Projet ANR VACSIM

Validation de la commande des systèmes critiques par couplage simulation et méthodes d'analyse formelle

Première réunion plénière à Bordeaux

15 et 16 mars 2012

AGENCE NATIONALE DE LA RECHERCHE
ANR

LURPA

EDF

INRIA
RENNES - BRETAGNE ATLANTIQUE

i3S
sophia antipolis

LaBRI

EDF
CHANGER L'ÉNERGIE ENSEMBLE

Le projet ANR VACSIM (2011 – 2014)

- Fait suite au projet ANR TESTEC (TEst des Systèmes Temps réel Embarqués Critiques – 07 TLOG 022)
- VACSIM : Validation de la commande des systèmes critiques par couplage simulation et méthodes d'analyse formelle
- Tirer profit des avantages respectifs des techniques de simulation, en incluant des modèles des processus commandés, et des méthodes d'analyse formelles, pour la validation de la commande des systèmes critiques
- 6 Tâches + 1 Démonstrateur et le traitement de cas industriels:
  - Validation par test progressif par parties de systèmes logiques
  - Validation par simulation de partie opérative
  - Apport des techniques d'identification des systèmes à événements discrets à la validation
  - Validation formelle de propriétés quantitatives : approche par automates
  - Validation formelle de propriétés quantitatives : approche par contraintes

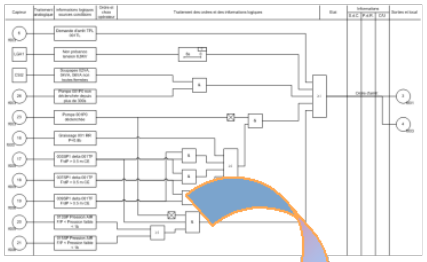
Tâche 1 : Validation par test progressif par parties de systèmes logiques

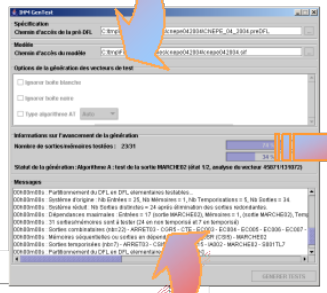
- Le projet ANR TESTEC a montré l'intérêt de l'utilisation des techniques de vérification pour réduire la taille des tests (TEst des Systèmes Temps réel Embarqués Critiques – 07 TLOG 022)
  - Application aux logiques de sécurité industrielles ou de sécurité de machines
  - Livrable L5.1 « Algorithmes de génération et d'exécution du test des systèmes logiques, séquentiels et temporisés non bouclés »
  - Livrable L6.2 « Etudes de cas industriels : mise en oeuvre du démonstrateur sur des applications industrielles, synthèse des études, préconisations de mise en œuvre et d'extensions »
  - Transformation du prototype TESTMINATOR d'EDF R&D en un logiciel de la suite ControlBuild de Geensoft / Dassault Systèmes (bénéficiant de plus d'une possibilité d'animation des spécifications logicielles par les cas de test générés)

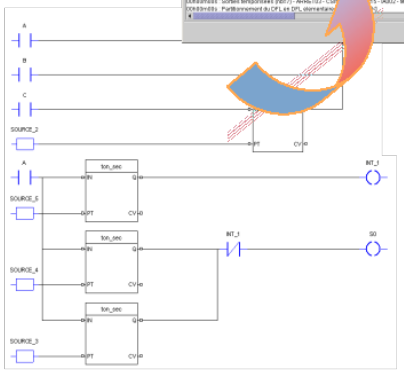
- Techniques utilisables pour des systèmes logiques critiques qui peuvent réaliser plusieurs milliers de fonctions simples utilisant typiquement de quelques entrées à une quinzaine d'entrées

Tâche 1 : Validation par test progressif par parties de systèmes logiques - Testminator

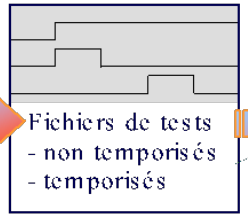
La spécification est traduite dans un formalisme non ambiguë.



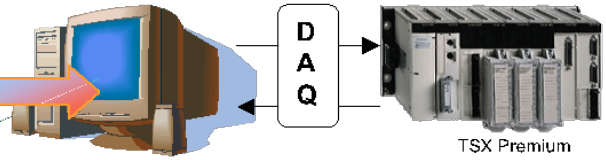




La réalisation est analysée pour vérifier un ensemble minimal d'hypothèses



L'outil génère un oracle de test en utilisant un ensemble de techniques de réduction permettant de maîtriser l'explosion combinatoire



Le jeu de test est exécuté :

- En simulation
- Ou directement implémenté sur la cible

Tâche 1 : Validation par test progressif par parties de systèmes logiques

- Problème de taille des tests pour plus d'une quinzaine d'entrées
  - Exemples de fonctions logiques complexes qui utilisent plus de quarante ou cinquante entrées: Synthèse d'alarmes de surveillance générale, Arrêt automatique général à une unité de production…
- Le projet TESTEC a été l'occasion de formaliser un algorithme de génération de tests et de vérifications minimales pour ces cas
  - Utilise en plus l'existence de sorties intermédiaires entre les sorties à tester et leurs entrées
  - Validation par un test progressif par parties (TPPP)
  - Le TPPP s'inspire d'une bonne pratique manuelle du test des systèmes hyper-critiques
- Le temps imparti au projet TESTEC n'a cependant pas permis de prototyper ces nouveaux algorithmes et de les mettre en œuvre sur des cas industriels
- Objectif de la tâche 1 : développer un prototype, intégré dans l'environnement ControlBuild, réalisant le test progressif par parties de systèmes logiques non-bouclés et le mettre en œuvre sur des cas industriels

Tâche 1 : Validation par test progressif par parties de systèmes logiques

- Durée : 36 mois (de T0 à T0+36)
- Responsable : EDF R&D
- Partenaires impliqués : Dassault Systèmes, I3S, LURPA

Tâche 1 : Validation par test progressif par parties de systèmes logiques

- Les livrables :

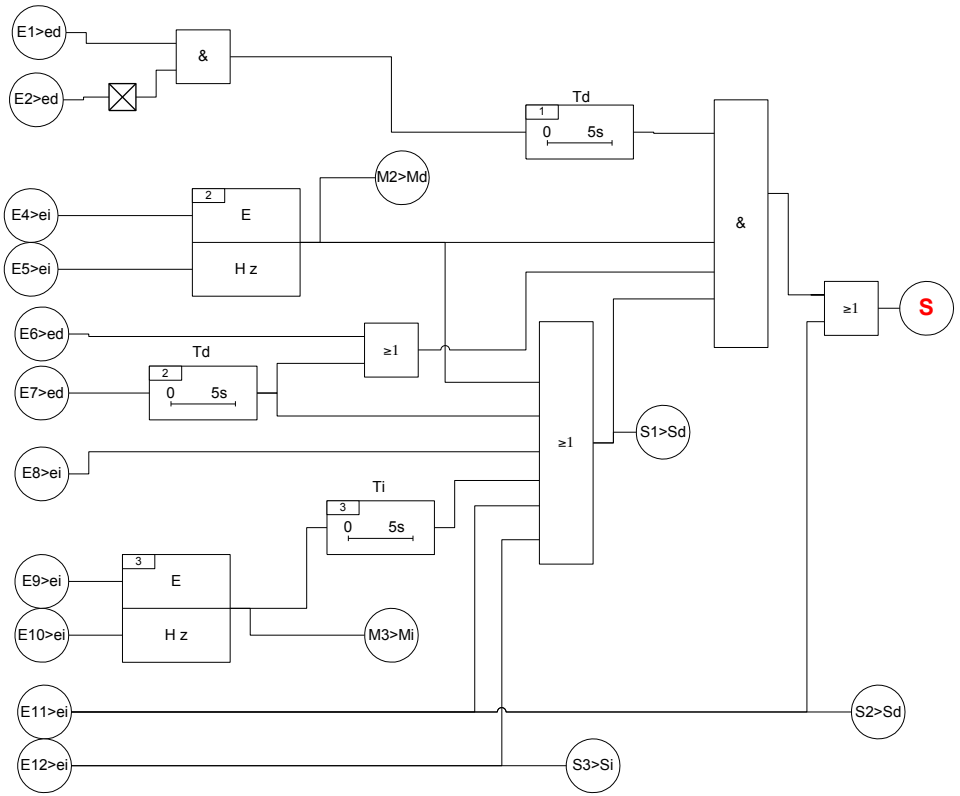
- L1.1 : Spécification et validation sur études de cas d'un algorithme de test progressif par parties de systèmes logiques non-bouclés (T0+12) – Document

- L1.2 : Développement d'un algorithme de test progressif par parties de systèmes logiques non-bouclés dans l'environnement ControlBuild (T0+24) – Logiciel

- L1.3 : Evaluation sur études de cas industriels d'un algorithme de test progressif par parties de systèmes logiques non-bouclés dans l'environnement ControlBuild (T0+36) - Document

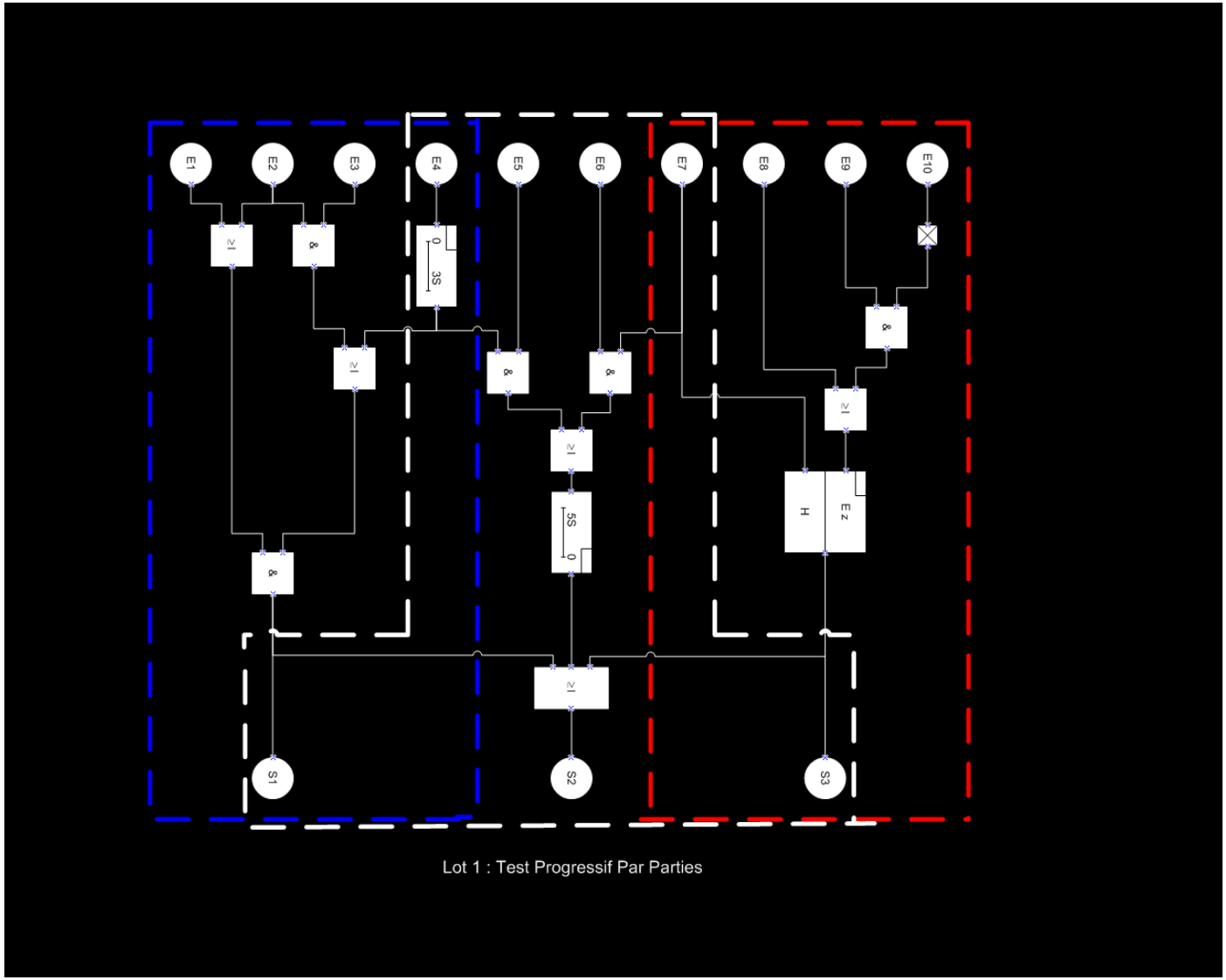
Tâche 1 : Validation par test progressif par parties de systèmes logiques

- Exemple:



- Vérifications :
  - Pour le test et la vérification de S : S = f(T1(E1, E2), M2, E6, T2(E7), S1, S2)
  - Pour le test et la vérification de M2 : M2 = f(M(E4, E5))
  - Pour le test et la vérification de S1 : S1 = f(M2, T2(E7), E8, T3(M3), S2, S3)
  - Pour le test et la vérification de M3 : M3 = f(M(E9, E10))
  - Pour le test et la vérification de S2 : S2 = f(E11)
  - Pour le test et la vérification de S3 : S3 = f(E12)

Tâche 1 : Validation par test progressif par parties de systèmes logiques



Tâche 1 : Validation par test progressif par parties de systèmes logiques

- Tests :
  - 1ère étape : Algorithme BTP
    - Détermination de l'ensemble des états atteignables du système (temporisations, mémoires, sorties intermédiaires en amont de la sortie S à tester), des séquences de vecteurs d'entrée, et des temps d'atteinte associés
  - 2ème étape : Positionnement du système dans un état E
  - 3ème étape : Jeu d'un vecteur candidat $V_{candidat}(e_i, e_d)$
    - L'état E évolue suivant $(E, E^1, \ldots, E^n)$ avant stabilisation
    - Le sous-état E_p des sorties, mémoires et temporisations directes évolue suivant $(E_p, E_p^1, \ldots, E_p^n)$
  - 4ème étape : Critère de sélection des vecteurs de test
    - Si $(E_p, e_d, E_p^1, \ldots, E_p^n)$ déjà connu et que la séquence $(E_p, E_p^1, \ldots, E_p^n)$ est plus longue que celle déjà sélectionnée, on ne retient pas cette séquence
    - Si $(E_p, e_d, E_p^1, \ldots, E_p^n)$ n'est pas déjà connu ou que la séquence $(E_p, E_p^1, \ldots, E_p^n)$ est déjà connue et est plus courte que celle déjà sélectionnée, on la retient en écrasant celle déjà sélectionnée si elle existe
  - 5ème étape : Optimisation de l'enchainement des vecteurs de test
    - Enchainement des vecteurs selon une séquence de type RII…IISII…IIRSRSRS…
    - Utilisation de l'état atteint lors du test comme état initial pour le test suivant