rtt'')[j] = &((\sigma(id_p)(''oat'')[j] = \texttt{BASIC} + \sigma(id_p)(''oat'')[j] = \texttt{TEST})\\ &.(\sigma(id_p)(''sm'') - \sigma(id_p)(''sots'') < \sigma(id_p)(''oaw'')[j])\\ &.(\sigma(id_p)(''sots'') > 0))\\ &+ \sigma(id_p)(''otf'')[j]\end{aligned} \tag{C.38}$$

Rewriting the goal with (C.38),

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

By construction, $<\texttt{output\_arcs\_types(j)} \Rightarrow \texttt{BASIC}> \in ipm_p$ and $<\texttt{output\_arcs\_weights(j)} \Rightarrow \omega> \in ipm_p$.
By property of the stabilize relation and $\texttt{comp}(id_p, ''place'', gm_p, ipm_p, opm_p) \in d.cs$:

$$\begin{aligned}\sigma'(id_p)(''oat'')[j] &= \texttt{BASIC} \qquad (C.39)\\ \sigma'(id_p)(''oaw'')[j] &= \omega \qquad (C.40)\end{aligned}$$

By property of $\gamma \vdash s \overset{\downarrow}{\sim} \sigma$:

$$\begin{aligned}\sigma(id_p)(''sm'') &= s.M(p) \qquad (C.41)\\ \sigma(id_p)(''sots'') &= \sum_{t_i \in Fired(s)} pre(p, t_i) \qquad (C.42)\end{aligned}$$

Rewriting the goal with (C.39), (C.40), (C.41) and (C.42), and simplifying the goal:

$$\boxed{(s.M(p) - \sum_{t_i \in Fired(s)} pre(p, t_i) < \omega \;.\; \sum_{t_i \in Fired(s)} pre(p, t_i) > 0)) + \sigma(id_t)(''fired'') = \texttt{true}}$$

Thanks to the hypotheses 1 and 3:

$$\begin{aligned}s.M(p) - \sum_{t_i \in Fired(s)} pre(p, t_i) < \omega &= \texttt{true} \qquad (C.43)\\ \sum_{t_i \in Fired(s)} pre(p, t_i) > 0 &= \texttt{true} \qquad (C.44)\\ & \qquad (C.45)\end{aligned}$$

Rewriting the goal with (C.43) and (C.44), and simplifying the goal, tautology.

- **CASE** such a place does not exist:
  Then, let us assume that, for all place $p \in P$
  1. $\sum_{t_i \in Fired(s)} pre(p, t_i) = 0$

2. or $\forall \omega \in \mathbb{N}^*,\ pre(p,t) = (\omega, \texttt{basic}) \vee pre(p,t) = (\omega, \texttt{test}) \Rightarrow s.M(p) - \sum_{t_i \in Fired(s)} pre(p,t_i) \geq \omega$.

In that case, by property of $\gamma \vdash s \overset{\downarrow}{\sim} \sigma$:

$$s'.reset_t(t) = \texttt{false} \tag{C.46}$$

Rewriting the goal with (C.46): $\boxed{\sum_{i=0}^{|input(t)|-1} \sigma'(id_t)(''rt'')[i] = \texttt{false}.}$

To prove the goal, let us show $\boxed{\forall i \in [0, |input(t)| - 1],\ \sigma'(id_t)(''rt'')[i] = \texttt{false}.}$
Given an $i \in [0, |input(t)| - 1]$, let us show $\boxed{\sigma'(id_t)(''rt'')[i] = \texttt{false}.}$

By construction, there exist a $p \in input(t)$, an $id_p \in Comps(\Delta)$, gm_p, ipm_p, opm_p, a $j \in [0, |output(p)| - 1]$, an $id_{ji} \in Sigs(\Delta)$ s.t. $\gamma(p) = id_p$ and $\texttt{comp}(id_p, ''place'', gm_p, ipm_p, opm_p) \in d.cs$ and $<\texttt{reinit\_transition\_time(j)} \Rightarrow \texttt{id}_{\texttt{ji}}> \in opm_p$ and $<\texttt{reinit\_time(i)} \Rightarrow \texttt{id}_{\texttt{ji}}> \in ipm_t$. Let us take such a p, id_p, gm_p, ipm_p, opm_p, j and id_{ji}.
By property of the stabilize relation, $<\texttt{reinit\_transition\_time(j)} \Rightarrow \texttt{id}_{\texttt{ji}}> \in opm_p$ and $<\texttt{reinit\_time(i)} \Rightarrow \texttt{id}_{\texttt{ji}}> \in ipm_t$:

$$\sigma'(id_t)(''rt'')[i] = \sigma'(id_{ji}) = \sigma'(id_p)(''rtt'')[j] \tag{C.47}$$

Rewriting the goal with (C.47): $\boxed{\sigma'(id_p)(''rtt'')[j] = \texttt{false}.}$

By property of the $\texttt{Inject}_{\uparrow}$, the $\mathcal{H}$-VHDL rising edge and the stabilize relations:

$$\begin{aligned}\sigma'(id_p)(''rtt'')[j] =& ((\sigma(id_p)(''oat'')[j] = \texttt{BASIC} + \sigma(id_p)(''oat'')[j] = \texttt{TEST})\\ &.(\sigma(id_p)(''sm'') - \sigma(id_p)(''sots'') < \sigma(id_p)(''oaw'')[j])\\ &.(\sigma(id_p)(''sots'') > 0))\\ &+ \sigma(id_p)(''otf'')[j]\end{aligned} \tag{C.48}$$

Rewriting the goal with (C.48),

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

By construction, there exists $id_{ft} \in Sigs(\Delta)$ s.t. $<\texttt{output\_transitions\_fired(j)} \Rightarrow \texttt{id}_{\texttt{ft}}> \in ipm_p$ and $<\texttt{fired} \Rightarrow \texttt{id}_{\texttt{ft}}> \in opm_t$. By property of state σ as being a stable state:

$$\sigma(id_t)(''fired'') = \sigma(id_{ft}) = \sigma(id_p)(''otf'')[j] \tag{C.49}$$

Rewriting the goal with (C.49),

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

By property of $\gamma \vdash s \overset{\downarrow}{\sim} \sigma$:

$$t \notin Fired(s) \Leftrightarrow \sigma(id_t)(''fired'') = \texttt{false} \tag{C.50}$$

Knowing that $t \notin Fired(s)$, we can rewrite the goal with the right side of (C.50) and simplify the goal (i.e, $\forall b \in \mathbb{B},\ b + \texttt{false} = b$):

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

Then, there are two cases:

1. **CASE** $\sum_{t_i \in Fired(s)} pre(p, t_i) = 0$:

   By property of $\gamma \vdash s \overset{\downarrow}{\sim} \sigma$:

   $$\sum_{t_i \in Fired(s)} pre(p, t_i) = \sigma(id_p)(''sots'') \tag{C.51}$$

   Rewriting the goal with (C.51) and $\sum_{t_i \in Fired(s)} pre(p, t_i) = 0$, simplifying the goal: tautology.

2. **CASE** $\forall \omega \in \mathbb{N}^*,\ pre(p,t) = (\omega, \texttt{basic}) \vee pre(p,t) = (\omega, \texttt{test}) \Rightarrow s.M(p) - \sum_{t_i \in Fired(s)} pre(p, t_i) \geq \omega$:

   Let us perform case analysis on $pre(p,t)$; there are two cases:

   (a) **CASE** $pre(p,t) = (\omega, \texttt{basic})$ or $pre(p,t) = (\omega, \texttt{basic})$:
   By construction, $<\texttt{output\_arcs\_weights(j)} \Rightarrow \omega> \in ipm_p$.
   By property of stable state σ and $\texttt{comp}(id_p, ''place'', gm_p, ipm_p, opm_p) \in d.cs$:

   $$\sigma(id_p)(''oaw'')[j] \quad = \quad \omega \tag{C.52}$$

   By property of $\gamma \vdash s \overset{\downarrow}{\sim} \sigma$:

   $$\sigma(id_p)(''sm'') \quad = \quad s.M(p) \tag{C.53}$$
   $$\sigma(id_p)(''sots'') \quad = \quad \sum_{t_i \in Fired(s)} pre(p, t_i) \tag{C.54}$$

By hypothesis, we know that $s.M(p) - \sum_{t_i \in Fired(s)} pre(p,t_i) \geq \omega$, and then we can deduce:

$$s.M(p) - \sum_{t_i \in Fired(s)} pre(p,t_i) < \omega = \texttt{false} \tag{C.55}$$

Rewriting the goal with (C.52), (C.53), (C.54), and (C.55), and simplifying the goal, tautology.

(b) **CASE** $pre(p,t) = (\omega, \texttt{inhib})$:
By construction, $<\texttt{output\_arcs\_types(j)} \Rightarrow \texttt{INHIB}> \in ipm_p$.
By property of stable state σ and $\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$:

$$\sigma(id_p)("oat")[j] = \texttt{INHIB} \tag{C.56}$$

Rewriting the goal with (C.56), and simplifying the goal, tautology.

□

C.3.5 Rising edge and action executions

Lemma 28 (Rising Edge Equal Action Executions). *For all sitpn, d, γ, E_c, E_p, τ, Δ, σ_e, s, s', σ, σ_i, $\sigma_\uparrow$, σ' that verify the hypotheses of Def. 9, then*
$\forall a \in \mathcal{A}, id_a \in Outs(\Delta)$ *s.t.* $\gamma(a) = id_a$, $s'.ex(a) = \sigma'(id_a)$.

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

By property of $E_c, \tau \vdash s \xrightarrow{\uparrow} s'$:

$$s.ex(a) = s'.ex(a) \tag{C.57}$$

By construction, id_a is an output port identifier of boolean type in the $\mathcal{H}$-VHDL design d assigned by the ‘‘action’’ process only during a falling edge phase.
By property of the $\mathcal{H}$-VHDL $\texttt{Inject}_\uparrow$, rising edge, stabilize relations, and the ‘‘action’’ process:

$$\sigma(id_a) = \sigma'(id_a) \tag{C.58}$$

Rewriting the goal with (C.57) and (C.58), $\boxed{s.ex(a) = \sigma(id_a).}$

By property of $E_c, \tau \vdash s \xrightarrow{\uparrow} s'$, $s.ex(a) = \sigma(id_a)$. □

C.3.6 Rising edge and function executions

Lemma 29 (Rising Edge Equal Function Executions). *For all sitpn, d, γ, E_c, E_p, τ, Δ, σ_e, s, s', σ, σ_i, $\sigma_\uparrow$, σ' that verify the hypotheses of Def. 9, then*
$\forall f \in \mathcal{F}, id_f \in Outs(\Delta)$ *s.t.* $\gamma(f) = id_f$, $s'.ex(f) = \sigma'(id_f)$.

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

By property of $E_c, \tau \vdash s \xrightarrow{\uparrow} s'$:

$$s'.ex(f) = \sum_{t \in Fired(s)} \mathbb{F}(t,f) \tag{C.59}$$

By construction, the ‘‘function’’ process is a part of design d's behavior, i.e $\mathtt{ps}(''function'', \emptyset, sl, ss) \in d.cs$.
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, $\mathtt{id_f} \Leftarrow \mathtt{false} \in ss_{\uparrow}$ 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, the stabilize relations and $\mathtt{ps}(''function'', \emptyset, sl, ss) \in d.cs$:

  $$\sigma'(id_f) = false \tag{C.60}$$

  By property of $\sum\limits_{t \in Fired(s)} \mathbb{F}(t, f)$ and $trs(f) = \emptyset$:

  $$\sum_{t \in Fired(s)} \mathbb{F}(t, f) = \mathtt{false} \tag{C.61}$$

  Rewriting the goal with (C.59), (C.60) and (C.61), tautology.

- **CASE** $trs(f) \neq \emptyset$:
  By construction, $\mathtt{id_f} \Leftarrow \mathtt{id_{ft_0}} + \cdots + \mathtt{id_{ft_n}} \in ss_{\uparrow}$, where $id_{ft_i} \in Sigs(\Delta)$, $ss_{\uparrow}$ is the part of the ‘‘function’’ process body executed during the rising edge phase, and $n = |trs(f)| - 1$.

  By property of the $\mathtt{Inject}_{\uparrow}$, the $\mathcal{H}$-VHDL rising edge, the stabilize relations, and $\mathtt{ps}(''function'', \emptyset, sl, ss) \in d.cs$:

  $$\sigma'(id_f) = \sigma(id_{ft_0}) + \cdots + \sigma(id_{ft_n}) \tag{C.62}$$

  Rewriting the goal with (C.59) and (C.62), $\boxed{\sum\limits_{t \in Fired(s)} \mathbb{F}(t, f) = \sigma(id_{ft_0}) + \cdots + \sigma(id_{ft_n}).}$

  Let us reason on the value of $\sigma(id_{ft_0}) + \cdots + \sigma(id_{ft_n})$; there are two cases:

  - **CASE** $\sigma(id_{ft_0}) + \cdots + \sigma(id_{ft_n}) = \mathtt{true}$:
    Then, we can rewrite the goal as follows: $\boxed{\sum\limits_{t \in Fired(s)} \mathbb{F}(t, f) = \mathtt{true}.}$

    To prove the above goal, let us show $\boxed{\exists t \in Fired(s)\ s.t.\ \mathbb{F}(t, f) = \mathtt{true}.}$
    Knowing that $\sigma(id_{ft_0}) + \cdots + \sigma(id_{ft_n}) = \mathtt{true}$, then $\exists id_{ft_i}\ s.t.\ \sigma(id_{ft_i}) = \mathtt{true}$. Let us take such an id_{ft_i}.
    By construction, for all id_{ft_i}, there exist a $t_i \in trs(f)$, an $id_{t_i} \in Comps(\Delta)$, gm_{t_i}, ipm_{t_i} and opm_{t_i} s.t. $\gamma(t_i) = id_{t_i}$ and $\mathtt{comp}(id_{t_i}, ''transition'', gm_{t_i}, ipm_{t_i}, opm_{t_i}) \in d.cs$ and $<\mathtt{fired} \Rightarrow \mathtt{id_{ft_i}}> \in opm_{t_i}$. Let us take such a t_i, id_{t_i}, gm_{t_i}, ipm_{t_i} and opm_{t_i}.
    By property of σ as being a stable design state, and $\mathtt{comp}(id_{t_i}, ''transition'', gm_{t_i}, ipm_{t_i}, opm_{t_i}) \in d.cs$:

    $$\sigma(id_{t_i})(''fired'') = \sigma(id_{ft_i}) \tag{C.63}$$

Thanks to (C.63) and $\sigma(id_{ft_i}) = \texttt{true}$, we can deduce that $\sigma(id_{t_i})(''fired'') = \texttt{true}$.

By property of $\gamma \vdash s \overset{\downarrow}{\sim} \sigma$:

$$t_i \in Fired(s) \Leftrightarrow \sigma(id_{t_i})(''fired'') = \texttt{true} \tag{C.64}$$

Thanks to (C.64), we can deduce $t_i \in Fired(s)$.

Let us use t_i to prove the goal: $\boxed{\mathbb{F}(t, f) = \texttt{true}.}$

By definition of $t_i \in trs(f)$, $\mathbb{F}(t, f) = \texttt{true}.$

- **CASE** $\sigma(id_{ft_0}) + \cdots + \sigma(id_{ft_n}) = \texttt{false}$:

  Then, we can rewrite the goal as follows: $\boxed{\sum\limits_{t \in Fired(s)} \mathbb{F}(t, f) = \texttt{false}.}$

  To prove the above goal, let us show $\boxed{\forall t \in Fired(s) \text{ s.t. } \mathbb{F}(t, f) = \texttt{false}.}$

  Given a $t \in Fired(s)$, let us show $\boxed{\mathbb{F}(t, f) = \texttt{false}.}$

  Let us perform case analysis on $\mathbb{F}(t, f)$; there are 2 cases:

  * **CASE** $\mathbb{F}(t, f) = \texttt{false}.$
  * **CASE** $\mathbb{F}(t, f) = \texttt{true}$:
    By construction, for all $t \in T$ s.t. $\mathbb{F}(t, f) = \texttt{true}$, there exist an $id_t \in Comps(\Delta)$, gm_t, ipm_t, opm_t and $id_{ft_i} \in Sigs(\Delta)$ s.t. $\gamma(t) = id_t$ and $\texttt{comp}(id_t, ''transition'', gm_t, ipm_t, opm_t) \in d.cs$ and $<\texttt{fired} \Rightarrow \texttt{id}_{\texttt{ft}_\texttt{i}}> \in opm_t$. Let us take such a id_t, gm_t, ipm_t, opm_t and id_{ft_i}.
    By property of stable design state σ and $\texttt{comp}(id_t, ''transition'', gm_t, ipm_t, opm_t) \in d.cs$, equation (C.63) holds.

    By property of $\gamma \vdash s \overset{\downarrow}{\sim} \sigma$, equation (C.64) holds.
    Thanks to (C.63) and (C.64), we can deduce that $\sigma(id_{ft_i}) = \texttt{true}$.

    Then, $\sigma(id_{ft_i}) = \texttt{true}$ contradicts $\sigma(id_{ft_0}) + \cdots + \sigma(id_{ft_n}) = \texttt{false}.$

□

C.3.7 Rising edge and sensitization

Lemma 30 (Rising Edge Equal Sensitized). *For all sitpn, d, γ, E_c, E_p, τ, Δ, σ_e, s, s', σ, σ_i, $\sigma_\uparrow$, σ' that verify the hypotheses of Def. 9, then*
$\forall t \in T, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t, t \in Sens(s'.M) \Leftrightarrow \sigma'(id_t)(''s\_enabled'') = \texttt{true}$.

Proof. Given a $t \in T$ and an $id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$, let us show
$\boxed{t \in Sens(s'.M) \Leftrightarrow \sigma'(id_t)(''s\_enabled'') = \texttt{true}.}$

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, the proof is in two parts:

1. Assuming that $t \in Sens(s'.M)$, let us show $\boxed{\sigma'(id_t)(''s\_enabled'') = \texttt{true}.}$

   By property of the stabilize relation and $\texttt{comp}(id_t, ''transition'', gm_t, ipm_t, opm_t) \in d.cs$:

$$\sigma'(id_t)(''se'') = \prod_{i=0}^{\Delta(id_t)(''ian'')-1} \sigma'(id_t)(''input\_arcs\_valid'')[i] \tag{C.65}$$

Rewriting the goal with (C.65), $\boxed{\prod_{i=0}^{\Delta(id_t)("ian")-1} \sigma'(id_t)("iav")[i] = \texttt{true}.}$

To prove the goal, let us show that $\boxed{\forall i \in [0, \Delta(id_t)("ian") - 1],\ \sigma'(id_t)("iav")[i] = \texttt{true}.}$

Given an $i \in [0, \Delta(id_t)("ian") - 1]$, let us show $\boxed{\sigma'(id_t)("iav")[i] = \texttt{true}.}$

Let us perform case analysis on $input(t)$.

- **CASE** $input(t) = \emptyset$:
  By construction, $<\texttt{input\_arcs\_number} \Rightarrow \texttt{1}> \in gm_t$ and
  $<\texttt{input\_arcs\_valid}(0) \Rightarrow \texttt{true}> \in ipm_t$.
  By property of the elaboration and stabilize relations and
  $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$:

$$\begin{aligned} \Delta(id_t)("ian") &= 1 && \text{(C.66)} \\ \sigma'(id_t)("iav")[0] &= \texttt{true} && \text{(C.67)} \end{aligned}$$

  Thanks to (C.66), we can deduce that $i = 0$. Rewriting the goal with (C.67), tautology.

- **CASE** $input(t) \neq \emptyset$:
  By construction, $<\texttt{input\_arcs\_number} \Rightarrow |input(t)|> \in gm_t$.
  By property of the elaboration relation and $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$:

$$\Delta(id_t)("ian") = |input(t)| \tag{C.68}$$

  Thanks to (C.68), we know that $i \in [0, |input(t)| - 1]$.
  By construction, there exist a $p \in input(t), id_p \in Comps(\Delta), gm_p, ipm_p, opm_p, j \in [0, |output(p)| - 1]$ and $id_{ji} \in Sigs(\Delta)$ s.t. $\gamma(p) = id_p$ and
  $\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$ and $<\texttt{output\_arcs\_valid(j)} \Rightarrow \texttt{id}_{\texttt{ji}}> \in opm_p$
  and $<\texttt{input\_arcs\_valid(i)} \Rightarrow \texttt{id}_{\texttt{ji}}> \in ipm_t$.
  By property of the stabilize relation, $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$ and $\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$:

$$\sigma'(id_t)("iav")[i] = \sigma'(id_{ji}) = \sigma'(id_p)("oav")[j] \tag{C.69}$$

  Rewriting the goal with (C.69), $\boxed{\sigma'(id_p)("oav")[j] = \texttt{true}.}$
  By property of the stabilize relation and $\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$:

$$\begin{aligned} \sigma'(id_p)("oav")[j] = &((\sigma'(id_p)("oat")[j] = \texttt{BASIC} + \sigma'(id_p)("oat")[j] = \texttt{TEST}) \\ &\quad .\ \sigma'(id_p)("sm") \geq \sigma'(id_p)("oaw")[j]) \\ &+ (\sigma'(id_p)("oat")[j] = \texttt{INHIB}\ .\ \sigma'(id_p)("sm") < \sigma'(id_p)("oaw")[j]) \end{aligned} \tag{C.70}$$

Rewriting the goal with (C.70),

$$\boxed{\begin{aligned}\texttt{true} =&((\sigma'(id_p)(''oat'')[j] = \texttt{BASIC} + \sigma'(id_p)(''oat'')[j] = \texttt{TEST})\\ &\quad . \, \sigma'(id_p)(''sm'') \geq \sigma'(id_p)(''oaw'')[j])\\ &+ (\sigma'(id_p)(''oat'')[j] = \texttt{INHIB} \, . \, \sigma'(id_p)(''sm'') < \sigma'(id_p)(''oaw'')[j])\end{aligned}}$$

Let us perform case analysis on $pre(p,t)$; there are 3 cases:

- **CASE** $pre(p,t) = (\omega, \texttt{BASIC})$:

  By construction, $<\texttt{output\_arcs\_types(j)} \Rightarrow \texttt{BASIC}> \in ipm_p$ and $<\texttt{output\_arcs\_weights(j)} \Rightarrow \omega> \in ipm_p$.
  By property of the stabilize relation and $\texttt{comp}(id_p, ''place'', gm_p, ipm_p, opm_p) \in d.cs$:

  $$\sigma'(id_p)(''oat'')[j] = \texttt{BASIC} \tag{C.71}$$
  $$\sigma'(id_p)(''oaw'')[j] = \omega \tag{C.72}$$

  Rewriting the goal with (C.71) and (C.72), and simplifying the goal:
  $\boxed{\sigma'(id_p)(''sm'') \geq \omega = \texttt{true}.}$
  Appealing to Lemma Rising Edge Equal Marking:

  $$s'.M(p) = \sigma'(id_p)(''sm'') \tag{C.73}$$

  Rewriting the goal with (C.73): $\boxed{s'.M(p) \geq \omega = \texttt{true}.}$

  By definition of $t \in Sens(s'.M)$, $s'.M(p) \geq \omega = \texttt{true}.$ [1]
- **CASE** $pre(p,t) = (\omega, \texttt{TEST})$: same as the preceding case.
- **CASE** $pre(p,t) = (\omega, \texttt{INHIB})$:
  By construction, $<\texttt{output\_arcs\_types(j)} \Rightarrow \texttt{INHIB}> \in ipm_p$ and $<\texttt{output\_arcs\_weights(j)} \Rightarrow \omega> \in ipm_p$.
  By property of the stabilize relation and $\texttt{comp}(id_p, ''place'', gm_p, ipm_p, opm_p) \in d.cs$:

  $$\sigma'(id_p)(''oat'')[j] = \texttt{INHIB} \tag{C.74}$$
  $$\sigma'(id_p)(''oaw'')[j] = \omega \tag{C.75}$$

  Rewriting the goal with (C.74) and (C.75), and simplifying the goal:
  $\boxed{\sigma'(id_p)(''sm'') < \omega = \texttt{true}.}$
  Appealing to Lemma Rising Edge Equal Marking, equation (C.73) holds.
  Rewriting the goal with (C.73): $\boxed{s'.M(p) < \omega = \texttt{true}.}$
  By definition of $t \in Sens(s'.M)$, $s'.M(p) < \omega = \texttt{true}.$

2. Assuming that $\sigma'(id_t)(''s\_enabled'') = \texttt{true}$, let us show $\boxed{t \in Sens(s'.M).}$

[1] Here $\geq$ denotes a boolean operator, i.e $\geq \in \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{B}$. As the $\geq \subseteq (\mathbb{N} \times \mathbb{B})$ relation is decidable for all pairs of natural numbers, we can interchange an expression $a \geq b = \texttt{true}$ with $a \geq b$ where $a, b \in \mathbb{N}$.

By definition of $t \in Sens(s'.M)$, let us show

$$\boxed{\forall p \in P, \omega \in \mathbb{N}^*, \ (pre(p,t) = (\omega, \texttt{basic}) \lor pre(p,t) = (\omega, \texttt{test}) \Rightarrow s'.M(p) \geq \omega) \land (pre(p,t) = (\omega, \texttt{inhib}) \Rightarrow s'.M(p) < \omega)}$$

Given a $p \in P$ and an $\omega \in \mathbb{N}^*$, let us show
$\boxed{pre(p,t) = (\omega, \texttt{basic}) \lor pre(p,t) = (\omega, \texttt{test}) \Rightarrow s'.M(p) \geq \omega}$ and
$\boxed{pre(p,t) = (\omega, \texttt{inhib}) \Rightarrow s'.M(p) < \omega.}$

(a) Assuming $pre(p,t) = (\omega, \texttt{basic}) \lor pre(p,t) = (\omega, \texttt{test})$, let us show $\boxed{s'.M(p) \geq \omega.}$

The proceeding is the same for $pre(p,t) = (\omega, \texttt{basic})$ and $pre(p,t) = (\omega, \texttt{test})$. Therefore, we will only cover the case where $pre(p,t) = (\omega, \texttt{basic})$.
By property of the stabilize relation and $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$, equation (C.65) holds.

Rewriting $\sigma'(id_t)("se") = \texttt{true}$ with (C.65), $\prod_{i=0}^{\Delta(id_t)("ian")-1} \sigma'(id_t)("input\_arcs\_valid")[i] = \texttt{true}$.
Then, we can deduce that $\forall i \in [0, \Delta(id_t)("ian") - 1]$, $\sigma'(id_t)("iav")[i] = \texttt{true}$.
By construction, there exist an $id_p \in Comps(\Delta)$, gm_p, ipm_p, opm_p, $i \in [0, |input(t)| - 1]$, $j \in [0, |output(p)| - 1]$ and $id_{ji} \in Sigs(\Delta)$ s.t. $\gamma(p) = id_p$ and $\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$ and $<\texttt{output\_arcs\_valid(j)} \Rightarrow \texttt{id}_{\texttt{ji}}> \in opm_p$ and $<\texttt{input\_arcs\_valid(i)} \Rightarrow \texttt{id}_{\texttt{ji}}> \in ipm_t$. Let us take such an $id_p \in Comps(\Delta)$, gm_p, ipm_p, opm_p, $i \in [0, |input(t)| - 1]$, $j \in [0, |output(p)| - 1]$ and $id_{ji} \in Sigs(\Delta)$.
By construction, $<\texttt{input\_arcs\_number} \Rightarrow |input(t)|> \in gm_t$.
By property of the elaboration relation and $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$, equation (C.68) holds.
Thanks to (C.68), we can deduce that $\forall i \in [0, |input(t)| - 1]$, $\sigma'(id_t)("iav")[i] = \texttt{true}$.
Having such an $i \in [0, |input(t)| - 1]$, we can deduce that $\sigma'(id_t)("iav")[i] = \texttt{true}$.
By property of the stabilize relation, $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$ and $\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$, equation (C.69) holds.
Thanks to (C.69), we can deduce that $\sigma'(id_p)("oav")[j] = \texttt{true}$.
By property of the stabilize relation and $\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$, equation (C.70) holds. Thanks to (C.70), we can deduce that:

$$\begin{aligned} \texttt{true} = &((\sigma'(id_p)("oat")[j] = \texttt{BASIC} + \sigma'(id_p)("oat")[j] = \texttt{TEST}) \\ &\quad . \ \sigma'(id_p)("sm") \geq \sigma'(id_p)("oaw")[j]) \\ &+ (\sigma'(id_p)("oat")[j] = \texttt{INHIB} \ . \ \sigma'(id_p)("sm") < \sigma'(id_p)("oaw")[j]) \end{aligned}$$

By construction, $<\texttt{output\_arcs\_types(j)} \Rightarrow \texttt{BASIC}> \in ipm_p$ and $<\texttt{output\_arcs\_weights(j)} \Rightarrow \omega> \in ipm_p$.
By property of the stabilize relation and $\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$, equations (C.71) and (C.72) hold.
Thanks to (C.71) and (C.72), we can deduce that $\sigma'(id_p)("sm") \geq \omega = \texttt{true}$.

Appealing to Lemma Rising Edge Equal Marking, $s'.M(p) \geq \omega$.

(b) Assuming $pre(p,t) = (\omega, \texttt{inhib})$, let us show $\boxed{s'.M(p) < \omega.}$

The proceeding is the same as the preceding case. Here, we will start the proof where the two cases are diverging, i.e:

By construction, $<\texttt{output\_arcs\_types(j)} \Rightarrow \texttt{INHIB}> \in ipm_p$ and $<\texttt{output\_arcs\_weights(j)} \Rightarrow \omega> \in ipm_p$.

By property of the stabilize relation and $\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$, equations (C.74) and (C.72) hold.

Thanks to (C.74) and (C.72), we can deduce that $\sigma'(id_p)("sm") < \omega = \texttt{true}$.

Appealing to Lemma Rising Edge Equal Marking, $s'.M(p) < \omega$.

□

Lemma 31 (Rising Edge Equal Not Sensitized)**.** *For all sitpn, d, γ, E_c, E_p, τ, Δ, σ_e, s, s', σ, σ_i, $\sigma_\uparrow$, σ' that verify the hypotheses of Def. 9, then*
$\forall t \in T, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t, t \notin Sens(s'.M) \Leftrightarrow \sigma'(id_t)("s\_enabled") = \texttt{false}$.

Proof. Proving the above lemma is trivial by appealing to Lemma Rising Edge Equal Sensitized and by reasoning on contrapositives. □

C.4 Falling Edge

Lemma 32 (Falling Edge)**.** *For all sitpn, d, γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_\downarrow$, σ' that verify the hypotheses of Def. 6, then $\gamma \vdash s' \overset{\downarrow}{\sim} \sigma'$.*

Proof. By definition of Post Falling Edge State Similarity, there are 12 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. $\forall c \in \mathcal{C}, id_c \in Ins(\Delta)$ 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. $\gamma(f) = id_f,\ s'.ex(f) = \sigma'(id_f)$.
7. $\forall t \in T, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t, t \in Firable(s') \Leftrightarrow \sigma'(id_t)("s\_firable") = \texttt{true}$.
8. $\forall t \in T, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t, t \notin Firable(s') \Leftrightarrow \sigma'(id_t)("s\_firable") = \texttt{false}$.
9. $\forall t \in T, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t, t \in Fired(s') \Leftrightarrow \sigma'(id_t)("fired") = \texttt{true}$.
10. $\forall t \in T, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t, t \notin Fired(s') \Leftrightarrow \sigma'(id_t)("fired") = \texttt{false}$.

11. $\forall p \in P, id_p \in Comps(\Delta)$ *s.t.* $\gamma(p) = id_p, \sum_{t \in Fired(s')} pre(p,t) = \sigma'(id_p)(''s\_output\_token\_sum'')$.

12. $\forall p \in P, id_p \in Comps(\Delta)$ *s.t.* $\gamma(p) = id_p, \sum_{t \in Fired(s')} post(t,p) = \sigma'(id_p)(''s\_input\_token\_sum'')$.

Each point is proved by a separate lemma:

- Apply Lemma Falling Edge Equal Marking to solve 1.
- Apply Lemma Falling Edge Equal Time Counters to solve 2.
- Apply Lemma Falling Edge Equal Reset Orders to solve 3.
- Apply Lemma Falling Edge Equal Condition Values to solve 4.
- Apply Lemma Falling Edge Equal Action Executions to solve 5.
- Apply Lemma Falling Edge Equal Function Executions to solve 6.
- Apply Lemma Falling Edge Equal Firable to solve 7.
- Apply Lemma Falling Edge Equal Not Firable to solve 8.
- Apply Lemma Falling Edge Equal Fired to solve 9.
- Apply Lemma Falling Edge Equal Not Fired to solve 10.
- Apply Lemma Falling Edge Equal Output Token Sum to solve 11.
- Apply Lemma Falling Edge Equal Input Token Sum to solve 12.

□

C.4.1 Falling Edge and marking

Lemma 33 (Falling Edge Equal Marking). *For all sitpn, d,* γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_\downarrow$, σ' *that verify the hypotheses of Def. 6, then* $\forall p \in P, id_p \in Comps(\Delta)$ *s.t.* $\gamma(p) = id_p,\ s'.M(p) = \sigma'(id_p)(''s\_marking'')$.

Proof. Given a $p \in P$ and an $id \in Comps(\Delta)$ s.t. $\gamma(p) = id_p$, let us show $\boxed{s'.M(p) = \sigma'(id_p)(''s\_marking'').}$

By definition of $E_c, \tau \vdash sitpn, s \xrightarrow{\downarrow} s'$:

$$s.M(p) = s'.M(p) \tag{C.76}$$

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

$$\sigma'(id_p)(''s\_marking'') = \sigma(id_p)(''s\_marking'') \tag{C.77}$$

Rewriting the goal with (C.76) and (C.77): $\boxed{s.M(p) = \sigma(id_p)(''s\_marking'').}$

By definition of $\gamma, E_c, \tau \vdash s \overset{\downarrow}{\sim} \sigma$: $s.M(p) = \sigma(id_p)(''s\_marking'')$.

□

Lemma 34 (Falling Edge Equal Output Token Sum). *For all sitpn, d, γ, Δ, σ_e, E_c, E_p, τ, s, s′, σ, σ_i, $\sigma_\downarrow$, σ' that verify the hypotheses of Def. 6, then $\forall p, id_p$ s.t. $\gamma(p) = id_p$, $\sum\limits_{t \in Fired(s')} pre(p,t) = \sigma'(id_p)(''s\_output\_token\_sum'')$.*

Proof. Given a $p \in P$ and an $id_p \in Comps(\Delta)$, let us show $\boxed{\sum\limits_{t \in Fired(s')} pre(p,t) = \sigma'(id_p)(''s\_output\_token\_sum'').}$

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 stabilize relation and $\texttt{comp}(id_p, ''place'', gm_p, ipm_p, opm_p) \in d.cs$:

$$\sigma'(id_p)(''sots'') = \sum_{i=0}^{\Delta(id_p)(''oan'')-1} \begin{cases} \sigma'(id_p)(''oaw'')[i] \texttt{ if } (\sigma'(id_p)(''otf'')[i] \\ \qquad\qquad . \, \sigma'(id_p)(''oat'')[i] = \texttt{BASIC}) \\ 0 \ otherwise \end{cases} \tag{C.78}$$

Rewriting the goal with (C.78):

$$\boxed{\sum_{t \in Fired(s')} pre(p,t) = \sum_{i=0}^{\Delta(id_p)(''oan'')-1} \begin{cases} \sigma'(id_p)(''oaw'')[i] \texttt{ if } (\sigma'(id_p)(''otf'')[i] \\ \qquad\qquad . \, \sigma'(id_p)(''oat'')[i] = \texttt{BASIC}) \\ 0 \ otherwise \end{cases}}$$

Let us unfold the definition of the left sum term:

$$\boxed{\begin{array}{c} \sum\limits_{t \in Fired(s')} \begin{cases} \omega \texttt{ if } pre(p,t) = (\omega, \texttt{basic}) \\ 0 \ otherwise \end{cases} \\ = \\ \sum\limits_{i=0}^{\Delta(id_p)(''oan'')-1} \begin{cases} \sigma'(id_p)(''oaw'')[i] \texttt{ if } (\sigma'(id_p)(''otf'')[i] \\ \qquad\qquad . \, \sigma'(id_p)(''oat'')[i] = \texttt{BASIC}) \\ 0 \ otherwise \end{cases} \end{array}}$$

To ease the reading, let us define functions $f \in Fired(s') \rightarrow \mathbb{N}$ and $g \in [0, |output(p)| - 1] \rightarrow \mathbb{N}$ s.t.

$$f(t) = \begin{cases} \omega \texttt{ if } pre(p,t) = (\omega, \texttt{basic}) \\ 0 \ otherwise \end{cases} \quad \text{and } g(i) = \begin{cases} \sigma'(id_p)(''oaw'')[i] \texttt{ if } (\sigma'(id_p)(''otf'')[i] \\ \qquad\qquad . \, \sigma'(id_p)(''oat'')[i] = \texttt{BASIC}) \\ 0 \ otherwise \end{cases}$$

Then, the goal is: $\boxed{\sum\limits_{t \in Fired(s')} f(t) = \sum\limits_{i=0}^{\Delta(id_p)(''oan'')-1} g(i)}$

Let us perform case analysis on $output(p)$; there are two cases:

1. $output(p) = \emptyset$:

   By construction, $<\texttt{output\_arcs\_number} \Rightarrow \texttt{1}> \in gm_p$, $<\texttt{output\_arcs\_types(0)} \Rightarrow \texttt{BASIC}> \in ipm_p$, $<\texttt{output\_transitions\_fired(0)} \Rightarrow \texttt{true}> \in ipm_p$, and $<\texttt{output\_arcs\_weights(0)} \Rightarrow \texttt{0}> \in ipm_p$.

   By property of the elaboration relation and $\texttt{comp}(id_p, ''place'', gm_p, ipm_p, opm_p) \in d.cs$:

$$\Delta(id_p)(''oan'') = 1 \tag{C.79}$$

By property of the stabilize relation and $\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$:

$$\begin{aligned} \sigma'(id_p)("oat")[0] &= \texttt{BASIC} && \text{(C.80)} \\ \sigma'(id_p)("otf")[0] &= \texttt{true} && \text{(C.81)} \\ \sigma'(id_p)("oaw")[0] &= 0 && \text{(C.82)} \end{aligned}$$

By property of $output(p) = \emptyset$:

$$\sum_{t \in Fired(s')} \begin{cases} \omega \ \texttt{if} \ pre(p,t) = (\omega, \texttt{basic}) \\ 0 \ otherwise \end{cases} = 0 \tag{C.83}$$

Rewriting the goal with (C.79), (C.80), (C.81), (C.82) and (C.83), tautology.

2. $output(p) \neq \emptyset$:

By construction, $<\texttt{output\_arcs\_number} \Rightarrow |output(p)|> \in gm_p$, and by property of the elaboration relation:

$$\Delta(id_p)("oan") = |output(p)| \tag{C.84}$$

Rewriting the goal with (C.84): $\boxed{\sum_{t \in Fired(s')} f(t) = \sum_{i=0}^{|output(p)|-1} g(i).}$

Let us reason by induction on the right sum term of the goal.

- **BASE CASE**:

  In that case, $0 > |output| - 1$ and $\sum_{i=0}^{|output(p)|-1} g(i) = 0$.

  As $0 > |output| - 1$, then $|output(p)| = 0$, thus contradicting $output(p) \neq \emptyset$.

- **INDUCTION CASE**:

  In that case, $0 \leq |output(p)| - 1$.

  $$\forall F \subseteq Fired(s'), g(0) + \sum_{t \in F} f(t) = g(0) + \sum_{i=1}^{|output(p)|-1} g(i)$$

  $$\boxed{\sum_{t \in Fired(s')} f(t) = g(0) + \sum_{i=1}^{|output(p)|-1} g(i)}$$

  By definition of g:

  $$g(0) = \begin{cases} \sigma'(id_p)("oaw")[0] \ \texttt{if} \ (\sigma'(id_p)("otf")[0] \\ \qquad\qquad\qquad\qquad . \ \sigma'(id_p)("oat")[0] = \texttt{BASIC}) \\ 0 \ otherwise \end{cases} \tag{C.85}$$

Let us perform case analysis on the value of $\sigma'(id_p)(''otf'')[0]\ .\ \sigma'(id_p)(''oat'')[0] = \texttt{BASIC}$; there are two cases:

(a) $(\sigma'(id_p)(''otf'')[0]\ .\ \sigma'(id_p)(''oat'')[0] = \texttt{BASIC}) = \texttt{false}$:
In that case, $g(0) = 0$, and then we can apply the induction hypothesis with $F = Fired(s')$ to solve the goal: $\displaystyle\sum_{t \in Fired(s')} f(t) = \sum_{i=1}^{|output(p)|-1} g(i)$.

(b) $(\sigma'(id_p)(''otf'')[0]\ .\ \sigma'(id_p)(''oat'')[0] = \texttt{BASIC}) = \texttt{true}$:
In that case, $g(0) = \sigma'(id_p)(''oaw'')[0]$, $\sigma'(id_p)(''otf'')[0] = \texttt{true}$ and $\sigma'(id_p)(''oat'')[0] = \texttt{BASIC}$.
By construction, there exist a $t \in output(t)$, $id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$. Let us take such a $t \in output(p)$.
By definition of id_t, there exist gm_t, ipm_t, opm_t s.t. $\texttt{comp}(id_t, ''transition'', gm_t, ipm_t, opm_t) \in d.cs$.
As $t \in output(p)$, there exist $\omega \in \mathbb{N}^*$ and $a \in \{\texttt{BASIC}, \texttt{TEST}, \texttt{INHIB}\}$ s.t. $pre(p,t) = (\omega, a)$.
Let us take an ω and a s.t. $pre(p,t) = (\omega, a)$.
By construction, $<\texttt{output\_arcs\_types}(0) \Rightarrow a> \in ipm_p$, $<\texttt{output\_arcs\_weights}(0) \Rightarrow \omega> \in ipm_p$, and there exists $id_{ft} \in Sigs(\Delta)$ s.t. $<\texttt{fired} \Rightarrow id_{ft}> \in opm_t$ and $<\texttt{output\_transitions\_fired}(0) \Rightarrow \texttt{id}_{\texttt{ft}}> \in ipm_p$
By property of the stabilize relation, $\sigma'(id_p)(''oat'')[0] = \texttt{BASIC}$ and $<\texttt{output\_arcs\_types}(0) \Rightarrow \texttt{a}> \in ipm_p$:

$$pre(p,t) = (\omega, \texttt{basic}) \tag{C.86}$$

By property of the stabilize relation, $<\texttt{fired} \Rightarrow \texttt{id}_{\texttt{ft}}> \in opm_t$, $<\texttt{output\_transitions\_fired}(0) \Rightarrow \texttt{id}_{\texttt{ft}}> \in ipm_p$ and $\sigma'(id_p)(''otf'')[0] = \texttt{true}$:

$$\sigma'(id_t)(''fired'') = \texttt{true} \tag{C.87}$$

Appealing to Lemma 4, we know $t \in Fired(s')$.
As $t \in Fired(s')$, we can rewrite the left sum term of the goal as follows:

$$\boxed{f(t) + \sum_{t' \in Fired(s') \setminus \{t\}} f(t') = g(0) + \sum_{i=1}^{|output(p)|-1} g(i)}$$

We know that $g(0) = \sigma'(id_p)(''oaw'')[0]$, and by property of the stabilize relation and $<\texttt{output\_arcs\_weights}(0) \Rightarrow \omega> \in ipm_p$:

$$\sigma'(id_p)(''oaw'')[0] = \omega \tag{C.88}$$

Rewriting the goal with (C.88):

$$\boxed{f(t) + \sum_{t' \in Fired(s') \setminus \{t\}} f(t') = \omega + \sum_{i=1}^{|output(p)|-1} g(i)}$$

By definition of f, and as $pre(p,t) = (\omega, \texttt{basic})$, then $f(t) = \omega$; thus, rewriting the goal:

$$\boxed{\omega + \sum_{t' \in Fired(s') \setminus \{t\}} f(t') = \omega + \sum_{i=1}^{|output(p)|-1} g(i)}$$

Then, knowing that $g(0) = \omega$, we can apply the induction hypothesis with $F = Fired(s') \setminus \{t\}$: $g(0) + \sum_{t' \in Fired(s') \setminus \{t\}} f(t') = g(0) + \sum_{i=1}^{|output(p)|-1} g(i)$.

□

Lemma 35 (Falling Edge Equal Input Token Sum)**.** *For all sitpn, d, γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_\downarrow$, σ' that verify the hypotheses of Def. 6, then $\forall p, id_p$ s.t. $\gamma(p) = id_p$, $\sum_{t \in Fired(s')} post(t,p) = \sigma'_p(''s\_input\_token\_sum'')$.*

Proof. Given a $p \in P$ and an $id_p \in Comps(\Delta)$, let us show $\boxed{\sum_{t \in Fired(s')} post(t,p) = \sigma'(id_p)(''s\_input\_token\_sum'').}$

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 stabilize relation and $\texttt{comp}(id_p, ''place'', gm_p, ipm_p, opm_p) \in d.cs$:

$$\sigma'(id_p)(''sits'') = \sum_{i=0}^{\Delta(id_p)(''ian'')-1} \begin{cases} \sigma'(id_p)(''iaw'')[i] \texttt{ if } \sigma'(id_p)(''itf'')[i] \\ 0 \textit{ otherwise} \end{cases} \tag{C.89}$$

Rewriting the goal with (C.89):

$$\boxed{\sum_{t \in Fired(s')} post(t,p) = \sum_{i=0}^{\Delta(id_p)(''ian'')-1} \begin{cases} \sigma'(id_p)(''iaw'')[i] \texttt{ if } \sigma'(id_p)(''otf'')[i] \\ 0 \textit{ otherwise} \end{cases}}$$

Let us unfold the definition of the left sum term:

$$\boxed{\begin{array}{c} \sum_{t \in Fired(s')} \begin{cases} \omega \texttt{ if } post(t,p) = \omega \\ 0 \textit{ otherwise} \end{cases} \\ = \\ \sum_{i=0}^{\Delta(id_p)(''ian'')-1} \begin{cases} \sigma'(id_p)(''iaw'')[i] \texttt{ if } \sigma'(id_p)(''itf'')[i] \\ 0 \textit{ otherwise} \end{cases} \end{array}}$$

Let us perform case analysis on $input(p)$; there are two cases:

1. $input(p) = \emptyset$:

   By construction, $<\texttt{input\_arcs\_number} \Rightarrow \texttt{1}> \in gm_p$, $<\texttt{input\_transitions\_fired(0)} \Rightarrow \texttt{true}> \in ipm_p$, and $<\texttt{input\_arcs\_weights(0)} \Rightarrow \texttt{0}> \in ipm_p$.

   By property of the elaboration relation and $\texttt{comp}(id_p, ''place'', gm_p, ipm_p, opm_p) \in d.cs$:

$$\Delta(id_p)(''ian'') = 1 \tag{C.90}$$

   By property of the stabilize relation and $\texttt{comp}(id_p, ''place'', gm_p, ipm_p, opm_p) \in d.cs$:

$$\sigma'(id_p)(''itf'')[0] = \texttt{true} \tag{C.91}$$

$$\sigma'(id_p)(''iaw'')[0] = 0 \tag{C.92}$$

By property of $input(p) = \emptyset$:

$$\sum_{t \in Fired(s')} \begin{cases} \omega \texttt{ if } post(t,p) = \omega \\ 0 \ otherwise \end{cases} = 0 \tag{C.93}$$

Rewriting the goal with (C.90), (C.91), (C.92), and (C.93), and simplifying the goal, tautology.

2. $input(p) \neq \emptyset$:

By construction, $<$`input_arcs_number` $\Rightarrow |input(p)|> \in gm_p$, and by property of the elaboration relation:

$$\Delta(id_p)(''ian'') = |input(p)| \tag{C.94}$$

To ease the reading, let us define functions $f \in Fired(s') \rightarrow \mathbb{N}$ and $g \in [0, |input(p)| - 1] \rightarrow \mathbb{N}$ s.t. $f(t) = \begin{cases} \omega \texttt{ if } post(t,p) = \omega \\ 0 \ otherwise \end{cases}$ and

$g(i) = \begin{cases} \sigma'(id_p)(''iaw'')[i] \texttt{ if } \sigma'(id_p)(''itf'')[i] \\ 0 \ otherwise \end{cases}$

Then, the goal is: $\boxed{\sum\limits_{t \in Fired(s')} f(t) = \sum\limits_{i=0}^{\Delta(id_p)(''ian'')-1} g(i)}$

Rewriting the goal with (C.94): $\boxed{\sum\limits_{t \in Fired(s')} f(t) = \sum\limits_{i=0}^{|input(p)|-1} g(i).}$

Let us reason by induction on the right sum term of the goal.

- **BASE CASE**:

  In that case, $0 > |input(p)| - 1$ and $\sum\limits_{i=0}^{|input(p)|-1} g(i) = 0$.

  As $0 > |input(p)| - 1$, then $|input(p)| = 0$, thus contradicting $input(p) \neq \emptyset$.

- **INDUCTION CASE**:
  In that case, $0 \leq |input(p)| - 1$.

  $$\boxed{\forall F \subseteq Fired(s'),\ g(0) + \sum_{t \in F} f(t) = g(0) + \sum_{i=1}^{|input(p)|-1} g(i)}$$

  $$\boxed{\sum_{t \in Fired(s')} f(t) = g(0) + \sum_{i=1}^{|input(p)|-1} g(i)}$$

  By definition of g:

$$g(0) = \begin{cases} \sigma'(id_p)(''iaw'')[0] \texttt{ if } \sigma'(id_p)(''itf'')[0] \\ 0 \ otherwise \end{cases} \tag{C.95}$$

Let us perform case analysis on the value of $\sigma'(id_p)(''itf'')[0]$; there are two cases:

(a) $\sigma'(id_p)(''itf'')[0] = \texttt{false}$:
In that case, $g(0) = 0$, and then we can apply the induction hypothesis with $F = Fired(s')$ to solve the goal: $\sum\limits_{t \in Fired(s')} f(t) = \sum\limits_{i=1}^{|input(p)|-1} g(i)$.

(b) $\sigma'(id_p)(''itf'')[0] = \texttt{true}$:
In that case, $g(0) = \sigma'(id_p)(''iaw'')[0]$ and $\sigma'(id_p)(''itf'')[0] = \texttt{true}$.
By construction, there exist a $t \in input(t)$, $id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$. Let us take such a $t \in input(p)$.
By definition of id_t, there exist gm_t, ipm_t, opm_t s.t. $\texttt{comp}(id_t, ''transition'', gm_t, ipm_t, opm_t) \in d.cs$.
As $t \in input(p)$, there exist $\omega \in \mathbb{N}^*$ s.t. $post(t,p) = \omega$. Let us take an ω s.t. $post(t,p) = \omega$.
By construction, $<\texttt{input\_arcs\_weights}(0) \Rightarrow \omega> \in ipm_p$, and there exists $id_{ft} \in Sigs(\Delta)$ s.t. $<\texttt{fired} \Rightarrow id_{ft}> \in opm_t$ and $<\texttt{input\_transitions\_fired}(0) \Rightarrow \texttt{id}_{\texttt{ft}}> \in ipm_p$
By property of the stabilize relation and $<\texttt{input\_arcs\_types}(0) \Rightarrow \texttt{a}> \in ipm_p$:

$$post(t,p) = \omega \tag{C.96}$$

By property of the stabilize relation, $<\texttt{fired} \Rightarrow \texttt{id}_{\texttt{ft}}> \in opm_t$, $<\texttt{input\_transitions\_fired}(0) \Rightarrow \texttt{id}_{\texttt{ft}}> \in ipm_p$ and $\sigma'(id_p)(''itf'')[0] = \texttt{true}$:

$$\sigma'(id_t)(''fired'') = \texttt{true} \tag{C.97}$$

Appealing to Lemma 4 and (C.97), we know $t \in Fired(s')$.
As $t \in Fired(s')$, we can rewrite the left sum term of the goal as follows:

$$\boxed{f(t) + \sum_{t' \in Fired(s') \setminus \{t\}} f(t') = g(0) + \sum_{i=1}^{|input(p)|-1} g(i)}$$

We know that $g(0) = \sigma'(id_p)(''iaw'')[0]$, and by property of the stabilize relation and $<\texttt{input\_arcs\_weights}(0) \Rightarrow \omega> \in ipm_p$:

$$\sigma'(id_p)(''iaw'')[0] = \omega \tag{C.98}$$

Rewriting the goal with (C.98):

$$\boxed{f(t) + \sum_{t' \in Fired(s') \setminus \{t\}} f(t') = \omega + \sum_{i=1}^{|input(p)|-1} g(i)}$$

By definition of f, and as $post(t,p) = \omega$, then $f(t) = \omega$; thus, rewriting the goal:

$$\boxed{\omega + \sum_{t' \in Fired(s') \setminus \{t\}} f(t') = \omega + \sum_{i=1}^{|input(p)|-1} g(i)}$$

Then, knowing that $g(0) = \omega$, we can apply the induction hypothesis with $F = Fired(s') \setminus \{t\}$: $g(0) + \sum\limits_{t' \in Fired(s') \setminus \{t\}} f(t') = g(0) + \sum\limits_{i=1}^{|input(p)|-1} g(i)$.

□

C.4.2 Falling edge and time counters

Lemma 36 (Falling Edge Equal Time Counters). *For all $sitpn$, d, γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_\downarrow$, σ' that verify the hypotheses of Def. 6, then $\forall t \in T_i, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$,*
$(upper(I_s(t)) = \infty \wedge s'.I(t) \leq lower(I_s(t)) \Rightarrow s'.I(t) = \sigma'(id_t)(''s\_time\_counter''))$
$\wedge(upper(I_s(t)) = \infty \wedge s'.I(t) > lower(I_s(t)) \Rightarrow \sigma'(id_t)(''s\_time\_counter'') = lower(I_s(t)))$
$\wedge(upper(I_s(t)) \neq \infty \wedge s'.I(t) > upper(I_s(t)) \Rightarrow \sigma'(id_t)(''s\_time\_counter'') = upper(I_s(t)))$
$\wedge(upper(I_s(t)) \neq \infty \wedge s'.I(t) \leq upper(I_s(t)) \Rightarrow s'.I(t) = \sigma'(id_t)(''s\_time\_counter''))$.

Proof. Given a $t \in T_i$ and an $id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$, let us show

$$\boxed{\begin{array}{l}(upper(I_s(t)) = \infty \wedge s'.I(t) \leq lower(I_s(t)) \Rightarrow s'.I(t) = \sigma'(id_t)(''s\_time\_counter''))\\ \wedge(upper(I_s(t)) = \infty \wedge s'.I(t) > lower(I_s(t)) \Rightarrow \sigma'(id_t)(''s\_time\_counter'') = lower(I_s(t)))\\ \wedge(upper(I_s(t)) \neq \infty \wedge s'.I(t) > upper(I_s(t)) \Rightarrow \sigma'(id_t)(''s\_time\_counter'') = upper(I_s(t)))\\ \wedge(upper(I_s(t)) \neq \infty \wedge s'.I(t) \leq upper(I_s(t)) \Rightarrow s'.I(t) = \sigma'(id_t)(''s\_time\_counter''))\end{array}}$$

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 elaboration, $\texttt{Inject}_\downarrow$, $\mathcal{H}$-VHDL rising edge and stabilize relations, and $\texttt{comp}(id_t, ''transition'', gm_t, ipm_t, opm_t) \in d.cs$:

$$\begin{aligned}\sigma(id_t)(''se'') = \texttt{true} \wedge \Delta(id_t)(''tt'') \neq \texttt{NOT\_TEMPORAL} \wedge \sigma(id_t)(''srtc'') = \texttt{false}\\ \wedge\sigma(id_t)(''stc'') < \Delta(id_t)(''mtc'') \Rightarrow \sigma'(id_t)(''stc'') = \sigma(id_t)(''stc'') + 1\end{aligned} \tag{C.99}$$

$$\begin{aligned}\sigma(id_t)(''se'') = \texttt{true} \wedge \Delta(id_t)(''tt'') \neq \texttt{NOT\_TEMPORAL} \wedge \sigma(id_t)(''srtc'') = \texttt{false}\\ \wedge\sigma(id_t)(''stc'') \geq \Delta(id_t)(''mtc'') \Rightarrow \sigma'(id_t)(''stc'') = \sigma(id_t)(''stc'')\end{aligned} \tag{C.100}$$

$$\begin{aligned}\sigma(id_t)(''se'') = \texttt{true} \wedge \Delta(id_t)(''tt'') \neq \texttt{NOT\_TEMPORAL}\\ \wedge\sigma(id_t)(''srtc'') = \texttt{true} \Rightarrow \sigma'(id_t)(''stc'') = 1\end{aligned} \tag{C.101}$$

$$\sigma(id_t)(''se'') = \texttt{false} \vee \Delta(id_t)(''tt'') = \texttt{NOT\_TEMPORAL} \Rightarrow \sigma'(id_t)(''stc'') = 0 \tag{C.102}$$

Then, there are 4 points to show:

1. $\boxed{upper(I_s(t)) = \infty \wedge s'.I(t) \leq lower(I_s(t)) \Rightarrow s'.I(t) = \sigma'(id_t)(''s\_time\_counter'')}$

   Assuming $upper(I_s(t)) = \infty$ and $s'.I(t) \leq lower(I_s(t))$, let us show $\boxed{s'.I(t) = \sigma'(id_t)(''s\_time\_counter'').}$

   Case analysis on $t \in Sens(s.M)$; there are two cases:

   (a) $t \notin Sens(s.M)$:

   By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have $\sigma(id_t)(''se'') = \texttt{false}$ (C.103).
   Appealing to (C.102) and (C.103), we have $\sigma'(id_t)(''stc'') = 0$ (C.104).
   By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s'.I(t) = 0$ (C.105).

Rewriting the goal with (C.104) and (C.105): tautology.

(b) $t \in Sens(s.M)$:

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have $\sigma(id_t)(''se'') = \texttt{true}$ (C.106).
By construction, and as $upper(I_s(t)) = \infty$, $<\texttt{transition\_type} \Rightarrow \texttt{TEMP\_A\_INF}> \in gm_t$. By property of the elaboration relation, we have $\Delta(id_t)(''tt'') = \texttt{TEMP\_A\_INF}$ (C.107).
Case analysis on $s.reset_t(t)$; there are two cases:

i. $s.reset_t(t) = \texttt{true}$:

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, $\sigma(id_t)(''srtc'') = \texttt{true}$ (C.108).
Appealing to (C.101), (C.106), (C.107) and (C.108), we have $\sigma'(id_t)(''stc'') = 1$ (C.109).
By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s'.I(t) = 1$ (C.110).
Rewriting the goal with (C.109) and (C.110): tautology.

ii. $s.reset_t(t) = \texttt{false}$:

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have $\sigma(id_t)(''srtc'') = \texttt{false}$ (C.111).
As $upper(I_s(t)) = \infty$, there exists an $a \in \mathbb{N}^*$ s.t. $I_s(t) = [a, \infty]$. Let us take such an $a \in \mathbb{N}^*$. By construction, $<\texttt{maximal\_time\_counter} \Rightarrow a> \in gm_t$, and by property of the elaboration relation, we have $\Delta(id_t)(''mtc'') = a$ (C.112).
By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, and knowing that $t \in Sens(s.M)$, $s.reset_t(t) = \texttt{false}$ and $upper(I_s(t)) = \infty$:

$$s'.I(t) = s.I(t) + 1 \tag{C.113}$$

Rewriting the goal with (C.113): $\boxed{s.I(t) + 1 = \sigma'(id_t)(''stc'').}$
We assumed that $s'.I(t) \leq lower(I_s(t))$, and as $s'.I(t) = s.I(t) + 1$, then $s.I(t) + 1 \leq lower(I_s(t))$, then $s.I(t) < lower(I_s(t))$, then $s.I(t) < a$ since $a = lower(I_s(t))$.
By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, and knowing that $s.I(t) < lower(I_s(t))$ and $upper(I_s(t)) = \infty$:

$$s.I(t) = \sigma(id_t)(''stc'') \tag{C.114}$$

Appealing to (C.112), (C.114) and $s.I(t) < a$:

$$\sigma(id_t)(''stc'') < \Delta(id_t)(''mtc'') \tag{C.115}$$

Appealing to (C.99), (C.115), (C.111) and (C.106):

$$\sigma'(id_t)(''stc'') = \sigma(id_t)(''stc'') + 1 \tag{C.116}$$

Rewriting the goal with (C.116) and (C.114): tautology.

2. $\boxed{upper(I_s(t)) = \infty \wedge s'.I(t) > lower(I_s(t)) \Rightarrow \sigma'(id_t)(''s\_time\_counter'') = lower(I_s(t)).}$

Assuming that $upper(I_s(t)) = \infty$ and $s'.I(t) > lower(I_s(t))$, let us show
$\boxed{\sigma'(id_t)(''s\_time\_counter'') = lower(I_s(t)).}$

As $upper(I_s(t)) = \infty$, there exists an $a \in \mathbb{N}^*$ s.t. $I_s(t) = [a, \infty]$. Let us take such an $a \in \mathbb{N}^*$. By construction, $<\texttt{maximal\_time\_counter} \Rightarrow a> \in gm_t$, and $<\texttt{transition\_type} \Rightarrow \texttt{TEMP\_A\_INF}> \in$

gm_t by property of the elaboration relation:

$$\Delta(id_t)(''mtc'') = a \tag{C.117}$$

$$\Delta(id_t)(''tt'') = \texttt{TEMP\_A\_INF} \tag{C.118}$$

Case analysis on $t \in Sens(s.M)$:

(a) $t \notin Sens(s.M)$:

By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, and knowing that $t \in Sens(s.M)$, then $s'.I(t) = 0$. Since $lower(I_s(t)) \in \mathbb{N}^*$, then $lower(I_s(t)) > 0$.

Contradicts $s'.I(t) > lower(I_s(t))$.

(b) $t \in Sens(s.M)$:

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$ and $t \in Sens(s.M)$:

$$\sigma(id_t)(''se'') = \texttt{true} \tag{C.119}$$

Case analysis on $s.reset_t(t)$; there are two cases:

i. $s.reset_t(t) = \texttt{true}$:

By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$: $s'.I(t) = 1$.
We assumed that $s'.I(t) > lower(I_s(t))$, then $1 > lower(I_s(t))$.

Contradicts $lower(I_s(t)) > 0$.

ii. $s.reset_t(t) = \texttt{false}$:

By property of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$ and $s.reset_t(t) = \texttt{false}$:

$$\sigma(id_t)(''srtc'') = \texttt{false} \tag{C.120}$$

By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, and knowing that $s'.I(t) > lower(I_s(t))$:

$$\begin{aligned} s'.I(t) = s.I(t) + 1 \Rightarrow s.I(t) + 1 &> lower(I_s(t)) \\ \Rightarrow s.I(t) &\geq lower(I_s(t)) \end{aligned} \tag{C.121}$$

Case analysis on $s.I(t) \geq lower(I_s(t))$:

A. $s.I(t) > lower(I_s(t))$: $\boxed{\sigma'(id_t)(''stc'') = lower(I_s(t)).}$

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$:

$$\sigma(id_t)(''stc'') = lower(I_s(t)) \tag{C.122}$$

Appealing to (C.100):

$$\sigma'(id_t)(''stc'') = \sigma(id_t)(''stc'') \tag{C.123}$$

Rewriting the goal with (C.122) and (C.123): tautology.

B. $s.I(t) = lower(I_s(t))$: $\boxed{\sigma'(id_t)(''stc'') = lower(I_s(t)).}$

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$:

$$s.I(t) = \sigma(id_t)(''stc'') \tag{C.124}$$

Appealing to (C.100):

$$\sigma'(id_t)(''stc'') = \sigma(id_t)(''stc'') \tag{C.125}$$

Rewriting the goal with (C.125), (C.124) and $s.I(t) = lower(I_s(t))$: tautology.

3. $\boxed{upper(I_s(t)) \neq \infty \wedge s'.I(t) > upper(I_s(t)) \Rightarrow \sigma'(id_t)(''s\_time\_counter'') = upper(I_s(t)).}$

Assuming that $upper(I_s(t)) \neq \infty$ and $s'.I(t) > upper(I_s(t))$, let us show $\boxed{\sigma'(id_t)(''s\_time\_counter'') = upper(I_s(t)).}$

As $upper(I_s(t)) \neq \infty$, there exists an $a \in \mathbb{N}^*$, and a $b \in \mathbb{N}^*$ s.t. $I_s(t) = [a, b]$. Let us take such an a and b. By construction, there exists $tt \in \{\texttt{TEMP\_A\_A}, \texttt{TEMP\_A\_B}\}$ s.t. $<\texttt{maximal\_time\_counter} \Rightarrow b> \in gm_t$, and $<\texttt{transition\_type} \Rightarrow tt> \in gm_t$; by property of the elaboration relation:

$$\begin{align} \Delta(id_t)(''mtc'') &= b = upper(I_s(t)) \tag{C.126} \\ \Delta(id_t)(''tt'') &\neq \texttt{NOT\_TEMP} \tag{C.127} \end{align}$$

Case analysis on $t \in Sens(s.M)$:

(a) $t \notin Sens(s.M)$:

By definition of $E_c, \tau \vdash s \overset{\downarrow}{\rightarrow} s'$, and knowing that $t \in Sens(s.M)$, then $s'.I(t) = 0$. Since $upper(I_s(t)) \in \mathbb{N}^*$, then $upper(I_s(t)) > 0$.

Contradicts $s'.I(t) > upper(I_s(t))$.

(b) $t \in Sens(s.M)$:

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$ and $t \in Sens(s.M)$:

$$\sigma(id_t)(''se'') = \texttt{true} \tag{C.128}$$

Case analysis on $s.reset_t(t)$; there are two cases:

i. $s.reset_t(t) = \texttt{true}$:

By definition of $E_c, \tau \vdash s \overset{\downarrow}{\rightarrow} s'$: $s'.I(t) = 1$.
We assumed that $s'.I(t) > upper(I_s(t))$, then $1 > upper(I_s(t))$.

Contradicts $upper(I_s(t)) > 0$.

ii. $s.reset_t(t) = \texttt{false}$:

By property of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$ and $s.reset_t(t) = \texttt{false}$:

$$\sigma(id_t)(''srtc'') = \texttt{false} \tag{C.129}$$

Case analysis on $s.I(t) > upper(I_s(t))$ or $s.I(t) \leq upper(I_s(t))$:

A. $s.I(t) > upper(I_s(t))$: $\boxed{\sigma'(id_t)(''stc'') = upper(I_s(t)).}$

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$:

$$s'.I(t) = s.I(t) \tag{C.130}$$

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$:

$$\sigma(id_t)(''stc'') = upper(I_s(t)) \tag{C.131}$$

Appealing to (C.100), we have $\sigma'(id_t)(''stc'') = \sigma(id_t)(''stc'')$.
Rewriting the goal with $\sigma'(id_t)(''stc'') = \sigma(id_t)(''stc'')$ and (C.131): tautology.

B. $s.I(t) \leq upper(I_s(t))$: $\boxed{\sigma'(id_t)(''stc'') = upper(I_s(t)).}$

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$:

$$s.I(t) = \sigma(id_t)(''stc'') \tag{C.132}$$

Case analysis on $s.I(t) \leq upper(I_s(t))$; there are two cases:

- $s.I(t) = upper(I_s(t))$:

  Appealing to (C.126), (C.132) and $s.I(t) = upper(I_s(t))$:

  $$\Delta(id_t)(''mtc'') \leq \sigma(id_t)(''stc'') \tag{C.133}$$

  Appealing to (C.133) and (C.100):

  $$\sigma'(id_t)(''stc'') = \sigma(id_t)(''stc'') \tag{C.134}$$

  Rewriting the goal with (C.134), (C.132) and $s.I(t) = upper(I_s(t))$: tautology.

- $s.I(t) < upper(I_s(t))$:

  By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$:

  $$s'.I(t) = s.I(t) + 1 \tag{C.135}$$

  From (C.135) and $s.I(t) < upper(I_s(t))$, we can deduce $s'.I(t) \leq upper(I_s(t))$; contradicts $s'.I(t) > upper(I_s(t))$.

4. $\boxed{upper(I_s(t)) \neq \infty \wedge s'.I(t) \leq upper(I_s(t)) \Rightarrow s'.I(t) = \sigma'(id_t)(''s\_time\_counter'').}$

Assuming that $upper(I_s(t)) \neq \infty$ and $s'.I(t) \leq upper(I_s(t))$, let us show $\boxed{s'.I(t) = \sigma'(id_t)(''s\_time\_counter'').}$

As $upper(I_s(t)) \neq \infty$, there exists an $a \in \mathbb{N}^*$, and a $b \in \mathbb{N}^*$ s.t. $I_s(t) = [a, b]$. Let us take such an a and b. By construction, there exists $tt \in \{\texttt{TEMP\_A\_A}, \texttt{TEMP\_A\_B}\}$ s.t. $<$`maximal_time_counter` $\Rightarrow b> \in gm_t$, and $<$`transition_type` $\Rightarrow tt> \in gm_t$; by property of the elaboration relation:

$$\Delta(id_t)(''mtc'') = b = upper(I_s(t)) \tag{C.136}$$

$$\Delta(id_t)(''tt'') \neq \texttt{NOT\_TEMP} \tag{C.137}$$

Case analysis on $t \in Sens(s.M)$:

(a) $t \notin Sens(s.M)$:

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have $\sigma(id_t)(''se'') = \texttt{false}$ (C.138).
Appealing (C.102) and (C.138), we have $\sigma'(id_t)(''stc'') = 0$ (C.139).

By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s'.I(t) = 0$ (C.140).
Rewriting the goal with (C.139) and (C.140): tautology.

(b) $t \in Sens(s.M)$:

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have $\sigma(id_t)(''se'') = \texttt{true}$ (C.141).
Case analysis on $s.reset_t(t)$:

i. $s.reset_t(t) = \texttt{true}$:

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have $\sigma(id_t)(''srtc'') = \texttt{true}$ (C.142).
Appealing to (C.101), (C.137), (C.141) and (C.142), we have $\sigma'(id_t)(''stc'') = 1$ (C.143).

By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s'.I(t) = 1$ (C.144).
Rewriting the goal with (C.143) and (C.144), tautology.

ii. $s.reset_t(t) = \texttt{false}$:

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have $\sigma(id_t)(''srtc'') = \texttt{false}$ (C.145).
Case analysis on $s.I(t) > upper(I_s(t))$ or $s.I(t) \leq upper(I_s(t))$:

A. $s.I(t) > upper(I_s(t))$:

By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s.I(t) = s'.I(t)$, and thus, $s'.I(t) > upper(I_s(t))$.
Contradicts $s'.I(t) \leq upper(I_s(t))$.

B. $s.I(t) \leq upper(I_s(t))$:

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have $s.I(t) = \sigma(id_t)(''stc'')$ (C.146).

- $s.I(t) < upper(I_s(t))$:
  From $s.I(t) < upper(I_s(t))$, (C.146) and (C.136), we can deduce $\sigma(id_t)(''stc'') < \Delta(id_t)(''mtc'')$ (C.147).
  From (C.99), (C.141), (C.137), (C.145) and (C.147), we can deduce:

$$\sigma'(id_t)(''stc'') = \sigma(id_t)(''stc'') + 1 \tag{C.148}$$

By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$:

$$s'.I(t) = s.I(t) + 1 \tag{C.149}$$

Rewriting the goal with (C.148) and (C.149), tautology.

- $s.I(t) = upper(I_s(t))$:

  By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we know that $s'.I(t) = s.I(t) + 1$. We assumed that $s'.I(t) \leq upper(I_s(t))$; thus, $s.I(t) + 1 \leq upper(I_s(t))$.
  Contradicts $s.I(t) = upper(I_s(t))$.

□

C.4.3 Falling edge and reset orders

Lemma 37 (Falling Edge Equal Reset Orders). *For all sitpn, d, γ, Δ, σ_e, E_c, E_p, τ, s, s′, σ, σ_i, $\sigma_\downarrow$, σ' that verify the hypotheses of Def. 6, then $\forall t \in T_i, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$, $s'.reset_t(t) = \sigma'(id_t)(''s\_reinit\_time\_counter'')$.*

Proof. Given a $t \in T_i$ and an $id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$, let us show $\boxed{s'.reset_t(t) = \sigma'(id_t)(''srtc'').}$
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 stabilize relation and $\texttt{comp}(id_t, ''transition'', gm_t, ipm_t, opm_t) \in d.cs$:

$$\sigma'(id_t)(''srtc'') = \sum_{i=0}^{\Delta(id_t)(''ian'')-1} \sigma'(id_t)(''rt'')[i] \tag{C.150}$$

□

C.4.4 Falling edge and condition values

Lemma 38 (Falling Edge Equal Condition Values). *For all sitpn, d, γ, Δ, σ_e, E_c, E_p, τ, s, s′, σ, σ_i, $\sigma_\downarrow$, σ' that verify the hypotheses of Def. 6, then $\forall c \in \mathcal{C}, id_c \in Ins(\Delta)$ s.t. $\gamma(c) = id_c$, $s'.cond(c) = \sigma'(id_c)$.*

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

By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s'.cond(c) = E_c(\tau, c)$ (C.151).
By property of the $\texttt{Inject}_\downarrow$, the $\mathcal{H}$-VHDL falling edge, the stabilize relations and $id_c \in Ins(\Delta)$, we have $\sigma'(id_c) = E_p(\tau, \downarrow)(id_c)$ (C.152).
Rewriting the goal with (C.151) and (C.152): $\boxed{E_c(\tau, c) = E_p(\tau, \downarrow)(id_c)}$
By definition of $\gamma \vdash E_p \overset{env}{=} E_c$: $E_c(\tau, c) = E_p(\tau, \downarrow)(id_c)$.

□

C.4.5 Falling and action executions

Lemma 39 (Falling Edge Equal Action Executions). *For all sitpn, d, γ, Δ, σ_e, E_c, E_p, τ, s, s′, σ, σ_i, $\sigma_\downarrow$, σ' that verify the hypotheses of Def. 6, then $\forall a \in \mathcal{A}, id_a \in Outs(\Delta)$ s.t. $\gamma(a) = id_a$, $s'.ex(a) = \sigma'(id_a)$.*

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

By property of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$:

$$s'.ex(a) = \sum_{p \in marked(s.M)} \mathbb{A}(p, a) \tag{C.153}$$

By construction, the ‘‘$\texttt{action}$’’ process is a part of design d's behavior, i.e there exist an $sl \subseteq Sigs(\Delta)$ and an $ss_a \in ss$ s.t. $\texttt{ps}(''action'', \emptyset, sl, ss) \in d.cs$.
By construction id_a is only assigned in the body of the ‘‘$\texttt{action}$’’ process. Let $pls(a)$ be the set of actions associated to action a, i.e $pls(a) = \{p \in P \mid \mathbb{A}(p,a) = true\}$. Then, depending on $pls(a)$, there are two cases of assignment of output port id_a:

- **CASE** $pls(a) = \emptyset$:
  By construction, $\texttt{id}_\texttt{a} \Leftarrow \texttt{false} \in ss_{a\downarrow}$ where $ss_{a\downarrow}$ is the part of the ‘‘$\texttt{action}$’’ process body executed during the falling edge phase.

  By property of the $\mathcal{H}$-VHDL falling edge, the stabilize relations and $\texttt{ps}(''action'', \emptyset, sl, ss_a) \in d.cs$:

  $$\sigma'(id_a) = false \tag{C.154}$$

  By property of $\sum\limits_{p \in marked(s.M)} \mathbb{A}(p,a)$ and $pls(a) = \emptyset$:

  $$\sum_{p \in marked(s.M)} \mathbb{A}(p,a) = \texttt{false} \tag{C.155}$$

  Rewriting the goal with (C.153), (C.154) and (C.155), tautology.

- **CASE** $pls(a) \neq \emptyset$:
  By construction, $\texttt{id}_\texttt{a} \Leftarrow \texttt{id}_{\texttt{mp}_0} + \cdots + \texttt{id}_{\texttt{mp}_\texttt{n}} \in ss_{a\downarrow}$, where $id_{mp_i} \in Sigs(\Delta)$, $ss_{a\downarrow}$ is the part of the ‘‘$\texttt{action}$’’ process body executed during the falling edge phase, and $n = |pls(a)| - 1$.

  By property of the $\texttt{Inject}_\downarrow$, the $\mathcal{H}$-VHDL falling edge, the stabilize relations, and $\texttt{ps}(''action'', \emptyset, sl, ss) \in d.cs$:

  $$\sigma'(id_a) = \sigma(id_{mp_0}) + \cdots + \sigma(id_{mp_n}) \tag{C.156}$$

  Rewriting the goal with (C.153) and (C.156), $\boxed{\sum\limits_{p \in marked(s.M)} \mathbb{A}(p,a) = \sigma(id_{mp_0}) + \cdots + \sigma(id_{mp_n}).}$

  Let us reason on the value of $\sigma(id_{mp_0}) + \cdots + \sigma(id_{mp_n})$; there are two cases:

  - **CASE** $\sigma(id_{mp_0}) + \cdots + \sigma(id_{mp_n}) = \texttt{true}$:
    Then, we can rewrite the goal as follows: $\boxed{\sum\limits_{p \in marked(s.M)} \mathbb{A}(p,a) = \texttt{true}.}$

    To prove the above goal, let us show $\boxed{\exists p \in marked(s.M) \ s.t. \ \mathbb{A}(p,a) = \texttt{true}.}$
    From $\sigma(id_{mp_0}) + \cdots + \sigma(id_{mp_n}) = \texttt{true}$, we can deduce that $\exists id_{mp_i} \ s.t. \ \sigma(id_{mp_i}) = \texttt{true}$. Let us take an id_{mp_i} s.t. $\sigma(id_{mp_i}) = \texttt{true}$.
    By construction, for all id_{mp_i}, there exist a $p_i \in pls(a)$, an $id_{p_i} \in Comps(\Delta)$, gm_{p_i}, ipm_{p_i} and opm_{p_i} s.t. $\gamma(p_i) = id_{p_i}$ and $\texttt{comp}(id_{p_i}, ''place'', gm_{p_i}, ipm_{p_i}, opm_{p_i}) \in d.cs$ and $<\texttt{marked} \Rightarrow \texttt{id}_{\texttt{mp}_\texttt{i}}> \in opm_{p_i}$. Let us take such a p_i, id_{p_i}, gm_{p_i}, ipm_{p_i} and opm_{p_i}.
    By property of stable σ, and $\texttt{comp}(id_{p_i}, ''place'', gm_{p_i}, ipm_{p_i}, opm_{p_i}) \in d.cs$:

    $$\sigma(id_{mp_i}) = \sigma(id_{p_i})(''marked'') \tag{C.157}$$
    $$\sigma(id_{p_i})(''marked'') = \sigma(id_{p_i})(''sm'') > 0 \tag{C.158}$$

From (C.157), (C.158) and $\sigma(id_{mp_i}) = \texttt{true}$, we can deduce that $\sigma(id_{p_i})(''marked'') = \texttt{true}$ and $(\sigma(id_{p_i})(''sm'') > 0) = \texttt{true}$.

By property of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$:

$$s.M(p_i) = \sigma(id_{p_i})(''sm'') \tag{C.159}$$

From (C.159) and $(\sigma(id_{p_i})(''sm'') > 0) = \texttt{true}$, we can deduce $p_i \in marked(s.M)$, i.e $s.M(p_i) > 0$.

Let us use p_i to prove the goal: $\boxed{\mathbb{A}(p,a) = \texttt{true}.}$

By definition of $p_i \in pls(a)$, $\mathbb{A}(p,a) = \texttt{true}.$

- **CASE** $\sigma(id_{mp_0}) + \cdots + \sigma(id_{mp_n}) = \texttt{false}$:

  Then, we can rewrite the goal as follows: $\boxed{\sum\limits_{p \in marked(s.M)} \mathbb{A}(p,a) = \texttt{false}.}$

  To prove the above goal, let us show $\boxed{\forall p \in marked(s.M) \text{ } s.t. \text{ } \mathbb{A}(p,a) = \texttt{false}.}$

  Given a $p \in marked(s.M)$, let us show $\boxed{\mathbb{A}(p,a) = \texttt{false}.}$

  Let us perform case analysis on $\mathbb{A}(p,a)$; there are 2 cases:

  * **CASE** $\mathbb{A}(p,a) = \texttt{false}.$
  * **CASE** $\mathbb{A}(p,a) = \texttt{true}$:
    By construction, for all $p \in P$ s.t. $\mathbb{A}(p,a) = \texttt{true}$, there exist an $id_p \in Comps(\Delta)$, gm_{tp}, ipm_p, opm_p and $id_{mp_i} \in Sigs(\Delta)$ s.t. $\gamma(p) = id_p$ and $\texttt{comp}(id_p, ''place'', gm_p, ipm_p, opm_p) \in d.cs$ and $<\texttt{marked} \Rightarrow \texttt{id}_{\texttt{mp}_\texttt{i}}> \in opm_p$. Let us take such a id_p, gm_p, ipm_p, opm_p and id_{mp_i}.
    By property of stable σ and $\texttt{comp}(id_p, ''place'', gm_p, ipm_p, opm_p) \in d.cs$:

    $$\sigma(id_{mp_i}) = \sigma(id_p)(''marked'') \tag{C.160}$$
    $$\sigma(id_p)(''marked'') = \sigma(id_p)(''sm'') > 0 \tag{C.161}$$

    From $\sigma(id_{mp_0}) + \cdots + \sigma(id_{mp_n}) = \texttt{false}$, we can deduce $\sigma(id_p)(''marked'') = \texttt{false}$, and thus that $(\sigma(id_p)(''sm'') > 0) = \texttt{false}$.

    By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have $s.M(p) = \sigma(id_p)(''sm'')$, and thus, we can deduce that $s.M(p) = 0$ (equivalent to $(s.M(p) > 0) = \texttt{false}$).
    Contradicts $p \in marked(s.M)$ (i.e, $s.M(p) > 0$).

□

C.4.6 Falling edge and function executions

Lemma 40 (Falling Edge Equal Function Executions). *For all $sitpn$, d, γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_\downarrow$, σ' that verify the hypotheses of Def. 6, then $\forall f \in \mathcal{F}, id_f \in Outs(\Delta)$ s.t. $\gamma(f) = id_f$, $s'.ex(f) = \sigma'(id_f)$.*

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

By property of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$:

$$s.ex(f) = s'.ex(f) \tag{C.162}$$

By construction, id_f is an output port identifier of boolean type in the $\mathcal{H}$-VHDL design d assigned by the ‘‘`function`’’ process only during a rising edge phase.
By property of the $\mathcal{H}$-VHDL $\texttt{Inject}_{\uparrow}$, rising edge, stabilize relations, and the ‘‘`function`’’ process:

$$\sigma(id_f) = \sigma'(id_f) \tag{C.163}$$

Rewriting the goal with (C.162) and (C.163), $\boxed{s.ex(f) = \sigma(id_f).}$

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, $s.ex(f) = \sigma(id_f).$ □

C.4.7 Falling edge and firable transitions

Lemma 41 (Falling Edge Equal Firable). *For all sitpn, d, γ, Δ, σ_e, E_c, E_p, τ, s, s′, σ, σ_i, $\sigma_{\downarrow}$, σ' that verify the hypotheses of Def. 6, then* $\forall t \in T, id_t \in Comps(\Delta)$ *s.t.* $\gamma(t) = id_t$, $t \in Firable(s') \Leftrightarrow \sigma'(id_t)(''s\_firable'') = \texttt{true}$.

Proof. Given a $t \in T$ and $id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$, let us show that $\boxed{t \in Firable(s') \Leftrightarrow \sigma'(id_t)(''s\_firable'') = \texttt{true}.}$

The proof is in two parts:

1. Assuming that $t \in Firable(s')$, let us show $\boxed{\sigma'(id_t)(''s\_firable'') = \texttt{true}.}$

   Apply Lemma Falling Edge Equal Firable 1 to solve the goal.

2. Assuming that $\sigma'(id_t)(''s\_firable'') = \texttt{true}$, let us show $\boxed{t \in Firable(s').}$

   Apply Lemma Falling Edge Equal Firable 2 to solve the goal.

□

Lemma 42 (Falling Edge Equal Firable 1). *For all sitpn, d, γ, Δ, σ_e, E_c, E_p, τ, s, s′, σ, σ_i, $\sigma_{\downarrow}$, σ' that verify the hypotheses of Def. 6, then* $\forall t \in T, id_t \in Comps(\Delta)$ *s.t.* $\gamma(t) = id_t$, $t \in Firable(s') \Rightarrow \sigma'(id_t)(''s\_firable'') = \texttt{true}$.

Proof. Given a $t \in T$ and $id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$, and assuming that $t \in Firable(s')$, let us show $\boxed{\sigma'(id_t)(''s\_firable'') = \texttt{true}.}$

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 $Inject_{\downarrow}$, the $\mathcal{H}$-VHDL falling edge, the stabilize relations and $\texttt{comp}(id_t, ''transition'', gm_t, ipm_t, opm_t) \in d.cs$:

$$\sigma'(id_t)(''sfa'') = \sigma(id_t)(''se'') \;.\; \sigma(id_t)(''scc'') \;.\; \texttt{checktc}(\Delta(id_t), \sigma(id_t)) \tag{C.164}$$

Let us define term $\texttt{checktc}(\Delta(id_t),\sigma(id_t))$ as follows:

$$\begin{aligned}\texttt{checktc}(\Delta(id_t),\sigma(id_t)) = \Big(&\texttt{not}\ \sigma(id_t)(''srtc'')\ .\\ &\big[(\Delta(id_t)(''tt'') = \texttt{TEMP\_A\_B}\ .\ (\sigma(id_t)(''stc'') \geq \sigma(id_t)(''A'') - 1)\\ &\qquad\qquad\qquad\qquad .\ (\sigma(id_t)(''stc'') \leq \sigma(id_t)(''B'') - 1))\\ &+ (\Delta(id_t)(''tt'') = \texttt{TEMP\_A\_A}\ .\ (\sigma(id_t)(''stc'') = \sigma(id_t)(''A'') - 1))\\ &+ (\Delta(id_t)(''tt'') = \texttt{TEMP\_A\_INF}\ .\ (\sigma(id_t)(''stc'') \geq \sigma(id_t)(''A'') - 1))\big]\Big)\\ &+ \big(\sigma(id_t)(''srtc'')\ .\ \Delta(id_t)(''tt'') \neq \texttt{NOT\_TEMP}\ .\ \sigma(id_t)(''A'') = 1\big)\\ &+ \Delta(id_t)(''tt'') = \texttt{NOT\_TEMP}\end{aligned} \tag{C.165}$$

Rewriting the goal with (C.164): $\boxed{\sigma(id_t)(''se'')\ .\ \sigma(id_t)(''scc'')\ .\ \texttt{checktc}(\Delta(id_t),\sigma(id_t)) = \texttt{true}.}$
Then, there are three points to prove:

1. $\boxed{\sigma(id_t)(''se'') = \texttt{true}}$:

   From $t \in Firable(s')$, we can deduce $t \in Sens(s'.M)$. By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s.M = s'.M$, and thus, we can deduce $t \in Sens(s.M)$.

   By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we know that $t \in Sens(s.M)$ implies $\sigma(id_t)(''se'') = \texttt{true}.$

2. $\boxed{\sigma(id_t)(''scc'') = \texttt{true}}$:

   By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$:

$$\sigma(id_t)(''scc'') = \prod_{c \in conds(t)} \begin{cases} E_c(\tau, c) & if\ \mathbb{C}(t,c) = 1 \\ \texttt{not}(E_c(\tau, c)) & if\ \mathbb{C}(t,c) = -1 \end{cases} \tag{C.166}$$

   where $conds(t) = \{c \in \mathcal{C} \mid \mathbb{C}(t,c) = 1 \vee \mathbb{C}(t,c) = -1\}$.

   Rewriting the goal with (C.166): $\boxed{\prod_{c \in conds(t)} \begin{cases} E_c(\tau, c) & if\ \mathbb{C}(t,c) = 1 \\ \texttt{not}(E_c(\tau, c)) & if\ \mathbb{C}(t,c) = -1 \end{cases} = \texttt{true}.}$

   To ease the reading, let us define $f(c) = \begin{cases} E_c(\tau, c) & if\ \mathbb{C}(t,c) = 1 \\ \texttt{not}(E_c(\tau, c)) & if\ \mathbb{C}(t,c) = -1 \end{cases}.$

   Let us reason by induction on the left term of the goal:

   - **BASE CASE**: $true = true.$
   - **INDUCTION CASE**:

$$\prod_{c' \in conds(t) \setminus \{c\}} f(c') = \texttt{true}$$

$\boxed{f(c) \,.\prod_{c' \in conds(t) \setminus \{c\}} f(c') = \texttt{true}.}$

Rewriting the goal with the induction hypothesis, and simplifying the goal, and unfolding the definition of $f(c)$: $\boxed{\begin{cases} E_c(\tau, c) & if\ \mathbb{C}(t,c) = 1 \\ \texttt{not}(E_c(\tau,c)) & if\ \mathbb{C}(t,c) = -1 \end{cases} = \texttt{true}.}$

As $c \in conds(t)$, let us perform case analysis on $\mathbb{C}(t,c) = 1 \vee \mathbb{C}(t,c) = -1$:

(a) $\mathbb{C}(t,c) = 1$: $\boxed{E_c(\tau,c) = \texttt{true}.}$

By definition of $t \in Firable(s')$, we can deduce that $s'.cond(c) = \texttt{true}$. By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s'.cond(c) = E_c(\tau,c)$. Thus, $E_c(\tau,c) = \texttt{true}.$

(b) $\mathbb{C}(t,c) = -1$: $\boxed{\texttt{not}\ E_c(\tau,c) = \texttt{true}.}$

By definition of $t \in Firable(s')$, we can deduce that $s'.cond(c) = \texttt{false}$. By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s'.cond(c) = E_c(\tau,c)$. Thus, $\texttt{not}\ E_c(\tau,c) = \texttt{true}.$

3. $\boxed{\texttt{checktc}(\Delta(id_t), \sigma(id_t)) = \texttt{true}}$:

By definition of $t \in Firable(s')$, we have $t \notin T_i \vee s'.I(t) \in I_s(t)$. Let us perform case analysis on $t \notin T_i \vee s'.I(t) \in I_s(t)$:

(a) $t \notin T_i$:

By construction, $<\texttt{transition\_type} \Rightarrow \texttt{NOT\_TEMP}> \in gm_t$, and by property of the elaboration relation, we have $\Delta(id_t)(''tt'') = \texttt{NOT\_TEMP}$.
From $\Delta(id_t)(''tt'') = \texttt{NOT\_TEMP}$, and the definition of $\texttt{checktc}(\Delta(id_t), \sigma(id_t))$, we can deduce $\texttt{checktc}(\Delta(id_t), \sigma(id_t)) = \texttt{true}.$

(b) $s'.I(t) \in I_s(t)$:

From $s'.I(t) \in I_s(t)$, we can deduce that $t \in T_i$. Thus, by construction, there exists $tt \in \{\texttt{TEMP\_A\_B}, \texttt{TEMP\_A\_A}, \texttt{TEMP\_A\_INF}\}$ s.t. $<\texttt{transition\_type} \Rightarrow tt> \in gm_t$. By property of the elaboration relation, we have $\Delta(id_t)(''tt'') = tt$, and thus, we know $\Delta(id_t)(''tt'') \neq$

NOT_TEMP. Therefore, we can simplfy the term $\texttt{checktc}(\Delta(id_t),\sigma(id_t))$ as follows:

$$\begin{aligned}\texttt{checktc}(\Delta(id_t),\sigma(id_t)) = \Big(&\texttt{not}\ \sigma(id_t)("srtc")\ .\\ &\big[\big(\Delta(id_t)("tt") = \texttt{TEMP\_A\_B}\ .\ (\sigma(id_t)("stc") \geq \sigma(id_t)("A") - 1)\\ &\qquad .\ (\sigma(id_t)("stc") \leq \sigma(id_t)("B") - 1)\big)\\ &+(\Delta(id_t)("tt") = \texttt{TEMP\_A\_A}\ .\\ &\quad(\sigma(id_t)("stc") = \sigma(id_t)("A") - 1))\\ &+(\Delta(id_t)("tt") = \texttt{TEMP\_A\_INF}\ .\\ &\quad(\sigma(id_t)("stc") \geq \sigma(id_t)("A") - 1))\big]\Big)\\ &+\big(\sigma(id_t)("srtc")\ .\ \sigma(id_t)("A") = 1\big)\end{aligned} \tag{C.167}$$

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have $s.reset_t(t) = \sigma(id_t)("srtc")$.
Let us perform case analysis on the value $s.reset_t(t)$:

i. $s.reset_t(t) = \texttt{true}$:

Then, from $s.reset_t(t) = \sigma(id_t)("srtc")$, we can deduce that $\sigma(id_t)("srtc") = \texttt{true}$. From $\sigma(id_t)("srtc") = \texttt{true}$, we can simplify the term $\texttt{checktc}(\Delta(id_t),\sigma(id_t))$ as follows:

$$\texttt{checktc}(\Delta(id_t),\sigma(id_t)) = \big(\sigma(id_t)("A") = 1\big) \tag{C.168}$$

Rewriting the goal with (C.168), and simplifying the goal: $\boxed{\sigma(id_t)("A") = 1.}$

By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, from $t \in Sens(s.M)$ and $s.reset_t(t) = \texttt{true}$, we can deduce $s'.I(t) = 1$. We know that $s'.I(t) \in I_s(t)$, and thus, we have $1 \in I_s(t)$. By definition of $1 \in I_s(t)$, there exist an $a \in \mathbb{N}^*$ and a $ni \in \mathbb{N}^* \sqcup \{\infty\}$ s.t. $I_s(t) = [a, ni]$ and $1 \in [a, ni]$.
By definition of $1 \in [a, ni]$, we have $a \leq 1$, and since $a \in \mathbb{N}^*$, we can deduce $a = 1$.
By construction, $<\texttt{time\_A\_value} \Rightarrow a> \in ipm_t$, and by property of stable σ, we have $\sigma(id_t)("A") = a = 1$.

ii. $s.reset_t(t) = \texttt{false}$:

Then, from $s.reset_t(t) = \sigma(id_t)("srtc")$, we can deduce that $\sigma(id_t)("srtc") = \texttt{false}$. From $\sigma(id_t)("srtc") = \texttt{false}$, we can simplify the term $\texttt{checktc}(\Delta(id_t),\sigma(id_t))$ as follows:

$$\begin{aligned}&\texttt{checktc}(\Delta(id_t),\sigma(id_t))\\ &=\\ &\big(\Delta(id_t)("tt") = \texttt{TEMP\_A\_B}\ .\ (\sigma(id_t)("stc") \geq \sigma(id_t)("A") - 1)\\ &\qquad .\ (\sigma(id_t)("stc") \leq \sigma(id_t)("B") - 1)\big)\\ &+(\Delta(id_t)("tt") = \texttt{TEMP\_A\_A}\ .\ (\sigma(id_t)("stc") = \sigma(id_t)("A") - 1))\\ &+(\Delta(id_t)("tt") = \texttt{TEMP\_A\_INF}\ .\ (\sigma(id_t)("stc") \geq \sigma(id_t)("A") - 1))\end{aligned} \tag{C.169}$$

Let us perform case analysis on $I_s(t)$; there are two cases:

- $I_s(t) = [a, b]$ where $a, b \in \mathbb{N}^*$; then, either $a = b$ or $a \neq b$:
  - $a = b$:
    Then, we have $I_s(t) = [a, a]$, and by construction <transition_type ⇒ TEMP_A_A> $\in gm_t$. By property of the elaboration relation, we have $\Delta(id_t)(''tt'') =$ TEMP_A_A; thus we can simplify the term checktc as follows:

    $$\texttt{checktc}(\Delta(id_t), \sigma(id_t)) = (\sigma(id_t)(''stc'') = \sigma(id_t)(''A'') - 1) \tag{C.170}$$

    Rewriting the goal with (C.170), and simplifying the goal:
    $\boxed{\sigma(id_t)(''stc'') = \sigma(id_t)(''A'') - 1.}$
    From $s'.I(t) \in [a, a]$, we can deduce that $s'.I(t) = a$. Let us perform case analysis on $s.I(t) < upper(I_s(t))$ or $s.I(t) \geq upper(I_s(t))$:
    - $s.I(t) < upper(I_s(t))$:
      By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have $s.I(t) = \sigma(id_t)(''stc'')$. By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s'.I(t) = s.I(t) + 1$. From $s'.I(t) = a$ and $s'.I(t) = s.I(t) + 1$, we can deduce $a - 1 = s.I(t)$.
      By construction, <time_A_value ⇒a> $\in ipm_t$, and by property of stable σ, we have $\sigma(id_t)(''A'') = a$.
      Rewriting the goal with $\sigma(id_t)(''A'') = a$ and $s.I(t) = \sigma(id_t)(''stc'')$:
      $\sigma(id_t)(''stc'') = \sigma(id_t)(''A'') - 1.$
    - $s.I(t) \geq upper(I_s(t))$:
      In the case where $s.I(t) > upper(I_s(t))$, then $s.I(t) > a$. By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s.I(t) = s'.I(t) = a$. Then, $a > a$ is a contradiction.

      In the case where $s.I(t) = upper(I_s(t))$, then $s.I(t) = a$. By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s'.I(t) = s.I(t) + 1$. Then, we have $s'.I(t) = a$ and $s'.I(t) = a + 1$. Then, $a = a + 1$ is a contradiction.
  - $a \neq b$:
    Then, we have $I_s(t) = [a, b]$, and by construction <transition_type ⇒ TEMP_A_B> $\in gm_t$. By property of the elaboration relation, we have $\Delta(id_t)(''tt'') =$ TEMP_A_B; thus we can simplify the term checktc as follows:

    $$\begin{gathered}\texttt{checktc}(\Delta(id_t), \sigma(id_t)) \\ = \\ (\sigma(id_t)(''stc'') \geq \sigma(id_t)(''A'') - 1)\ .\ (\sigma(id_t)(''stc'') \leq \sigma(id_t)(''B'') - 1)\end{gathered} \tag{C.171}$$

    Rewriting the goal with (C.171), and simplifying the goal:
    $\boxed{(\sigma(id_t)(''stc'') \geq \sigma(id_t)(''A'') - 1) \wedge (\sigma(id_t)(''stc'') \leq \sigma(id_t)(''B'') - 1).}$

    Let us perform case analysis on $s.I(t) < upper(I_s(t))$ or $s.I(t) \geq upper(I_s(t))$:
    - $s.I(t) < upper(I_s(t))$:
      By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have $s.I(t) = \sigma(id_t)(''stc'')$. By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s'.I(t) = s.I(t) + 1$. By definition of $s'.I(t) \in [a, b]$:
      $\Rightarrow a \leq s'.I(t) \leq b$.

$\Rightarrow a \leq s'.I(t) \wedge s'.I(t) \leq b$
$\Rightarrow a \leq s.I(t) + 1 \wedge s.I(t) + 1 \leq b$
$\Rightarrow a - 1 \leq s.I(t) \wedge s.I(t) \leq b - 1$
By construction, $<\texttt{time\_A\_value} \Rightarrow a> \in ipm_t$ and $<\texttt{time\_B\_value} \Rightarrow b> \in ipm_t$, and by property of stable σ, we have $\sigma(id_t)(''A'') = a$ and $\sigma(id_t)(''B'') = b$.
Rewriting the goal with $\sigma(id_t)(''A'') = a, \sigma(id_t)(''B'') = b$ and $s.I(t) = \sigma(id_t)(''stc'')$:
$a - 1 \leq s.I(t) \wedge s.I(t) \leq b - 1.$

∗ $s.I(t) \geq upper(I_s(t))$:
In the case where $s.I(t) > upper(I_s(t))$, then $s.I(t) > b$. By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s.I(t) = s'.I(t) = b$. Then, $b > b$ is a contradiction.

In the case where $s.I(t) = upper(I_s(t))$, then $s.I(t) = b$. By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s'.I(t) = s.I(t) + 1$.
By definition of $s'.I(t) \in [a, b]$, we have $s'.I(t) \leq b$:
$\Rightarrow s.I(t) + 1 \leq b$
$\Rightarrow$ $b + 1 \leq b$ is contradiction.

- $I_s(t) = [a, \infty]$ where $a \in \mathbb{N}^*$:
By construction $<\texttt{transition\_type} \Rightarrow \texttt{TEMP\_A\_INF}> \in gm_t$. By property of the elaboration relation, we have $\Delta(id_t)(''tt'') = \texttt{TEMP\_A\_INF}$; thus we can simplify the term `checktc` as follows:

$$\texttt{checktc}(\Delta(id_t), \sigma(id_t)) = (\sigma(id_t)(''stc'') \geq \sigma(id_t)(''A'') - 1) \tag{C.172}$$

Rewriting the goal with (C.172), and simplifying the goal:
$\boxed{\sigma(id_t)(''stc'') \geq \sigma(id_t)(''A'') - 1.}$
From $s'.I(t) \in [a, \infty]$, we can deduce $a \leq s'.I(t)$. Then, let us perform case analysis on $s.I(t) \leq lower(I_s(t))$ or $s.I(t) > lower(I_s(t))$:

  – $s.I(t) \leq lower(I_s(t))$:
  By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have $s.I(t) = \sigma(id_t)(''stc'')$.
  By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s'.I(t) = s.I(t) + 1$:
  $\Rightarrow a \leq s'.I(t)$
  $\Rightarrow a \leq s.I(t) + 1$
  $\Rightarrow a - 1 \leq s.I(t)$
  By construction, $<\texttt{time\_A\_value} \Rightarrow a> \in ipm_t$, and by property of stable σ, we have $\sigma(id_t)(''A'') = a$.
  Rewriting the goal with $\sigma(id_t)(''A'') = a$ and $s.I(t) = \sigma(id_t)(''stc'')$:
  $a - 1 \leq s.I(t).$

  – $s.I(t) > lower(I_s(t))$:
  By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have $\sigma(id_t)(''stc'') = lower(I_s(t)) = a$.
  By construction, $<\texttt{time\_A\_value} \Rightarrow a> \in ipm_t$, and by property of stable σ, we have $\sigma(id_t)(''A'') = a$.
  Rewriting the goal with $\sigma(id_t)(''stc'') = a$ and $\sigma(id_t)(''A'') = a$: $a - 1 \leq a.$

□

Lemma 43 (Falling Edge Equal Firable 2)**.** *For all* $sitpn$, d, γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_{\downarrow}$, σ' *that verify the hypotheses of Def. 6, then* $\forall t \in T, id_t \in Comps(\Delta)$ *s.t.* $\gamma(t) = id_t$, $\sigma'(id_t)(''s\_firable'') = \mathtt{true} \Rightarrow t \in Firable(s')$*.*

Proof. Given a $t \in T$ and $id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$, and assuming that $\sigma'(id_t)(''s\_firable'') = \mathtt{true}$, let us show $\boxed{t \in Firable(s').}$

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 $Inject_{\downarrow}$, the $\mathcal{H}$-VHDL falling edge, the stabilize relations and $\mathtt{comp}(id_t, ''transition'', gm_t, ipm_t, opm_t) \in d.cs$:

$$\sigma'(id_t)(''sfa'') = \sigma(id_t)(''se'')\ .\ \sigma(id_t)(''scc'')\ .\ \mathtt{checktc}(\Delta(id_t), \sigma(id_t)) = \mathtt{true} \tag{C.173}$$

From (C.173), we can deduce:

$$\sigma(id_t)(''se'') = \mathtt{true} \tag{C.174}$$

$$\sigma(id_t)(''scc'') = \mathtt{true} \tag{C.175}$$

$$\mathtt{checktc}(\Delta(id_t), \sigma(id_t)) = \mathtt{true} \tag{C.176}$$

Term $\mathtt{checktc}(\Delta(id_t), \sigma(id_t))$ as the same definition as in Lemma Falling Edge Equal Firable 1. By definition of $t \in Firable(s')$, there are three points to prove:

1. $\boxed{t \in Sens(s'.M)}$

2. $\boxed{t \notin T_i \lor s'.I(t) \in I_s(t)}$

3. $\boxed{\forall c \in \mathcal{C},\ \mathbb{C}(t,c) = 1 \Rightarrow s'.cond(c) = \mathtt{true} \text{ and } \mathbb{C}(t,c) = -1 \Rightarrow s'.cond(c) = \mathtt{false}}$

Let us prove these three points:

1. $\boxed{t \in Sens(s'.M)}$:

   By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s.M = s'.M$. Rewriting the goal with $s.M = s'.M$: $\boxed{t \in Sens(s.M).}$

   By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have $\sigma(id_t)(''se'') = \mathtt{true} \Leftrightarrow t \in Sens(s.M)$.

   $t \in Sens(s.M)$.

2. $\boxed{\forall c \in \mathcal{C},\ \mathbb{C}(t,c) = 1 \Rightarrow s'.cond(c) = \mathtt{true} \text{ and } \mathbb{C}(t,c) = -1 \Rightarrow s'.cond(c) = \mathtt{false}}$

   Given a $c \in \mathcal{C}$, there are two points to prove:

   (a) $\boxed{\mathbb{C}(t,c) = 1 \Rightarrow s'.cond(c) = \mathtt{true}.}$

   (b) $\boxed{\mathbb{C}(t,c) = -1 \Rightarrow s'.cond(c) = \mathtt{false}.}$

   Let us prove these two points:

(a) Assuming that $\mathbb{C}(t,c) = 1$, let us show $\boxed{s'.cond(c) = \texttt{true}.}$

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have:

$$\sigma(id_t)(''scc'') = \prod_{c \in conds(t)} \begin{cases} E_c(\tau,c) & if\ \mathbb{C}(t,c) = 1 \\ \texttt{not}(E_c(\tau,c)) & if\ \mathbb{C}(t,c) = -1 \end{cases} = \texttt{true} \tag{C.177}$$

where $conds(t) = \{c \in \mathcal{C} \mid \mathbb{C}(t,c) = 1 \vee \mathbb{C}(t,c) = -1\}$.
As $c \in conds(t)$ and $\mathbb{C}(t,c) = 1$, and by definition of the product expression, we have:

$$E_c(\tau,c)\ .\prod_{c' \in conds(t)\setminus\{c\}} \begin{cases} E_c(\tau,c') & if\ \mathbb{C}(t,c') = 1 \\ \texttt{not}(E_c(\tau,c')) & if\ \mathbb{C}(t,c') = -1 \end{cases} = \texttt{true} \tag{C.178}$$

From (C.178), we can deduce that $E_c(\tau,c) = \texttt{true}$.

By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s'.cond(c) = E_c(\tau,c)$.

Rewriting the goal with $s'.cond(c) = E_c(\tau,c)$ and $E_c(\tau,c) = \texttt{true}$: tautology.

(b) Assuming that $\mathbb{C}(t,c) = -1$, let us show $\boxed{s'.cond(c) = \texttt{false}.}$

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have:

$$\sigma(id_t)(''scc'') = \prod_{c \in conds(t)} \begin{cases} E_c(\tau,c) & if\ \mathbb{C}(t,c) = 1 \\ \texttt{not}(E_c(\tau,c)) & if\ \mathbb{C}(t,c) = -1 \end{cases} = \texttt{true} \tag{C.179}$$

where $conds(t) = \{c \in \mathcal{C} \mid \mathbb{C}(t,c) = 1 \vee \mathbb{C}(t,c) = -1\}$.
As $c \in conds(t)$ and $\mathbb{C}(t,c) = -1$, and by definition of the product expression, we have:

$$\texttt{not}\ E_c(\tau,c)\ .\prod_{c' \in conds(t)\setminus\{c\}} \begin{cases} E_c(\tau,c') & if\ \mathbb{C}(t,c') = 1 \\ \texttt{not}(E_c(\tau,c')) & if\ \mathbb{C}(t,c') = -1 \end{cases} = \texttt{true} \tag{C.180}$$

From (C.180), we can deduce that $E_c(\tau,c) = \texttt{false}$.

By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s'.cond(c) = E_c(\tau,c)$.

Rewriting the goal with $s'.cond(c) = E_c(\tau,c)$ and $E_c(\tau,c) = \texttt{false}$: tautology.

3. $\boxed{t \notin T_i \vee s'.I(t) \in I_s(t)}$

Reasoning on $\texttt{checktc}(\Delta(id_t), \sigma(id_t)) = \texttt{true}$, there are 3 cases:

(a) $\left(\texttt{not}\ \sigma(id_t)(''srtc'')\ .\ [\ldots]\right) = \texttt{true}$[2]

(b) $\left(\sigma(id_t)(''srtc'')\ .\ \Delta(id_t)(''tt'') \neq \texttt{NOT\_TEMP}\ .\ \sigma(id_t)(''A'') = 1\right) = \texttt{true}$

(c) $\left(\Delta(id_t)(''tt'') = \texttt{NOT\_TEMP}\right) = \texttt{true}$

(a) $\left(\texttt{not}\ \sigma(id_t)(''srtc'')\ .\ [\ldots]\right) = \texttt{true}$:

[2] See equation (C.165) for the full definition

Then, we can deduce $\texttt{not}\ \sigma(id_t)(''srtc'') = \texttt{true}$ and $[\dots] = \texttt{true}$. From $\texttt{not}\ \sigma(id_t)(''srtc'') = \texttt{true}$, we can deduce $\sigma(id_t)(''srtc'') = \texttt{false}$, and from $[\dots] = \texttt{true}$, we have three other cases:

i. $(\Delta(id_t)(''tt'') = \texttt{TEMP\_A\_B} \,.\, (\sigma(id_t)(''stc'') \geq \sigma(id_t)(''A'') - 1) \,.\, (\sigma(id_t)(''stc'') \leq \sigma(id_t)(''B'') - 1)) = \texttt{true}$

ii. $(\Delta(id_t)(''tt'') = \texttt{TEMP\_A\_A} \,.\, (\sigma(id_t)(''stc'') = \sigma(id_t)(''A'') - 1)) = \texttt{true}$

iii. $(\Delta(id_t)(''tt'') = \texttt{TEMP\_A\_INF} \,.\, (\sigma(id_t)(''stc'') \geq \sigma(id_t)(''A'') - 1)) = \texttt{true}$

Let us prove the goal is these three contexts:

i. $(\Delta(id_t)(''tt'') = \texttt{TEMP\_A\_B} \,.\, (\sigma(id_t)(''stc'') \geq \sigma(id_t)(''A'') - 1) \,.\, (\sigma(id_t)(''stc'') \leq \sigma(id_t)(''B'') - 1)) = \texttt{true}$:

Then, converting boolean equalities into intuitionistic predicates, we have:

- $\Delta(id_t)(''tt'') = \texttt{TEMP\_A\_B}$
- $\sigma(id_t)(''stc'') \geq \sigma(id_t)(''A'') - 1$
- $\sigma(id_t)(''stc'') \leq \sigma(id_t)(''B'') - 1$

By property of the elaboration relation, and $\Delta(id_t)(''tt'') = \texttt{TEMP\_A\_B}$, there exist $a, b \in \mathbb{N}^*$ s.t. $I_s(t) = [a, b]$. Let us take such an a and b. Then, let us show $\boxed{s'.I(t) \in I_s(t).}$

Rewriting the goal with $I_s(t) = [a, b]$: $\boxed{s'.I(t) \in [a, b].}$

By construction, $<\texttt{time\_A\_value} \Rightarrow a>$ and $<\texttt{time\_B\_value} \Rightarrow b>$, and by property of stable σ, we have $\sigma(id_t)(''A'') = a$ and $\sigma(id_t)(''B'') = b$.

Rewriting the goal with $\sigma(id_t)(''A'') = a$ and $\sigma(id_t)(''B'') = b$, and by definition of $\in$: $\boxed{\sigma(id_t)(''A'') \leq s'.I(t) \leq \sigma(id_t)(''B'').}$

Now, let us perform case analysis on $s.I(t) \leq upper(I_s(t))$ or $s.I(t) > upper(I_s(t))$:

- $s.I(t) \leq upper(I_s(t))$:

  By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have $s.I(t) = \sigma(id_t)(''stc'')$.
  From $\sigma(id_t)(''se'') = \texttt{true}$, we can deduce $t \in Sens(s.M)$, and from $\sigma(id_t)(''srtc'') = \texttt{false}$, we can deduce $s.reset_t(t) = \texttt{false}$. Then, by definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s'.I(t) = s.I(t) + 1$.

  $\Rightarrow \boxed{\sigma(id_t)(''A'') \leq s.I(t) + 1 \leq \sigma(id_t)(''B'')}$ (by $s'.I(t) = s.I(t) + 1$)

  $\Rightarrow \boxed{\sigma(id_t)(''A'') \leq \sigma(id_t)(''stc'') + 1 \leq \sigma(id_t)(''B'')}$ (by $s.I(t) = \sigma(id_t)(''stc'')$)

  $\Rightarrow \sigma(id_t)(''A'') - 1 \leq \sigma(id_t)(''stc'') \leq \sigma(id_t)(''B'') - 1$

- $s.I(t) > upper(I_s(t))$:

  By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have $\sigma(id_t)(''stc'') = upper(I_s(t)) = b$.
  Then, from $\sigma(id_t)(''stc'') \leq \sigma(id_t)(''B'') - 1$, $\sigma(id_t)(''stc'') = upper(I_s(t)) = b$ and $\sigma(id_t)(''B'') = b$, we can deduce the following contradiction:

  $\sigma(id_t)(''B'') \leq \sigma(id_t)(''B'') - 1.$

ii. $(\Delta(id_t)(''tt'') = \texttt{TEMP\_A\_A} \,.\, (\sigma(id_t)(''stc'') = \sigma(id_t)(''A'') - 1)) = \texttt{true}$:

Then, converting boolean equalities into intuitionistic predicates, we have:

- $\Delta(id_t)(''tt'') = \texttt{TEMP\_A\_A}$
- $\sigma(id_t)(''stc'') = \sigma(id_t)(''A'') - 1$

By property of the elaboration relation, and $\Delta(id_t)(''tt'') = \texttt{TEMP\_A\_A}$, there exist $a \in \mathbb{N}^*$ s.t. $I_s(t) = [a, a]$. Let us take such an a. Then, let us show $\boxed{s'.I(t) \in I_s(t).}$

Rewriting the goal with $I_s(t) = [a, a]$: $\boxed{s'.I(t) \in [a, a].}$
By construction, $<\texttt{time\_A\_value} \Rightarrow a>$, and by property of stable σ, we have $\sigma(id_t)(''A'') = a$.
Rewriting the goal with $\sigma(id_t)(''A'') = a$, unfolding the definition of $\in$, and simplifying the goal: $\boxed{s'.I(t) = \sigma(id_t)(''A'').}$
Now, let us perform case analysis on $s.I(t) \leq upper(I_s(t))$ or $s.I(t) > upper(I_s(t))$:

- $s.I(t) \leq upper(I_s(t))$:

  By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have $s.I(t) = \sigma(id_t)(''stc'')$.
  From $\sigma(id_t)(''se'') = \texttt{true}$, we can deduce $t \in Sens(s.M)$, and from $\sigma(id_t)(''srtc'') = \texttt{false}$, we can deduce $s.reset_t(t) = \texttt{false}$. Then, by definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s'.I(t) = s.I(t) + 1$.
  $\Rightarrow \boxed{s.I(t) + 1 = \sigma(id_t)(''A'')}$ (by $s'.I(t) = s.I(t) + 1$)
  $\Rightarrow \boxed{\sigma(id_t)(''stc'') + 1 = \sigma(id_t)(''A'')}$ (by $s.I(t) = \sigma(id_t)(''stc'')$)
  $\Rightarrow \sigma(id_t)(''stc'') = \sigma(id_t)(''A'') - 1$

- $s.I(t) > upper(I_s(t))$:

  By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have $\sigma(id_t)(''stc'') = upper(I_s(t)) = a$.
  Then, from $\sigma(id_t)(''stc'') = \sigma(id_t)(''A'') - 1, \sigma(id_t)(''stc'') = upper(I_s(t)) = a, \sigma(id_t)(''A'') = a$, and $a \in \mathbb{N}^*$, we can deduce the following contradiction:
  $\sigma(id_t)(''A'') = \sigma(id_t)(''A'') - 1.$

iii. $(\Delta(id_t)(''tt'') = \texttt{TEMP\_A\_INF}\ .\ (\sigma(id_t)(''stc'') \geq \sigma(id_t)(''A'') - 1)) = \texttt{true}$:
Then, converting boolean equalities into intuitionistic predicates, we have:

- $\Delta(id_t)(''tt'') = \texttt{TEMP\_A\_INF}$
- $\sigma(id_t)(''stc'') \geq \sigma(id_t)(''A'') - 1$

By property of the elaboration relation, and $\Delta(id_t)(''tt'') = \texttt{TEMP\_A\_INF}$, there exist $a \in \mathbb{N}^*$ s.t. $I_s(t) = [a, \infty]$. Let us take such an a. Then, let us show $\boxed{s'.I(t) \in I_s(t).}$
Rewriting the goal with $I_s(t) = [a, \infty]$: $\boxed{s'.I(t) \in [a, \infty].}$
By construction, $<\texttt{time\_A\_value} \Rightarrow a>$, and by property of stable σ, we have $\sigma(id_t)(''A'') = a$.
Rewriting the goal with $\sigma(id_t)(''A'') = a$, unfolding the definition of $\in$, and simplifying the goal: $\boxed{\sigma(id_t)(''A'') \leq s'.I(t).}$
Now, let us perform case analysis on $s.I(t) \leq lower(I_s(t))$ or $s.I(t) > lower(I_s(t))$:

- $s.I(t) \leq lower(I_s(t))$:

  By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$, we have $s.I(t) = \sigma(id_t)(''stc'')$.
  From $\sigma(id_t)(''se'') = \texttt{true}$, we can deduce $t \in Sens(s.M)$, and from $\sigma(id_t)(''srtc'') = \texttt{false}$, we can deduce $s.reset_t(t) = \texttt{false}$. Then, by definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s'.I(t) = s.I(t) + 1$.
  $\Rightarrow \boxed{\sigma(id_t)(''A'') \leq s.I(t) + 1}$ (by $s'.I(t) = s.I(t) + 1$)
  $\Rightarrow \boxed{\sigma(id_t)(''A'') \leq \sigma(id_t)(''stc'') + 1}$ (by $s.I(t) = \sigma(id_t)(''stc'')$)
  $\Rightarrow \sigma(id_t)(''A'') - 1 \leq \sigma(id_t)(''stc'')$

- $s.I(t) > lower(I_s(t))$:

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\leadsto} \sigma$, we have $\sigma(id_t)("stc") = lower(I_s(t)) = a$.
From $\sigma(id_t)("se") = \texttt{true}$, we can deduce $t \in Sens(s.M)$, and from $\sigma(id_t)("srtc") = \texttt{false}$, we can deduce $s.reset_t(t) = \texttt{false}$. Then, by definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, we have $s'.I(t) = s.I(t) + 1$.

$\Rightarrow \boxed{\sigma(id_t)("A") \leq s.I(t) + 1}$ (by $s'.I(t) = s.I(t) + 1$)

$\Rightarrow \boxed{a \leq s.I(t) + 1}$ (by $\sigma(id_t)("A") = a$)

$\Rightarrow \boxed{a < s.I(t)}$

$\Rightarrow lower(I_s(t)) < s.I(t)$

(b) $\big(\sigma(id_t)("srtc")\ .\ \Delta(id_t)("tt") \neq \texttt{NOT\_TEMP}\ .\ \sigma(id_t)("A") = 1\big) = \texttt{true}$

Then, converting boolean equalities into intuitionistic predicates, we have:

- $\sigma(id_t)("srtc") = \texttt{true}$
- $\Delta(id_t)("tt") \neq \texttt{NOT\_TEMP}$
- $\sigma(id_t)("A") = 1$

By property of the elaboration relation, and $\Delta(id_t)("tt") \neq \texttt{NOT\_TEMP}$, there exist an $a \in \mathbb{N}^*$ and a $ni \in \mathbb{N}^* \sqcup \{\infty\}$ s.t. $I_s(t) = [a, ni]$. Let us take such an a and ni.
By construction, $<\texttt{time\_A\_value} \Rightarrow a> \in ipm_t$, and by property of stable σ, we have $\sigma(id_t)("A") = a$. Thus, we can deduce $a = 1$ and $I_s(t) = [1, ni]$.

By definition of $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\leadsto} \sigma$, from $\sigma(id_t)("se") = \texttt{true}$, we can deduce $t \in Sens(s.M)$, and from $\sigma(id_t)("srtc") = \texttt{true}$, we can deduce $s.reset_t(t) = \texttt{true}$.

By definition of $E_c, \tau \vdash s \xrightarrow{\downarrow} s'$, $t \in Sens(s.M)$ and $s.reset_t(t) = \texttt{true}$, we have $s'.I(t) = 1$.

Now, let us show $\boxed{s'.I(t) \in I_s(t)}$.

Rewriting the goal with $s'.I(t) = 1$ and $I_s(t) = [1, ni]$: $1 \in [1, ni]$.

(c) $\big(\Delta(id_t)("tt") = \texttt{NOT\_TEMP}\big) = \texttt{true}$

Let us show $\boxed{t \notin T_i.}$

By property of the elaboration relation and $\Delta(id_t)("tt") = \texttt{NOT\_TEMP}$, we have $t \notin T_i$.

□

Lemma 44 (Falling Edge Equal Not Firable). *For all sitpn, d, γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_\downarrow$, σ' that verify the hypotheses of Def. 6, then $\forall t \in T, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$, $t \in Firable(s') \Leftrightarrow \sigma'(id_t)("s\_firable") = \texttt{true}$.*

Proof. Proving the above lemma is trivial by appealing to Lemma Falling Edge Equal Firable and by reasoning on contrapositives. □

Bibliography

[1] Karima Berramla, El Abbassia Deba, and Mohammed Senouci. “Formal Validation of Model Transformation with Coq Proof Assistant”. In: *2015 First International Conference on New Technologies of Information and Communication (NTIC)*. 2015 First International Conference on New Technologies of Information and Communication (NTIC). Nov. 2015, pp. 1–6. DOI: 10.1109/NTIC.2015.7368755.

[2] Sandrine Blazy, Zaynah Dargaye, and Xavier Leroy. “Formal Verification of a C Compiler Front-End”. In: *FM 2006: Formal Methods*. International Symposium on Formal Methods. Springer, Berlin, Heidelberg, Aug. 21, 2006, pp. 460–475. DOI: 10.1007/11813040_31. URL: https://link.springer.com/chapter/10.1007/11813040_31 (visited on 05/25/2020).

[3] Thomas Bourgeat et al. “The Essence of Bluespec: A Core Language for Rule-Based Hardware Design”. In: *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*. PLDI 2020. New York, NY, USA: Association for Computing Machinery, June 11, 2020, pp. 243–257. ISBN: 978-1-4503-7613-6. DOI: 10.1145/3385412.3385965. URL: https://doi.org/10.1145/3385412.3385965 (visited on 05/05/2021).

[4] Timothy Bourke et al. “A Formally Verified Compiler for Lustre”. In: (), p. 17.

[5] Thomas Braibant and Adam Chlipala. “Formal Verification of Hardware Synthesis”. In: *Computer Aided Verification*. Ed. by Natasha Sharygina and Helmut Veith. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2013, pp. 213–228. ISBN: 978-3-642-39799-8. DOI: 10.1007/978-3-642-39799-8_14.

[6] Daniel Calegari et al. “A Type-Theoretic Framework for Certified Model Transformations”. In: *Formal Methods: Foundations and Applications*. Ed. by Jim Davies, Leila Silva, and Adenilso Simao. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, pp. 112–127. ISBN: 978-3-642-19829-8. DOI: 10.1007/978-3-642-19829-8_8.

[7] Adam Chlipala. “A Verified Compiler for an Impure Functional Language”. In: *ACM SIGPLAN Notices* 45.1 (Jan. 17, 2010), pp. 93–106. ISSN: 0362-1340. DOI: 10.1145/1707801.1706312. URL: https://doi.org/10.1145/1707801.1706312 (visited on 05/22/2020).

[8] Benoît Combemale et al. “Essay on Semantics Definition in MDE. An Instrumented Approach for Model Verification”. In: *Journal of Software* 4 (Nov. 1, 2009). DOI: 10.4304/jsw.4.9.943-958.

[9] Johannes Dyck, Holger Giese, and Leen Lambers. “Automatic Verification of Behavior Preservation at the Transformation Level for Relational Model Transformation”. In: *Software & Systems Modeling* 18.5 (5 Oct. 1, 2019), pp. 2937–2972. ISSN: 1619-1374. DOI: 10.1007/s10270-018-00706-9. URL: https://link.springer.com/article/10.1007/s10270-018-00706-9 (visited on 05/22/2020).

[10] Lukasz Fronc and Franck Pommereau. “Towards a Certified Petri Net Model-Checker”. In: *Programming Languages and Systems*. Ed. by Hongseok Yang. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, pp. 322–336. ISBN: 978-3-642-25318-8. DOI: 10.1007/978-3-642-25318-8_24.

[11] A. Habibi and S. Tahar. “Design and Verification of SystemC Transaction-Level Models”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 14.1 (Jan. 2006), pp. 57–68. ISSN: 1557-9999. DOI: 10.1109/TVLSI.2005.863187.

[12] Xavier Leroy. “A Formally Verified Compiler Back-End”. In: *Journal of Automated Reasoning* 43.4 (Nov. 4, 2009), p. 363. ISSN: 1573-0670. DOI: 10.1007/s10817-009-9155-4. URL: https://doi.org/10.1007/s10817-009-9155-4 (visited on 01/21/2020).

[13] Andreas Lööw. “Lutsig: A Verified Verilog Compiler for Verified Circuit Development”. In: *Proceedings of the 10th ACM SIGPLAN International Conference on Certified Programs and Proofs*. CPP 2021. New York, NY, USA: Association for Computing Machinery, Jan. 17, 2021, pp. 46–60. ISBN: 978-1-4503-8299-1. DOI: 10.1145/3437992.3439916. URL: https://doi.org/10.1145/3437992.3439916 (visited on 05/04/2021).

[14] Said Meghzili et al. “On the Verification of UML State Machine Diagrams to Colored Petri Nets Transformation Using Isabelle/HOL”. In: *2017 IEEE International Conference on Information Reuse and Integration (IRI)*. 2017 IEEE International Conference on Information Reuse and Integration (IRI). Aug. 2017, pp. 419–426. DOI: 10.1109/IRI.2017.63.

[15] Martin Strecker. “Formal Verification of a Java Compiler in Isabelle”. In: *Automated Deduction—CADE-18*. International Conference on Automated Deduction. Springer, Berlin, Heidelberg, July 27, 2002, pp. 63–77. DOI: 10.1007/3-540-45620-1_5. URL: https://link.springer.com/chapter/10.1007/3-540-45620-1_5 (visited on 06/08/2020).

[16] Yong Kiam Tan et al. “A New Verified Compiler Backend for CakeML”. In: (Sept. 4, 2016). DOI: 10.17863/CAM.6525.

[17] Yong Kiam Tan et al. “A New Verified Compiler Backend for CakeML”. In: (), p. 14.

[18] Zhibin Yang et al. “From AADL to Timed Abstract State Machines: A Verified Model Transformation”. In: *Journal of Systems and Software* 93 (July 1, 2014), pp. 42–68. ISSN: 0164-1212. DOI: 10.1016/j.jss.2014.02.058. URL: http://www.sciencedirect.com/science/article/pii/S0164121214000727 (visited on 01/16/2020).

[19] Zhibin Yang et al. “Towards a Verified Compiler Prototype for the Synchronous Language SIGNAL”. In: *Frontiers of Computer Science* 10.1 (Feb. 1, 2016), pp. 37–53. ISSN: 2095-2236. DOI: 10.1007/s11704-015-4364-y. URL: https://doi.org/10.1007/s11704-015-4364-y (visited on 01/21/2020).