Projet ANR VALMEM

Délivrable : D3.2

Titre : Model-checking temporisé spécialisé pour les circuits mémoire

Auteurs : E. André, E. Encrenaz, L. Fribourg

Version : 1

Date : 30 juin 2008

VALMEM : *Validation fonctionnelle et temporelle des mémoires embarquées décrites au niveau transistor par des méthodes formelles*

Model-Checking temporisé spécialisé pour les circuits mémoire Livrable D3.2

É. André[1], E. Encrenaz[2], L. Fribourg[1]
1 LSV, ENS de Cachan & CNRS
2 LIP6, Université Paris 6 & CNRS

30 juin 2008

1 Introduction

Ce document correspond à la fourniture D3.2 du projet VALMEM. Il présente différentes pistes de spécialisation du model-checking temporisé pour l'analyse de circuits mémoire et compare leur application pour l'analyse d'une portion de la mémoire SPSMALL.

2 Modèle d'automates temporisé et particularités

Le modèle à analyser est représenté sous la forme d'un ensemble d'automates temporisés [1]. Ces derniers sont des automates composés de localités et de transitions, et pourvus d' *horloges*.

- les horloges sont des variables sur $\mathbb{R}$
- elles apparaissent dans des expressions d'invariants de localité et gardes des transitions.

Un état d'un automate est donné par sa localité (*locations* en anglais) et la valeur courante de chacune de ses horloges. On distingue deux changements d'états possibles :

- changement d'état induit par une action (le franchissement d'une transition de l'automate) : la transition est franchissable si l'expression de sa garde est VRAIE. Pendant le franchissement d'une transition d'action, des ensembles d'horloges peuvent être réinitialisées à 0.
- changement d'état induit par l'écoulement du temps : les valeurs de toutes les horloges progressent avec la même dérivée.

Une exécution d'un automate temporisé est représentée par la suite des états du système obtenue en appliquant ces deux règles de changement d'états.

Les modèles que nous considérons ont en outre les particularités suivantes :

- chaque automate $\mathcal{A}_i$ a une horloge distincte x_i ;
- chaque automate est déterministe *en action et en temps* ;
- chaque localité est d'une des deux formes suivantes :
  - ATTENTE PASSIVE : l'invariant de la localité est trivial (= VRAI) ; de la localité, il peut y avoir plusieurs transitions sortantes de la forme munies d'une garde triviale (= VRAI), ces transitions sont impérativement synchronisées sur étiquette partagée avec d'autres transitions d'autres automates temporisés ;

- ATTENTE ACTIVE : l'invariant de la localité est de la forme $x_i \leq p_i$; de cette localité, il existe une transition sortante de la forme garde $= x_i = p_i$ avec synchronisation sur étiquette partagée, et plusieurs transitions sortantes avec une garde triviale avec synchronisation sur étiquette partagée ;

– chaque transition synchronisée a une unique garde non triviale.

En conséquences, le modèle est *quasi-déterministe* : pour une instanciation ponctuelle des paramètres, seules les transitions concurrentes provenant d'automates différents et franchissables au même instant peuvent provoquer de l'indéterminisme.

Pour une instanciation ponctuelle des paramètres, les sources d'indéterminisme sont très réduites.

Le système est analysé sur des séquences bornées, représentant un nombre déterminé (et borné) de cycles d'horloge : le graphe des états accessibles temporisé est sans boucle.

3 Particularisation du model-checking

Les propriétés à vérifier sont des propriétés de sûreté ou de vivacité, exprimées dans une logique temporisée. Ces propriétés sont analysables en examinant le graphe des états accessibles obtenu en réalisant le produit des automates temporisés. Dans ce graphe, chaque état est composé d'un ensemble de localités, décrivant l'état courant de chaque automate du produit, et d'une région temporelle définissant la portion de temps pendant laquelle le système peut rester dans cet état.

On distingue deux types d'analyse :

– Modèle instancié : déterminer si un temps de réponse est correct.
– Modèle paramétré : synthétiser des contraintes entre les délais garantissant une plage de fonctionnement correct (i.e. un temps de réponse donné).

Dans le cas général, le model-checking d'un produit d'automates temporisés instanciés est décidable, alors que ce n'est pas le cas pour un produit d'automates temporisés paramétrés [2]. Dans notre cas, le graphe des états accessibles étant sans boucle, le model-checking paramétré est décidable.

De plus, en tenant compte des particularités de quasi-déterminisme des modèles, nous proposons de synthétiser des contraintes garantissant des plages de bon fonctionnement en partant soit d'une trace de référence, soit d'un point de bon fonctionnement. On cherche alors à obtenir des contraintes généralisant ces instanciations de référence, et garantissant que toute trace d'exécution d'un système dont les paramètres satisfont ces contraintes correspond à un fonctionnement correct du système.

3.1 Méthode 1 : Analyse à partir d'une trace

On suppose, pour un système $\mathcal{A}$, un état initial s_{init} et une propriété à analyser, la donnée d'une trace de référence T correspondant à une exécution possible pour une instance des paramètres du modèle. On construit la contrainte $\mathcal{C}$ généralisant la trace T tout en garantissant que

la propriété reste vérifiée.

- Déterminer une zone Bad : un ensemble d'états invalidant la propriété ;
- Déterminer l'instant d'occurence de chaque événement de T (parcours arrière des automates) ;
- Synthétiser le système de contraintes $\mathcal{C}$ autorisant T ;
- Vérifier que $Post^*(Init \cap \mathcal{C}) \not\ni Bad$.

La synthèse du système de contraintes $\mathcal{C}$ est réalisée manuellement (soit par examen du graphe des états temporisés paramétrés, cf. section 4.7.1, soit par adjonction de variables, et lancement de plusieurs parcours successifs, cf. section 4.8.3). Dans les deux cas, l'extraction pourrait être automatisée.

3.2 Méthode 2 : Analyse à partir d'un point de fonctionnement

On suppose, pour un système $\mathcal{A}$, un état initial s_{init} et une propriété à analyser, la donnée d'une instanciation de tous les paramètres correspondant à un point de bon focntionnement. On construit la contrainte $\mathcal{K}$ généralisant le point de fonctionnement tout en garantissant que la propriété est vérifiée.

Les variables utilisées par l'algorithme sont les suivantes :

Variable	Type	Description	Initialement
i	Entier	Etape courante	$i := 0$
$\mathcal{K}$	Contrainte	Résultat	$\mathcal{K} := \top$
$\mathcal{S}$	Ensemble d'états	Ensemble des états accessibles	$\mathcal{S} := \{s_{init}\}$
$\mathcal{S}_{prev}$	Ensemble d'états	Ensemble des états accessibles à l'étape précédente	$\mathcal{S}_{prev} := \{s_{init}\}$

L'algorithme est donné ci-après :

DO
  $\mathcal{S} := Post^i_{\mathcal{A}(\mathcal{K})}(s_{init})$
  if $\mathcal{S} = \mathcal{S}_{prev}$
    then return $\mathcal{K}$
  fi
  DO until $\mathcal{S}$ NE contient PAS d'état π_0-incompatible **:**
    Selection d'une disjonction π_0-incompatible dan sun état de § et
      d'un atome J de la P-contrainte associée, tel que $\pi_0 \models \mathcal{K} \wedge \neg J$
    $\mathcal{K} := \mathcal{K} \wedge \neg J$
    $\mathcal{S}_{prev} := \mathcal{S}$
    $\mathcal{S} := Post^i_{\mathcal{A}(\mathcal{K})}(s_{\ init})$
  OD
  $i := i + 1$
OD

Cet algorithme a été implanté dans un script python pilotant l'outil HYTECH. Ce script réalise itérativement :

- Préparation des fichiers à exécuter par HYTECH (modification de la partie "commande",

– Lancement de l'exécution de HYTECH,
– Récupère des états (paramétrés) visités,
– Repérage des contraintes incompatibles avec le point d'instanciation de référence (appel à Prolog), et
– Renforcement de la contrainte courante par adjonction de l anégation de la contrainte incompatible détectée précédemment.
Remarque : La variable $\mathcal{S}_{prev}$ n'est pas représentée dans le script Python ; on garde en mémoire uniquement le nombre de localités (composées)et le nombre de disjonctions de deux étapes successives.

4 Analyse d'une portion de SPSMALL

4.1 Description

La mémoire SPSMALL est un circuit permettant la mémorisation de 3 mots de 2 bits. Sa description en transistors est obtenue à partir d'un générateur de mémoire paramétrable, propriété de STMicroelectronics. Après l'étape d'abstraction fonctionnelle et temporelle (réalisée par les outils YAGLE et TAS pour cette étude préliminaire), le LIP6 fournit deux fichiers :

– `SPSMALL9gp_3x2_nsL.vhd` : description fonctionnelle VHDL de la SPSMALL.
– `tempo.dat` : temps de propagation MAXIMAUX pour les signaux décrits dans `SPSMALL9gp_3x2_nsL.vhd`.

Nous décrivons dans la section suivante la modélisation sous forme d'automates temporisés d'une portion de ce circuit.

4.2 Schéma en portes de D0 à v_18_E_data_delay_int

Le schema logique (après abstraction des chaînes d'inverseurs) représentant l'écriture du signal `D0` dans le latch d'entrée `REG12` est donné sur la figure 1.

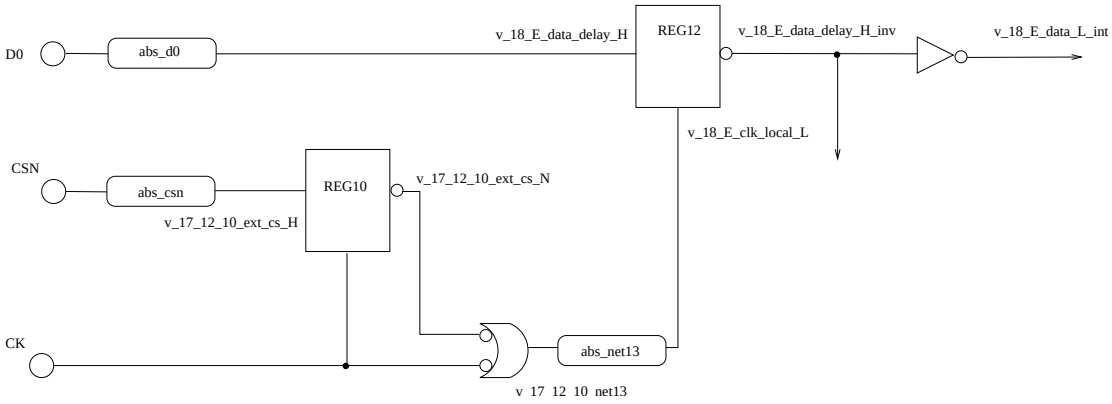

FIG. 1 – Schéma logique de l'écriture de D0 dans le latch d'entrée.

nom du signal	délai falling ($d^{\downarrow}$)	délai rising ($d^{\uparrow}$)
abs_d0	781.1	1035.7
abs_csn	138.2	120.7
abs_net13	182.5	251.3

TAB. 1 – Délais associés aux éléments retard.

4.3 Modèle des portes

4.3.1 Retard (avec délai inertiel)

Cet élément résulte de l'abstraction de chaînes d'inverseurs :
– abs_d0 est l'abstraction de `DO -> v_18_E_net81 -> v_18_E_net85 -> v_18_E_net83 -> v_18_E_data_delay_H`,
– abs_csn est l'abstraction de `CSN -> v_17_12_10_net96 -> v_17_12_10_ext_cs_H`,
– abs_net13 est l'abstraction de `v_17_12_10_net13 -> clk_sig_H -> net41 -> CLK_H ->v_18_E_clk_local_L`.
L'automate temporisé associé est donné sur la figure 2.

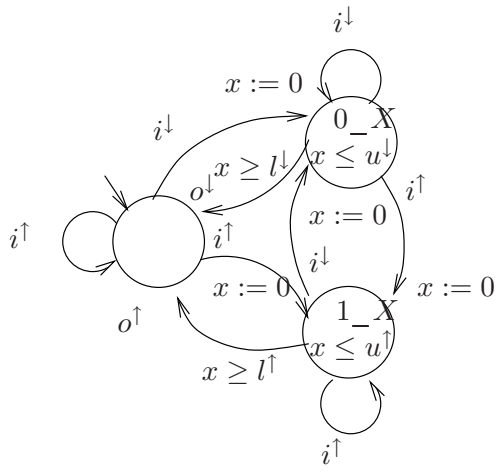

FIG. 2 – Automate temporisé modélisant un retard inertiel compris dans $[l^{\uparrow}, u^{\uparrow}]$ et dans $[l^{\downarrow}, u^{\downarrow}]$.

Les délais associés aux éléments retards (en ps) sont donnés dans la table 1. Ils sont obtenus en réalisant la somme alternée des délais des fronts montants et descendants pour les signaux abstraits de long de la chaine d'inverseurs.

4.3.2 Porte NOT

Cet élément modélise un inverseur isolé : celui pilotant v_18_E_data_L_int par exemple. L'automate temporisé associé a la même structure que celui modélisant un retard inertiel, mais la valeur de sortie est inversée (cf. figure 3).
Les délais associés à la porte NOT (en ps) sont donnés dans la table 2.

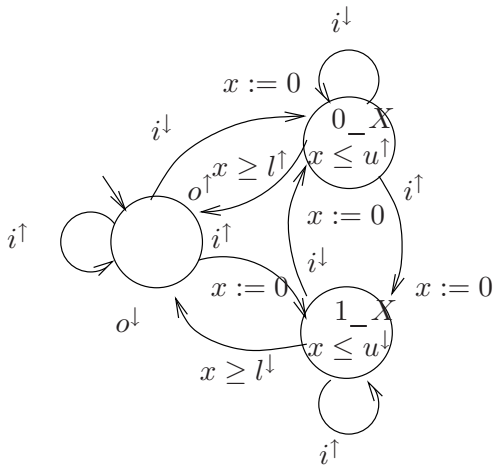

FIG. 3 – Automate temporisé modélisant une porte NOT avec un délai compris dans $[l^{\uparrow}, u^{\uparrow}]$ et dans $[l^{\downarrow}, u^{\downarrow}]$.

nom du signal	délai falling ($d^{\downarrow}$)	délai rising ($d^{\uparrow}$)
v18_E_data_L_int	107.3	114.3

TAB. 2 – Délais associés aux portes NOT.

4.3.3 Bloc combinatoire à 2 entrées

Cet élément modélise un bloc combinatoire composé d'une simple porte ou d'un assemblage (non cyclique) de portes logiques. La porte pilotant v_17_12_10_net13 en est un exemple : c'est un OR avec deux entrées inversées.
L'automate temporisé associé est présenté sur la figure 4.
Les délais associés à la porte OR (en ps) sont donnés dans la table 3.

nom du signal	délai falling ($d^{\downarrow}$)	délai rising ($d^{\uparrow}$)
v17_12_10_net13	[31.2,34.8]	[91.3,101.6]

TAB. 3 – Délais associés à la porte OR.

4.4 Latchs

Les latchs et autres éléments mémorisants sont représentés sous la forme de processus (`process` en VHDL). La mémoire SPSMALL comprend plusieurs modèles syntaxiques différents.
Le modèle décrivant les latchs 10 et 12 est le suivant :

```
process(d,e)
begin
  if e = constante then
    q <= not d
  endif;
end process;
```

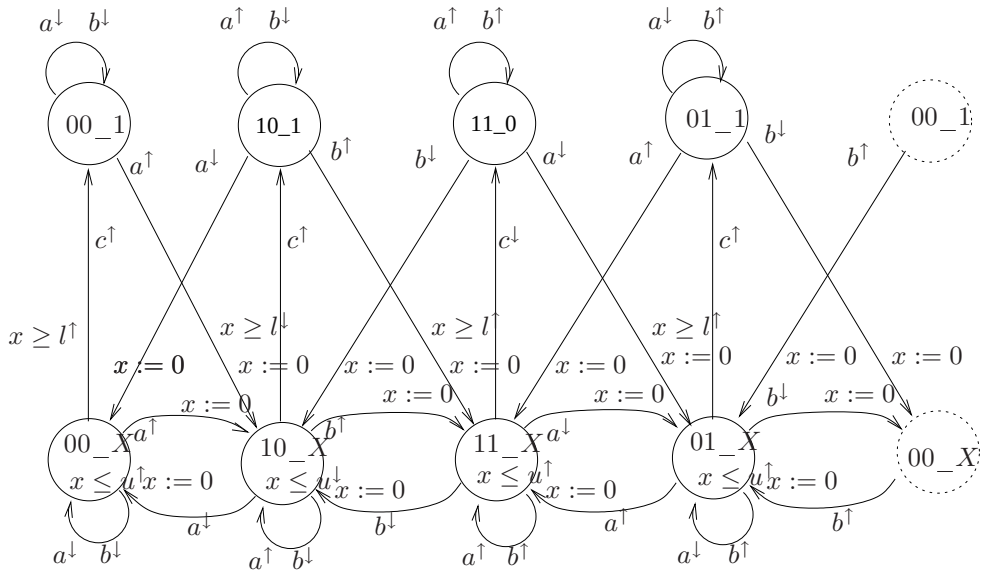

FIG. 4 – Automate temporisé modélisant l'équation c <= (not a) or (not b) avec un délai compris dans $[l^{\uparrow}, u^{\uparrow}]$ et dans $[l^{\downarrow}, u^{\downarrow}]$.

La condition d'écriture porte uniquement sur le signal `e`, qui est comparée à la constante `'0'` (pour REG10) ou `'1'` (pour REG12). Lorsque l'écriture est autorisée, la sortie `q` recopie la valeur inversée de l'entrée `d`.
L'automate temporisé associé est décrit sur la figure 5. Les délais associés aux latchs (en ps) sont donnés dans la table 4.

nom du signal	délai falling ($d^{\downarrow}$)	délai rising ($d^{\uparrow}$)
v17_12_10_ext_cs_N (REG10)	70.4	96.3
v18_E_ext_data_delay_H_inv (REG12)	142.8	176.7

TAB. 4 – Délais associés aux latchs.

4.5 Modélisation de l'environnement

L'environnement produit les variations des signaux CK, D et CSN. Ces variations sont modélisées sur deux cycles, dans un seul automate, suivant le chronogramme de la figure 6.

4.6 Propriétés à vérifier

On s'intéresse à deux propriétés de sûreté décrivant l'écriture du signal `D` dans le latch `REG12`.
P1 : Si D est positionné à 1 et CSN à 0 sur le front montant de CK marquant le premier cycle, alors v_18_E_data_delay_inv vaut 0 au bout de deux cycles.
P2 : Si D est positionné à 0 et CSN à 0 sur le front montant de CK marquant le premier cycle, alors v_18_E_data_delay_inv vaut 1 au bout de deux cycles.

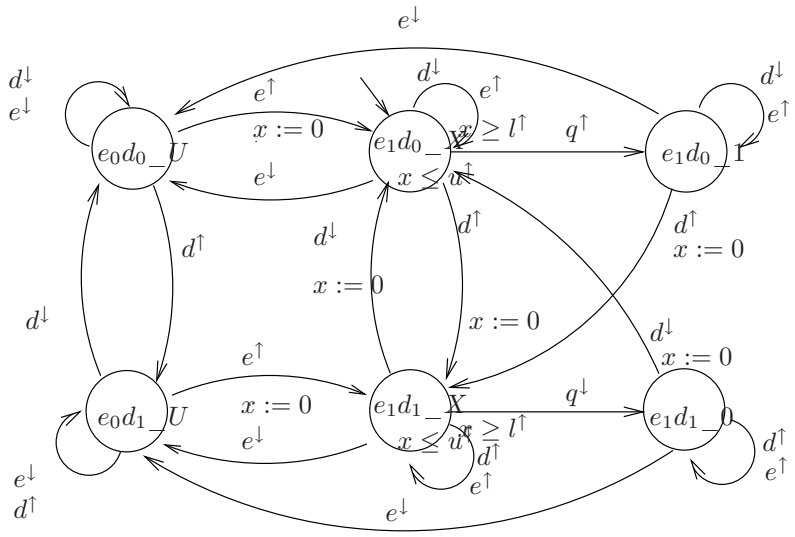

FIG. 5 – Automate temporisé modélisant un latch avec la condition d'écriture `e = '1'` et un délai compris dans $[l^{\uparrow}, u^{\uparrow}]$ et dans $[l^{\downarrow}, u^{\downarrow}]$.

4.7 Vérification

Le programme Hytech correspondant au schéma de la figure 1, pour la vérification de la propriété 1, est donné en annexe 7. Dans ce modèle, tous les délais sont paramétrés ; les délais associés aux fronts montants ou descendants ne sont pas distingués.

4.7.1 Analyse par la Méthode 1 : élargissement des paramètres autour d'une trace.

Sur ce modèle totalement paramétré, le calcul de $Post*(Init)$ ne termine pas. L'instanciation des paramètres à une constante (la valeur maximale de l'intervalle associé à chaque délai) permet d'obtenir l'ensemble des états accessibles (pour cette instanciation des paramètres) très rapidement. Ce graphe des états accessibles est présenté en annexe 7.
Ce modèle instancié présente quatre traces d'exécution, T_1,T_2,T_3 et T_4 décrites en annexe. Pour chaque trace T_i (ou groupe de traces), on construit le polyèdre P_i représentant les contraintes sur les paramètres autorisant cette trace. Le système étant quasi-déterministe (mais pas déterministe), on vérifie *a posteriori* que le modèle paramétré, restreint par le polyèdre, ne peut engendrer de mauvais états (i.e. $Post^*(Init \cap P_i) \cap Bad = \emptyset$). Le détail de l'analyse des quatre traces ainsi que les polyèdres engendrés sont donnés en annexe 7.
L'instanciation des paramètres à des valeurs comprises dans des intervalles (les valeurs données dans les tables 1 à 5 du présent document) a été testée : le calcul de Post*(Init) termine en 12h (sans l'affichage de tous les accessibles), et produit 168 traces. Toutes terminent dans un état final correct.

4.7.2 Analyse par la Méthode 2 : élargissement autour d'un point de fonctionnement

L'algorithme d'extraction des contraintes sur le système complètement paramétré, à partir du point d'instanciation donné en annexe, produit un ensemble de contraintes *EN 3 JOURS....* Il reste à prouver que cet ensemble ne conduit pas dans *Bad*.

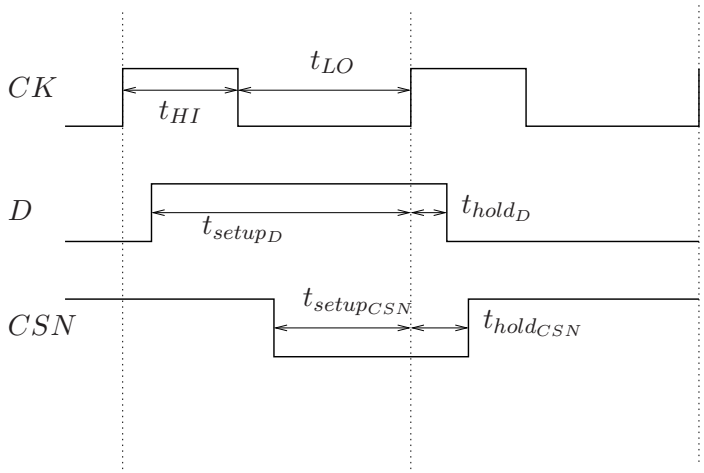

FIG. 6 – Chronogramme décrivant l'évolution des signaux `D`, `CK` et `CSN`.

4.8 Améliorations du modèle

4.8.1 Distinction fronts montants / fronts descendants

L'analyse précédente montre qu'il est plus judicieux de distinguer les délais de propagation des fronts montants et descendants variant dans deux intervalles très serrés (éventuellement ponctuels), plutôt que de les associer en un seul délai, variant dans un large intervalle. En effet, la modélisation bi-ponctuelle est plus conforme à la réalité ; la dispersion des délais de propagation dans une porte logique n'est pas uniforme, mais correspond à deux pics : l'un pour la propagation d'un front descedant et l'autre pour la propagation d'un front montant. De plus, cette modélisation bi-ponctuelle réduit l'indéterminisme temporel du modèle, ce qui réduit également l'explosion combinatoire de l'espace d'états à analyser.

4.8.2 minimisation des calculs de la porte "or"

Dans la modélisation précédente, l'automate associé à la porte "or" est générique : sa structure est identique pour toutes les blocs combinatoires à deux entrées, et son mode d'évaluation est simple : à chaque changement de configuration, la sortie est ré-évaluée, que son résultat diffère de la sortie courante ou non. Seuls les labels de sortie des états d'attente passive et des transitions y menant dépendent de la fonctionalité de la porte analysée. On peut réduire l'évaluation de la sortie aux cas où cette sortie va effectivement changer. La structure de l'automate n'est plus générique car elle dépend de la fonctionalité de la porte, mais elle reste assez simple à déterminer. A partir d'un état d'attente passive, lors d'un changement de configuration des entrées, si la nouvelle configuration n'induit pas de changement de valeur de la sortie, alors la transition mène dans l'état d'attente passive associé à la nouvelle configuration. Si par contre la nouvelle configuration induit un changement de la valeur du signal de sortie, alors la transition mène dans l'état d'attente active associé à cette nouvelle configuration, correspondant au calcul de cette nouvelle sortie. La nouvelle modélisation de l'affectation `c <= (not a) or (not b) ;` est donnée sur la figure 7.

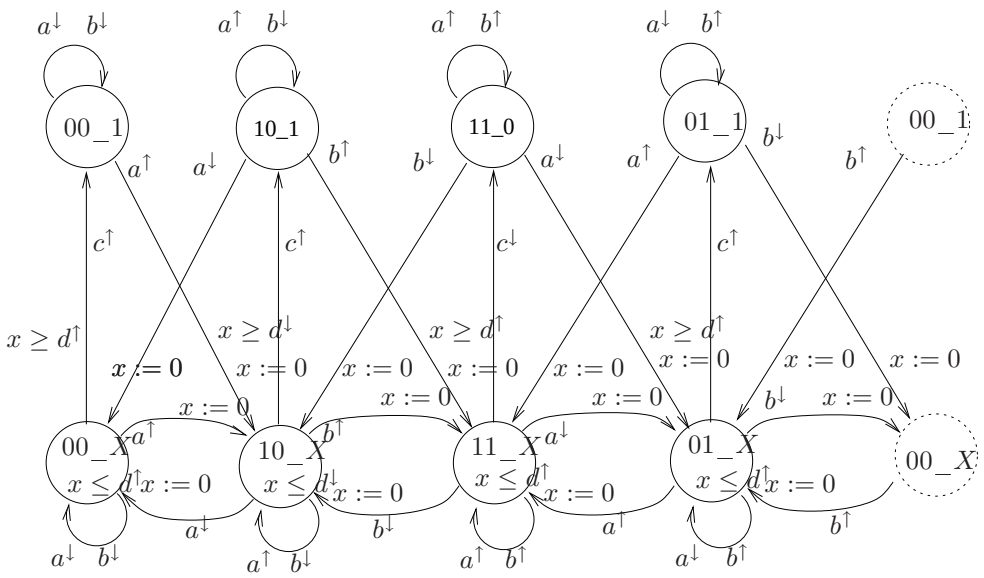

FIG. 7 – Automate temporisé modélisant l'équation c <= (not a) or (no b) avec un délai égal à $d^{\uparrow}$ pour la production d'un front montant et $d^{\downarrow}$ pour la production d'un front descendant. Seuls les changements de configurations induisant un changement de la valeur de sortie induisent une réévaluation de la sortie.

4.8.3 Analyse par les méthodes 1 et 2

Le nouveau programme Hytech, intégrant ces améliorations, est décrit ci-après (cf. optim.hy, donné en annexe). Il est quasi-déterministe pour l'instanciation ponctuelle des paramètres (on obtient 2 traces reconvergeantes, cf. optim.log_onion). Le calcul de post sur le système totalement paramétré ne termine pas (cf. optim_parametre.log : out of memory). On peut instancier tous les délais internes et laisser en paramètres les délais associés au temps de setup et hold des signaux d'entrée `D` et `CSN`. Dans ce cas, on calcule l'ensemble des états accessibles (paramétrés en setup et hold) en une vingtaine de secondes (cf. optim.log). Il est alors aisé d'extraire les régions finales dans lesquelles la propriété est vérifiée : ce sont les régions menant dans un état final où la propriété est vérifiée, et ne menant pas dans un état final défectueux : $S = \{(l,c) \in Post*(Init) \wedge (l,c) \in Final \cap Good \wedge Post*(Init \wedge c) \not\ni Bad\}$ (cf. optim_region_good.log, et optim_good_verif.hy).
De même, l'algorithme élargissant un point de fonctionnement permet de trouver des contraintes garantissant le bon fonctionnement en quelques minutes (lorsque les délais internes sont instanciés). La contrainte obtenue est compatible avec les régions trouvées par la méthode précédente.
Ces régions garantissant le bon fonctionnement sont données sur la figure 8. Sur ce diagramme, on distingue différentes zones :

- la région en rouge a été trouvée par la méthode 2 sur le système paramétré en `setup` et `hold`. Elle garantit le bon fonctionnement du système.
- les régions hachurées ont été obtenues par un simple calcul d'accessibilité (paramétré en `setup` et `hold`) et rendent possible le bon fonctionnement du système (elles peuvent mener dans un état final satisfait la propriété). Toutes ne garantissent pas le bon fonctionnement du système :

– les régions hachurées avec un fond bleu peuvent mener dans un état final invalidant la propriété.
– les autres régions hachurées garantissent garantissent le bon fonctionnement du système. Parmi celles-ci, les régions hachurées avec un fond rouge avec un fond rouge coïncident avec la région obtenue par la méthode 2.

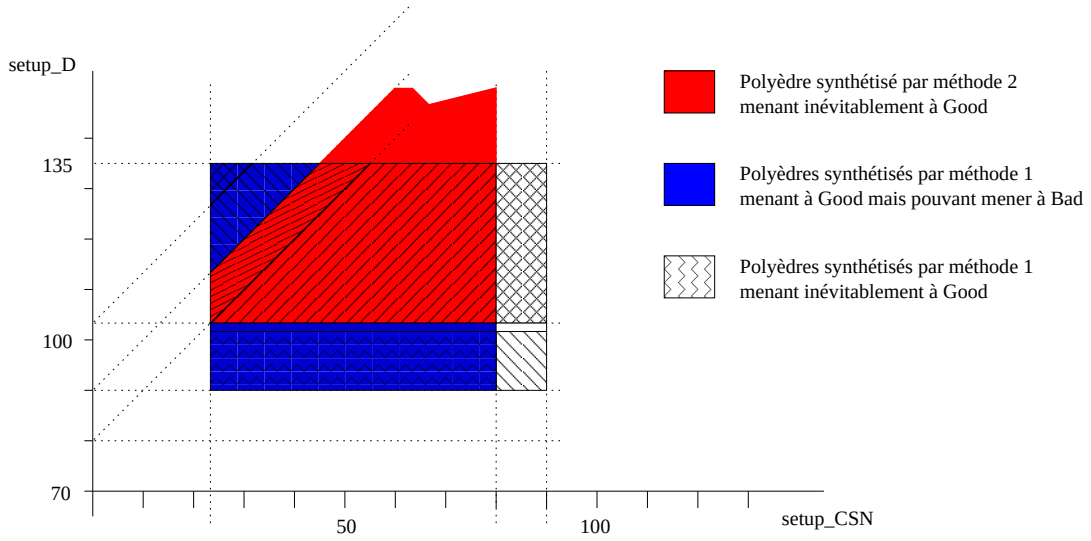

FIG. 8 – Régions temporelles associées aux délais $setup_D$ et $setup_{CSN}$ garantissant que la propriété P1 est satisfaite.

5 Autres Expérimentations

Cet exemple reprend la modélisation de la mémoire SPSMALL pour la détermination de la valeur du temps d'accès en écriture *taaw*. Ce modèle et les expériences y afférant sont décrites dans [3]. La figure 9 illustre le modèle. Chaque porte ou latch est pourvu de deux délais variant dans des intervalles (en général étroits) $[l^{\downarrow}, u^{\downarrow}]$ et $[l^{\uparrow}, u^{\uparrow}]$. Les fonctionalités des portes sont simplifiées : la propagation des seuls fronts pertinents vis-à-vis du scénario testé est modélisée. Le fichier Hytech correspondant à ce modèle est donné en annexe 9.

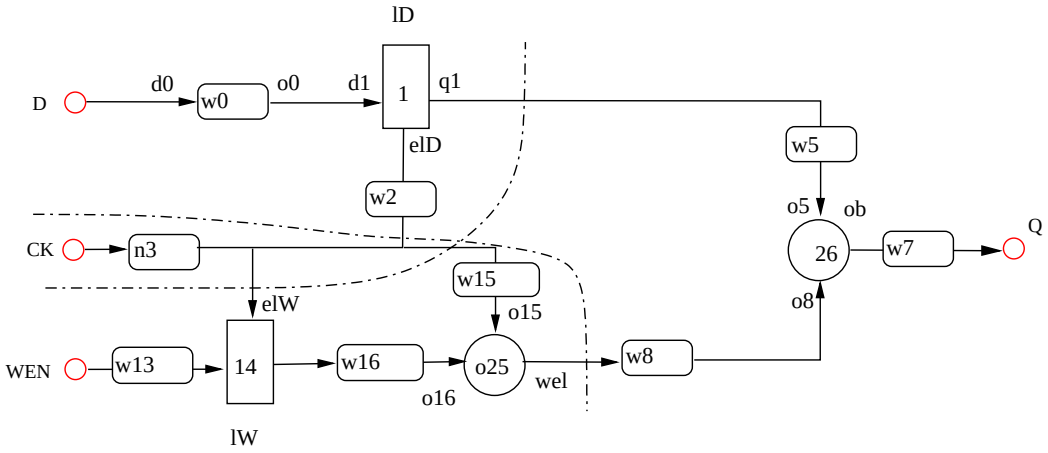

FIG. 9 – Graphe Fonctionnel et Temporel Abstrait d'une portion de SPSMALL, construit suivant la démarche BLUEBERRIES.

Sur ce modèle, les deux approches sont appliquées et comparées.

5.1 Analyse par élargissement autour d'une trace

On suppose donné le point de référence π_0. Les ensembles d'états $Init$ et $Final$ caractérisent les extrémités du scénario que l'on analyse. On note S un état paramétré et S_{π_0} un état pour lequel les paramétres sont instanciés selon π_0.

1. Le modèle est enrichi : on introduit les variables discrètes t_i indiquant la date de franchissement de chaque transition i. L'action de chaque transition i chaque transition i affecte t_i à la date courante.[1]
2. Calcul de $Post^*(Init_{\pi_0})$, détermination de la trace T définissant l'ordre de franchissement des transitions pour cette exécution. Soit $\mathcal{O}$ la contrainte (en t_i) associée à cet ordre.
3. Calcul de $Pre^*(Final \cap \mathcal{O}) \cap Init$. On obtient en $Init$ une contrainte $\mathcal{C}$ (en l,u) rendant la trace T possible.
4. On vérifie que $Post^*(Init \cap \mathcal{C})Bad = \emptyset$.
5. On relâche la contrainte $\mathcal{C}$ (tout en vérifiant que les états Bad restent inaccessibles).

Notons ici que les étapes 3, 4 et 5 sont effectuées sur le modèle complètement paramétré. Les étapes 2, 3 et 4 s'effectuent en quelques minutes. L' étape 5 s'effectue en 1h. La contrainte obtenue est donnée en annexe.

[1] Cette démarche s'applique uniquement si les transitions sont franchissables une seule fois. C'est le cas dans ce modèle. Si ce n'est pas le cas, il faut déplier les automates.

5.2 Analyse par élargissement du point de fonctionnement

L'application directe de l'algorithme d'élargissement à partir de π_0 produit en 10 minutes une contrainte (donnée en annexe). On vérifie que $Post^*(Init \cap \mathcal{C})Bad = \emptyset$ en quelques secondes. Un travail d'élargissement de la contrainte peut également être mené (cf. étape 5 de la méthode précédente).

6 Conclusion

Dans ce rapport, nous avons proposé une démarche de modélisation d'un circuit décrit en VHDL dans le formalisme des automates temporisés. Nous avons montré comment les délais de propagation des signaux dans les portes logiques pouvaient être associés aux automates temporisés et avons adopté le modèle bi-ponctuel pour les délais des portes internes. Nous avons également proposé plusieurs modélisations possibles pour l'évaluation de la valeur de sortie d'une porte : un premier modèle, générique, est simple à mettre en oeuvre mais nécessite l'évaluation de la sortie pour chaque changement de configuration ; le second modèle présente une structure d'automate moins régulière mais évite les évaluations superflues et optimise la construction du graphe des états accessibles du système. C'est ce second modèle que nous avons retenu.
Dans une seconde partie nous avons proposé et évalué deux méthodes différentes pour synthétiser des contraintes sur les délais paramétrés garantissant le bon fonctionnement du modèle. La première nécessite la donnée d'une trace d'exécution de référence, la seconde requiert un point d'instanciation de référence. La démarche consiste à élargir le domaine de variations des paramètres tout en respectant la donnée de réference.
Nous avons expérimenté ces approches sur deux exemples :

- une portion du circuit mémoire SPSMALL fourni par STMicroelectronics, extraite par le LIP6, avec instanciation des délais internes.
- une autre portion du même circuit, extrait manuellement dans le cadre du projet MEDEA+ BLUEBERRIES, avec délais paramétrés.

Les deux méthodes d'extraction des contraintes sont efficaces pour de petites portions de circuits, présentant des délais internes bi-ponctuels et instanciés. La méthode basée sur l'élargissement de traces semble plus prometteuse pour des systèmes bi-bornés (et non bi-ponctuels) avec un grand nombre de paramètres : les temps de calculs obtenus par la méthode 2 pour la portion de SPSMALL avec délais bi-bornés paramétrés sont rédhibitoires pour espérer passer à l'échelle sur ce type de modèles.

Références

[1] Rajeev Alur and David Dill. A theory of timed automata. *Theor. Comp. Sci.*, 126(2) :183–235, 1994.

[2] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric real-time reasoning. In *STOC '93 : Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 592–601, New York, NY, USA, 1993. ACM.

[3] R. Chevallier, E. Encrenaz, L. Fribourg, and W. Xu. Verification of the generic architecture of a memory circuit with parametric timed automata. *FORMATS*, 2006.

7 Annexe : SPSMALL – portion menant de D à REG12 – modèle bi-borné

```
-- spsmall : de D/CK/CSN à REG12
-- D_reg12_T1.hy

var  x_abs_d0, x_abs_csn, x_abs_net13, x_reg_10,
x_reg_12, x_or_net13, x_not_v18_E,
x_CK,  s : clock;

q, qD, qDb,
t0, t1, t2, t3, t4, t7, t5, t13, t14, t12, t8, t9, t10,
       t11, t15, t16, t19, t18, t21, t20, t17, t24, t25,
t29, t26, t22, t23 : discrete;

tHI, tLO, d_setup_D, d_hold_D, d_setup_CSN, d_hold_CSN,
d_abs_d0, d_abs_csn, d_abs_net13, d_reg_10, d_reg_12,
d_or_net13, d_not_v18_E : parameter;




automaton input
synclabs: up_CK, down_CK, up_D, down_D, up_CSN, down_CSN;
initially init_input;

loc init_input: while x_CK <=tHI+tLO - d_setup_D  wait {}
when x_CK=tHI+tLO - d_setup_D sync up_D do {} goto A_input;

loc A_input: while x_CK <= tHI wait {}
when x_CK=tHI sync down_CK do {} goto B_input;

loc B_input: while x_CK <= tHI+tLO - d_setup_CSN wait {}
when x_CK= tHI+tLO - d_setup_CSN sync down_CSN do {} goto C_input;

loc C_input: while x_CK <=tHI+tLO wait {}
when x_CK = tHI+tLO sync up_CK do {} goto D_input;

loc D_input: while x_CK <= tHI+tLO+d_hold_D  wait{}
when x_CK=tHI+tLO+d_hold_D sync down_D do {} goto E_input;

loc E_input: while x_CK <= tHI+tLO+d_hold_CSN  wait{}
```

```
when x_CK=tHI+tLO+d_hold_CSN sync up_CSN do {} goto F_input;

loc F_input: while x_CK <= 2tHI+tLO  wait{}
when x_CK=2tHI+tLO sync down_CK do {} goto G_input;

loc G_input: while x_CK <= 2tHI+2tLO  wait{}
when x_CK=2tHI+2tLO            do {} goto H_input;

loc H_input: while x_CK>=0 wait {}
when True do {} goto H_input;

end -- input


automaton abs_d0
synclabs: down_D, up_D,  -- inputs
  down_v18_E_data_delay_H, up_v18_E_data_delay_H; -- outputs
initially init_abs_d0;

loc init_abs_d0 : while True wait {}
when True sync up_D do {x_abs_d0'=0} goto A_abs_d0;
when True sync down_D do {x_abs_d0'=0} goto B_abs_d0;

loc A_abs_d0 : while x_abs_d0 <= d_abs_d0 wait {}
when True sync down_D do {x_abs_d0'=0} goto B_abs_d0;
when x_abs_d0 = d_abs_d0 sync up_v18_E_data_delay_H do {} goto init_abs_d0;

loc B_abs_d0 : while x_abs_d0 <= d_abs_d0 wait {}
when True sync up_D do {x_abs_d0'=0} goto A_abs_d0;
when x_abs_d0 = d_abs_d0 sync down_v18_E_data_delay_H do {} goto init_abs_d0;

end -- abs_d0

automaton abs_net13
synclabs: down_net13, up_net13,  -- inputs
  down_v18_E_clk_local_L, up_v18_E_clk_local_L; -- outputs
initially init_abs_net13;

loc init_abs_net13 : while True wait {}
when True sync up_net13 do {x_abs_net13'=0} goto A_abs_net13;
when True sync down_net13 do {x_abs_net13'=0} goto B_abs_net13;

loc A_abs_net13 : while x_abs_net13 <= d_abs_net13 wait {}
when True sync down_net13 do {x_abs_net13'=0} goto B_abs_net13;
when True sync up_net13 do {} goto A_abs_net13;
when x_abs_net13 = d_abs_net13 sync up_v18_E_clk_local_L do {} goto init_abs_net13;

loc B_abs_net13 : while x_abs_net13 <= d_abs_net13 wait {}
when True sync up_net13 do {x_abs_net13'=0} goto A_abs_net13;
when True sync down_net13 do {} goto B_abs_net13;
when x_abs_net13 = d_abs_net13 sync down_v18_E_clk_local_L do {} goto init_abs_net13;

end -- abs_net13
```

```
automaton abs_csn
synclabs: down_CSN, up_CSN,  -- inputs
  down_v17_ext_cs_H, up_v17_ext_cs_H; -- outputs
initially init_abs_csn;

loc init_abs_csn : while True wait {}
when True sync up_CSN do {x_abs_csn'=0} goto A_abs_csn;
when True sync down_CSN do {x_abs_csn'=0} goto B_abs_csn;

loc A_abs_csn : while x_abs_csn <= d_abs_csn wait {}
when True sync down_CSN do {x_abs_csn'=0} goto B_abs_csn;
when True sync up_CSN do {} goto A_abs_csn;
when x_abs_csn = d_abs_csn sync up_v17_ext_cs_H do {} goto init_abs_csn;

loc B_abs_csn : while x_abs_csn <= d_abs_csn wait {}
when True sync up_CSN do {x_abs_csn'=0} goto A_abs_csn;
when True sync down_CSN do {} goto B_abs_csn;
when x_abs_csn = d_abs_csn sync down_v17_ext_cs_H do {} goto init_abs_csn;

end -- abs_csn


automaton reg_10
synclabs: up_v17_ext_cs_H, down_v17_ext_cs_H, up_CK, down_CK,  -- inputs  (data + INVERTED enable)
  up_v17_ext_cs_N, down_v17_ext_cs_N ; -- outputs (INVERTED)
initially e0d1_U_reg_10;

loc e0d0_U_reg_10: while True wait {}
when True sync down_CK do {x_reg_10'=0} goto e1d0_X_reg_10;
when True sync up_v17_ext_cs_H do {} goto e0d1_U_reg_10;

loc e1d0_X_reg_10: while x_reg_10 <= d_reg_10 wait {}
when True sync up_CK do {} goto e0d0_U_reg_10;
when True sync up_v17_ext_cs_H do {x_reg_10'=0} goto e1d1_X_reg_10;
when x_reg_10 = d_reg_10 sync up_v17_ext_cs_N goto e1d0_1_reg_10;

loc e1d0_1_reg_10: while True wait {}
when True sync up_CK do {} goto e0d0_U_reg_10;
when True sync up_v17_ext_cs_H do {x_reg_10'=0} goto e1d1_X_reg_10;

loc e0d1_U_reg_10: while True wait {}
when True sync down_CK do {x_reg_10'=0} goto e1d1_X_reg_10;
when True sync down_v17_ext_cs_H do {} goto e0d0_U_reg_10;

loc e1d1_X_reg_10: while  x_reg_10 <= d_reg_10 wait {}
when True sync up_CK do {} goto e0d1_U_reg_10;
when True sync down_v17_ext_cs_H do {x_reg_10'=0} goto e1d0_X_reg_10;
when x_reg_10 = d_reg_10 sync down_v17_ext_cs_N do {q' = 1} goto e1d1_0_reg_10;

loc e1d1_0_reg_10: while True wait {}
when True sync up_CK do {} goto e0d1_U_reg_10;
when True sync down_v17_ext_cs_H do {x_reg_10'=0} goto e1d0_X_reg_10;

end -- reg_10
```

```
automaton reg_12
synclabs: up_v18_E_data_delay_H, down_v18_E_data_delay_H,  -- inputs (data)
  up_v18_E_clk_local_L, down_v18_E_clk_local_L; -- inputs (enable)
initially e0d0_U_reg_12;

loc e0d0_U_reg_12: while True wait {}
when True sync up_v18_E_clk_local_L do {x_reg_12'=0} goto e1d0_X_reg_12;
when True sync up_v18_E_data_delay_H do {} goto e0d1_U_reg_12;

loc e1d0_X_reg_12: while x_reg_12 <= d_reg_12 wait {}
when True sync down_v18_E_clk_local_L do {} goto e0d0_U_reg_12;
when True sync up_v18_E_clk_local_L do {} goto e1d0_X_reg_12;
when True sync up_v18_E_data_delay_H do {x_reg_12'=0} goto e1d1_X_reg_12;
when x_reg_12 = d_reg_12 do {qD'=1} goto e1d0_1_reg_12;

loc e1d0_1_reg_12: while True wait {}
when True sync down_v18_E_clk_local_L do {} goto e0d0_U_reg_12;
when True sync up_v18_E_clk_local_L do {} goto e1d0_1_reg_12;
when True sync up_v18_E_data_delay_H do {x_reg_12'=0} goto e1d1_X_reg_12;

loc e0d1_U_reg_12: while True wait {}
when True sync up_v18_E_clk_local_L do {x_reg_12'=0} goto e1d1_X_reg_12;
when True sync down_v18_E_data_delay_H do {} goto e0d0_U_reg_12;

loc e1d1_X_reg_12: while x_reg_12 <= d_reg_12 wait {}
when True sync down_v18_E_clk_local_L do {} goto e0d1_U_reg_12;
when True sync up_v18_E_clk_local_L do {} goto e1d1_X_reg_12;
when True sync down_v18_E_data_delay_H do {x_reg_12'=0} goto e1d0_X_reg_12;
when x_reg_12 = d_reg_12 do {qDb'=1} goto e1d1_0_reg_12;

loc e1d1_0_reg_12: while True wait {}
when True sync down_v18_E_clk_local_L do {} goto e0d1_U_reg_12;
when True sync up_v18_E_clk_local_L do {} goto e1d1_0_reg_12;
when True sync down_v18_E_data_delay_H do {x_reg_12'=0} goto e1d0_X_reg_12;
 end -- reg_12


automaton or_net13
synclabs: up_v17_ext_cs_N, down_v17_ext_cs_N,  -- INVERTED inputs
  up_CK, down_CK,-- INVERTED inputs
  up_net13, down_net13; -- outputs

-- codage des états : 1re entree (v_17_ext_cs_N), 2e entree (CK) _ sortie net13 (0,1,X)
initially e_01_1_net13;

loc e_00_1_net13: while True wait{}
when True sync up_v17_ext_cs_N  do {x_or_net13'=0} goto e_10_X_net13;
when True sync up_CK  do {x_or_net13'=0} goto e_01_X_net13;
when True sync down_v17_ext_cs_N  do {} goto e_00_1_net13;
when True sync down_CK  do {} goto e_00_1_net13;

loc e_01_1_net13: while True wait{}
when True sync up_v17_ext_cs_N  do {x_or_net13'=0} goto e_11_X_net13;
```

```
when True sync down_CK  do {x_or_net13'=0} goto e_00_X_net13;
when True sync down_v17_ext_cs_N  do {} goto e_01_1_net13;
when True sync up_CK  do {} goto e_01_1_net13;

loc e_10_1_net13: while True wait{}
when True sync down_v17_ext_cs_N  do {x_or_net13'=0} goto e_00_X_net13;
when True sync up_CK  do {x_or_net13'=0} goto e_11_X_net13;
when True sync up_v17_ext_cs_N  do {} goto e_10_1_net13;
when True sync down_CK  do {} goto e_10_1_net13;

loc e_11_0_net13: while True wait{}
when True sync down_v17_ext_cs_N  do {x_or_net13'=0} goto e_01_X_net13;
when True sync down_CK  do {x_or_net13'=0} goto e_10_X_net13;
when True sync up_v17_ext_cs_N  do {} goto e_11_0_net13;
when True sync up_CK  do {} goto e_11_0_net13;

loc e_00_X_net13: while x_or_net13 <= d_or_net13 wait{}
when True sync up_v17_ext_cs_N  do {x_or_net13'=0} goto e_10_X_net13;
when True sync up_CK  do {x_or_net13'=0} goto e_01_X_net13;
when True sync down_v17_ext_cs_N  do {} goto e_00_X_net13;
when True sync down_CK  do {} goto e_00_X_net13;
when x_or_net13 = d_or_net13 sync up_net13  do {}  goto e_00_1_net13;

loc e_01_X_net13: while x_or_net13 <= d_or_net13 wait{}
when True sync up_v17_ext_cs_N  do {x_or_net13'=0} goto e_11_X_net13;
when True sync down_CK  do {x_or_net13'=0} goto e_00_X_net13;
when True sync down_v17_ext_cs_N  do {} goto e_01_X_net13;
when True sync up_CK  do {} goto e_01_X_net13;
when x_or_net13 = d_or_net13 sync up_net13  do {} goto e_01_1_net13;

loc e_10_X_net13: while x_or_net13 <= d_or_net13 wait{}
when True sync down_v17_ext_cs_N  do {x_or_net13'=0} goto e_00_X_net13;
when True sync up_CK  do {x_or_net13'=0} goto e_11_X_net13;
when True sync up_v17_ext_cs_N  do {} goto e_10_X_net13;
when True sync down_CK  do {} goto e_10_X_net13;
when x_or_net13 = d_or_net13 sync up_net13  do {} goto e_10_1_net13;

loc e_11_X_net13: while x_or_net13 <= d_or_net13 wait{}
when True sync down_v17_ext_cs_N  do {x_or_net13'=0} goto e_01_X_net13;
when True sync down_CK  do {x_or_net13'=0} goto e_10_X_net13;
when True sync up_v17_ext_cs_N  do {} goto e_11_X_net13;
when True sync up_CK  do {} goto e_11_X_net13;
when x_or_net13 = d_or_net13 sync down_net13  do {} goto e_11_0_net13;

end -- or_net13

-- analysis commands

var init_reg, post_reg : region;

------------------------------------------------------------------------
-- calcul des états accessibles pour le modèle instancié
------------------------------------------------------------------------
init_reg :=
```

```
  loc[input]=init_input
& loc[abs_d0]=init_abs_d0
& loc[abs_csn]=init_abs_csn
& loc[abs_net13]=init_abs_net13
& loc[reg_10]=e0d1_U_reg_10
& loc[reg_12]=e0d0_U_reg_12
& loc[or_net13]=e_01_1_net13

& x_abs_d0 = 0 & x_abs_csn = 0 & x_abs_net13 = 0 & x_reg_10 = 0
& x_reg_12 = 0 & x_or_net13 = 0 & x_not_v18_E = 0 & x_CK = 0 &   s = 0

& q = 0 & qD=2 & qDb = 0


& 0 < tHI & 0 < tLO & 0 < d_setup_D & 0 < d_hold_D & 0 < d_setup_CSN
& 0 < d_hold_CSN & 0 < d_abs_d0 & 0 < d_abs_csn & 0 < d_abs_net13
& 0 < d_reg_10 & 0 < d_reg_12 & 0 < d_or_net13 & 0 < d_not_v18_E

-- instanciation bi-bornée

& tHI = 45
& tLO = 90
& d_setup_D = 130
& d_hold_D = 1
& d_hold_CSN =2
& d_setup_CSN =50
& 78 < d_abs_d0 & d_abs_d0 < 104
& 12 < d_abs_csn & d_abs_csn < 14
& 18 < d_abs_net13 & d_abs_net13 < 25
& 7 < d_reg_10 & d_reg_10 < 10
& 14 < d_reg_12 & d_reg_12 < 18
& 3 < d_or_net13 & d_or_net13 < 10
& 10 < d_not_v18_E & d_not_v18_E < 12

-- instanciation ponctuelle

--& tHI = 45
--& tLO = 90
--& d_setup_D = 130
--& d_hold_D = 1
--& d_setup_CSN = 50
--& d_hold_CSN = 2
--& d_abs_d0 = 103
--& d_abs_csn = 14
--& d_abs_net13 = 25
--& d_reg_10 = 10
--& d_reg_12 = 18
--& d_or_net13 = 10
--& d_not_v18_E = 12
;

prints "hi there: initial region";
print init_reg;
prints "";
```

```
prints "iterated succ";
post_reg := reach forward from init_reg endreach;

-- affichage des états accessibles et des états finaux invalidant la propriété

--prints "REACHED";
--print (hide
--  x_abs_csn, x_abs_net13, x_reg_10,
--x_abs_d0, x_reg_12,
--x_or_net13, x_not_v18_E,
--x_CK,
---- s,
--tHI, tLO, d_setup_D, d_hold_D, d_setup_CSN, d_hold_CSN,
--d_abs_d0,
--d_abs_csn, d_abs_net13, d_reg_10, d_reg_12,
--d_or_net13, d_not_v18_E in post_reg endhide);

--loc_final := loc[input]=H_input;
--final_reg := post_reg &  loc_final;

--bad_reg := final_reg & q=0;

--prints "BAD :";
--print (bad_reg);
```
Le graphe des états accessibles (pour le modèle instancié) est présenté sur la figure 10.

7.1 Trace T1

Trace T_1 : 3-4-7 et 24-25

Location	temps	input	abs_d0	abs_net13	abs_csn	reg_10	reg_12	or_net13	transition sortante
1	$[0, 5]$	init	init	init	init	e0d1_U	e0d0_U	e_01_1	up_D
2	$[5, 45]$	A	A	init	init	e0d1_U	e0d0_U	e_01_1	down_CK
3	$[45, 55]$	B	A	init	init	e1d1_X	e0d0_U	e_00_X	down_v17_ext_cs_N
4	55	B	A	init	init	e1d1_0	e0d0_U	e_00_X	up_net13
7	$[55, 80]$	B	A	A	init	e1d1_0	e0d0_U	e_00_1	up_v18_E_clk_local_L
5	$[80, 85]$	B	A	init	init	e1d1_0	e1d0_X	e_00_1	down_CSN
13	$[85, 98]$	C	A	init	B	e1d1_0	e1d0_X	e_00_1	up_v18_data_delay_H_inv
14	$[98, 99]$	C	A	init	B	e1d1_0	e1d0_1	e_00_1	up_v17_ext_cs_H
12	$[99, 108]$	C	A	init	init	e1d0_X	e1d0_1	e_00_1	up_v18_data_delay_H
8	$[108, 109]$	C	init	init	init	e1d0_X	e1d1_X	e_00_1	up_v17_ext_cs_N
9	$[109, 119]$	C	init	init	init	e1d0_1	e1d1_X	e_10_X	up_net13
10	$[119, 126]$	C	init	A	init	e1d0_1	e1d1_X	e_10_1	down_v18_data_delay_H_inv
11	$[126, 135]$	C	init	A	init	e1d0_1	e1d1_0	e_10_1	up_CK
15	$[135, 136]$	D	init	A	init	e0d0_U	e1d1_0	e_11_X	down_D
16	$[136, 137]$	E	B	A	init	e0d0_U	e1d1_0	e_11_X	up_CSN
19	$[137, 144]$	F	B	A	A	e0d0_U	e1d1_0	e_11_X	up_v18_E_clk_local_L
18	$[144, 145]$	F	B	init	A	e0d0_U	e1d1_0	e_11_X	down_net13
21	$[145, 151]$	F	B	B	A	e0d0_U	e1d1_0	e_11_0	up_v17_ext_cs_H
20	$[151, 170]$	F	B	B	init	e0d1_U	e1d1_0	e_11_0	down_v18_E_clk_local_L
17	$[170, 180]$	F	B	init	init	e0d1_U	e0d1_U	e_11_0	down_CK
24	$[180, 190]$	G	B	init	init	e1d1_X	e0d1_U	e_10_X	down_v17_ext_cs_N
25	$[190, 200]$	G	B	init	init	e1d1_0	e0d1_U	e_00_X	up_net13
29	$[200, 225]$	G	B	A	init	e1d1_0	e0d1_U	e_00_1	up_v18_E_clk_local_L
26	$[225, 239]$	G	B	init	init	e1d1_0	e1d1_X	e_00_1	down_v18_E_data_delay_H
22	$[239, 257]$	G	init	init	init	e1d1_0	e1d0_X	e_00_1	up_v18_E_data_delay_H_inv
23	$[257, 270]$	G	init	init	init	e1d1_0	e1d0_1	e_00_1	end
31	$[270, \infty[$	H	init	init	init	e1d1_0	e1d0_1	e_00_1	

7.1.1 graphe d'événements associé à T_1

La figure 11 présente une portion du graphe d'événements associé à la trace T_1. Il s'agit du préfixe pour les événements datés de t_1 à t_7.

7.1.2 Système de contraintes associé à T_1

1. $t_1 = t_0 + tHI + tLO - d_{setup_D}$
2. $t_2 = t_0 + tHI$
3. $t_3 = t_2 + d\_reg\_10$
4. $t_4 = t_2 + d\_or\_net13$
5. $t_7 = t_4 + d\_abs\_net13$
6. $t_5 = t_0 + tHI + tLO - d_{setup_{CSN}}$
7. $t_{13} = t_7 + d\_reg\_12$
8. $t_{14} = t_5 + d\_abs\_CSN$
9. $t_{12} = t_1 + d\_abs\_d0$
10. $t_8 = t_{14} + d\_abs\_reg\_10$
11. $t_9 = t_8 + d\_or\_net13$
12. $t_{10} = t_{12} + d\_reg\_12$
13. $t_{11} = t0 + tHI + TLO$
14. $t_{15} = t0 + THI + TLO + d_{hold_D}$
15. $t_{16} = t_0 + t_HI + tLO + d_{hold_{CSN}}$
16. $t_{19} = t_9 + d\_abs\_net13$
17. $t_{18} = t_{11} + d\_or\_net13$
18. $t_{21} = t_{16} + d\_abs\_csn$
19. $t_{20} = t_{18} + d\_abs\_net13$
20. $t_{17} = t_0 + 2\ tHI + tLO$
21. $t_{24} = t_{17} + d\_reg\_10$
22. $t_{25} = t_{24} + d\_or\_net13$
23. $t_{29} = t_{25} + d\_abs\_net13$
24. $t_{26} = t_{15} + d\_abs\_d0$
25. $t_{22} = t_{26} + d\_reg\_12$
26. $t_{23} = t_0 + 2\ tHI + 2\ tLO$

7.1.3 Polyêdre associé à T_1

Trace T_1 : $t_1 < t_2 < t_3 < t_4 < t_7 < t_5 < t_{13} < t_{14} < t_{12} < t_8 < t_9 < t_{10} < t_{11} < t_{15} < t_{16} < t_{19} < t_{18} < t_{21} < t_{20} < t_{17} < t_{24} < t_{25} < t_{29} < t_{26} < t_{22} < t_{23}$
polyêdre P_1 :

$d_{setup_D} + d\_abs\_csn + d\_reg\_10 + d\_or\_net13 < d_{setup_{CSN}} + d\_abs\_d0 + d\_reg\_12$
$\wedge\ \ d_{setup_CSN} + d_{hold_{CSN}} < d\_abs\_csn + d\_abs\_net13 + d\_reg\_10 + d\_or\_net13$
$\wedge\ \ d_{hold_D} < d_{hold_CSN}$

$\wedge \;\; d\_abs\_d0 + d\_reg\_12 < d_{setup_D}$
$\wedge \;\; d\_abs\_csn + d\_abs\_net13 + d\_reg\_10 < d_{setup_{CSN}}$
$\wedge \;\; d\_abs\_net13 + d\_or\_net13 < tHI$
$\wedge \;\; tLO < d_{setup_D}$
$\wedge \;\; d_{hold_{CSN}} + d\_abs\_csn < d\_abs\_net13 + d\_or\_net13$
$\wedge \;\; d\_reg\_10 < d\_or\_net13$
$\wedge \;\; tLO < d_{setup_{CSN}} + d\_abs\_net13 + d\_reg\_12 + d\_or\_net13$
$\wedge \;\; d_setup_D + d\_abs\_csn < d_{setup_{CSN}} + d\_abs\_d0$
$\wedge \;\; d_{setup_{CSN}} + d\_abs\_d0 < d_setup_D + d\_abs\_csn + d\_reg\_10$
$\wedge \;\; d\_or\_net13 < d_{hold_{CSN}} + d\_abs\_csn$
$\wedge \;\; d_{setup_{CSN}} + d\_abs\_net13 + d\_reg\_12 + d\_or\_net13 < tLO + d\_abs\_csn$
$\wedge \;\; d_{setup_{CSN}} + d\_abs\_net13 + d\_or\_net13 < tLO$
$\wedge \;\; tHI + d\_abs\_net13 + d\_reg\_10 + d\_or\_net13 < d_{hold_D} + d\_abs\_d0$
$\wedge \;\; d_{hold_D} + d\_abs\_d0 + d\_reg\_12 < tHI + tLO$

7.1.4 Vérification sur le modèle paramétré contraint par le polyèdre P_1

Le calcul de $Post * (Init \cap P_1)$ s'effectue en 1mn (cf. D-reg12-T1.log). Il engendre quatre traces. On a bien $Post * (Init \cap P_1) \cap Bad = \emptyset$.

7.2 Trace T_2

Trace T_2 : 3-6-7 et 24-25

7.2.1 Système de contraintes associé à T_2

1. $t_1 = t_0 + tHI + tLO - d_{setup_D}$

Location	temps	input	abs_d0	abs_net13	abs_csn	reg_10	reg_12	or_net13	transition sortante
1	$[0, 5]$	init	init	init	init	e0d1_U	e0d0_U	e_01_1	up_D
2	$[5, 45]$	A	A	init	init	e0d1_U	e0d0_U	e_01_1	down_CK
3	$[45, 55]$	B	A	init	init	e1d1_X	e0d0_U	e_00_X	up$_net13$
6	55	B	A	A	init	e1d1_X	e0d0_U	e_00_1	down_v17_ext_cs_N
7	$[55, 80]$	B	A	A	init	e1d1_0	e0d0_U	e_00_1	up_v18_E_clk_local_L
5	$[80, 85]$	B	A	init	init	e1d1_0	e1d0_X	e_00_1	down_CSN
13	$[85, 98]$	C	A	init	B	e1d1_0	e1d0_X	e_00_1	up_v18_data_delay_H_inv
14	$[98, 99]$	C	A	init	B	e1d1_0	e1d0_1	e_00_1	up_v17_ext_cs_H
12	$[99, 108]$	C	A	init	init	e1d0_X	e1d0_1	e_00_1	up_v18_data_delay_H
8	$[108, 109]$	C	init	init	init	e1d0_X	e1d1_X	e_00_1	up_v17_ext_cs_N
9	$[109, 119]$	C	init	init	init	e1d0_1	e1d1_X	e_10_X	up_net13
10	$[119, 126]$	C	init	A	init	e1d0_1	e1d1_X	e_10_1	down_v18_data_delay_H_inv
11	$[126, 135]$	C	init	A	init	e1d0_1	e1d1_0	e_10_1	up_CK
15	$[135, 136]$	D	init	A	init	e0d0_U	e1d1_0	e_11_X	down_D
16	$[136, 137]$	E	B	A	init	e0d0_U	e1d1_0	e_11_X	up_CSN
19	$[137, 144]$	F	B	A	A	e0d0_U	e1d1_0	e_11_X	up_v18_E_clk_local_L
18	$[144, 145]$	F	B	init	A	e0d0_U	e1d1_0	e_11_X	down_net13
21	$[145, 151]$	F	B	B	A	e0d0_U	e1d1_0	e_11_0	up_v17_ext_cs_H
20	$[151, 170]$	F	B	B	init	e0d1_U	e1d1_0	e_11_0	down_v18_E_clk_local_L
17	$[170, 180]$	F	B	init	init	e0d1_U	e0d1_U	e_11_0	down_CK
24	$[180, 190]$	G	B	init	init	e1d1_X	e0d1_U	e_10_X	down_v17_ext_cs_N
25	$[190, 200]$	G	B	init	init	e1d1_0	e0d1_U	e_00_X	up_net13
29	$[200, 225]$	G	B	A	init	e1d1_0	e0d1_U	e_00_1	up_v18_E_clk_local_L
26	$[225, 239]$	G	B	init	init	e1d1_0	e1d1_X	e_00_1	down_v18_E_data_delay_H
22	$[239, 257]$	G	init	init	init	e1d1_0	e1d0_X	e_00_1	up_v18_E_data_delay_H_inv
23	$[257, 270]$	G	init	init	init	e1d1_0	e1d0_1	e_00_1	end
31	$[270, \infty[$	H	init	init	init	e1d1_0	e1d0_1	e_00_1	

2. $t_2 = t_0 + tHI$
3. $t_3 = t_2 + d\_or\_net13$
4. $t_6 = t_2 + d\_reg\_10$
5. $t_7 = t_6 + d\_abs\_net13$
6. $t_5 = t_0 + tHI + tLO - d_{setup_{CSN}}$
7. $t_{13} = t_7 + d\_reg\_12$
8. $t_{14} = t_5 + d\_abs\_CSN$
9. $t_{12} = t_1 + d\_abs\_d0$
10. $t_8 = t_{14} + d\_abs\_reg\_10$
11. $t_9 = t_8 + d\_or\_net13$
12. $t_{10} = t_{12} + d\_reg\_12$
13. $t_{11} = t0 + tHI + TLO$
14. $t_{15} = t0 + THI + TLO + d_{hold_D}$
15. $t_{16} = t_0 + t_HI + tLO + d_{hold_{CSN}}$
16. $t_{19} = t_9 + d\_abs\_net13$
17. $t_{18} = t_{11} + d\_or\_net13$
18. $t_{21} = t_{16} + d\_abs\_csn$
19. $t_{20} = t_{18} + d\_abs\_net13$
20. $t_{17} = t_0 + 2\ tHI + tLO$
21. $t_{24} = t_{17} + d\_reg\_10$
22. $t_{25} = t_{24} + d\_or\_net13$
23. $t_{29} = t_{25} + d\_abs\_net13$
24. $t_{26} = t_{15} + d\_abs\_d0$
25. $t_{22} = t_{26} + d\_reg\_12$
26. $t_{23} = t_0 + 2\ tHI + 2\ tLO$

7.2.2 Polyêdre associé à T_2

Trace T_1 : $t_1 < t_2 < t_3 < t_6 < t_7 < t_5 < t_{13} < t_{14} < t_{12} < t_8 < t_9 < t_{10} < t_{11} < t_{15} < t_{16} < t_{19} < t_{18} < t_{21} < t_{20} < t_{17} < t_{24} < t_{25} < t_{29} < t_{26} < t_{22} < t_{23}$
Polyêdre P_2 :

$d_{setup_D} + d\_abs\_csn + d\_reg\_10 + d\_or\_net13 < d_{setup_{CSN}} + d\_abs_d0 + d\_reg\_12$
$\wedge\ \ 0 < d_{hold_D}$
$\wedge\ \ d_{hold_D} < d_{hold_{CSN}}$
$\wedge\ \ d_{setup_{CSN}} + d_{hold_{CSN}} < d\_abs\_csn + d\_abs\_net13 + d\_reg\_10 + d\_or\_net13$
$\wedge\ \ d\_abs\_net13 + d\_or\_net13 < tHI$
$\wedge\ \ d\_abs_d0 + d\_reg\_12 < d_{setup_D}$
$\wedge\ \ tHI + d\_abs\_net13 + d\_reg\_10 + d\_or\_net13 < d_{hold_D} + d\_abs_d0$
$\wedge\ \ tLO < d_{setup_D}$
$\wedge\ \ d\_abs\_csn + d\_abs\_net13 + d\_reg\_10 < d_{setup_{CSN}}$
$\wedge\ \ d_{setup_D} + d\_abs\_csn < d_{setup_{CSN}} + d\_abs_d0$

$\wedge \;\; d\_or\_net13 < d\_reg\_10$
$\wedge \;\; d_{hold_{CSN}} + d\_abs\_csn < d\_abs\_net13 + d\_or\_net13$
$\wedge \;\; d\_or\_net13 < d_{hold_{CSN}} + d\_abs\_csn$
$\wedge \;\; d_{hold_D} + d\_abs_d0 + d\_reg\_12 < tHI + tLO$
$\wedge \;\; tLO < d_{setup_{CSN}} + d\_abs\_net13 + d\_reg\_10 + d\_reg\_12$
$\wedge \;\; d_{setup_{CSN}} + d\_abs_d0 < d_{setup_D} + d\_abs\_csn + d\_reg\_10$
$\wedge \;\; d_{setup_{CSN}} + d\_abs\_net13 + d\_reg\_10 + d\_reg\_12 < tLO + d\_abs\_csn$
$\wedge \;\; d_{setup_{CSN}} + d\_abs\_net13 + d\_reg\_10 < tLO$

7.2.3 Vérification sur le modèle paramétré contraint par le polyèdre P_2

Le calcul de $Post * (Init \cap P_2)$ ne termine pas (exceeded number of rays in cherni a la 25e iteration) (cf. D-reg12-T2.log). Une instanciation partielle permet d'aboutir (les délais des composants internes sont initialisés à leur valeur max, les temps de cycle, de setup et hold sont paramétrés) ; tous les états finaux sont corrects (cf. D-reg-12-T2-semi-inst.log1 et .log2).

7.3 Trace T3

Trace T_3 : 3-4-7 et 24-28

Location	temps	input	abs_d0	abs_net13	abs_csn	reg_10	reg_12	or_net13	transition sortante
1	$[0, 5]$	init	init	init	init	e0d1_U	e0d0_U	e_01_1	up_D
2	$[5, 45]$	A	A	init	init	e0d1_U	e0d0_U	e_01_1	down_CK
3	$[45, 55]$	B	A	init	init	e1d1_X	e0d0_U	e_00_X	down_v17_ext_cs_N
4	55	B	A	init	init	e1d1_0	e0d0_U	e_00_X	up_net13
7	$[55, 80]$	B	A	A	init	e1d1_0	e0d0_U	e_00_1	up_v18_E_clk_local_L
5	$[80, 85]$	B	A	init	init	e1d1_0	e1d0_X	e_00_1	down_CSN
13	$[85, 98]$	C	A	init	B	e1d1_0	e1d0_X	e_00_1	up_v18_data_delay_H_inv
14	$[98, 99]$	C	A	init	B	e1d1_0	e1d0_1	e_00_1	up_v17_ext_cs_H
12	$[99, 108]$	C	A	init	init	e1d0_X	e1d0_1	e_00_1	up_v18_data_delay_H
8	$[108, 109]$	C	init	init	init	e1d0_X	e1d1_X	e_00_1	up_v17_ext_cs_N
9	$[109, 119]$	C	init	init	init	e1d0_1	e1d1_X	e_10_X	up_net13
10	$[119, 126]$	C	init	A	init	e1d0_1	e1d1_X	e_10_1	down_v18_data_delay_H_inv
11	$[126, 135]$	C	init	A	init	e1d0_1	e1d1_0	e_10_1	up_CK
15	$[135, 136]$	D	init	A	init	e0d0_U	e1d1_0	e_11_X	down_D
16	$[136, 137]$	E	B	A	init	e0d0_U	e1d1_0	e_11_X	up_CSN
19	$[137, 144]$	F	B	A	A	e0d0_U	e1d1_0	e_11_X	up_v18_E_clk_local_L
18	$[144, 145]$	F	B	init	A	e0d0_U	e1d1_0	e_11_X	down_net13
21	$[145, 151]$	F	B	B	A	e0d0_U	e1d1_0	e_11_0	up_v17_ext_cs_H
20	$[151, 170]$	F	B	B	init	e0d1_U	e1d1_0	e_11_0	down_v18_E_clk_local_L
17	$[170, 180]$	F	B	init	init	e0d1_U	e0d1_U	e_11_0	down_CK
24	$[180, 190]$	G	B	init	init	e1d1_X	e0d1_U	e_10_X	up_net13
28	190	G	B	A	init	e1d1_X	e0d1_U	e_10_1	down_v17_ext_cs_N
30	$[190, 200]$	G	B	A	init	e1d1_0	e0d1_U	e_00_X	up_net13
29	$[200, 215]$	G	B	A	init	e1d1_0	e0d1_U	e_00_1	up_v18_E_clk_local_L
26	$[215, 233]$	G	B	init	init	e1d1_0	e1d1_X	e_00_1	down_v18_E_data_delay_H_inv
27	$[233, 239]$	G	B	init	init	e1d1_0	e1d1_0	e_00_1	down_v18_E_data_delay_H
22	$[239, 257]$	G	init	init	init	e1d1_0	e1d0_X	e_00_1	up_v18_E_data_delay_H_inv
23	$[257, 270]$	G	init	init	init	e1d1_0	e1d0_1	e_00_1	end
31	$[270, \infty[$	H	init	init	init	e1d1_0	e1d0_1	e_00_1	

7.3.1 Système de contraintes associé à T_3

1. $t_1 = t_0 + tHI + tLO - d_{setup_D}$
2. $t_2 = t_0 + tHI$
3. $t_3 = t_2 + d\_reg\_10$
4. $t_4 = t_2 + d\_or\_net13$
5. $t_7 = t_4 + d\_abs\_net13$
6. $t_5 = t_0 + tHI + tLO - d_{setup_{CSN}}$
7. $t_{13} = t_7 + d\_reg\_12$
8. $t_{14} = t_5 + d\_abs\_csn$
9. $t_{12} = t_1 + d\_abs\_d0$
10. $t_8 = t_{14} + d\_abs\_reg\_10$
11. $t_9 = t_8 + d\_or\_net13$
12. $t_{10} = t_{12} + d\_reg\_12$
13. $t_{11} = t0 + tHI + tLO$
14. $t_{15} = t0 + tHI + tLO + d_{hold_D}$
15. $t_{16} = t_0 + t_HI + tLO + d_{hold_{CSN}}$
16. $t_{19} = t_9 + d\_abs\_net13$
17. $t_{18} = t_{11} + d\_or\_net13$
18. $t_{21} = t_{16} + d\_abs\_csn$
19. $t_{20} = t_{18} + d\_abs\_net13$
20. $t_{17} = t_0 + 2\ tHI + tLO$
21. $t_{24} = t_{17} + d\_or\_net13$
22. $t_{28} = t_{17} + d\_reg\_10$
23. $t_{30} = t_{28} + d\_or\_net13$
24. $t_{29} = t_{30} + d\_abs\_net13$
25. $t_{26} = t_{29} + d\_reg\_12$
26. $t_{27} = t_{15} + d\_abs\_d0$
27. $t_{22} = t_{27} + d\_reg\_12$
28. $t_{23} = t_0 + 2\ tHI + 2\ tLO$

7.3.2 Polyêdre associé à T_3

Trace T_1 : $t_1 < t_2 < t_3 < t_4 < t_7 < t_5 < t_{13} < t_{14} < t_{12} < t_8 < t_9 < t_{10} < t_{11} < t_{15} < t_{16} < t_{19} < t_{18} < t_{21} < t_{20} < t_{17} < t_{24} < t_{28} < t_{30} < t_{29} < t_{26} < t_{27} < t_{22} < t_{23}$
Polyêdre P_3 :

$\emptyset$
La trace T3 est impossible.

7.4 Trace T_4

Trace T_4 : 3-6-7 et 24-28.

Location	temps	input	abs_d0	abs_net13	abs_csn	reg_10	reg_12	or_net13	transition sortante
1	$[0, 5]$	init	init	init	init	e0d1_U	e0d0_U	e_01_1	up_D
2	$[5, 45]$	A	A	init	init	e0d1_U	e0d0_U	e_01_1	down_CK
3	$[45, 55]$	B	A	init	init	e1d1_X	e0d0_U	e_00_X	up$_net13$
6	55	B	A	A	init	e1d1_X	e0d0_U	e_00_1	down_v17_ext_cs_N
7	$[55, 80]$	B	A	A	init	e1d1_0	e0d0_U	e_00_1	up_v18_E_clk_local_L
5	$[80, 85]$	B	A	init	init	e1d1_0	e1d0_X	e_00_1	down_CSN
13	$[85, 98]$	C	A	init	B	e1d1_0	e1d0_X	e_00_1	up_v18_data_delay_H_inv
14	$[98, 99]$	C	A	init	B	e1d1_0	e1d0_1	e_00_1	up_v17_ext_cs_H
12	$[99, 108]$	C	A	init	init	e1d0_X	e1d0_1	e_00_1	up_v18_data_delay_H
8	$[108, 109]$	C	init	init	init	e1d0_X	e1d1_X	e_00_1	up_v17_ext_cs_N
9	$[109, 119]$	C	init	init	init	e1d0_1	e1d1_X	e_10_X	up_net13
10	$[119, 126]$	C	init	A	init	e1d0_1	e1d1_X	e_10_1	down_v18_data_delay_H_inv
11	$[126, 135]$	C	init	A	init	e1d0_1	e1d1_0	e_10_1	up_CK
15	$[135, 136]$	D	init	A	init	e0d0_U	e1d1_0	e_11_X	down_D
16	$[136, 137]$	E	B	A	init	e0d0_U	e1d1_0	e_11_X	up_CSN
19	$[137, 144]$	F	B	A	A	e0d0_U	e1d1_0	e_11_X	up_v18_E_clk_local_L
18	$[144, 145]$	F	B	init	A	e0d0_U	e1d1_0	e_11_X	down_net13
21	$[145, 151]$	F	B	B	A	e0d0_U	e1d1_0	e_11_0	up_v17_ext_cs_H
20	$[151, 170]$	F	B	B	init	e0d1_U	e1d1_0	e_11_0	down_v18_E_clk_local_L
17	$[170, 180]$	F	B	init	init	e0d1_U	e0d1_U	e_11_0	down_CK
24	$[180, 190]$	G	B	init	init	e1d1_X	e0d1_U	e_10_X	up_net13
28	190	G	B	A	init	e1d1_X	e0d1_U	e_10_1	down_v17_ext_cs_N
30	$[190, 200]$	G	B	A	init	e1d1_0	e0d1_U	e_00_X	up_net13
29	$[200, 215]$	G	B	A	init	e1d1_0	e0d1_U	e_00_1	up_v18_E_clk_local_L
26	$[215, 233]$	G	B	init	init	e1d1_0	e1d1_X	e_00_1	down_v18_E_data_delay_H_inv
27	$[233, 239]$	G	B	init	init	e1d1_0	e1d1_0	e_00_1	down_v18_E_data_delay_H
22	$[239, 257]$	G	init	init	init	e1d1_0	e1d0_X	e_00_1	up_v18_E_data_delay_H_inv
23	$[257, 270]$	G	init	init	init	e1d1_0	e1d0_1	e_00_1	end
31	$[270, \infty[$	H	init	init	init	e1d1_0	e1d0_1	e_00_1	

7.4.1 Système de contraintes associé à T_4

- $t_1 = t_0 + tHI + tLO - d\_setup\_D$
- $t_2 = t_0 + tHI$
- $t_3 = t_2 + d\_or\_net13$
- $t_6 = t_2 + d\_reg\_10$
- $t_7 = t_6 + d\_abs\_net13$
- $t_5 = t_0 + tHI + tLO - d\_setup\_CSN$
- $t_13 = t_7 + d\_reg\_12$
- $t_14 = t_5 + d\_abs\_csn$
- $t_12 = t_1 + d\_abs\_d0$
- $t_8 = t_{14} + d\_reg\_10$
- $t_9 = t_8 + d\_or\_net13$
- $t_{10} = t_{12} + d\_reg\_12$
- $t_{11} = t_0 + tHI + tLO$
- $t_{15} = t_0 + tHI + tLO + d\_hold\_D$
- $t_{16} = t_0 + tHI + tLO + d\_hold\_CSN$
- $t_{19} = t_9 + d\_abs\_net13$
- $t_{18} = t_{11} + d\_or\_net13$
- $t_{21} = t_{16} + d\_abs\_csn$
- $t_{20} = t_{18} + d\_abs\_net13$
- $t_{17} = t_0 + 2tHI + tLO$
- $t_{24} = t_{17} + d\_or\_net13$
- $t_{28} = t_{17} + d\_reg\_10$
- $t_{30} = t_{28} + d\_or\_net13$
- $t_{29} = t + d\_abs\_net13$
- $t_{26} = t_{29} + d\_reg\_12$
- $t_{27} = t_{15} + d\_abs\_d0$
- $t_{22} = t_{27} + d\_reg\_12$
- $t_{23} = t_0 + 2tHI + 2tLO$

7.4.2 Polyêdre associé à T_4

Trace T_1 : $t_1 < t_2 < t_3 < t_6 < t_7 < t_5 < t_{13} < t_{14} < t_{12} < t_8 < t_9 < t_{10} < t_{11} < t_{15} < t_{16} < t_{19} < t_{18} < t_{21} < t_{20} < t_{17} < t_{24} < t_{28} < t_{30} < t_{29} < t_{26} < t_{27} < t_{22} < t_{23}$
Polyêdre P_4 :

$\wedge\ d\_abs\_csn + d\_abs\_net13 + d\_reg\_10 < d\_setup_CSN$
$\wedge\ d\_hold\_D < d\_hold\_CSN$
$\wedge\ d\_setup_CSN + d\_hold\_CSN < d\_abs\_csn + d\_abs\_net13 + d\_reg\_10 + d\_or\_net13$
$\wedge\ d\_reg\_10 < d\_abs\_net13$
$\wedge\ d\_or\_net13 <= d\_reg\_10$
$\wedge\ d\_abs\_d0 + d\_reg\_12 < d\_setup_D$
$\wedge\ tHI + d\_abs\_net13 + d\_reg\_12 + d\_or\_net13 < d\_hold\_D + d\_abs\_d0$
$\wedge\ tLO < d\_setup_D$
$\wedge\ tLO < d\_setup_CSN + d\_abs\_net13 + d\_reg\_10 + d\_reg\_12$
$\wedge\ d\_hold\_CSN + d\_abs\_csn < d\_abs\_net13 + d\_or\_net13$
$\wedge\ d\_setup_CSN + d\_abs\_d0 < d\_setup_D + d\_abs\_csn + d\_reg\_10$
$\wedge\ d\_setup_CSN + d\_abs\_net13 + d\_reg\_10 < tLO$

$\wedge\ \ d\_abs\_net13 + d\_or\_net13 < tHI$
$\wedge\ \ d\_or\_net13 < d\_hold\_CSN + d\_abs\_csn$
$\wedge\ \ d\_setup_D + d\_abs\_csn < d\_setup_CSN + d\_abs\_d0$
$\wedge\ \ d\_hold\_D + d\_abs\_d0 + d\_reg\_12 < tHI + tLO$
$\wedge\ \ d\_setup_D + d\_abs\_csn + d\_reg\_10 + d\_or\_net13 < d\_setup_CSN + d\_abs\_d0 + d\_reg\_12$
$\wedge\ \ d\_setup_CSN + d\_abs\_net13 + d\_reg\_10 + d\_reg\_12 < tLO + d\_abs\_csn$

7.4.3 Vérification sur le modèle paramétré contraint par le polyèdre P_4

Le calcul de $Post * (Init \cap P_4)$ termine en 30 itérations et 35 mn. Il y a 4 traces. Tous les états finaux sont corrects (cf. D-reg-12-T4.log).

8 Annexe : SPSMALL – portion de D à REG12 – modèle bi-ponctuel optimisé

Le code Hytech du modèle à délais bi-ponctuels et intégrant les optimisations d'évaluation des sorties des portes logiques est le suivant :

```
-- spsmall : de D/CK/CSN à REG12
-- optim.hy : 2 delais ponctuels par porte/latch
--optimisation de l'automate or : seules les
--configurations modifiant la valeur de sortie
--entrainent une évaluation de la valeur de sortie.

var  x_abs_d0, x_abs_csn, x_abs_net13, x_reg_10, x_reg_12,
x_or_net13, x_not_v18_E, x_CK,  s : clock;

q, qD, qDb : discrete;

tHI, tLO, d_setup_D, d_hold_D, d_setup_CSN, d_hold_CSN,
d_abs_d0_u, d_abs_d0_d, d_abs_csn_u, d_abs_csn_d,
d_abs_net13_u, d_abs_net13_d, d_reg_10_u, d_reg_10_d,
d_reg_12_u, d_reg_12_d, d_or_net13_u, d_or_net13_d,
d_not_v18_E_u, d_not_v18_E_d : parameter;




automaton input
synclabs: up_CK, down_CK, up_D, down_D, up_CSN, down_CSN;
initially init_input;

loc init_input: while x_CK <=tHI+tLO - d_setup_D  wait {}
when x_CK=tHI+tLO - d_setup_D sync up_D do {} goto A_input;

loc A_input: while x_CK <= tHI wait {}
when x_CK=tHI sync down_CK do {} goto B_input;

loc B_input: while x_CK <= tHI+tLO - d_setup_CSN wait {}
```

```
when x_CK= tHI+tLO - d_setup_CSN sync down_CSN do {} goto C_input;

loc C_input: while x_CK <=tHI+tLO wait {}
when x_CK = tHI+tLO sync up_CK do {} goto D_input;

loc D_input: while x_CK <= tHI+tLO+d_hold_D  wait{}
when x_CK=tHI+tLO+d_hold_D sync down_D do {} goto E_input;

loc E_input: while x_CK <= tHI+tLO+d_hold_CSN  wait{}
when x_CK=tHI+tLO+d_hold_CSN sync up_CSN do {} goto F_input;

loc F_input: while x_CK <= 2tHI+tLO  wait{}
when x_CK=2tHI+tLO sync down_CK do {} goto G_input;

loc G_input: while x_CK <= 2tHI+2tLO  wait{}
when x_CK=2tHI+2tLO            do {} goto H_input;

loc H_input: while x_CK>=0 wait {}
when True do {} goto H_input;

end -- input


automaton abs_d0
synclabs: down_D, up_D,  -- inputs
  down_v18_E_data_delay_H, up_v18_E_data_delay_H; -- outputs
initially init_abs_d0;

loc init_abs_d0 : while True wait {}
when True sync up_D do {x_abs_d0'=0} goto A_abs_d0;
when True sync down_D do {x_abs_d0'=0} goto B_abs_d0;

loc A_abs_d0 : while x_abs_d0 <= d_abs_d0_u wait {}
when True sync down_D do {x_abs_d0'=0} goto B_abs_d0;
when x_abs_d0 = d_abs_d0_u sync up_v18_E_data_delay_H do {} goto init_abs_d0;

loc B_abs_d0 : while x_abs_d0 <= d_abs_d0_d wait {}
when True sync up_D do {x_abs_d0'=0} goto A_abs_d0;
when x_abs_d0 = d_abs_d0_d sync down_v18_E_data_delay_H do {} goto init_abs_d0;

end -- abs_d0

automaton abs_net13
synclabs: down_net13, up_net13,  -- inputs
  down_v18_E_clk_local_L, up_v18_E_clk_local_L; -- outputs
initially init_abs_net13;

loc init_abs_net13 : while True wait {}
when True sync up_net13 do {x_abs_net13'=0} goto A_abs_net13;
when True sync down_net13 do {x_abs_net13'=0} goto B_abs_net13;

loc A_abs_net13 : while x_abs_net13 <= d_abs_net13_u wait {}
when True sync down_net13 do {x_abs_net13'=0} goto B_abs_net13;
when True sync up_net13 do {} goto A_abs_net13;
```

```
when x_abs_net13 = d_abs_net13_u sync up_v18_E_clk_local_L do {} goto init_abs_net13;

loc B_abs_net13 : while x_abs_net13 <= d_abs_net13_d wait {}
when True sync up_net13 do {x_abs_net13'=0} goto A_abs_net13;
when True sync down_net13 do {} goto B_abs_net13;
when x_abs_net13 = d_abs_net13_d sync down_v18_E_clk_local_L do {} goto init_abs_net13;

end -- abs_net13

automaton abs_csn
synclabs: down_CSN, up_CSN,  -- inputs
  down_v17_ext_cs_H, up_v17_ext_cs_H; -- outputs
initially init_abs_csn;

loc init_abs_csn : while True wait {}
when True sync up_CSN do {x_abs_csn'=0} goto A_abs_csn;
when True sync down_CSN do {x_abs_csn'=0} goto B_abs_csn;

loc A_abs_csn : while x_abs_csn <= d_abs_csn_u wait {}
when True sync down_CSN do {x_abs_csn'=0} goto B_abs_csn;
when True sync up_CSN do {} goto A_abs_csn;
when x_abs_csn = d_abs_csn_u sync up_v17_ext_cs_H do {} goto init_abs_csn;

loc B_abs_csn : while x_abs_csn <= d_abs_csn_d wait {}
when True sync up_CSN do {x_abs_csn'=0} goto A_abs_csn;
when True sync down_CSN do {} goto B_abs_csn;
when x_abs_csn = d_abs_csn_d sync down_v17_ext_cs_H do {} goto init_abs_csn;

end -- abs_csn


automaton reg_10
synclabs: up_v17_ext_cs_H, down_v17_ext_cs_H, up_CK, down_CK,  -- inputs  (data + INVERTED enable)
  up_v17_ext_cs_N, down_v17_ext_cs_N ; -- outputs (INVERTED)
initially e0d1_U_reg_10;

loc e0d0_U_reg_10: while True wait {}
when True sync down_CK do {x_reg_10'=0} goto e1d0_X_reg_10;
when True sync up_v17_ext_cs_H do {} goto e0d1_U_reg_10;
when True sync up_CK do {} goto e0d0_U_reg_10;
when True sync down_v17_ext_cs_H do {} goto e0d0_U_reg_10;

loc e1d0_X_reg_10: while x_reg_10 <= d_reg_10_u wait {}
when True sync up_CK do {} goto e0d0_U_reg_10;
when True sync up_v17_ext_cs_H do {x_reg_10'=0} goto e1d1_X_reg_10;
when True sync down_CK do {} goto e1d0_X_reg_10;
when True sync down_v17_ext_cs_H do {} goto e1d0_X_reg_10;
when x_reg_10 = d_reg_10_u sync up_v17_ext_cs_N goto e1d0_1_reg_10;

loc e1d0_1_reg_10: while True wait {}
when True sync up_CK do {} goto e0d0_U_reg_10;
when True sync up_v17_ext_cs_H do {x_reg_10'=0} goto e1d1_X_reg_10;
when True sync down_CK do {} goto e1d0_1_reg_10;
when True sync down_v17_ext_cs_H do {} goto e1d0_1_reg_10;
```

```
loc e0d1_U_reg_10: while True wait {}
when True sync down_CK do {x_reg_10'=0} goto e1d1_X_reg_10;
when True sync down_v17_ext_cs_H do {} goto e0d0_U_reg_10;
when True sync up_CK do {} goto e0d1_U_reg_10;
when True sync up_v17_ext_cs_H do {} goto e0d1_U_reg_10;

loc e1d1_X_reg_10: while  x_reg_10 <= d_reg_10_d wait {}
when True sync up_CK do {} goto e0d1_U_reg_10;
when True sync down_v17_ext_cs_H do {x_reg_10'=0} goto e1d0_X_reg_10;
when True sync down_CK do {} goto e1d1_X_reg_10;
when True sync up_v17_ext_cs_H do {} goto e1d1_X_reg_10;
when x_reg_10 = d_reg_10_d sync down_v17_ext_cs_N do {q' = 1} goto e1d1_0_reg_10;

loc e1d1_0_reg_10: while True wait {}
when True sync up_CK do {} goto e0d1_U_reg_10;
when True sync down_v17_ext_cs_H do {x_reg_10'=0} goto e1d0_X_reg_10;
when True sync down_CK do {} goto e1d1_0_reg_10;
when True sync up_v17_ext_cs_H do {} goto e1d1_0_reg_10;

end -- reg_10

automaton reg_12
synclabs: up_v18_E_data_delay_H, down_v18_E_data_delay_H,  -- inputs (data)
  up_v18_E_clk_local_L, down_v18_E_clk_local_L; -- inputs (enable)
initially e0d0_U_reg_12;

loc e0d0_U_reg_12: while True wait {}
when True sync up_v18_E_clk_local_L do {x_reg_12'=0} goto e1d0_X_reg_12;
when True sync up_v18_E_data_delay_H do {} goto e0d1_U_reg_12;
when True sync down_v18_E_clk_local_L do {} goto e0d0_U_reg_12;
when True sync down_v18_E_data_delay_H do {} goto e0d0_U_reg_12;

loc e1d0_X_reg_12: while x_reg_12 <= d_reg_12_u wait {}
when True sync down_v18_E_clk_local_L do {} goto e0d0_U_reg_12;
when True sync up_v18_E_clk_local_L do {} goto e1d0_X_reg_12;
when True sync up_v18_E_data_delay_H do {x_reg_12'=0} goto e1d1_X_reg_12;
when True sync down_v18_E_data_delay_H do {} goto e1d0_X_reg_12;
--when x_reg_12 = d_reg_12_u sync up_v18_data_delay_H_inv goto e1d0_1_reg_12;
when x_reg_12 = d_reg_12_u do {qD'=1} goto e1d0_1_reg_12;

loc e1d0_1_reg_12: while True wait {}
when True sync down_v18_E_clk_local_L do {} goto e0d0_U_reg_12;
when True sync up_v18_E_clk_local_L do {} goto e1d0_1_reg_12;
when True sync up_v18_E_data_delay_H do {x_reg_12'=0} goto e1d1_X_reg_12;
when True sync down_v18_E_data_delay_H do {} goto e1d0_1_reg_12;

loc e0d1_U_reg_12: while True wait {}
when True sync up_v18_E_clk_local_L do {x_reg_12'=0} goto e1d1_X_reg_12;
when True sync down_v18_E_data_delay_H do {} goto e0d0_U_reg_12;
when True sync down_v18_E_clk_local_L do {} goto e0d1_U_reg_12;
when True sync up_v18_E_data_delay_H do {} goto e0d1_U_reg_12;
```

```
loc e1d1_X_reg_12: while x_reg_12 <= d_reg_12_d wait {}
when True sync down_v18_E_clk_local_L do {} goto e0d1_U_reg_12;
when True sync up_v18_E_clk_local_L do {} goto e1d1_X_reg_12;
when True sync down_v18_E_data_delay_H do {x_reg_12'=0} goto e1d0_X_reg_12;
when True sync up_v18_E_data_delay_H do {} goto e1d1_X_reg_12;
--when x_reg_12 = d_reg_12_d sync down_v18_data_delay_H_inv goto e1d1_0_reg_12;
when x_reg_12 = d_reg_12_d do {qDb'=1} goto e1d1_0_reg_12;

loc e1d1_0_reg_12: while True wait {}
when True sync down_v18_E_clk_local_L do {} goto e0d1_U_reg_12;
when True sync up_v18_E_clk_local_L do {} goto e1d1_0_reg_12;
when True sync down_v18_E_data_delay_H do {x_reg_12'=0} goto e1d0_X_reg_12;
when True sync up_v18_E_data_delay_H do {} goto e1d1_0_reg_12;
 end -- reg_12


automaton or_net13
synclabs: up_v17_ext_cs_N, down_v17_ext_cs_N,  -- INVERTED inputs
  up_CK, down_CK,-- INVERTED inputs
  up_net13, down_net13; -- outputs

-- codage des états : 1re entree (v_17_ext_cs_N), 2e entree (CK) _ sortie net13 (0,1,X)
initially e_01_1_net13;

loc e_00_1_net13: while True wait{}
when True sync up_v17_ext_cs_N  do {} goto e_10_1_net13;
when True sync up_CK  do {} goto e_01_1_net13;
when True sync down_v17_ext_cs_N  do {} goto e_00_1_net13;
when True sync down_CK  do {} goto e_00_1_net13;

loc e_01_1_net13: while True wait{}
when True sync up_v17_ext_cs_N  do {x_or_net13'=0} goto e_11_X_net13;
when True sync down_CK  do {} goto e_00_1_net13;
when True sync down_v17_ext_cs_N  do {} goto e_01_1_net13;
when True sync up_CK  do {} goto e_01_1_net13;

loc e_10_1_net13: while True wait{}
when True sync down_v17_ext_cs_N  do {} goto e_00_1_net13;
when True sync up_CK  do {x_or_net13'=0} goto e_11_X_net13;
when True sync up_v17_ext_cs_N  do {} goto e_10_1_net13;
when True sync down_CK  do {} goto e_10_1_net13;

loc e_11_0_net13: while True wait{}
when True sync down_v17_ext_cs_N  do {x_or_net13'=0} goto e_01_X_net13;
when True sync down_CK  do {x_or_net13'=0} goto e_10_X_net13;
when True sync up_v17_ext_cs_N  do {} goto e_11_0_net13;
when True sync up_CK  do {} goto e_11_0_net13;

loc e_00_X_net13: while x_or_net13 <= d_or_net13_u wait{}
when True sync up_v17_ext_cs_N  do {x_or_net13'=0} goto e_10_X_net13;
when True sync up_CK  do {x_or_net13'=0} goto e_01_X_net13;
when True sync down_v17_ext_cs_N  do {} goto e_00_X_net13;
when True sync down_CK  do {} goto e_00_X_net13;
when x_or_net13 = d_or_net13_u sync up_net13  do {}  goto e_00_1_net13;
```

```
loc e_01_X_net13: while x_or_net13 <= d_or_net13_u wait{}
when True sync up_v17_ext_cs_N  do {x_or_net13'=0} goto e_11_X_net13;
when True sync down_CK  do {x_or_net13'=0} goto e_00_X_net13;
when True sync down_v17_ext_cs_N  do {} goto e_01_X_net13;
when True sync up_CK  do {} goto e_01_X_net13;
when x_or_net13 = d_or_net13_u sync up_net13  do {} goto e_01_1_net13;

loc e_10_X_net13: while x_or_net13 <= d_or_net13_u wait{}
when True sync down_v17_ext_cs_N  do {x_or_net13'=0} goto e_00_X_net13;
when True sync up_CK  do {x_or_net13'=0} goto e_11_X_net13;
when True sync up_v17_ext_cs_N  do {} goto e_10_X_net13;
when True sync down_CK  do {} goto e_10_X_net13;
when x_or_net13 = d_or_net13_u sync up_net13  do {} goto e_10_1_net13;

loc e_11_X_net13: while x_or_net13 <= d_or_net13_d wait{}
when True sync down_v17_ext_cs_N  do {x_or_net13'=0} goto e_01_X_net13;
when True sync down_CK  do {x_or_net13'=0} goto e_10_X_net13;
when True sync up_v17_ext_cs_N  do {} goto e_11_X_net13;
when True sync up_CK  do {} goto e_11_X_net13;
when x_or_net13 = d_or_net13_d sync down_net13  do {} goto e_11_0_net13;

end -- or_net13
```

9 Annexe : SPSMALL – portion "temps de réponse en écriture" (BLUEBERRIES) – modèle bi-borné

Le code hytech du modèle est donné ci-après.

```
-- taawpretest.hy
var  x25, x26, x1, z0, s, z1,z5, z7,z8, zD : clock;
t, q0, q1, q2, q7, q8, q26, d,e, d1, e1, o5, o15,
o16, qDdec, oDdec, o8 : discrete;

tHI, tLO, b3up, b3down, b2up, b2down, tDdec,
tupc1, tdownc1, tupE, tdownc2, tdownE, tupD1,
tdownwen, toutQZ1, toutQA0, toutQA1, toutDdec, toutQE1,
tout25, tout26, tout8, tout7, tout5,
down_d15, tsetupwen,
lbl13, ubl13, lbl14,ubl14, lbl7,ubl7, lbl8,ubl8, lbl5,ubl5,
lbl16,ubl16,lbl1, ubl1  : parameter;




automaton clock1
synclabs: upc1, downc1,
downc2, upE, downE;
initially Highc1pre;

loc Highc1pre: while x1 <=tHI+b3up & x1>=0 wait {}
when x1=tHI+b3up sync upc1 do {tupc1'=s,e'=s} goto LowE;

loc LowE: while x1 <= tHI+b3up+b2up & x1>=0 wait {}
```

```
when x1=tHI+b3up+b2up sync upE do {tupE'=s,e1'=s} goto Lowc2;

loc Lowc2: while x1 <= tHI+tLO+b3down & x1>=0 wait {}
when x1= b3down+tHI+tLO sync downc1
do {tdownc1'=s} goto Highc1bis;

loc Highc1bis: while x1 <=tHI+tLO+b3down+down_d15 & x1>=0 wait {}
when x1 = tHI+tLO+b3down+down_d15 sync downc2 do {tdownc2'=s,o15'=s} goto YY;

loc YY: while x1 <= tHI+tLO+b3down+b2down & x1>=0 wait{}
when x1=tHI+tLO+b3down+b2down sync downE do {tdownE'=s} goto PreEndc1;

loc PreEndc1: while x1 <= 2tHI+2tLO & x1>=0 wait {}
when x1=2tHI+2tLO -- sync upc1
----& tout7=tHI+tLO+b3down+down_d15+lbl7+lbl8

--& tupc1 < tdownwen
--& tupc1<tupE
--& tdownwen < toutQZ1
--& tupE<toutQZ1
--& toutQZ1<toutQA1
--& toutQA1<toutQE1
--& toutQE1<tupD1
--& tupD1<toutDdec
--& tupD1<tdownc1
--& toutDdec<tdownc2
--& tdownc1<tdownc2
--& tdownc2<tdownE
--& tdownc2<tout5
--& tdownE<tout8
--& tout5<tout8
--& tout8<tout7

--do {e'=s}
goto Endc1;

loc Endc1: while x1>=0 wait {}
when True do {} goto Endc1;

end -- clock1


automaton wen
synclabs: downwen ;
initially Highwen;

loc Highwen: while s <= tHI+tLO-tsetupwen & s>=0 wait {}
when s=tHI+tLO-tsetupwen sync downwen do {tdownwen'=s,q0'=1} goto End_d1;

loc End_d1: while s>=0 wait {}
when True do {} goto End_d1;

end -- wen
```

```
automaton D
synclabs: upD1 ;
initially Ddec_0;

loc Ddec_0: while s <= tDdec & s>=0 wait {}
when s=tDdec sync upD1 do {tupD1'=s,qDdec'=1, d1'=s} goto End_Ddec;

loc End_Ddec: while s>=0 wait {}
when True do {} goto End_Ddec;

end -- wen


automaton delay13
synclabs: downwen,
outQZ1;
initially D0Z;

loc D0Z: while z0>=0 wait {}
when True sync downwen do {z0'=0} goto D1Z;

loc D1Z: while z0 <= ubl13 & z0>=0 wait {}
when z0 >= lbl13 sync outQZ1 do {toutQZ1'=s,q1'=1,d'=s} goto end0; -- D1bisZ;

loc end0: while z0>=0  wait {}
when True do {} goto end0;

end -- delay13

automaton latchW
synclabs: upc1, downc1,
outQZ1, outQA0, outQA1;
initially D2c0A;

loc D2c1A:while z1>=0 wait {}
when True sync downc1 do {} goto D2c0A;
when True sync outQZ1 do {z1'=0} goto D1c1A;

loc D2c0A: while z1>=0 wait{}
when True sync upc1 do{} goto D2c1A;
when True sync outQZ1 do {} goto D1c0A;

loc D0c0A: while z1>=0 wait {}
  when True sync upc1 do {z1'=0} goto D0c1A;
  when True sync outQZ1 do {} goto D1c0A;

loc D1c0A: while z1>=0 wait {}
  when True sync upc1  do {z1'=0} goto D1c1A;
  when True sync outQZ1 do {} goto D1c0A;

loc D0c1A: while z1 <= ubl14 & z1>=0 wait {}
  when True sync downc1 do {} goto D0c0A;
  when True sync outQZ1 do {z1'=0} goto D1c1A;
```

```
  when z1 >= lbl14 sync outQA0 do {toutQA0'=s,q1'=0}
         goto D0c1bisA; -- D2c1A;


loc D1c1A: while z1 <= ubl14  & z1>=0 wait {}
  when True sync downc1 do {} goto D1c0A;
  when True sync outQZ1 do {} goto D1c1A;
  when z1 >= lbl14 &  d>e
sync outQA1 do {toutQA1'=s,q1'=1}
         goto D1c1bisA; -- D2c1A;

 loc D0c1bisA: while z1>=0 wait {}
  when True sync downc1 do {} goto D0c0A;
     when True sync outQZ1 do {z1'=0} goto D1c1A;

 loc D1c1bisA: while z1>=0  wait {}
  when True sync downc1 do {} goto D1c0A;
when True sync outQZ1 do {} goto D1c1bisA;

end -- latchW

automaton latchD
synclabs: upD1, upE, downE, outDdec;

initially D2E0;

loc D2E1: while zD>=0 wait {}
when True sync downE do {} goto D2E0;
when True sync upD1 do {zD'=0} goto D1E1;

loc D2E0: while zD>=0 wait{}
when True sync upD1 do{} goto D1E0;
when True sync upE do {} goto D2E1;

loc D1E0: while zD>=0 wait {}
  when True sync upE do {zD'=0} goto D1E1;


loc D1E1: while zD <= ubl1  wait {}
  when True sync downE do {} goto D1E0;
  when zD >= lbl1 &  d1>e1
sync outDdec do {toutDdec'=s,oDdec'=s} goto EndlatchD;

 loc EndlatchD: while zD>=0  wait {}
  when True do {} goto EndlatchD;
  when True sync downE do {} goto EndlatchD;

end -- latchD


automaton delay5
 synclabs: outDdec, out5 ;
 initially E5_0;
```

```
loc E5_0: while z5>=0 wait {}
 when True sync outDdec do {z5'=0} goto E5_1;

loc E5_1: while z5 <= ubl5 & z5>=0 wait {}
 when z5 >= lbl5 sync out5 do {tout5'=s,o5'=s} goto end5;

loc end5: while z5>=0  wait {}
 when True do {} goto end5;

end -- delay5


automaton delay16
synclabs: outQA1, outQE1 ;
initially E0Z;

loc E0Z: while z5>=0 wait {}
 when True sync outQA1 do {z5'=0} goto E1Z;

loc E1Z: while z5 <= ubl16 & z5>=0 wait {}
 when z5 >= lbl16 sync outQE1 do {toutQE1'=s,q2'=1, o16'=s} goto endE;

loc endE: while z5>=0  wait {}
 when True do {} goto endE;

end -- delay16

automaton delay8
synclabs: out25, out8 ;
initially E8_0;

loc E8_0: while z8>=0 wait {}
 when True sync out25 do {z8'=0} goto E8_1;

loc E8_1: while z8 <= ubl8 & z8>=0 wait {}
 when z8 >= lbl8 sync out8 do {q8'=1, tout8'=s, o8'=s} goto end8;

loc end8: while z8>=0  wait {}
 when True do {} goto end8;

end -- delay8


automaton delay7
synclabs: out26, outQ ;
initially E7_0;

loc E7_0: while z7>=0 wait {}
 when True sync out26 do {z7'=0} goto E7_1;

loc E7_1: while z7 <= ubl7 & z7>=0 wait {}
 when z7 >= lbl7 sync outQ do {q7'=1, tout7'=s} goto end7;

loc end7: while z7>=0  wait {}
```

```
  when True do {} goto end7;

 end -- delay7


automaton gate_25
synclabs:outQE1,out25,downc2;
initially init_or;

loc init_or : while x25>=0 wait {}
when True sync outQE1 do{} goto A;
when True sync downc2 do{} goto B;

loc A: while x25>=0 wait {}
when True sync downc2 do{q2'=1,x25'=0} goto D;

loc B: while x25>=0 wait {}
when True sync outQE1 do{q2'=1,x25'=0} goto D;


loc D: while x25>=0 wait {}
when o15>o16 & x25=0 sync out25 do {tout25'=s} goto End_or;

loc End_or: while x25>=0 wait {}
when True do{} goto End_or;

end -- gate_25


automaton gate_26
synclabs:out8,out5,out26;
initially init26;

loc init26 : while x26>=0 wait {}
when True sync out8 do{} goto A26;
when True sync out5 do{} goto B26;

loc A26: while x26>=0 wait {}
when True sync out5 do{q26'=1,x26'=0} goto D26;

loc B26: while x26>=0 wait {}
when True sync out8 do{q26'=1,x26'=0} goto D26;

loc D26: while x26>=0 wait {}
when o8>oDdec & x26=0 sync out26 do {tout26'=s} goto End26;

loc End26: while x26>=0 wait {}
when True do {} goto End26;

end -- gate_26

-- analysis commands

var init_reg,
```

```
pre_reg, post_reg, bad_reg1, bad_reg2, nZ, interEnd , approx,
reg1, reg2, loc_final, goal_reg, ult_reg : region;

init_reg :=
loc[clock1]=Highc1pre
& loc[wen]=Highwen
& loc[latchW]=D2c0A
& loc[delay5]=E5_0
& loc[delay7]=E7_0
& loc[delay8]=E8_0
& loc[delay13]=D0Z
& loc[delay16]=E0Z
  & loc[gate_25]=init_or
& loc[gate_26]=init26
& loc[latchD]=D2E0
& loc[D]=Ddec_0

&  x1=0 & x25=0 & x26=0 & z0=0 & z1=0 & z5=0 & z7=0 & z8=0 & zD=0 & s=0

& b3down>0 & b3up>0
& b2down>0 & b2up>0

        & 0<lbl13 & lbl13=ubl13
        & 0<lbl14 & lbl14=ubl14
& 0<lbl16 & lbl16=ubl16
& 0<lbl1 & lbl1=ubl1
& 0<lbl5 & lbl5=ubl5
& 0<lbl7 & lbl7=ubl7
& 0<lbl8 & lbl8=ubl8
& 0<down_d15

& t=0 & q0=2 & q1=2 & q2=2  & q7=2 & q8=2 & q26=2 & qDdec=2
& d=0 & e=0 & d1=0 & e1=0 & o15=0 & o16=0 & o8=0 & oDdec=0
--& tupc1=0 & tdownc1=0 & tupE=0 & tdownc2=0 & tdownE=0
--& tdownwen=0 &  tupD1=0
--& tdownwen=0 & toutQZ1=0 & toutQA0=0 & toutQA1=0
--& toutDdec=0 & tout5=0
--& toutQE1=0 & tout25=0 & tout26=0 & tout8=0 & tout7=0 & tout5=0

& 0<tsetupwen

        & tHI>0 & tLO>0 & tDdec>0 & down_d15>0 & tsetupwen>0

-- SP1 obtenu par Pre* avec ordre total --
----  &   tHI + tLO + lbl13 + lbl14 + lbl16 < tDdec + tsetupwen
--& tDdec + ubl1 < tHI + tLO + b3down
-- retrait donne: tsetupD in ]99,107] and tsetupW in ]46,54[
--& tLO < b3up + b2up + tsetupwen
-- retrait supplementaire permet de trouver des valeurs tsetupW<46
----& tHI + tLO + b3down + down_d15 < tDdec + ubl5 + ubl1
--& tDdec + ubl5 + ubl1 < tHI + tLO + b3down + b2down
----& b3down + down_d15 + lbl7 + lbl8 < tHI + tLO
----& b3up + b2up + tsetupwen < tLO + lbl13
----& b3up + tsetupwen < tLO
```

```
----& b2down < down_d15 + lbl8;
;
------------------------------------------------------------------------
prints "hi there: initial region";
print init_reg;
prints "";

prints "iterated succ";
post_reg := reach forward from init_reg endreach;

loc_final := post_reg
& loc[clock1]=Endc1
& loc[wen]=End_d1
& loc[delay13]=end0
& loc[latchW]=D1c0A
& loc[delay16]=endE
  & loc[gate_25]=End_or
& loc[delay7]=end7
& loc[delay8]=end8
& loc[delay5]=end5
& loc[gate_26]=End26
& loc[latchD]=EndlatchD
& loc[D]=End_Ddec
& q1=1 & q2=1 & q0=1 & q2=1 & q7=1 & q8=1 & q26=1
&  x1>=0 & x25>=0 & x26>=0
& z0>=0 & z1>=0 & z5>=0 & z7>=0 & z8>=0 & zD>=0
& tHI>0 & tLO>0
& s=2tHI+2tLO
        & 0<lbl13 & lbl13=ubl13
        & 0<lbl14 & lbl14=ubl14
& 0<lbl16 & lbl16=ubl16
& 0<lbl1 & lbl1=ubl1
& 0<lbl5 & lbl5=ubl5
& 0<lbl7 & lbl7=ubl7
& 0<lbl8 & lbl8=ubl8
& 0<down_d15 & b2up>0 & b2down>0
& tsetupwen>0 & tDdec>0
& tupc1>0 & tdownc1>0 & tupE>0 & tdownc2>0 & tdownE>0
& tdownwen>0 &  tupD1>0
& tdownwen>0 & toutQZ1>0 & toutQA0>0 & toutQA1>0
& toutDdec>0
& toutQE1>0 & tout25>0 & tout26>0 & tout8>0 & tout7>0 & tout5>0

-- ordonnancement total  des evenements sur SP1

--& tupc1 < tdownwen & tdownwen < tupE & tupE<toutQZ1 & toutQZ1<toutQA1
--& toutQA1<toutQE1 & toutQE1<tupD1 & tupD1<toutDdec
--& toutDdec<tdownc1 & tdownc1<tdownc2
--& tdownc2<tout5 & tout5<tdownE & tdownE<tout8 & tout8<tout7
& tout7>tHI+tLO+b3down+down_d15+lbl7+lbl8 -- <2tHI+2tLO;
;

goal_reg := hide x25, x26, x1, z0, z1,z5, z7,z8, zD,
t, q0, q1, q2, q7, q8, q26, d,e, d1, e1, o5, o15, o16, qDdec, oDdec,o8,
```

```
tsetupwen, tDdec,
tupc1, tdownc1, tupE, tdownc2, tdownE,
tdownwen,  tupD1,
tdownwen, toutQZ1, toutQA0, toutQA1,
toutDdec,
toutQE1, tout25, tout26, tout8, tout5,

--tout7,
s
--,a,b,c,d,e,f,g,h
in
loc_final
endhide
;

print(goal_reg);
```
Le polyèdre obtenu par l'analyse de trace (méthode Post-Pre-Post) est le suivant :

```
 tHI + tLO + lbl13 + lbl14 + lbl16 < tDdec + tsetupwen  (*)
 tHI + tLO + b3down + down_d15 < tDdec + ubl5 + ubl1
 tDdec + ubl1 < tHI + tLO + b3down                      (*)
 tDdec + ubl5 + ubl1 < tHI + tLO + b3down + b2down
 b3down + down_d15 + lbl7 + lbl8 < tHI + tLO
 b3up + b2up + tsetupwen < tLO + lbl13
 tLO < b3up + b2up + tsetupwen
 b3up + tsetupwen < tLO
 b2down < down_d15 + lbl8;
```
On vérifie a posteriori qu'on ne peut atteindre d'état défectueux en une dizaine de secondes. Certaines contraintes peuvent être relâchées (ce qui permet d'améliorer les bornes de `setupD` et `setupWEN`). Les contraintes marquées d'un astérisque peuvent être éliminées. La vérification est réalisée en 1h30.
le polyèdre obtenu par l'élargissemnt du point de fonctionnement (méthode 2, pour la même instanciation de référence des paramètres) est le suivant :

```
Atoms selected:
     & b2down < lbl8 + down_d15
     & ubl14 + ubl13 < tsetupwen + b3down
     & lbl5 + tDdec + lbl1 < tLO + lbl8 + tHI + b3down + down_d15
     & down_d15 + lbl8 + b3down + lbl7 < tLO + tHI
     & ubl14 + ubl16 + ubl13 < tsetupwen + b3down
     & lbl16 + tLO + lbl13 + tHI + ubl14 < tsetupwen + tDdec
     & tLO + tHI < tsetupwen + tDdec
     & tsetupwen < tLO + tHI
     & tsetupwen + b3up < tLO
     & tLO + tHI + lbl13 < tsetupwen + tDdec
     & tLO + b3down + tHI < ubl5 + tDdec + lbl1
     & b3up + tHI < tDdec
     & tLO < b2up + b3up + tsetupwen
     & tLO + lbl14 + lbl13 + tHI < tsetupwen + tDdec
     & down_d15 + b3down + lbl8 < tLO + tHI
```

```
     & lbl5 + lbl1 + tDdec < tLO + b2down + tHI + b3down
     & down_d15 + tLO + tHI + b3down < lbl5 + tDdec + lbl1
     & b2up + tsetupwen + b3up < lbl13 + tLO
     & lbl13 < tsetupwen + b3down
     & tDdec < tLO + b3down + tHI
     & lbl1 + tDdec < tLO + tHI + b3down

Script ended after 606.18 seconds
  Python execution time : 6.46s
  HyTech execution time : 584.67s (40 executions)
  Prolog execution time : 9.67s (375 executions)
*********************************************
```

On vérifie a posteriori qu'on ne peut atteindre d'état défectueux en une dizaine de secondes. Le même travail de relâchement pourrait être réalisé.

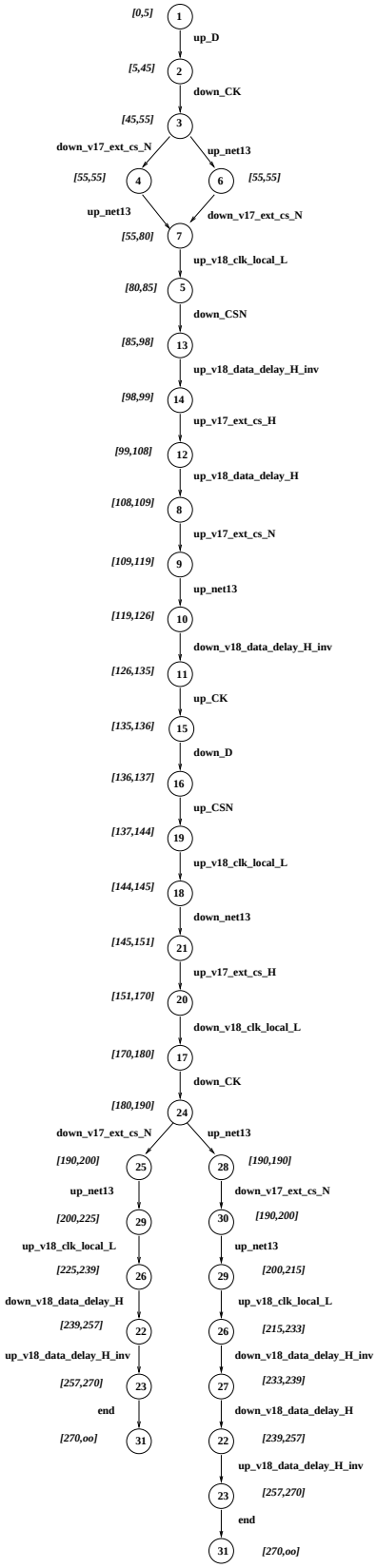

FIG. 10 – Graphe des états accessibles du programme pour les paramètres instanciés.

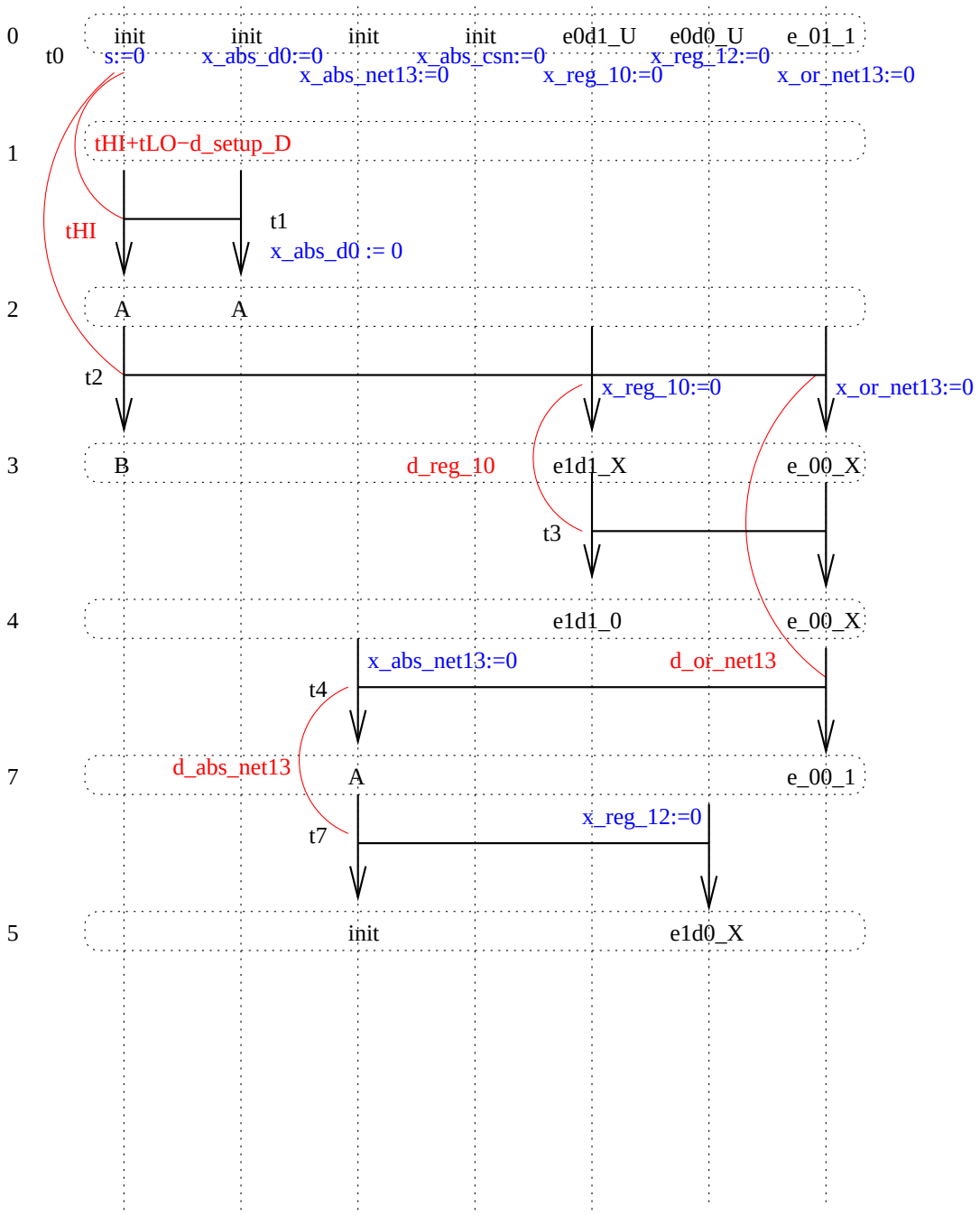

FIG. 11 – Portion de graphe d'événements associé à la trace T_1.