Conception sûre de systèmes de contrôle-commande matériels à base de COTS

Salam Hajjar, Emil Dumitrescu, Eric Niel

Conception sûre de systèmes de contrôle-commande matériels à base de COTS

Salam Hajjar, Emil Dumitrescu, Eric Niel

Laboratoire Ampère
25, rue Jean Capelle
F-69621, Villeurbanne
{salam.hajjar,emil.dumitrescu,eric.niel}@insa-lyon.fr

RÉSUMÉ. *Cet article propose une méthode de conception sûre de systèmes de contrôle-commande embarqués à base de COTS ControlBuild. La méthode commence par l'élaboration des COTS ControlBuild, puis elle utilise la vérification formelle pour découvrir les erreurs de conception. Ensuite la technique de la synthèse du contrôleur discret (SCD) est employée pour corriger automatiquement les erreurs détectées. La méthode proposée est illustrée sur un l'exemple d'un système ferroviaire d'accès voyageurs pour le train NAT.*

ABSTRACT. *This paper proposes an approach for safe design of hardware embedded control systems. The approach is based on a combination of formal verification and discrete controller synthesis techniques. Formal verification is solicited to detect design errors and provide counterexamples, while the Discrete Controller Synthesis technique is used to correct those error since it attempts to enforce previously verified specifications which do not hold. It automatically produces control code, which is correct by construction with respect to the specification to enforce. We illustrate the approach on a real train controller subsystem.*

MOTS-CLÉS : *COTS, vérification formelle, synthèse de contrôleurs discrets, système à événements discrets, propriété de sûreté, propriété de vivacité.*

KEYWORDS: *COTS, formal verification, discret controller synthesis, discret event systems, safety property, liveness property.*

1. Introduction

Dans les systèmes embarqués, les erreurs de conception peuvent avoir des conséquences graves, surtout quand elles apparaissent dans les systèmes de contrôle commande critiques, telles que les usines robotisées, usines de production d'énergie, ou les systèmes de transport. De telles erreurs peuvent coûter des vies humaines, ou, au mieux, une importante somme d'argent pour la refonte du système. Ces systèmes sont souvent modélisés comme des systèmes réactifs synchrones, en utilisant des machines à états finis communicantes. Leur réalisation nécessite des méthodes et des techniques de conception spécifiques, afin de garantir une fonctionnalité correcte, conformément au cahier des charges.

En plus de la simulation, la vérification formelle et en particulier la technique intitulée "model checking", présentée par Clarke (2008a), Clarke (2008b) est devenue incontournable pour détecter les "bugs" subtils, qui sont très difficiles à détecter par simulation. Elle été utilisée avec succès dans le domaine industriel pour détecter automatiquement de telles erreurs de conception, notamment dans les systèmes avioniques (Laurent *et al.* (2001)). Toutefois, le concepteur doit corriger ces erreurs manuellement, un processus délicat où l'on s'expose à de nouvelles erreurs de conception. En effet, en tentant de corriger manuellement une erreur, une autre erreur peut être involontairement introduite, ce qui crée un cercle vicieux. Toutefois, ce processus de conception permet une maturation progressive de blocs de construction réutilisables, que l'on finit par considérer comme sûrs. Ces blocs réutilisables peuvent être gérés en interne ou acquis auprès de fournisseurs. Leur nom générique est "composant sur étagère", ("composants off-the shelf" dits aussi "COTS Commercial"). Un COTS englobe à la fois un comportement et un cahier des charges. Les COTS peuvent être assemblés en fonction de leurs caractéristiques, afin de créer de nouveaux comportements, satisfaisant des spécifications plus complexes. L'avantage de cette démarche réside en un gain considérable de temps de développement, grâce à la réutilisation du code. Cependant, même si chaque COTS a été individuellement et soigneusement vérifié, ils sont rarement conçus pour fonctionner parfaitement ensemble. Des erreurs de conception peuvent apparaître lors du simple assemblage d'une collection de COTS sûrs et leur correction manuelle peut être source d'erreurs et engendrer une importante perte de temps. Ainsi, l'intégration de COTS demeure une tâche coûteuse.

Dans ce contexte, la technique de synthèse de contrôleurs discrets (SCD) est une solution prometteuse. Elle permet de construire des composants corrects par construction, en générant automatiquement un composant nommé *superviseur*. Cette méthode a été d'abord proposée par (Ramadge, Wonham, 1989), avec de nombreuses applications dans le contrôle automatisé des chaînes de fabrication. Dans cet article, nous proposons une approche basée sur les méthodes présentées dans (Hajjar *et al.*, 2012) et (Hajjar *et al.*, 2013), et appliquée aux systèmes de contrôle-commande sûrs à base de COTS. Cette approche utilise le "model checking" en synergie avec la synthèse de contrôleurs discrets. Nous démontrons la validité de cette approche sur un système de contrôle-commande issu d'un applicatif ferroviaire. Cette validation s'appuie sur la proposition d'une chaîne d'outils établissant une synergie entre un logiciel industriel

de conception d'automatismes et les techniques de "model checking" et de synthèse de contrôleurs discrets.

Le reste du papier est organisé comme suit. La section 2 définit la notion de COTS. La section 3 rappelle brièvement les notions de base du "model-checking". La section 4 rappelle le principe de la synthèse de contrôleurs discrets. La section 5 présente notre méthode de conception sûre pour les systèmes matériels à base de composants. La section 6 illustre la méthode sur un système de contrôle-commande des trains, elle présente une application de la méthode proposée et les résultats obtenus. Une étude bibliographique portant sur les travaux connexes dans le domaine de la conception à base de composants est proposée dans la section 7.

2. Définitions

Le comportement M d'un COTS est donné par un programme réactif, dont le modèle sémantique sous-jacent est celui des machines à états finis synchrones *(FSM)* Booléennes, avec entrées et sorties. Une FSM M est définie par un n-uplet $M = \langle q_0, X, Q, \delta, Y, \lambda \rangle$ où q_0 est l'état initial, X est l'ensemble des variables Booléennes d'entrée, Q est l'ensemble des états, δ est la fonction de transition, Y est l'ensemble des variables Booléennes de sortie, et λ est la fonction de sortie de M.

Un composant COTS de contrôle-commande est caractérisé par quatre éléments : (1) une interface $I = X \cup Y$; (2) des hypothèses de l'environnement, ou pré-conditions A; (3) des garanties de comportement ou post-conditions G; (4) un comportement fonctionnel M. Les hypothèses et garanties de C sont exprimées en logique temporelle.

Dans le cadre de ce travail, les COTS sont des briques de base, servant à la construction d'un système de contrôle-commande qui délivre sa fonction en interagissant avec un processus physique. Ceci est illustré dans la figure 1 où CC et PH représentent respectivement les parties contrôle-commande et physique du système final.

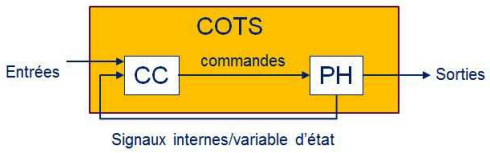

Figure 1. Architecture du COTS, la partie commande avec la partie physique

3. La technique du "model checking"

Il s'agit d'une technique de vérification formelle. Elle permet d'établir une relation de satisfaction entre le modèle FSM Booléen d'un programme réactif, et sa spécification, écrite en logique temporelle (Lamport, 1983). L'algorithme de vérification effectue une recherche exhaustive dans l'espace d'états du modèle étudié, et calcule

l'ensemble de tous les états dans lesquels la spécification est vraie. De par son exhaustivité, la méthode de "model checking" est plus puissante que la simulation. En outre, le "model checking" est capable de fournir un contre-exemple diagnostique en cas de non satisfaction d'une spécification. Ce contre-exemple s'assimile à un scénario de simulation automatiquement généré et directement visualisable, menant le système étudié de son état initial vers l'état où la spécification désirée n'est plus satisfaite.

La performance du "model checking" diminue considérablement selon la taille du programme vérifié, en raison de sa complexité exponentielle par rapport au nombre de variables d'état contenues dans le modèle FSM sous-jacent. C'est pourquoi le "model checking" reste surtout adapté pour la vérification de systèmes de taille moyenne. Cependant, pour de tels systèmes, même si la vérification formelle peut trouver de nombreuses erreurs de conception, leur correction manuelle reste une mission difficile, à la charge du concepteur.

4. La synthèse de contrôleurs discrets (SCD)

Cette technique est développée par Marchand *et al.* (2000). La principale fonctionnalité utilisée dans le cadre de ce travail concerne le renforcement de l'invariance d'ensembles d'états. Pour tout système modélisé par une FSM M et pour toute exigence fonctionnelle P qui se traduit par un ensemble d'états au sein duquel M doit rester impérativement, la SCD calcule le plus grand ensemble d'états de M satisfaisant P et qui peut être rendu invariant. Cet ensemble est appelé un invariant sous contrôle IUC. L'ensemble des transitions menant à l'ensemble IUC constitue la solution du problème de contrôle. Cet ensemble solution est appelé un superviseur SUP.

Le calcul du superviseur dépend de la contrôlabilité de M. Pour réaliser ce contrôle, le superviseur doit agir sur la valeur de certaines variables d'entrée de M désignées comme étant contrôlables (dont la valeur peut être affectée par le superviseur). Ainsi, préalablement au calcul du superviseur, l'utilisateur de la SCD doit proposer une partition du vecteur d'entrée X de M en deux sous-ensembles disjoints : les entrées incontrôlables X_{uc} et les entrées contrôlables X_c. Cette partition doit se faire de telle sorte que $X = X_c \cup C_{uc}$.

La *SCD* a été utilisée dans de nombreuses applications, notamment pour synthétiser certaines propriétés temporelles pour la conception d'un robot (Altisen *et al.*, 2003), et pour résoudre les discordances entre les protocoles d'interaction (Roop *et al.*, 2009). Nous intégrons cette technique dans la méthode de conception développée, afin de corriger automatiquement les erreurs de conception préalablement détectées par la technique du "model checking".

5. Flot de conception à base de COTS

Comme mentionné ci-dessus, chaque COTS possède des hypothèses d'environnement locales A et des garanties locales G. La correction d'un COTS, considéré

séparément, s'exprime par la satisfaction des exigences contenues dans l'ensemble G, en supposant que les pré-conditions contenues dans l'ensemble A sont vraies.

Le fait d'assembler des COTS peut entraîner un comportement global incorrect de l'ensemble. Ceci arrive principalement à cause de contradictions entre les hypothèses (pré-conditions) d'un COTS et les garanties d'un autre COTS avec lequel il doit y avoir une intéraction. Par exemple, dans la figure 2, si G_2, l'ensemble des garanties de C_2, contredit A_1, l'ensemble des hypothèses de C_1, alors cela entraîne la non-satisfaction de certaines garanties G_2. Ainsi, des erreurs globales de conception apparaissent lors de l'assemblage des COTS, bien que le comportement local de chaque COTS soit sûr.

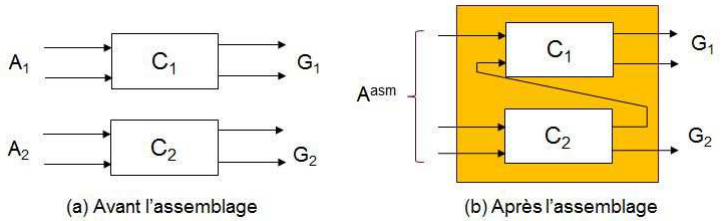

Figure 2. Assemblage de COTS

La conception d'un système critique par assemblage de COTS nécessite donc d'établir des garanties de correction. La figure 3 présente notre flot de conception. La méthode de conception proposée est une méthode semi-automatique ayant deux objectifs principaux. Le premier est de fournir une bibliothèque formelle de COTS qui contient non seulement le modèle exécutable du composant mais aussi une documentation des exigences du composant, exprimée formellement. Le deuxième objectif est de réaliser des systèmes de contrôle-commande matériels corrects par construction. La première étape de la méthode de conception est de sélectionner le(s) composant(s) nécessaire(s) pour la conception et formaliser (1) les hypothèses d'environnement, (2) les garanties du comportement et les nouvelles propriétés supplémentaires pouvant être exigées sur l'assemblage.

Deux ou plusieurs COTS peuvent être assemblés, selon la fonctionnalité ciblée. Cette tâche se traduit par une composition synchrone entre les machines à états finis sous-jacentes aux COTS assemblés. Dans la deuxième étape de la conception, l'assemblage des COTS est vérifié formellement par rapport aux spécifications formelles de chaque composant, afin de détecter d'éventuelles erreurs de conception dûes à l'assemblage. Si la vérification réussit, le nouvel assemblage vérifié sera ajoutée à la bibliothèque COTS en tant que nouveau composant réutilisable avec sa documentation formelle. Si la vérification échoue, l'outil de vérification fournit un contre-exemple qui illustre de quelle manière la propriété est violée. Typiquement, ce contre-exemple montre un ensemble de variables ayant un impact sur la non-satisfaction d'une spécification, ainsi que leurs séquences de valeurs à partir l'état initial et jusqu'à l'état ou la spécification est violée. Si l'erreur détectée est liée à une propriété de vivacité, le concepteur doit corriger cette erreur manuellement. Par contre, si l'erreur est liée à une

propriété de sûreté, elle peut être corrigée automatiquement, lors de l'étape suivante, basée sur la technique SCD.

Pour appliquer la SCD à un système matériel, il est nécessaire d'identifier un ensemble d'entrées contrôlables que le superviseur devra affecter. Dans le contexte de la conception matérielle, cette partition d'un vecteur d'entrées vis-à-vis de la contrôlabilité n'est pas une opération intuitive pour les concepteurs. Dans le cadre de notre méthode, nous proposons au concepteur d'utiliser le contre-exemple fourni par l'outil de vérification pour guider son choix d'un ensemble de variables contrôlables, en s'inspirant par la cause même de l'erreur. En effet, le contre-exemple montre un ensemble de variables d'entrée dont les valeurs sont responsables de la violation de la propriété. En choisissant parmi ces variables il est possible de construire un ensemble de variables candidates à être contrôlées.

Cette procédure de sélection est donnée par un algorithme de décision, que le concepteur applique de façon manuelle, car il nécessite une connaissance approfondie du système que l'on s'apprête à contrôler. Pour construire l'ensemble des variables contrôlables X_c, le concepteur doit sélectionner l'ensemble des variables d'entrée fourni par le contre-exemple, et supprimer certaines de ces variables qui ne peuvent pas être contrôlées. Certaines règles doivent être respectées lors de la suppression des ces variables de la liste des candidates :

- X_c initiale = l'ensemble total des variable fournit dans le contre-exemple.
- exclure de X_c toute variable désignant un capteur physique ;
- exclure de X_c toute variable qui représente une donnée à traiter ;
- exclure de X_c toute variable qui représente une alerte ;
- exclure de X_c toute variable d'état interne.

Ces règles peuvent être appliquées de manière systématique à condition de pouvoir identifier les types de signaux mentionnés ci-dessus.

Une fois l'ensemble X_c construit, l'application de la SCD permet de construire un superviseur qui renforce la satisfaction d'une/des spécifications en affectant les variables contrôlables choisies précédemment. Dans le cas où la méthode de synthèse ne trouve pas de solution, le concepteur conclut que la/les spécifications ne peuvent pas être satisfaites par ce système.

Parfois, l'outil de synthèse produit des superviseurs très restreints qui assurent ces spécifications en désactivant la plupart des comportements intéressants du système. Pour éviter de mettre dans la bibliothèque un composant (ou un système) trop restrictif, nous proposons de vérifier, après l'étape de synthèse, certaines propriétés clé exprimant la vivacité et prédéfinies par le concepteur, afin de s'assurer que le système conserve après synthèse le comportement désiré.

Il est également nécessaire de s'assurer de la *passivité* du superviseur. Un superviseur passif ne doit pas "inventer" des événements. Son rôle se limite à retarder ou empêcher des événements arrivant depuis l'environnement.

La dernière étape de notre méthode préconise une simulation classique, qui a pour but une validation dynamique et visuelle du système contrôlé. Cette étape est généralement requise par les concepteurs, car elle apporte un moyen supplémentaire, jugé incontournable, de validation visuelle du nouveau système, avant sa mise en œuvre sur une puce FPGA.

Dans la suite de cet article, cette méthode de conception est illustrée sur un exemple industriel.

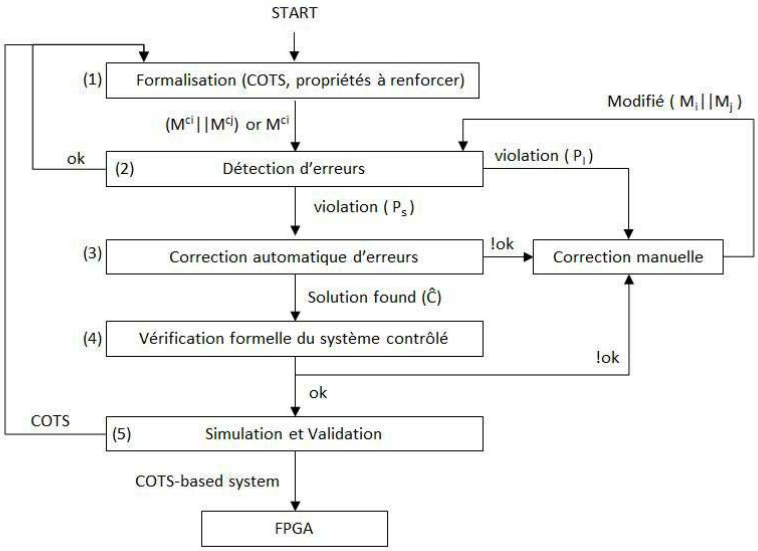

Figure 3. Méthode de conception de systèmes matériels embarqués

6. Système d'accès voyageur

Le déroulement de la méthode de conception proposée est illustré sur le sous-système (Porte/Emmarchement mobile) étudié dans le cadre du projet FerroCOTS (Bombardier, 2007). L'objectif de cette étude de cas est de montrer de quelle manière la synthèse de contrôleurs permet de générer automatiquement du code de contrôle-commande correct par construction, apportant un gain de temps et des garanties de correction, par rapport à un codage manuel. Le système est composé de trois composants COTS.

Le COTS gerer_ouverture_fermeture représenté dans la figure 4 se charge de la commande des vantaux, en fonction d'une demande d'ouverture/fermeture, et des valeurs fournies par les capteurs physiques.

D'autres contraintes dites opérationnelles, spécifiques au contexte d'utilisation du train, conduisent à la définition de nouveaux COTS. Par exemple, au moment de la fermeture, il faut attendre une seconde entre la demande de fermeture et la commande

des vantaux, délai correspondant à l'émission du signal sonore. Cette contrainte opérationnelle est mise en œuvre sous la forme d'un nouveau composant modélisant ce comportement d'attente. Le COTS SEQ_PORTE est illustré dans la figure 7.

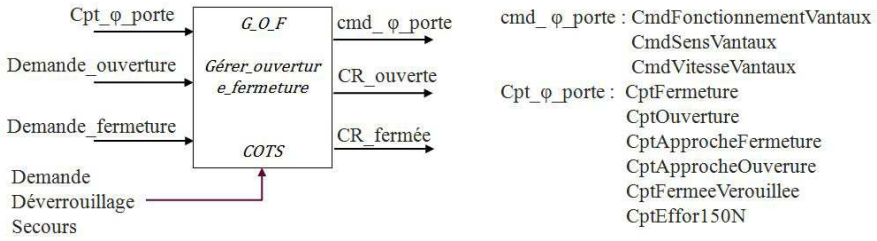

Figure 4. COTS gerer_ouverture_fermeture

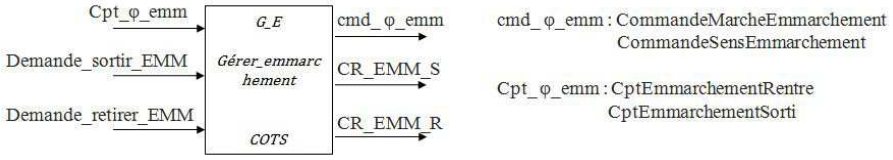

Figure 5. COTS gerer_emmarchement

Les fonctions de contrôle commande de l'emmarchement mobile disposent également de deux COTS élémentaires : gérer_emmarchement et SEQ_EMM. Le composant gérer_emmarchement active l'emmarchement mobile et renvoie des signaux de compte rendu de fin d'opération : emmarchement mobile sorti CR_EMM_S et emmarchement moblie rentré CR_EMM_R. Le COTS "Autorisation ouverture" présenté dans la figure 6 répond à une contrainte fonctionnelle préalable à l'action d'ouverture. Il commande l'ouverture ou de fermeture des portes selon des demandes émises à partir du pupitre de contrôle du train. Le comportement de ce COTS est purement combinatoire :

$$\begin{aligned} Ouvrir :=& (DemandeOuvExt \vee demandeOuvInt) \wedge \\ & LT\_AutorisationOuverture \wedge \neg LT\_DemandeFermeture; \\ Fermer :=& LT\_DemandeFermeture; \end{aligned}$$

On note que la distinction entre contraintes fonctionnelles et opérationnelles relève d'un partitionnement adapté lors de l'élaboration du cahier des charges. Dans le cadre de notre travail, cette distinction a été faite par une étude réalisée en amont.

6.1. Assemblage des COTS

L'assemblage des composants (originaux) est illustré dans la figure 7. L'objectif de cet assemblage est de réaliser une fonction d'accès des voyageurs au train, permettant

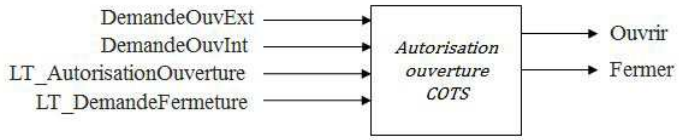

Figure 6. Le COTS Autorisation d'ouverture des portes

de garantir des conditions de sécurité stipulées dans le cahier de charges : les portes ne doivent pas s'ouvrir si l'emmarchement mobile n'est pas déployé. De manière duale, le retrait de l'emmarchement mobile ne peut se faire que si les portes sont fermées et verrouillées.

L'assemblage obtenu ne permet pas, dans l'état, de satisfaire ces exigences sécuritaires. En effet, il n'existe aucune synergie entre les fonctions "porte" et "emmarchement mobile", qui n'ont pas été conçues pour interagir. Cette synergie est établie grâce à la SCD, qui génère un superviseur renforçant cette exigence de sécurité. L'assemblage des composants contrôlé est illustré dans la figure 8.

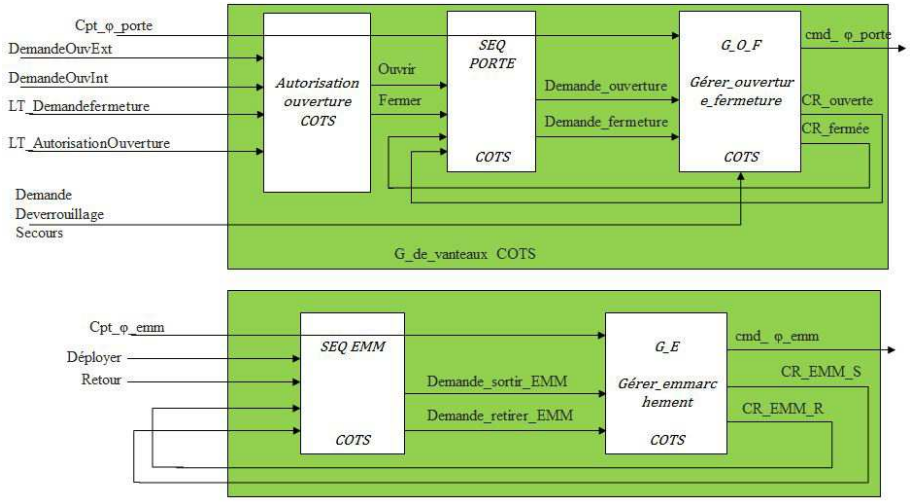

Figure 7. Assemblage des COTS originaux

6.1.1. Les propriétés d'assemblage

Pour cet assemblage particulier, nous nous focalisons sur le renforcement des deux exigences spécifiques à cet assemblage. Elles sont exprimées dans la logique temporelle LTL, en utilisant les opérateurs G ($always$) et F ($eventually$). L'operateur *before* est une abréviation permettant d'exprimer une contrainte de précédence. Il est défini de la manière suivante :

$$a \text{ before } b = \neg b \text{ until } (a \wedge \neg b)$$

L'expression des deux exigences est la suivante :

1. Hors situation d'urgence, la porte de ne doit pas s'ouvrir avant déploiement complet de l'emmarchement mobile :

$$prop1 : G(\neg DemandeDeverrouillageSecours \rightarrow (CR\_EmmSorti \text{ before } CR\_ouverte))$$

2. L'emmarchement mobile ne se retire pas avant la fermeture complète de la porte :

$$prop2 : G(CR\_ferme \text{ before } CR\_EmmRentre)$$

L'étape de la vérification formelle montre que les propriétés ci-dessus ne sont pas respectées. Ceci était prévisible, car aucune coordination n'existe initialement entre les fonctions porte et emmarchement mobile. Cette coordination sera assurée par synthèse de contrôleurs, en renforçant les propriétés $prop1$ et $prop2$.

6.1.2. Contrôlabilité de l'assemblage

Les propriétés $prop1$ et $prop2$ sont renforcées en contrôlant certaines variables d'entrée de l'assemblage. Le choix des variables d'entrée à contrôler est guidé par l'examen du contre-exemple fourni par l'outil de vérification formelle, en respectant les règles de sélection proposées ci-dessus, applicables à cet exemple : tout signal en provenance d'un capteur physique sera considéré comme incontrôlable. Ainsi, le choix de la contrôlabilité s'est porté sur les signaux entrants suivants : $DemandeOuvertureExt$, $DemandeOuvertureInt$, $LT\_DemandeFermeture$, $Dployer$, $Retirer$.

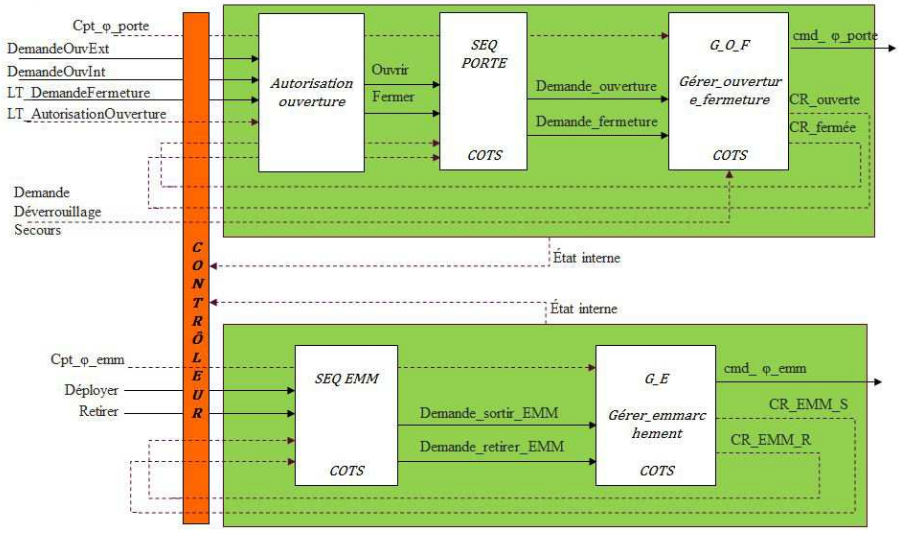

Figure 8. Assemblage des COTS avec le contrôleur

Le comportement du superviseur est décrit de la manière suivante. Lorsque les portes sont fermées et que l'on reçoit un signal d'ouverture, ce signal sera retardé jusqu'à l'arrivée du signal CR_EMM_S qui confirme que l'emmarchement mobile est déployé. Ceci renforce la satisfaction de $prop1$. Lorsque l'emmarchement mobile est déployé et que l'on reçoit la demande de le retirer, ce signal sera retardé jusqu'à l'arrivée du signal CR_fermée qui confirme le fait que la porte est fermée et verrouillée. Ceci renforce la satisfaction de $prop2$.

6.1.3. Vivacité de l'assemblage

Le besoin de vérifier formellement la correction de l'assemblage de COTS contrôlé est lié à la possibilité de perdre certaines garanties locales à travers l'opération d'assemblage/synthèse. Par ailleurs, l'ajout d'un superviseur à l'assemblage peut mener à des situations de blocage, dans lesquelles les exigences ciblées pour la synthèse se retrouvent satisfaites de manière triviale, par un système devenu immobile et/ou inopérant. C'est pourquoi il est nécessaire de s'assurer que l'assemblage contrôlé reste réactif par rapport à son environnement. Par exemple, étant donné que le signal $DmdOuvertureExt$ est contrôlé par le superviseur, il faut prouver qu'à chaque fois qu'un passager demande une ouverture de porte, si les conditions sont réunies, cela aboutira toujours à une ouverture effective des portes. De la même manière, on doit confirmer la conservation de cette réactivité vis-à-vis des autres signaux contrôlés. Cela nécessite la vérification des propriétés suivantes :
$prop\_asm1 : G((DemandeOuvInt \vee DemandeOuvExt) \rightarrow F\ CR\_ouverte)$
$prop\_asm2 : G(Dployer \rightarrow F\ CR\_Emm\_sorti)$
$prop\_asm3 : G(LT\_DemandeFermeture \rightarrow F\ CR\_Ferme)$
$prop\_asm4\ :\ G(retirer \rightarrow F\ CR\_Emm_rentr)$

A noter que la satisfaction de ces propriétés repose sur le maintien des valeurs des demandes entrantes jusqu'à ce qu'elles soient satisfaites (condition suffisante).

6.1.4. Passivité du superviseur

Pour vérifier la passivité du superviseur, il est nécessaire de s'assurer que, tout en garantissant les propriétés de l'assemblage $prop1$ et $prop2$, celui-ci n'effectue pas d'actions non désirées. Dans le cas présent, une telle action correspond à l'activation d'une variable contrôlable. Pour cela on vérifie systématiquement que toute variable contrôlable ne prend pas la valeur 1 si la valeur qu'elle reçoit de l'environnement est égale à 0 :

$$passif : G\neg(x_c \wedge \neg x_c^{env})$$

Par exemple, on veut s'assurer que le contrôleur ne commandera jamais l'ouverture d'une porte, si cette demande n'a pas été préalablement faite par l'environnement (passager). Il en va de même avec l'emmarchement mobile. Ces exigences se traduisent par les propriétés suivantes :

$prop\_asm5\ :\ G\neg(\neg DemandeOuvertureExt \wedge o\_DemandeOuvertureExt)$
$prop\_asm6 : G\neg(\neg DemandeOuvertureInt \wedge o\_DemandeOuvertureInt)$

$prop\_asm7 \ : \ G\neg(\neg Deployer \wedge o\_Deployer)$
$prop\_asm8 \ : \ G\neg(\neg Retirer \wedge o\_Retirer)$
$prop\_asm9 \ : \ G\neg(\neg LT\_DemandeFermeture \wedge o\_LT\_DemandeFermeture)$

Ces propriétés ont été formellement vérifiées et sont respectées indépendamment de l'environnement physique de l'assemblage.

6.1.5. Comparaison : assemblage contrôlé par synthèse/assemblage initial

Contrairement à un processus de conception manuel, qui n'apporte pas de garanties de correction, la conception par assemblage de COTS intégrant la SCD renforce formellement les propriétés désirées.

A noter cependant que la solution contrôlée présente deux signaux entrants supplémentaires : les demandes de déploiement/retrait de l'emmarchement mobile peuvent être pilotées par l'environnement dans le cas où le contrôleur est non-déterministe. Un choix possible (et trivial) de mise en œuvre serait d'activer simultanément les demandes d'ouverture/déploiement d'une part, et de fermeture/retrait d'autre part, le superviseur garantissant la bonne séquence de ces actions. Par ailleurs, la présence du superviseur permet de demander l'ouverture et le déploiement dans n'importe quel ordre, avec la garantie d'une séquence correcte.

6.1.6. Simulation

L'étape de simulation permet de visualiser le bon fonctionnement de l'assemblage contrôlé. Elle a été réalisée en modélisant manuellement le comportement des capteurs physiques et de la partie opérative en général. La figure 9 représente en partie haute l'ensemble des valeurs des capteurs physiques. La simulation ne représente que la demande d'ouverture de l'intérieur de la cabine, le signal de demande extérieure ayant une fonction identique. En partie basse du chronogramme, on représente les signaux de commande de la partie opérative : vantaux, sens d'ouverture et vitesse, ainsi que l'emmarchement mobile. Les signaux clk et rst représentent respectivement l'horloge matérielle et l'initialisation du système, lors de sa mise sous tension.

6.2. Implémentation des COTS sous ControlBuild

Un COTS ControlBuild est un composant de contrôle-commande construit au sein de la plateforme ControlBuild (GeenSoft, 2010). Son modèle de comportement est créé en utilisant des langages standardisés (IEC 61131-3 : Grafcet, Schémas à relais, Logigrammes, Texte structuré). La mise en œuvre finale de ces COTS se fait sur une puce FPGA.

Le flot d'élaboration et de conception est présenté dans la figure 10. Il repose sur un ensemble d'outils, certains libres, d'autres commerciaux. Certaines étapes, bien qu'automatisables, font, dans l'état d'avancement actuel, l'objet de traitements manuels. L'outil HDLGEN permet de traduire les modèles ControlBuild en VHDL synthétisable. A ce stade, le compilateur RTL (Synomsys) permet d'extraire à partir d'un

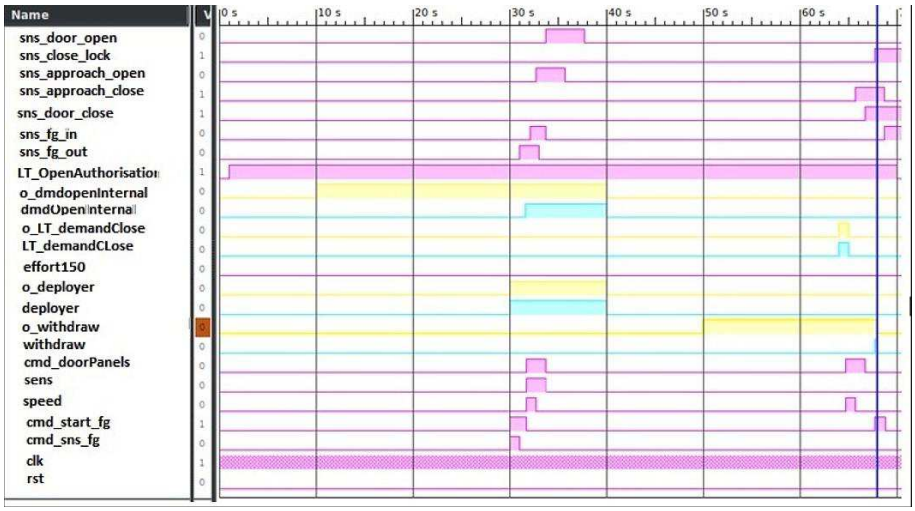

Figure 9. Simulation du système contrôlé

programme VHDL un modèle FSM exploitable pour la SDC et le “model checking”. Ainsi, quelque soit le style d’écriture du développeur (logigramme, Grafcet, etc.), on obtient grâce à la synthèse RTL une représentation mise à plat sous forme de réseau de portes logiques génériques.

L’outil dc2z3z, développé par Ampère, transforme ce résultat dans le langage d’entrée des outils de synthèse et de vérification.

Dans le cadre du démonstrateur, la synthèse de contrôleurs discrets et la vérification formelle sont réalisées grâce à des outils universitaires, libres d’accès : Sigali pour la synthèse, et Cadence SMV (ou alternativement, NUSMV) pour la vérification formelle.

Le superviseur synthétisé est assemblé manuellement avec le reste de l’assemblage de COTS. Il peut être retraduit en Controlbuild grâce à l’outil cont2comp (développé par Ampère) en vue d’une simulation globale.

Dans l’état actuel du démonstrateur, l’exigence fonctionnelle à vérifier formellement ou à synthétiser est associée manuellement, par le concepteur, au moment de la vérification ou de la synthèse. A moyen terme, cette association peut se faire directement dans le code ControlBuild du composant, par le biais des sections pré- et post-condition d’un composant. Cet aspect est purement technique et ne pénalise pas la pertinence de la chaîne d’outils mise en place.

7. Conclusion

Dans cet article nous avons présenté une méthode de conception à base de COTS ControlBuild, intégrant la Synthèse de contrôleurs discrets et la vérification formelle

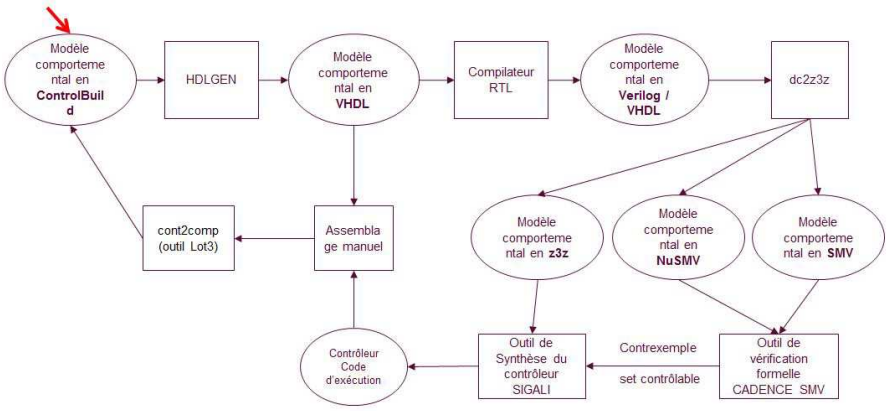

Figure 10. Chaîne d'outils de conception

au sein de la même démarche. Les résultats obtenus permettent actuellement de générer un modèle VHDL d'un assemblage de COTS contrôlé par synthèse, compatible avec la norme VHDL pour la synthèse (matérielle) RTL, et donc prêt à être implémenté sur une puce FPGA. Les perspectives de recherche liées à ce travail concernent l'automatisation du choix des signaux contrôlables, afin de fournir aux concepteurs un cadre entièrement automatisé d'utilisation de la SCD.

Bibliographie

Altisen K., Clodic A., Maraninchi F., Rutten E. (2003). Using controller-synthesis techniques to build property-enforcing layers. In *Proceedings of the 12th european conference on programming*, p. 174–188. Warsaw, Poland, Springer-Verlag. Consulté sur http://portal.acm.org/citation.cfm?id=1765712.1765727 (ACM ID: 1765727)

Besnard L., Marchand H., Rutten E. (2006, july). The sigali tool box environment. In *Discrete event systems, 2006 8th international workshop on*, p. 465–466.

Bombardier. (2007). *http://eurailmag.com/from-cable-to-chip-ferrocots-takes-command-control/ ferrocots.* Consulté sur http://eurailmag.com/from-cable-to-chip-ferrocots-takes-command-control/

Clarke E. M. (2008a). 25 years of model checking. In O. Grumberg, H. Veith (Eds.),, p. 1–26. Springer-Verlag. Consulté sur http://dx.doi.org/10.1007/978-3-540-69850-0_1 (ACM ID: 1423536)

Clarke E. M. (2008b). 25 years of model checking. In O. Grumberg, H. Veith (Eds.),, p. 1–26. Berlin, Heidelberg, Springer-Verlag. Consulté sur http://dx.doi.org/10.1007/978-3-540-69850-0_1

GeenSoft. (2010). *Design, simulate & deploy automation & embedded control systems with higher efficiency.* Consulté sur http://www.geensoft.com/en/article/controlbuild

Hajjar S., Dumitrescu E., Niel E. (2012, Februray). A component-based safe design method for train control systems. In *Embedded real time software and systems erts.*

Hajjar S., Dumitrescu E., Niel E. (2013, July). Safe design method of embedded control systems : case study. In *5 èmes journées doctorales / journées nationales macs, jd-jn-macs 2013,*.

Lamport L. (1983). What good is temporal logic. *Information processing*, vol. 83, p. 657–668. Consulté sur http://research.microsoft.com/en-us/um/people/lamport/pubs/what-good.pdf

Laurent O., Michel P., Wiels V. (2001). Using formal verification techniques to reduce simulation and test effort. In *Proceedings of the international symposium of formal methods europe on formal methods for increasing software productivity*, p. 465–477. Springer-Verlag. Consulté sur http://portal.acm.org/citation.cfm?id=647540.730020 (ACM ID: 730020)

Marchand H., Bournai P., Borgne M. L., Guernic P. L. (2000, octobre). Synthesis of Discrete-Event controllers based on the SignalEnvironment. *Discrete Event Dynamic Systems*, vol. 10, n° 4, p. 325–346. Consulté sur http://portal.acm.org/citation.cfm?id=593615.593670 (ACM ID: 593670)

McMillan K. L. (1992). *Symbolic model checking: an approach to the state explosion problem*. Thèse de doctorat non publiée, Pittsburgh, PA, USA. (UMI Order No. GAX92-24209)

Ramadge P., Wonham W. (1989, janvier). The control of discrete event systems. *Proceedings of the IEEE*, vol. 77, n° 1, p. 81 –98.

Roop P., Girault A., Sinha R., Goessler G. (2009). Specification enforcing refinement for convertibility verification. In *Proceedings of the 2009 ninth international conference on application of concurrency to system design*, p. 148–157. IEEE Computer Society. Consulté sur http://dx.doi.org/10.1109/ACSD.2009.25 (ACM ID: 1673101)