Langages synchrones

Emmanuelle Encrenaz
(emmanuelle.encrenaz@lip6.fr)

Master d'Informatique
Spécialité SAR
Langages Synchrones

Sept . 2018

Contenu du cours et Planning

- **Introduction aux langages synchrones**
  - Systèmes réactifs
  - Hypothèses synchrones
  - Modèle sous-jacent
- **Deux langages synchrones**
  - Flot de données : Lustre
  - Automates concurrents : Esterel
- **Analyse statique et Compilation**
  - Causalité, calcul d'horloges
  - Code séquentiel, Compilation en circuit
- **Vérification de propriétés de sûreté**
  - Observateur

- **Lien vers les TP:**
  http://www-soc.lip6.fr/~ema/enseignement/
- **Cours: UPMC, salle SAR : 14-15 508**

Date		
13/02	C1	**Introduction Syst. réactifs, modèle sous-jacent, programmes Lustre**
20/02	C2	**Compilation en automate et en circuit**
21/02	TD/TP 1	Ecriture de programmes Lustre, simulation et vérification
27/02	C3	**Lustre : vérification par observateurs.**
28/02	TD/TP 2	Différentes représentations d'un noeud

1. Introduction aux langages synchrones

a) Ingénierie des systèmes critiques
b) Définition de l'approche synchrone
   a) Système réactif synchrone
   b) Modèle sous forme d'automate
c) Différents langages et utilisation industrielle

Master d'Informatique
Spécialité SAR
Langages Synchrones

Syst. temps-réel, embarqués, critiques

- **Temps-réel**
  - Temps de chaque traitement connu ou borné dans un intervalle
  - Temps réel dur : contraintes temporelles à respecter impérativement
- **Embarqué**
  - Dispositif logiciel + matériel « portable » (automobile, avion, humain)
  - Consommation, poids, surchauffe sont des contraintes importantes
- **Critique**
  - Défaillance d'une partie du système a des conséquences importantes (humaines, environnementales, sociétales, économiques, …)

Trois domaines fortement liés:

- Temps-réel et embarqué: horloges, alarmes, ..
- Embarqué et critique: systèmes de pilotage, robots chirurgiens, etc.

- **Objectifs de ce cours: étudier l'approche synchrone**

Systèmes, Domaines et Contraintes

Contraintes de conception d'un système dépendent :

- du domaine d'applications (vocabulaire, techniques, normes)
- de types de contraintes à respecter :
  - *fiabilité, maintenabilité, sécurité, intégrité, testabilité*

Domaine d'applications	Applications	Criticité	Types de contraintes	Respect des Contraintes
Avionique	Pilotage, frein, distribution électrique	Safety-critical	Fiabilité, déterminisme	Certifications
Spatial	Contrôle de trajectoires, altitude, imagerie, transmission	Mission-critical	Fiabilité, intégrité, déterminisme	Qualité de code

Particularités des systèmes critiques

- **Fontionnement permanent**
  - Durée de vie : mois / années / décennies
- **Interactions fortes**
  - Avec l'environnement : Capteurs, horloges, utilisateur, environnement physique, … [variables discrètes ou continues]
  - Avec d'autres entités du système : Un élément d'une chaîne: centrale inertielle d'un lanceur Ariane V
- **Contraintes de performances**
  - Temps d'exécution, consommation, poids …
- **Robustesse / résilience aux erreurs Nécessaire gestion des entrées erronées**
  - Entrées-sorties erronées
  - Erreurs dues à l'environnement; bit-flip du aux radiations solaires
- **Recours à des méthodes de conception spécifiquement adaptées**

Environnements de développement

- **Circuits**
  - IPs : blocs réutilisables, programmés en HDL (Hardware Description Language)
    - *Ex: Verilog, VHDL, SystemC*
  - Connectés par bus et réseaux (eux même des IP)
  - Standardisation et réutilisation => SoC, System on Chip
- **Logiciels**
  - C/C++, Java, Ada,
  - Système d'exploitation, exécutif assure la coordination
  - Standardisation et réutilisation => bibliothèques
  - Modèles à base de composants et environnement de déploiement [Fractal, BIP]
- **Logiciel/matériel**
  - Plate-forme de conception mixte logiciel/matériel :
    - *Modèles d'architecture multiprocesseurs + noyau système + code applicatif*
    - *UML-Marte / SysML, AADL, Dysident, plateforme SOCLIB (Lip6)*

Niveaux de certification

Critères communs pour la sécurité des technologies de l'information (ISO/CEI 15408)

Niveaux d'évaluation (Evaluation Assurance Level) :

EAL1 : testé fonctionnellement
EAL2 : testé structurellement
EAL3 : testé et vérifié méthodiquement
EAL4 : conçu, testé et vérifié méthodiquement
EAL5 : conçu de façon semi-formelle et testé
EAL6 : conception vérifiée de façon semi-formelle et testée
EAL7 : conception vérifiée de façon formelle et testée

Niveaux de certification

Normes de conception pour l'aéronautique

Development Assurance Level « Un défaut du système ou sous-système étudié peut provoquer un problème …»

DAL-A	catastrophique - Sécurité du vol ou atterrissage compromis - Crash de l'avion
DAL-B	majeur entraînant des dégâts sérieux voire la mort de quelques occupants
DAL-C	sérieux entraînant un dysfonctionnement des équipements vitaux de l'appareil
DAL-D	pouvant perturber la sécurité du vol
DAL-E	sans effet sur la sécurité du vol

Niveaux de certification

Normes de conception pour l'aéronautique

DO-178 B & C (1992,2012) *Software considerations in airborne systems and equipment certification*

DO-254 (2000) *Design assurance guidance for airbone electronic hardware*

DO-331 *Model-Based Development and Verification Supplement to DO-178C and DO-278* - addressing Model-Based Development (MBD) and verification and the ability to use modeling techniques to improve development and verification while avoiding pitfalls inherent in some modeling methods

DO-333 *Formal Methods Supplement to DO-178C and DO-278A* – addressing formal methods to complement (but not replace) testing

Réduire les erreurs de conception

- **Améliorer les procédures de test**
  - Coût prohibitif, pouvant atteindre 60% du budget du projet
- **Améliorer les techniques de conception, adaptées au niveau de certification visé**
  - Spécification des besoins
  - Identification d'une architecture matérielle / logicielle
  - Spécification fonctionnelle des différentes entités + contraintes spécifiques (non fonctionnelles : temps, consommation, dimensionnement)
    - *Cohérence et complétude de la spécification : fiable et réutilisable*
    - *Modèle de calcul simple, apte à l'analyse*
  - Réduire la distance architecte/concepteur/utilisateur, matériel/logiciel
    - *Intégration matériel/logiciel dans un même environnement, avec des modèles formels communs ou compatibles*
- **Besoin d'outils pour :**
  - Spécifier : Modèles de haut niveau, spécifications formelles
  - Vérifier / Prouver : Vérification automatique ou guidée de propriétés
  - Compiler du code vérifié : Bibliothèques, compilateurs certifiés

Positionnement des langages synchrones

Les langages synchrones permettent de

- ***décrire, simuler, vérifier*** **des systèmes réactifs,**

et de

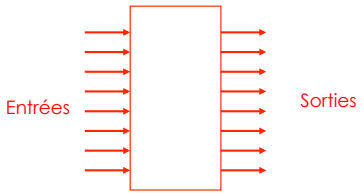
- ***compiler*** **du code ou du matériel garantissant le même comportement que le système décrit, si l'hypothèse synchrone est vérifiée**

2. Définition de l'approche synchrone

Master d'Informatique
Spécialité SAR
Langages Synchrones

Système réactif

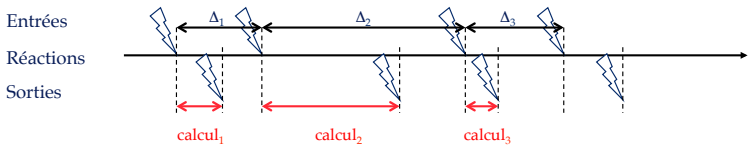
- **Un système à fonctionnement permanent, qui réagit à des sollicitations de son environnement :**
  - Programme sensible à des événements (en entrée) et produisant des événements (en sortie).
  - Cycle d'exécution :
    - *Attend un événement extérieur*
    - *Déclenche un calcul*
    - *Produit un résultat qu'il transmet à l'environnement*
  - Peut être constitué de différents éléments évoluant en parallèle



- **Une partie des systèmes temps-réels**
  - Interface interactive (gestion interruptions, conversion E/S physiques ↔ logiques)
  - *Noyau réactif* (sélection réaction suite à un événement logique)
  - Manipulation de données (transformations, calculs pilotés par le noyau réactif)

Système réactif

- **Vue temporelle de l'exécution du système**



- **Le calcul dépend :**
  - De l'événement déclencheur courant (de la configuration d'entrée courante)
  - De l'historique du système (de la séquence d'événements antérieurs)

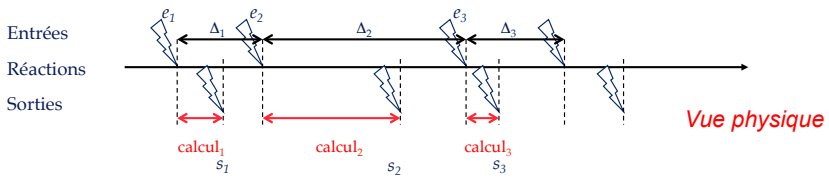
Système synchrone

- **Système réactif muni des propriétés suivantes :**
  - Le calcul est borné statiquement en temps et en mémoire
    - *Pas de structures de données dynamiques dont la taille dépend de l'entrée ou du pas de calcul*
    - *Pas de récursion*
  - Le calcul est déterministe
    - *Pas d'ordonnancement indéterministe interne*
  - Une étape de calcul ne peut pas être interrompue par la survenue d'un nouvel événement en entrée
    - *Borne temporelle de chaque calcul → le calcul le plus long fixe la fréquence maximale de sollicitation de l'environnement*

- **Impératif supplémentaire dans le contexte des systèmes critiques**
  - Sûreté de fonctionnement : le système ne peut pas présenter de « mauvais comportements »

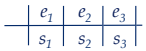
hypothèse synchrone

- **Introduction d'un *temps logique*, décomptant les événements physiques auxquels le système réagit (mais pas leur instant d'occurrence)**
- **Le calcul induit par un événement d'entrée e_i se produit en temps logique nul et produit l'événement de sortie s_i.**



Vue physique

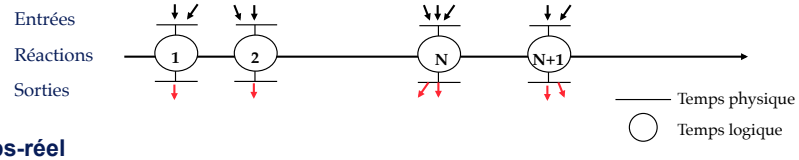
- **Une réaction = un couple (e_i,s_i)**
- **Une exécution : une liste de couples**



Vue logique

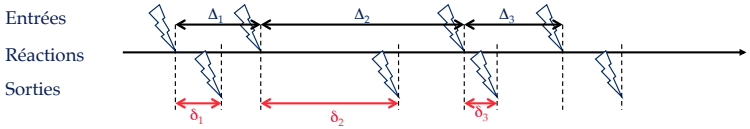
Fonctionnalité et temps-réel

- **Fonctionnalité**
  - En fonction d'une séquence d'entrées donnée, le programme calcule les bonnes séquences de sorties.



- **Temps-réel**
  - Le programme produit les sorties au bon moment (ni trop tôt, ni trop tard).
  - Échelles de temps très variables (microsecondes / secondes)

  → abstraction : temps de calcul < temps de réaction de l'environnement



Déterminisme

- **Sources de non-déterminisme**
  - Implantation multi-tâche : ordonnancement dynamique / préemptions

  → imprévisibilité des temps de réponse : Le non-déterminisme nuit au respect des contraintes temps-réel.
- **Déterminisme : Pour un état initial donné, une séquence d'entrées donnée produit toujours la même séquence de sorties**
  - Conséquence : la sortie du système est complètement déterminée par la séquence d'entrées et l'état initial

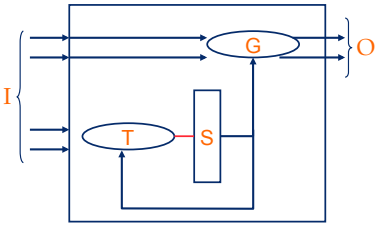
    $\forall$ i : $O_i = \phi(I_1, I_2, \ldots I_i, S_0)$
  - Contrainte supplémentaire : mémoire bornée S représentant l'historique du système

    $\forall S_0, T,\ O_i = G(S_i, I_i)$ *et* $S_{i+1} = T(S_i, I_i)$

Modèle d'un composant réactif synchrone

- **Modèle sous-jacent d'un système synchrone :**

  une *Machine de Mealy Déterministe*
- **M = < S, I, O, T, G, S_0>**
  - *S = ensemble fini d'états (états de contrôle)*
  - *I = ensemble fini des signaux d'entrée*
  - *O = ensemble fini des signaux de sortie*
  - *T = fonction de transition : $S \times 2^I \rightarrow S$*
  - *G = fonction de génération : $S \times 2^I \rightarrow 2^O$*
  - *S_0 = état initial*



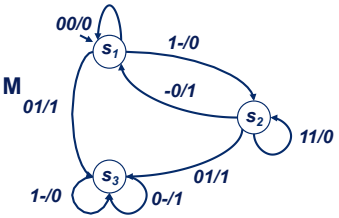
- **Un événement d'entrée (resp. de sortie) : un élément de 2^I (resp. 2^O)**
- **Machine déterministe et complète : T et G sont des fonctions (et non pas des relations) totales → Un état initial et une séquence d'événements d'entrée donnés définissent un *unique* état de contrôle et un *unique* événement de sortie.**

Exemple

- **Une machine de Mealy M = < S, I, O, T, G, S_0>**
  - S = {s_1, s_2, s_3}, S_0 = {s_1}
  - I = {i_1,i_2}, O = {o}
  - T = <s_1,00> → s_1, <s_1,01> → s_3, <s_1,1-> → s_2
    <s_2,-0> → s_1, <s_2,11> → s_2, <s_2,01> → s_3
    <s_3,--> → s_3
  - G = <s_1,00> → 0, <s_1,01> → 1, <s_1, 1-> → 0
    <s_2,-0> → 1, <s_2,11> → 0, <s_2,01> → 1
    <s_3,1-> → 0, <s_3,0-> → 1

Représentation graphique

- **Graphe étiqueté : <E,V,L_V, E_0>**
  - E : sommets étiquetés par les états de M
  - V : arcs : transitions ⊆ E × E
    v = (s,s') ∈V ⇔∃ <s,e_i,s'> ∈ S ×2^I × S et s' = T(s,e_i)
  - L_V : étiquettes des arcs : V → 2^I × 2^O
    ∀ v = (s,s') ∈V, L_V(v) = e_i / e_o ⇔ s' = T(s,e_i) et e_o = G(s,e_i)
  - E_0 : sommet correspondant à l'état initial
- **Représentation graphique de la machine M**



Exemple

- **Le digicode :**
  Soit un digicode déclenchant l'ouverture d'une porte lorsque le code saisi est correct. La saisie du code n'a lieu que lorsque la porte est fermée. Le code est composé de trois digits. Une erreur de saisie provoque un bip sonore. Dans ce cas, le digicode est réinitialisé par un délai de garde (Temporisation). Un capteur détecte la fermeture de la porte, un autre l'état de la temporisation.

Composition

- **La mise en parallèle de modules synchrones correspond à la composition des machines de Mealy équivalentes.**
- **M1 || M2 = M3 est une machine de Mealy *si certaines hypothèses sont respectées* : analyse de causalité**
  → rejet des programmes ne respectant pas ces hypothèses

→**Description modulaire**
Réutilisabilité / remplacement

→ **Vérification formelle à partir du modèle en Machine de Mealy**

3. Différents langages et utilisation industrielle

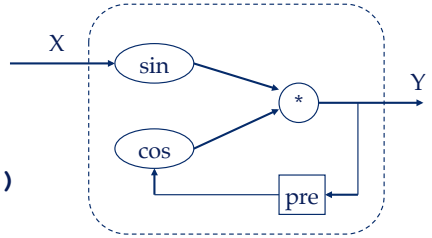
Master d'Informatique
Spécialité SAR
Langages Synchrones

Modèle basé flot de données

- **Le traitement des donnée est constant au cours du temps, les données passent à travers un réseau d'opérateurs à un rythme pré-établi**
  - Chemins de données, contrôle continu, traitement du signal
  - Lustre/SCADE: langages fonctionnels d'équations séquentielles
  - Cf Simulink (Matlab), Ptolemy, …

$Y_t = \sin(X_t) * \cos(Y_{t-1})$

```
Y = sin(X) * cos(pre(Y))
```

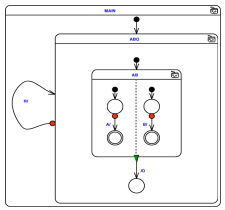


boîtes = opérateurs
flèches = flots de données

Modèle basé flot de contrôle

- **Le comportement change fréquemment (et en fonction des données), peu de données**
  - Protocoles, IHM, mode de fonctionnement, contrôle mémoire
  - Esterel/SyncCharts: impératif + hiérarchie
  - Cf. aussi StateCharts, StateFlow, …

```
abort
     sustain DMAReq
when DMAOk;
abort
     abort
          every ByteIn do
               emit ByteOut (?ByteIn)
          end every
     when DMAEnd
when 10 MilliSecond do
     emit TimeOut
end abort
```

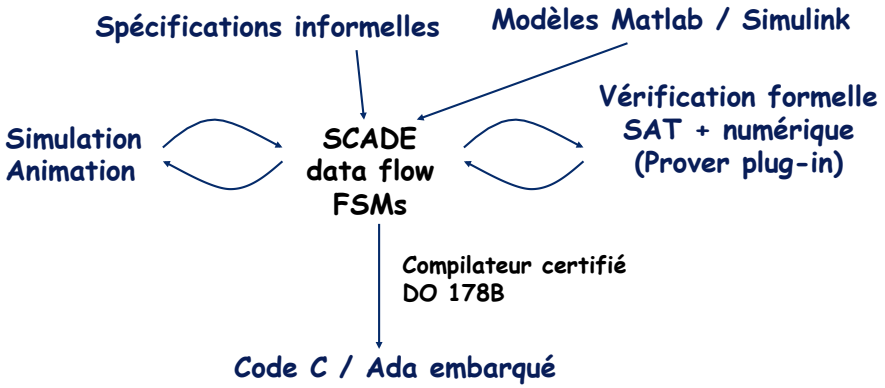


boîtes = états
flèches = transitions
noms = signaux
hiérarchie = préemption

Langages synchrones et industrie

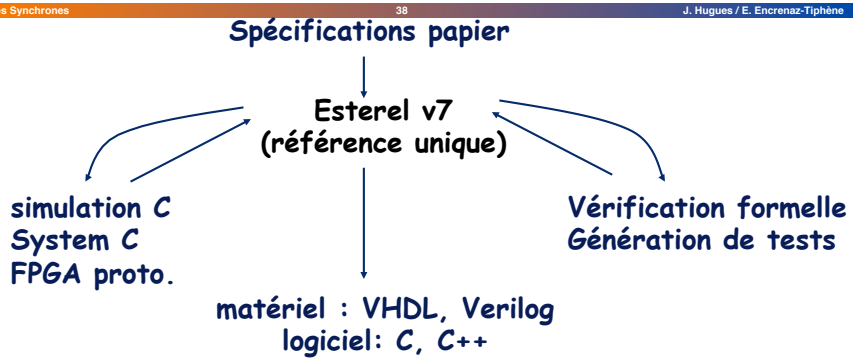
- **Créés dans les années 1980**
  - Esterel: Ecole des Mines/INRIA, SyncCharts: U. de Nice
  - Lustre: IMAG (Grenoble), Signal: INRIA Rennes
  - Ptolemy (Berkeley), TCCP (Xerox), Lava (Chalmers), …
- **Utilisation industrielles dès les années 1990**
  - Lustre/SCADE: nucléaire (Schneider), avionique (Airbus)
  - Esterel; avionique (Dassault), telecom
  - Signal/SILDEX; contrôle continu
- **Développement se poursuit**
  - Avionique; Airbus, Dassault, Eurocopter, SNECMA, Thales, …
  - Automobile: GM, PSA, …

Flot de conception d'un logiciel embarqué



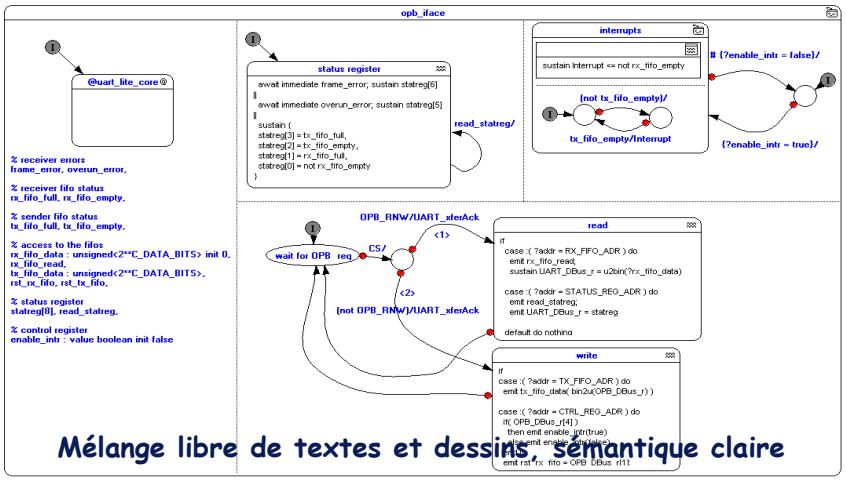
Autorise l'échange de spécifications formelles

Le flot circuits Esterel v7



La sémantique est exactement la même pour le matériel et le logiciel

UART with OPB Interface (Xilinx)



Mélange libre de textes et dessins, sémantique claire

Conclusion

- **Systèmes embarqués: un domaine en croissance**
  - Besoins applicatifs nombreux: grand public, défense, médical, transport,
  - Domaine techniques: automatique, informatique, circuits
- **Fortes contraintes de qualités**
  - Coût de bugs, des tests, de la validation, de la certification
- **Besoin de techniques spécifiques de conception et vérification**
  - Prise en compte du parallélisme et déterminisme
- **Définition du modèle synchrone**
  - Langage et outils spécifiques
  - Sémantique mathématique complète
  - Compilation vers logiciel et circuit électronique
- **Synthèse et vérification formelle**