# Spécification, vérification et optimisation des systèmes temps réel distribués embarqués

## Méthodologie
## Adéquation Algorithme Architecture

**Yves.Sorel@inria.fr**

**www-rocq.inria.fr/syndex**

# INRIA

Institut National de Recherche en Informatique et Automatique

http://www.inria.fr

- **Depuis 1968**
- **3500 personnes**
- **6 Centres**
- **Budget : 135 MEuro**
- **20 Startups**

- **Recherche**
- **Transfert de connaissance**
- **Formation**
- **W3C**
- **Collaboration internationale**

# Cinq thèmes de recherche

1. **Systèmes communicants**
2. **Systèmes cognitifs**
3. **Systèmes symboliques**
4. **Systèmes numériques**
5. **Systèmes biologiques**

# Contexte INRIA

- **Projet AOSTE**
- **Thème Com C : Systèmes embarqués et mobilité**
- **Modèles et méthodes pour l'Analyse et l'Optimisation des Systèmes Temps réel Embarqués**
- **Implantation sur plate-formes embarquées :**
  **Méthodologie Adéquation Algorithme-Architecture pour l'implantation optimisée d'application distribuées temps réel embarquées (logiciel SynDEx)**

# Applications

- **Automobile (AEE, EAST, ECLIPSE)**
- **Robotique mobile (CyCab, SAFE)**
- **Traitement signal : radio logicielle (Mitsubishi ITE)**
- **Télécommunications : SoC UMTS (PROMPT)**
- **Traitement d'image : JPEG2000 (P2I), MPEG4 (INSA), guidage automatique (MBDA)**
- **…**

# Objectifs

- **Sécurité de conception**
- **Prototypage rapide**
- **Optimisation**
- **Génération de code de série**
- ***Réduction du cycle de développement***

# Caractéristiques

- **Algorithmes** : contrôle-commande, TSI
- **Réactif** : *stimulus - opérations - réaction*
- **Temps réel** : temps de réaction borné
  contraintes : latence, cadence
- **Distribué** : puissance calcul, modularité, câblage réduit

⇒ ***Architecture multicomposant hétérogène***

  - Réseau processeurs + circuits intégrés spécifiques
  - Circuit intégré spécifique (ASIC, ASIP, FPGA, IP)

- **Embarqué** : minimisation des ressources

# Méthodologie Adéquation Algorithme Architecture

- **Approche globale** fondée sur la sémantique des Langages Synchrones et le modèle matériel RTL
- **Modèle unifié :** graphes orientés
  - **A**lgorithme : opération / dépendance
  - **A**rchitecture : FSM / connexion
  - Implantation : transformations de graphes
- **Adéquation :** implantation optimisée
- **Macro-génération :**
  - Exécutifs pour multicomposant
  - VHDL structurel pour synthèse de circuit intégré

# Approches hors-ligne versus en-ligne

**Méthodologie AAA**

Hors-ligne → En-ligne

(le plus possible) (quand inévitable)

**Conception classique**

En-ligne → Hors-ligne

**Hors-ligne :** distrib./ordo. hors-ligne sans préemtion

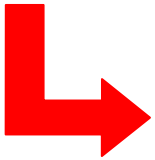
**En-ligne :** distrib./ordo. en-ligne, ou hors-ligne avec préemption

*Avantages*
- HL : déterminisme, surcoût d'exéc. faible
- EL : événements apériodiques

*Inconvénients*
- HL : pas toujours possible, nécessite connaissance application
- EL : surcoût d'exéc. élevé, indéterminisme

# Spécification Algorithme : hyper-graphe orienté

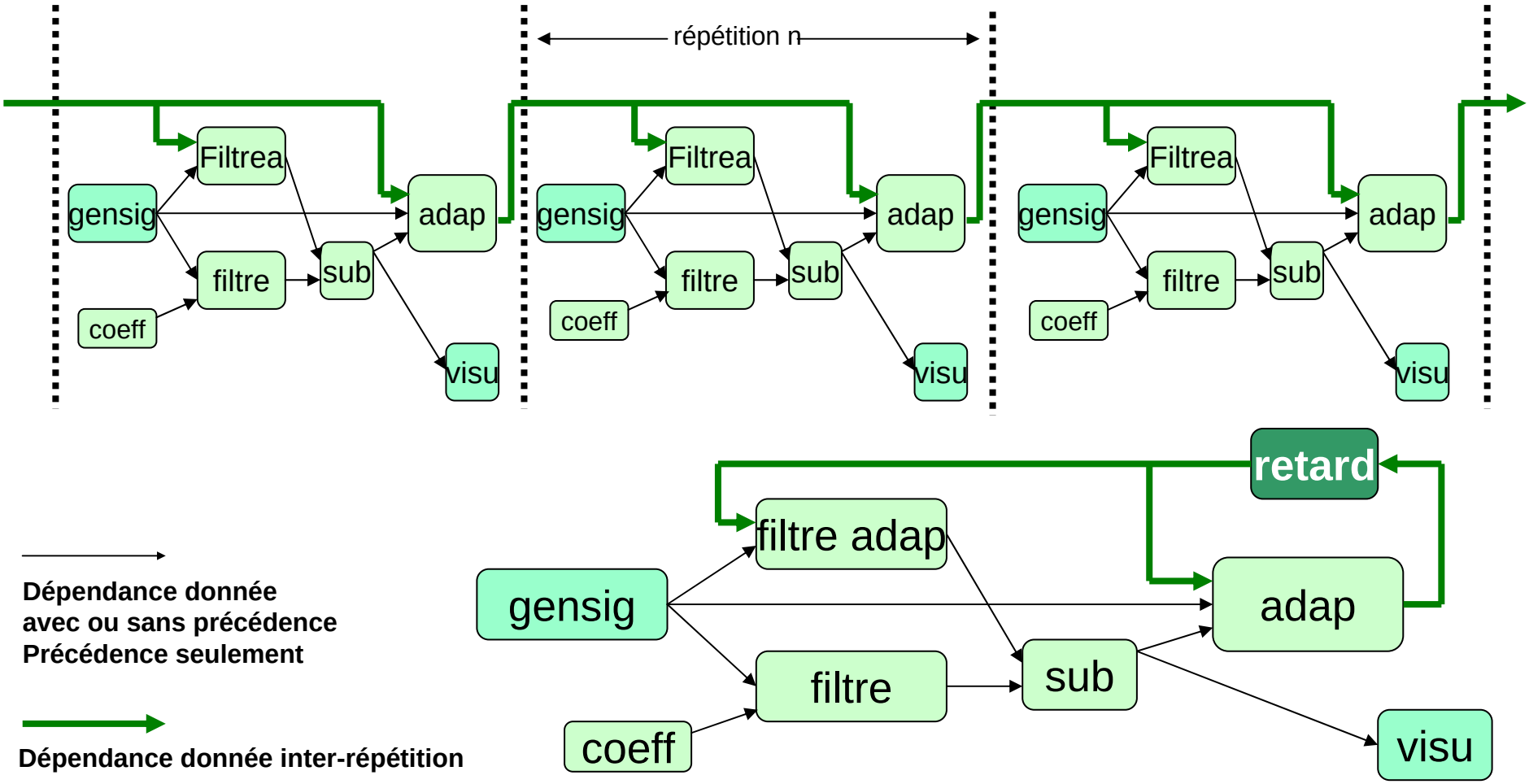
- **Sommet :** opération conditionnée : entrées-calculs-sorties
- **Hyper-arc orienté :** dépendance avec diffusion : données avec ou sans précédence, précédence seulement, conditionnement (exécution ou non)
  définit un ordre partiel = parallélisme potentiel
- **Répétition infinie d'un sous-graphe :** réactif boucle inf.
- **Répétition finie d'un sous-graphe :** boucle finie



**Graphe factorisé conditionné de dépendances de données**

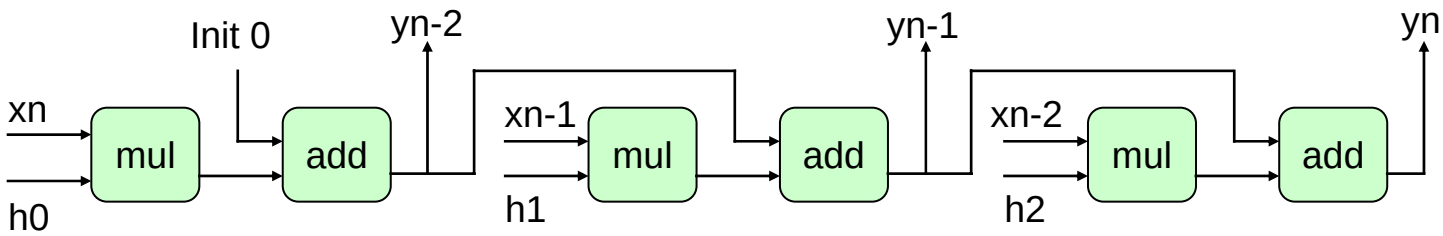
- **Issu de langages orientés métiers : Langages Synchrones, AIL, Scicos, AVS, CamlFlow, ...**

# Exemple d'algorithme : égaliseur adaptatif



# Filtre à 3 coefficients : répétition finie de 3

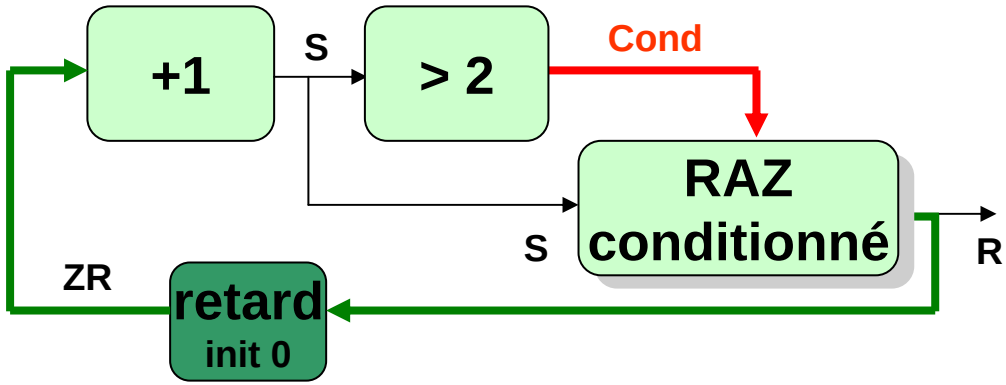
$Yn = \Sigma(hi*Xn-i)$ pour i=0 à 2

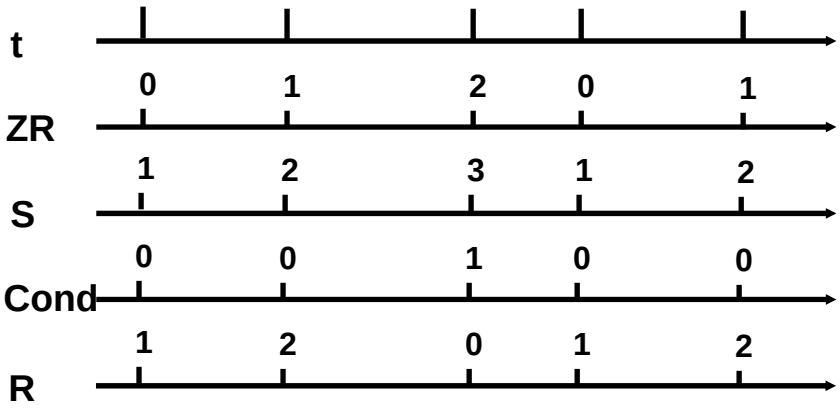


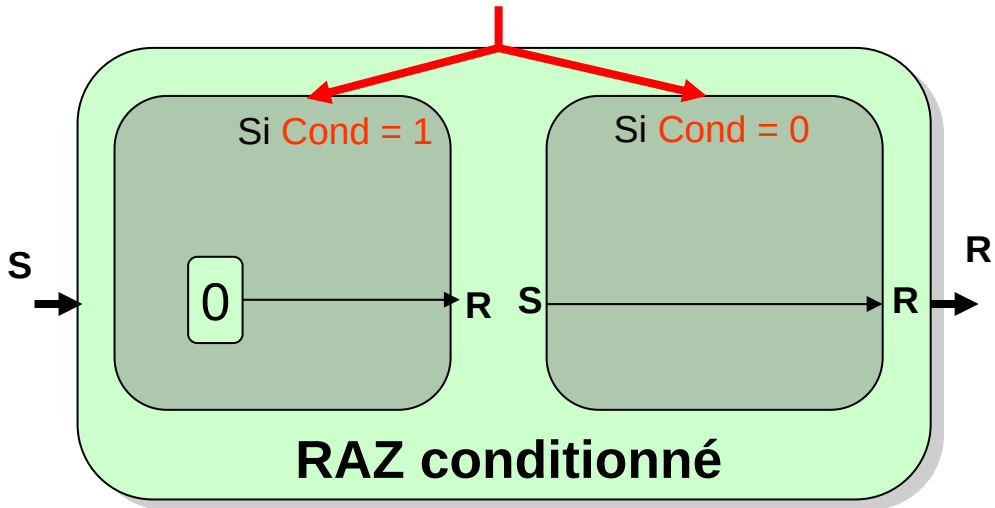

Fork : H(h0, h1, h2) et X(xn, xn-1, xn-2)

Join : Y(yn, yn-1, yn-2)

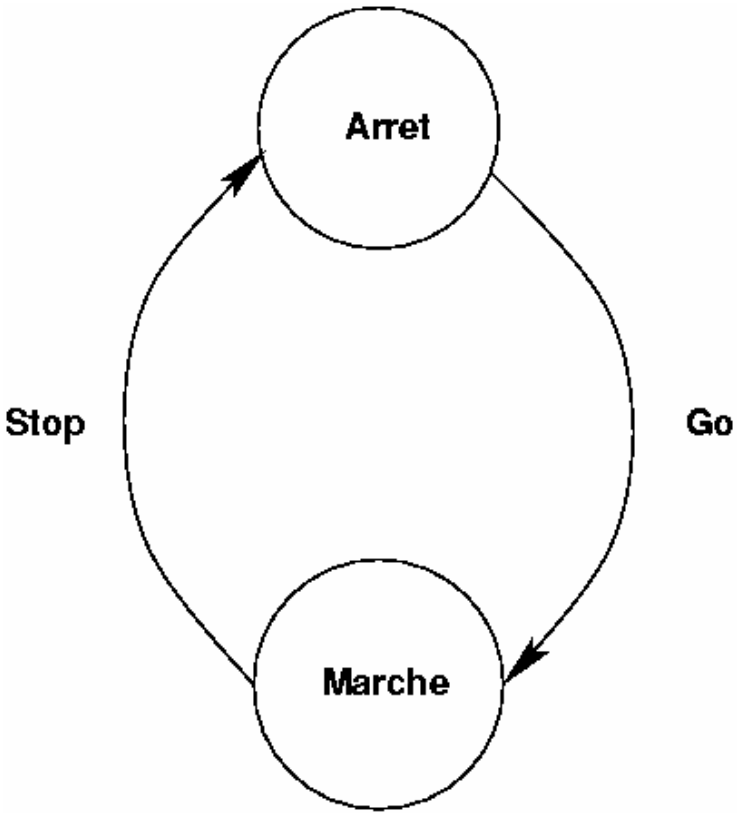
# Algorithme conditionné : compteur modulo 3

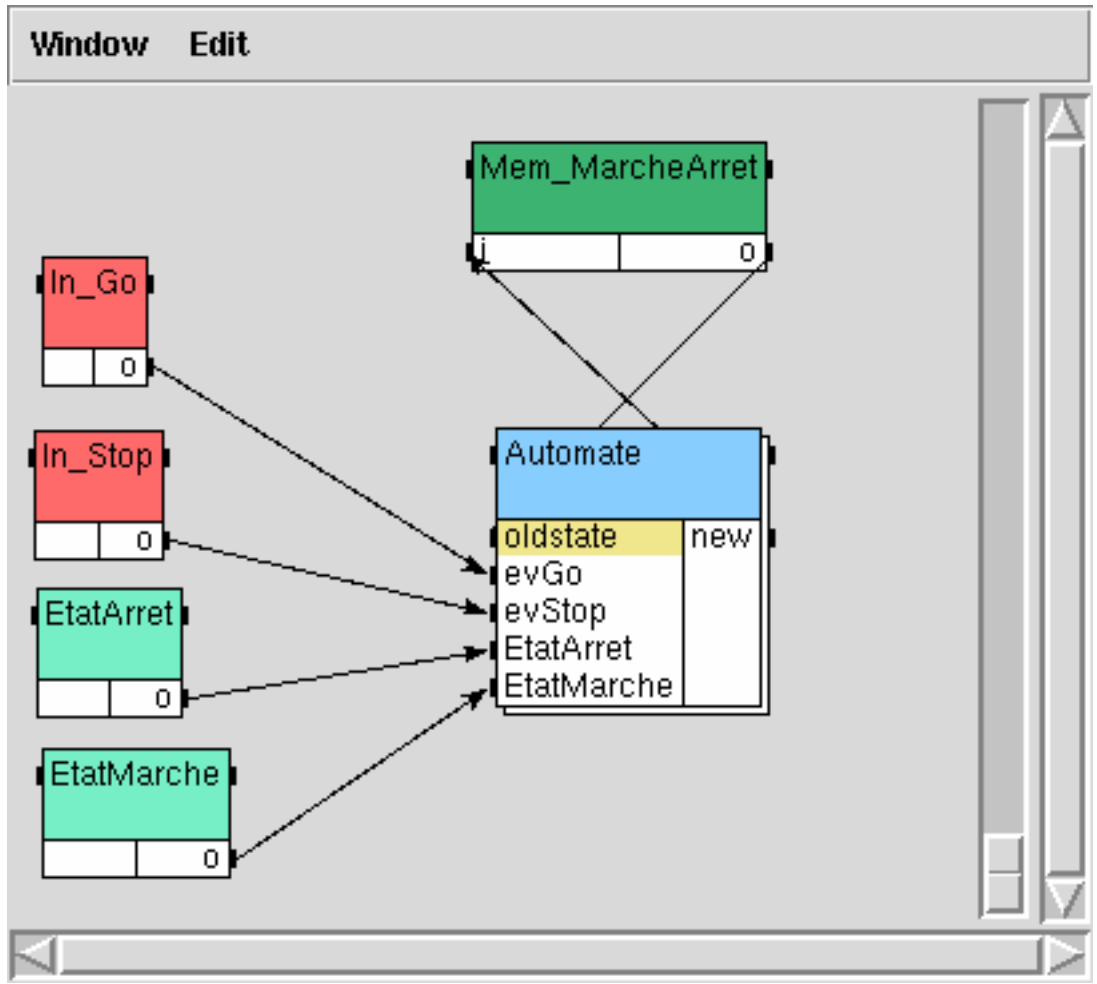


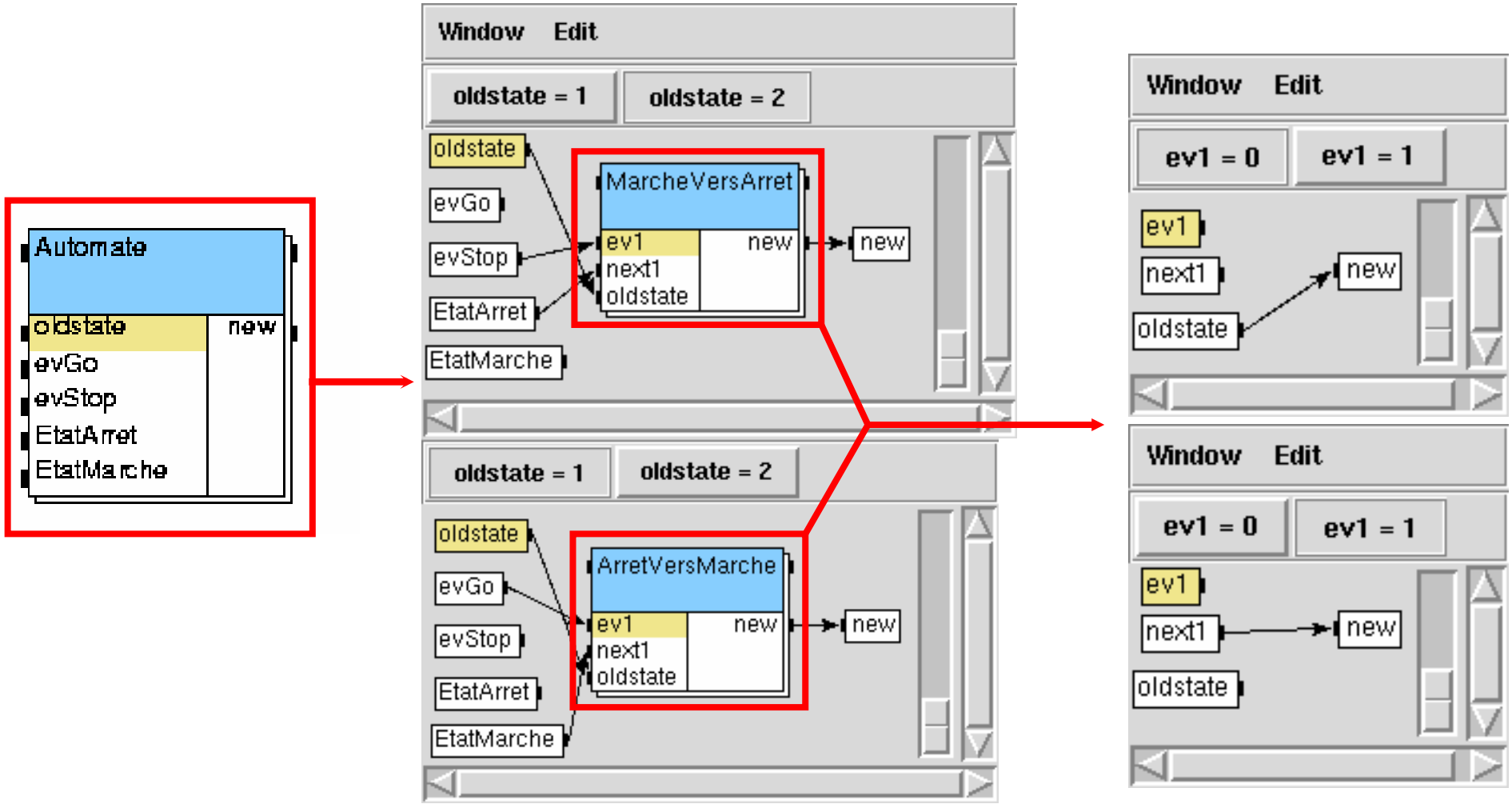



# Machine à état avec AAA/SynDEx (1/2)

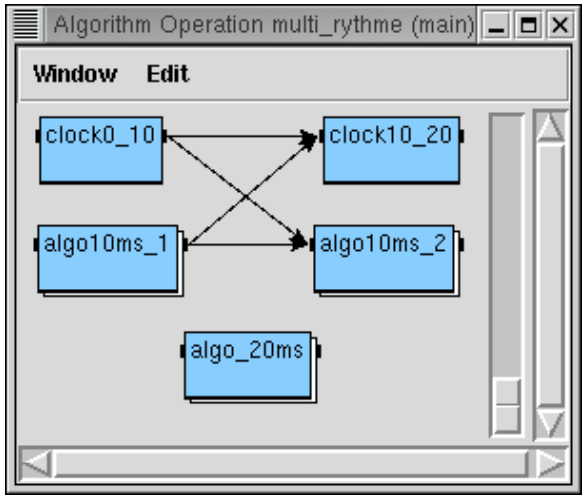


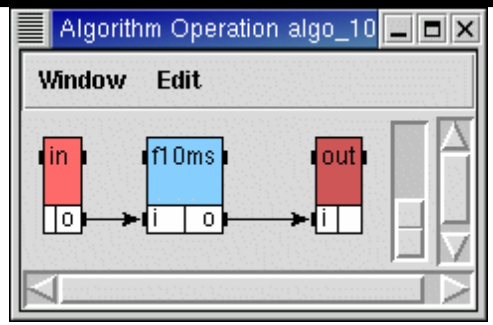



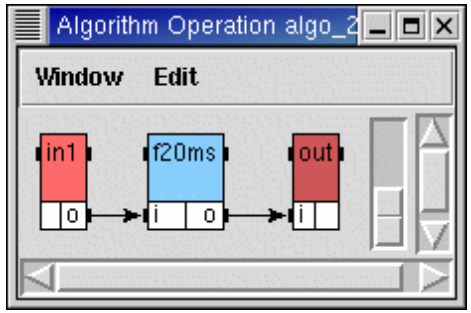
# Machine à état avec AAA/SynDEx (2/2)

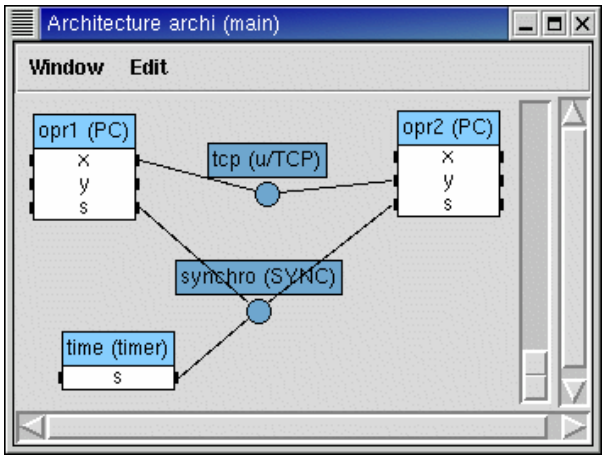


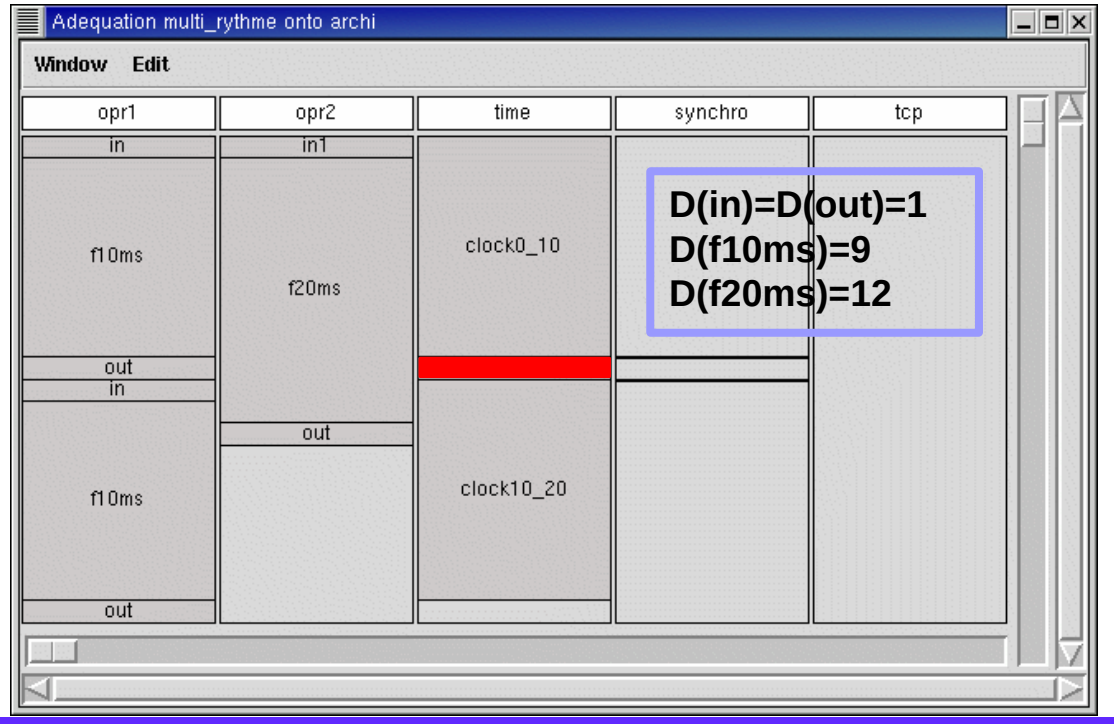
# Multi-période



# Vérification Algorithme

- **Langages Synchrones**
  - Esterel, Lustre, Signal, SyncCharts ...
  - Spécification modulaire
  - Hors contraintes matérielles, indépendante des durées : temps logique, ordre sur événements
  - Réaction ***simultanée*** *avec* Stimulus
  - Vérifications :
    - Cycles de dépendance **seulement avec** retard
    - Cohérence ordre sur événements
    - Vérification de propriétés temporelles logiques

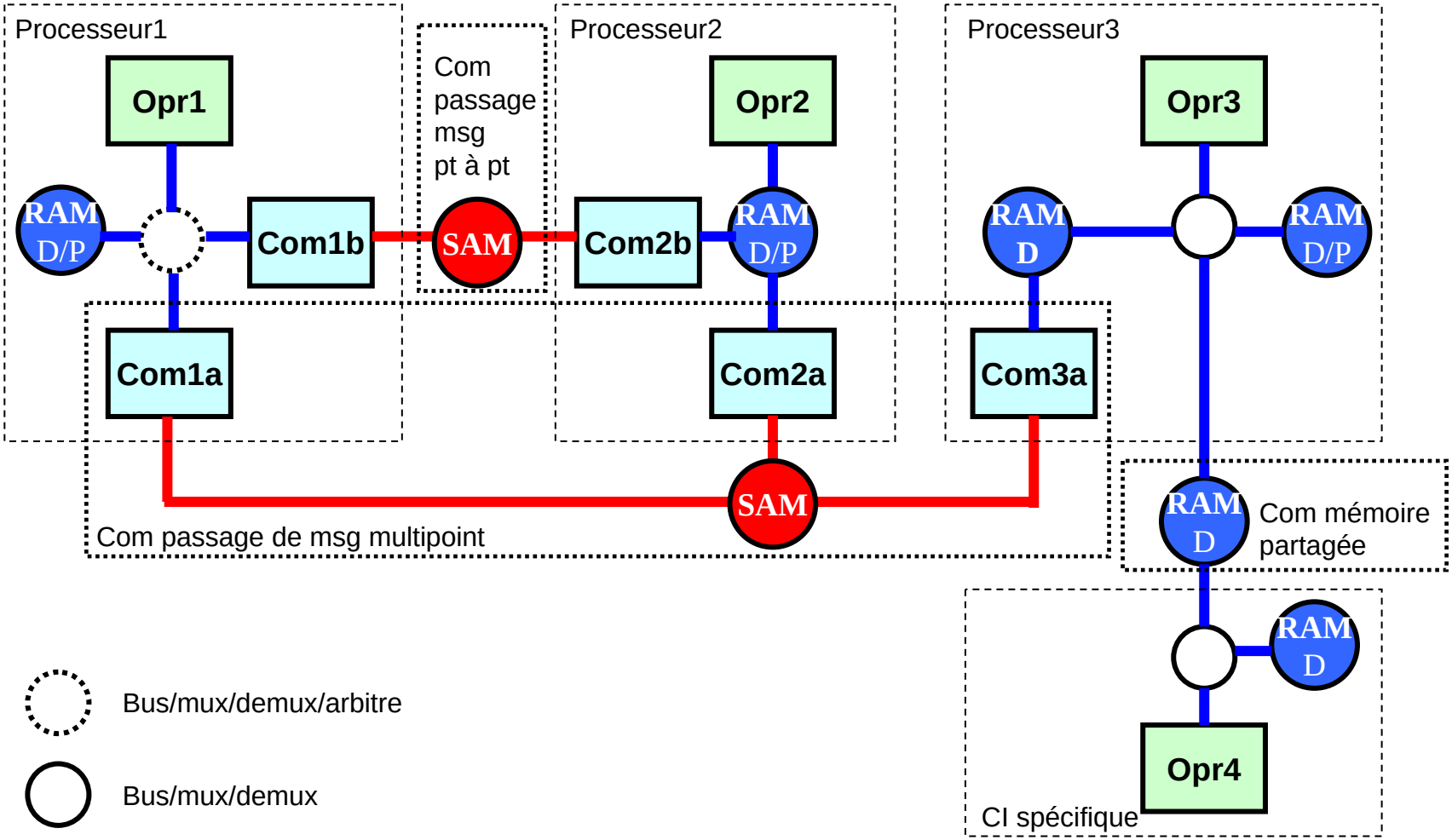
# Spécification Architecture : graphe orienté

- **Sommets (FSM machine séquentielle)**
  - Opérateur : exécution séquentielle des opérations
  - Communicateur : exécution séquentielle opérations comm.
  - Mémoire avec ou sans arbitre :
    - Accès aléatoire (RAM) : comm. non synchronisées
      - Programmes, données, comm. par mémoire partagée
    - Accès séquentiel (SAM) : comm. synchronisées, ordre R/W
      - Données seulement, point-à-point, multi-point avec ou sans diffusion matérielle, comm. par passage de message
  - Bus/mux/démux avec ou sans arbitre :
    - Bus/mux/démux : sélection d'une mémoire parmi plusieurs
    - Bus/mux/démux/arbitre : arbitre l'accès aux mém. partagées

# Spécification Architecture

- **Arcs orientés**
  - Connexions entre ces sommets pour modéliser les transferts de données
  - Connexions suivant un ensemble de règles :
    - Des opérateurs ne peuvent pas être connectés entre eux
    - Des communicateurs ne peuvent être pas connectés entre eux
    - Des mémoires ne peuvent pas être connectés entre elles
    - Des sommets bus/mux/demux avec ou sans arbitre peuvent être connectés entre eux
    - ...
- **Modèle Macro-RTL :** opération, macro-registre

# Exemple d'Architecture



# Implantation multicomposant (1)

**Toutes les implantations décrites en intention comme la composition de trois relations binaires :**

(Gal, Gar) $\xrightarrow{\textit{rout o distrib o ordon}}$ (Gal', Gar')

- **Routage** : création de toutes les routes inter-opérateur
- **Distribution** : *allocation spatiale* :
  - Partition et allocation des opérations sur les opérateurs
  - Partition des arcs inter-partition en fonction des routes
  - Création et allocation :
    - Sommets opérations de communication sur communicateurs de la route
    - Sommets allocation sur mémoires
    - Sommets identité sur bus/mux/démux/ avec ou sans arbitres

# Implantation multicomposant (2)

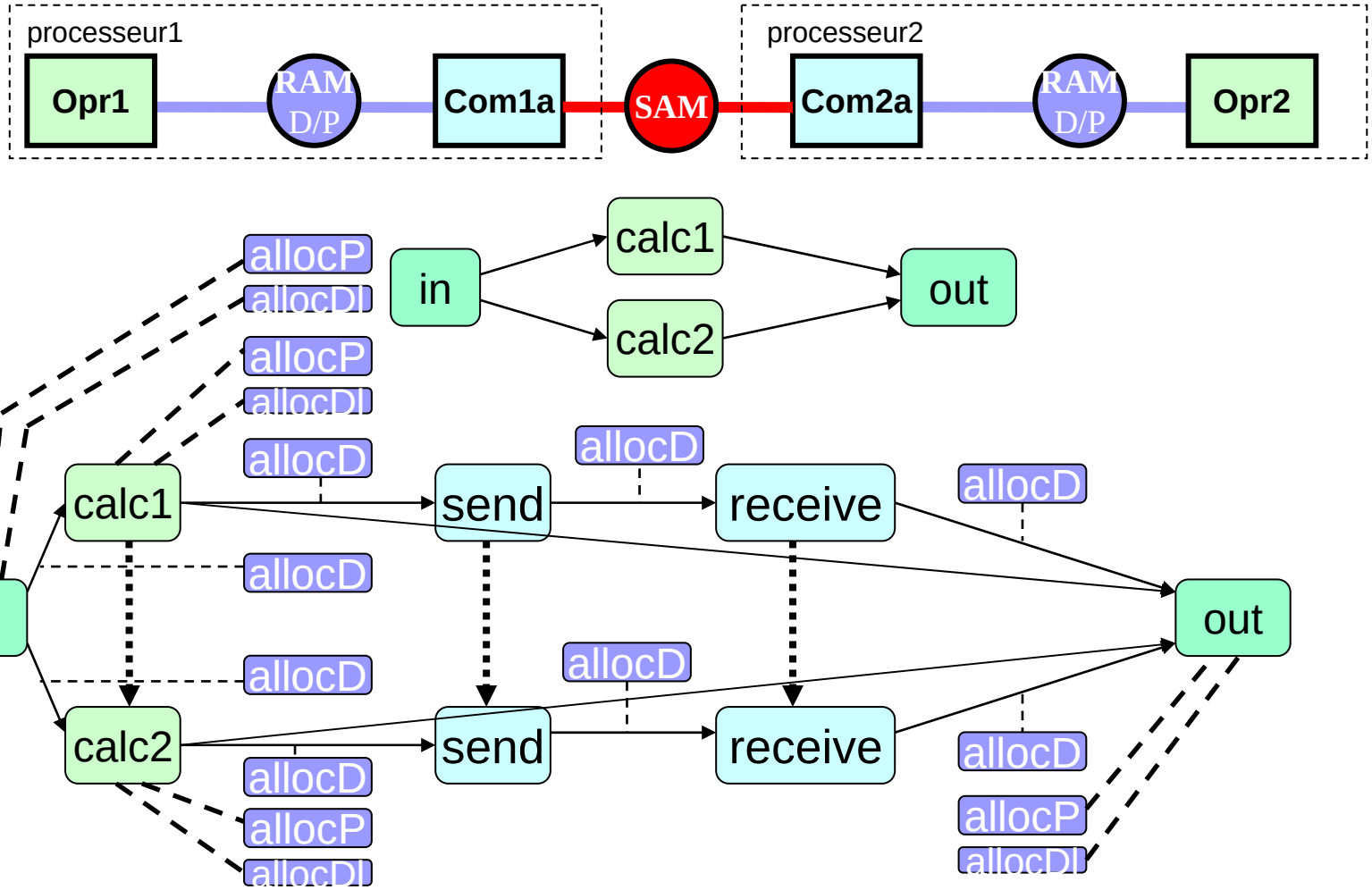
- **Ordonnancement** : *allocation temporelle*
  - Ordre partiel → total pour :
    - chaque partition d'opérations allouée à un opérateur
    - chaque partition d'opération de communications allouée à un communicateur

**Le routage, la distribution et l'ordonnancement conduisent à un ordre partiel cohérent avec l'ordre partiel initial du graphe de l'algorithme**

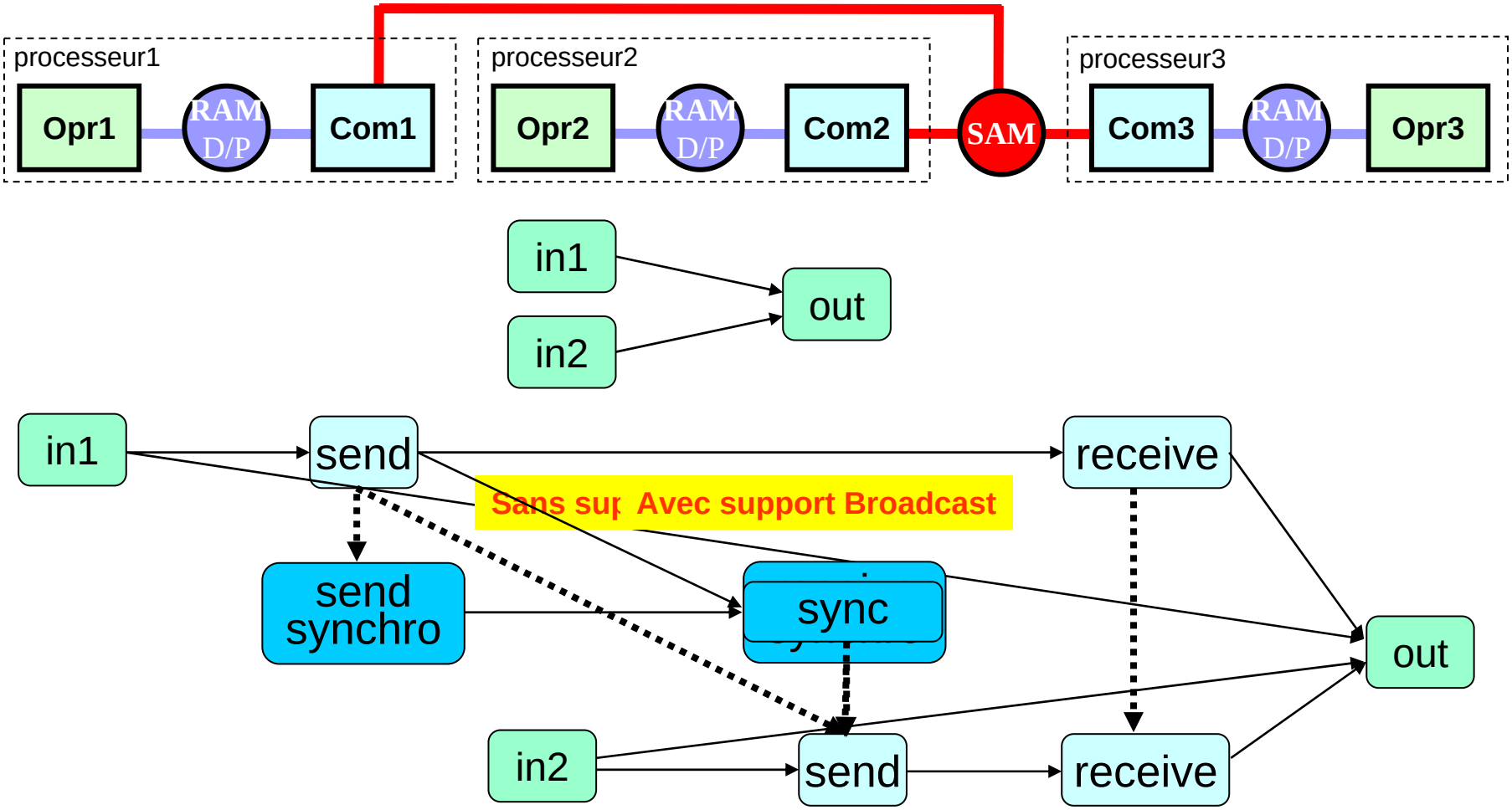
Transformation de graphes : loi de composition externe

Le graphe d'implantation Gal' = Gal*Gar

# Exemple d'implantation : SAM point à point



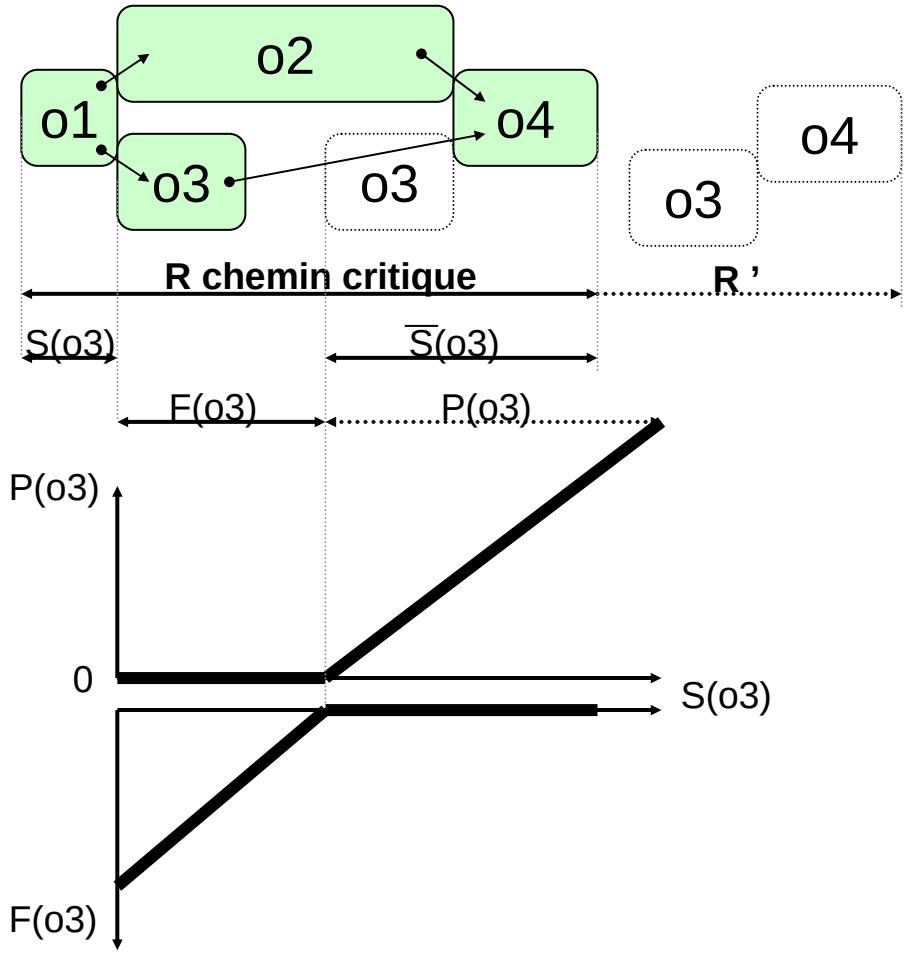
# Exemple d'implantation : SAM multipoint



# Optimisation : principes

- **Caractérisation opérations/opérateurs**
  - Mesures : durée, mémoire, interférences calculs/com.
- **Choix d'une implantation parmi toutes**
  - Qui respecte la contrainte temps réel (latence = cadence) et les contraintes de distribution
  - Qui minimise les ressources
- **Distribution/ordonnancement :** hors-ligne avec ou sans préemption
- **Problèmes NP-difficiles :** heuristiques
  - Rapides (prototypage) : gloutonne, ordonnancement liste
  - Lentes itératives : voisinage local, global, Tabou, Recuit, etc

# Optimisation : exemple latence=cadence hors ligne sans préemption




- Date de **début au plus tôt depuis le début** :

$$S(o_i) = \max_{\forall x_j \in pred(o)} E(x_j) \text{ (ou 0 si } pred(o_i) = \varnothing \text{ )}$$

- Date de **fin au plus tôt depuis le début** :

$$E(o_i) = S(o_i) + \Delta(o_i)$$

- Date de **fin au plus tard depuis la fin** :

$$\overline{E}(o_i) = \max_{\forall x_j \in succ(o)} \overline{S}(x_j) \text{ (ou 0 si } succ(o_i) = \varnothing \text{ )}$$

- Date de **début au plus tard depuis la fin** :

$$\overline{S}(o_i) = \overline{E}(o_i) + \Delta(o_i)$$

- **Flexibilité d’ordonnancement** :

$$F(o_i) = R - S(o_i) - \overline{S}(o_i)$$

- **Pénalité d’ordonnancement** :

$$P(o_i) = R - R'$$

- **Pression d’ordonnancement** :

$$\sigma(o_i) = P(o_i) - F(o_i)$$

# Heuristique gloutonne ordonnancement liste

- Initialiser la liste de candidats avec les opérations sans prédécesseur :

$$Cand = \{o_i / pred(o_i) = \varnothing\}$$

- Tant que la liste n'est pas vide :

  ❶ pour chaque opération $o_i$ de la liste rechercher le meilleur opérateur (avec prise en compte du coût des communications),

$$opr_{o_i} = \mathbf{min}_{\forall p_i \in Proc} \sigma(o_i, p_i)$$

  ❷ sélectionner l'opération la plus urgente à ordonnancer,

$$o_{urgente} = \mathbf{max}_{\forall o_i \in Cand} \sigma(o_i, opr_{o_i})$$

  ❸ retirer l'opération de la liste et ajouter tous ses successeurs devenus ordonnaçables,

$$Cand = Cand - o_{urgente} + Succ_ordo(o_{urgente})$$

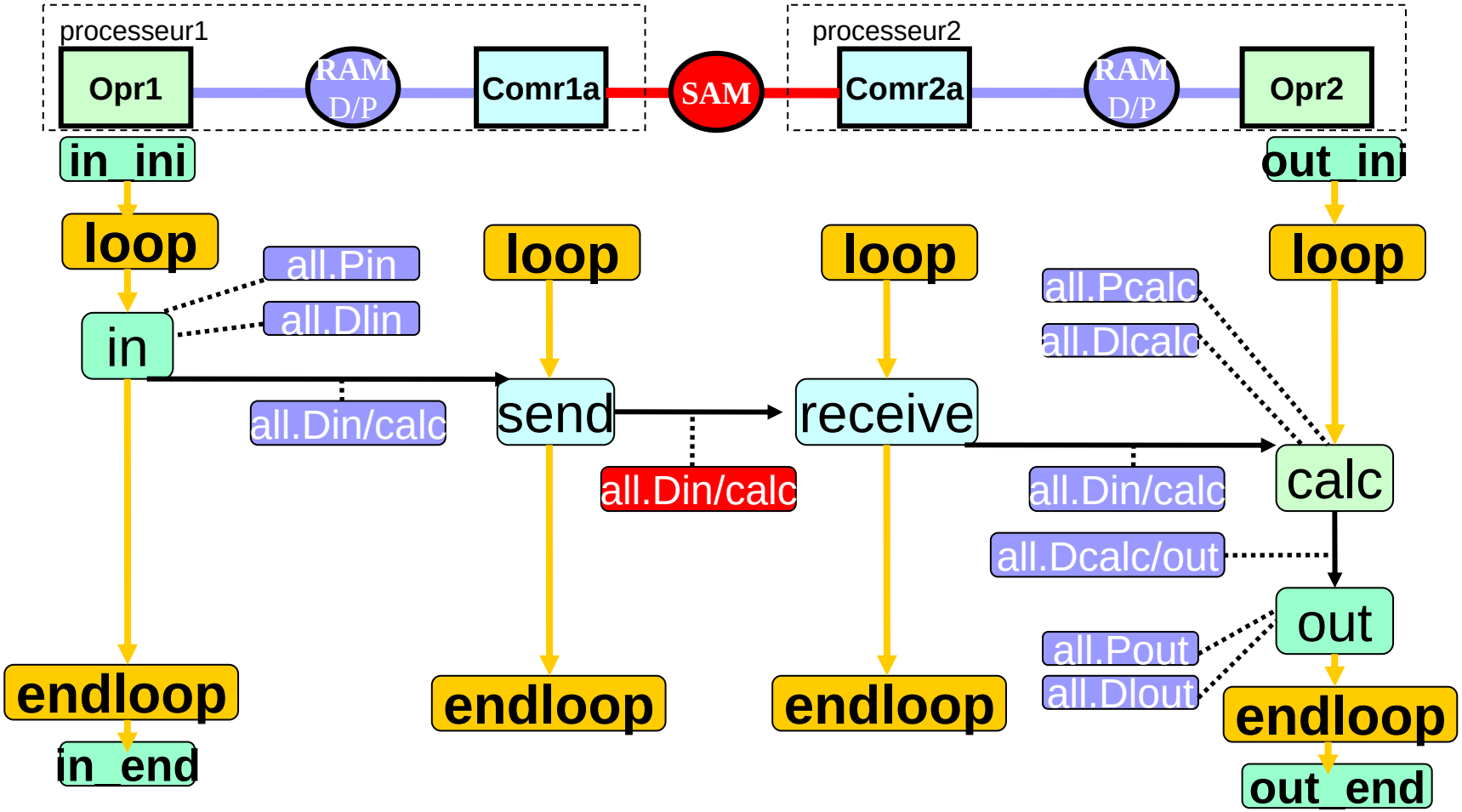
# Génération d'exécutifs dédiés

- **Sans interblocage et à faible surcoût**
- **Macro-code indépendant du processeur (.m4)**
  - Extensible, facilement portable (C, asm, …)
- **Noyau d'exécutif dépendant du processeur (.m4x) définitions macros pour :**
  - Microcontrôleurs : MPC555, MC68332, 80C196
  - DSP : ADSP2160, TMS320C40, TMS320C60
  - Microprocesseurs : i80x86, PPC G4, C/Unix
  - Moyen de comm. : links, CAN, RS232, TCP/IP

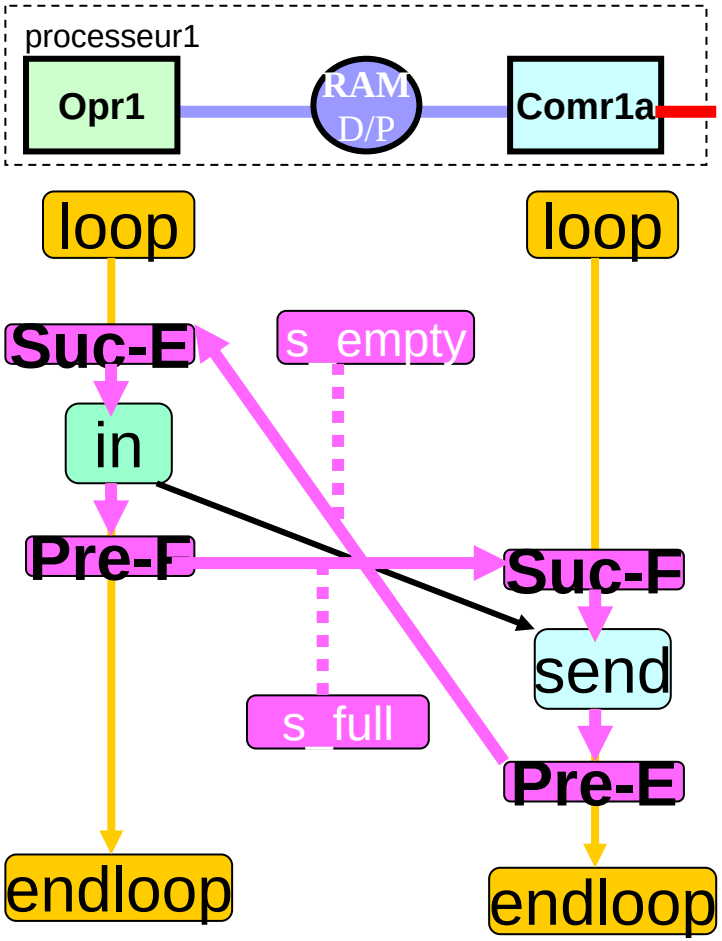
# Transformations de graphes

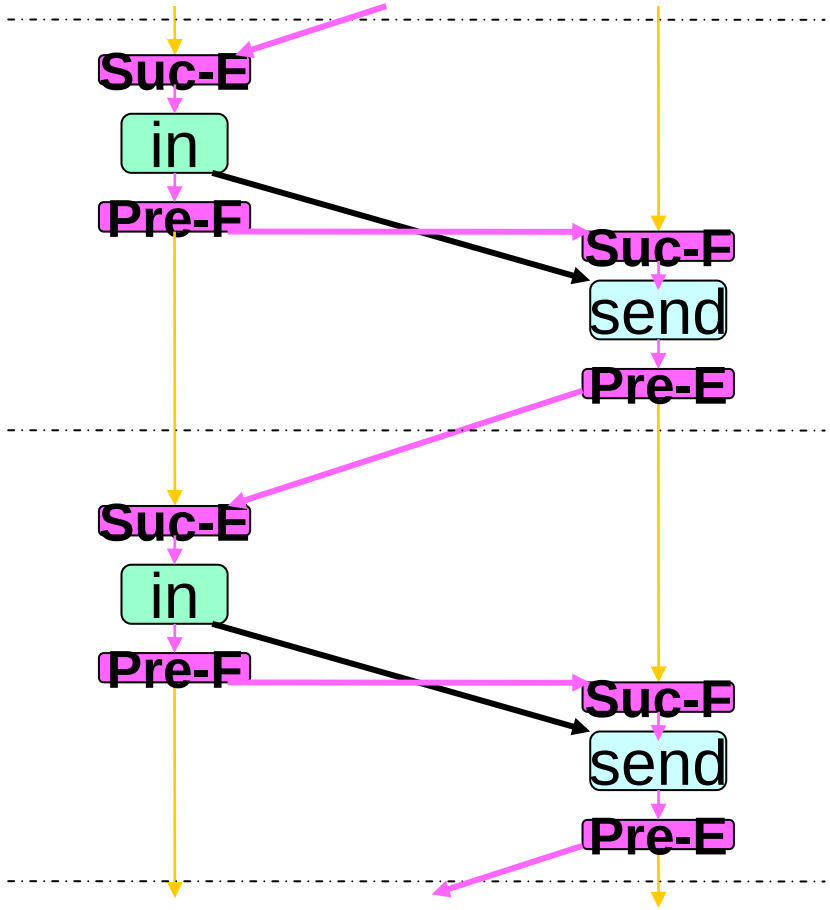
- **Graphes d'implantation optimisé *en* graphe d'exécution par ajout de sommets « *systèmes* » :**
  - Extraction du motif répété infiniment
  - Répétition explicite par ajout de sommets ***loop/endloop***
  - Ajout de sommets d'***init./finalisation*** pour E/S
  - Synchronisation calc/com par ajout de sommets ***Pre/Suc*** utilisant des sémaphores
- **Graphe d'exécution *en* macro-code**
- **Macro-code (fichier.m4) *en* programme source compilable :** macro-processeur (gm4) + bibliothèques d'exécutif (définitions de macros fichier.m4x)

# Graphe d'exécution : répétition explicite

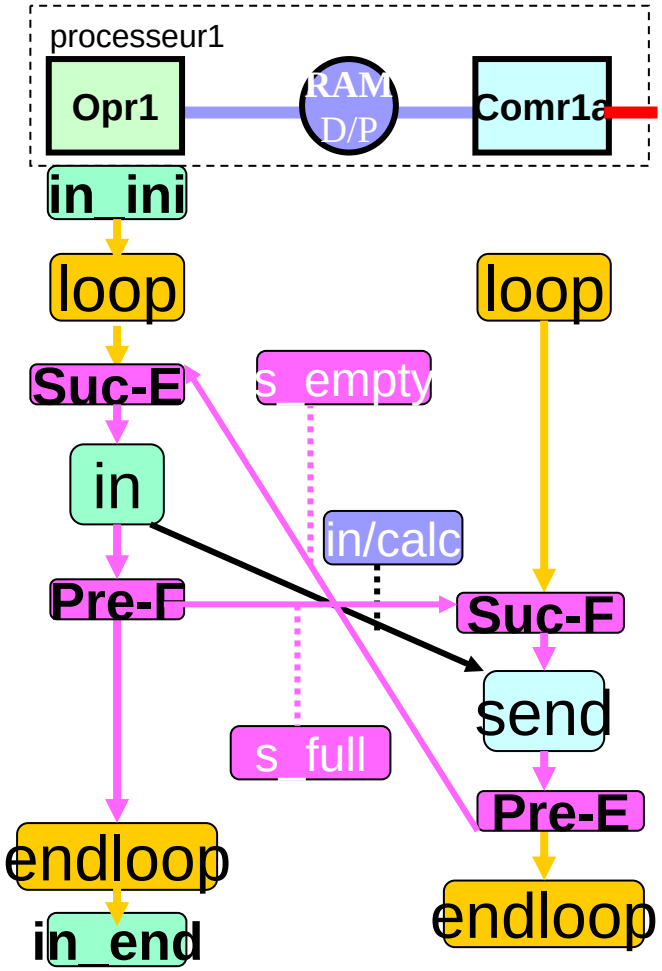


# Graphe d'exécution : synchronisations



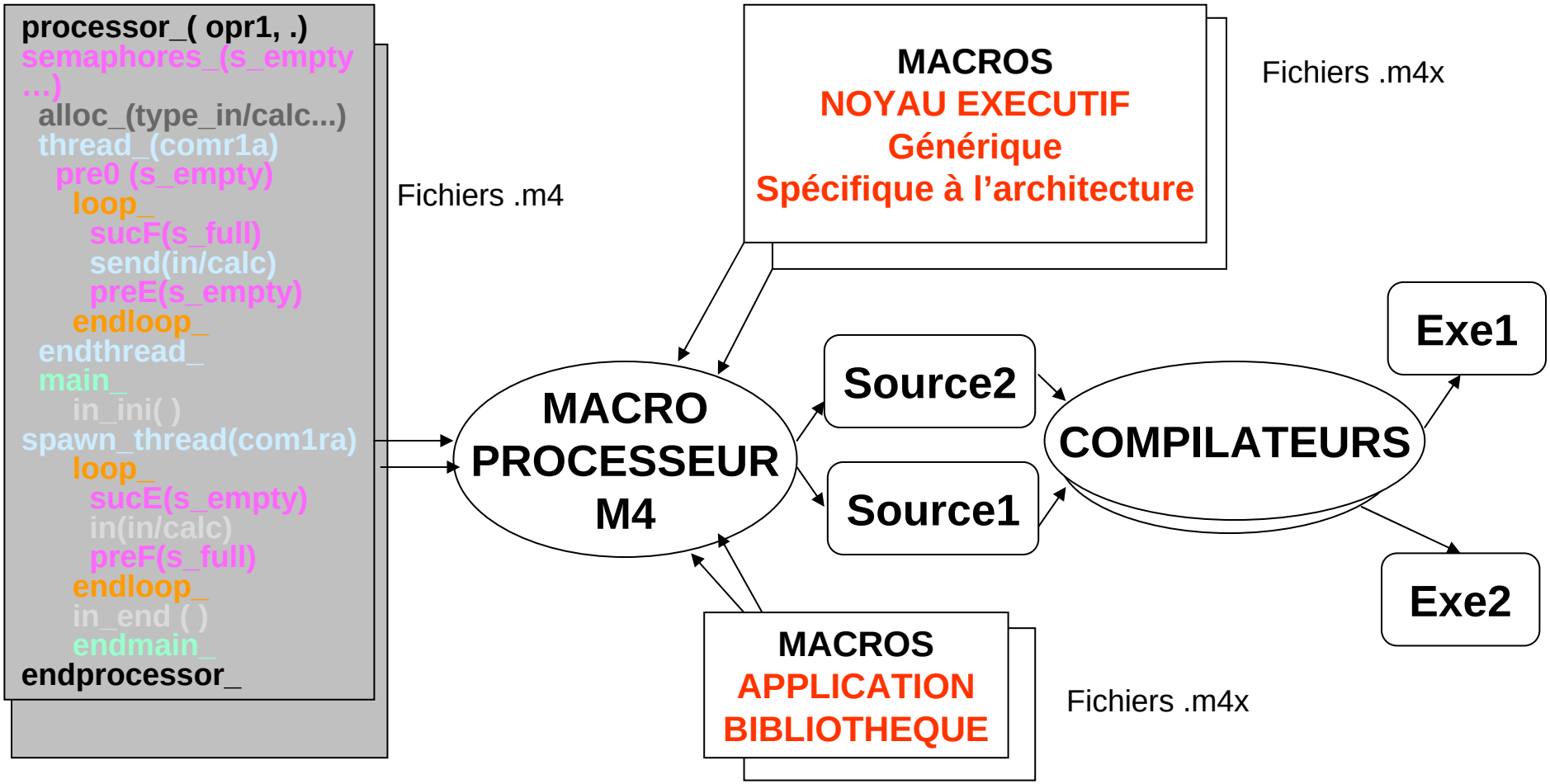
# Génération de macro-code



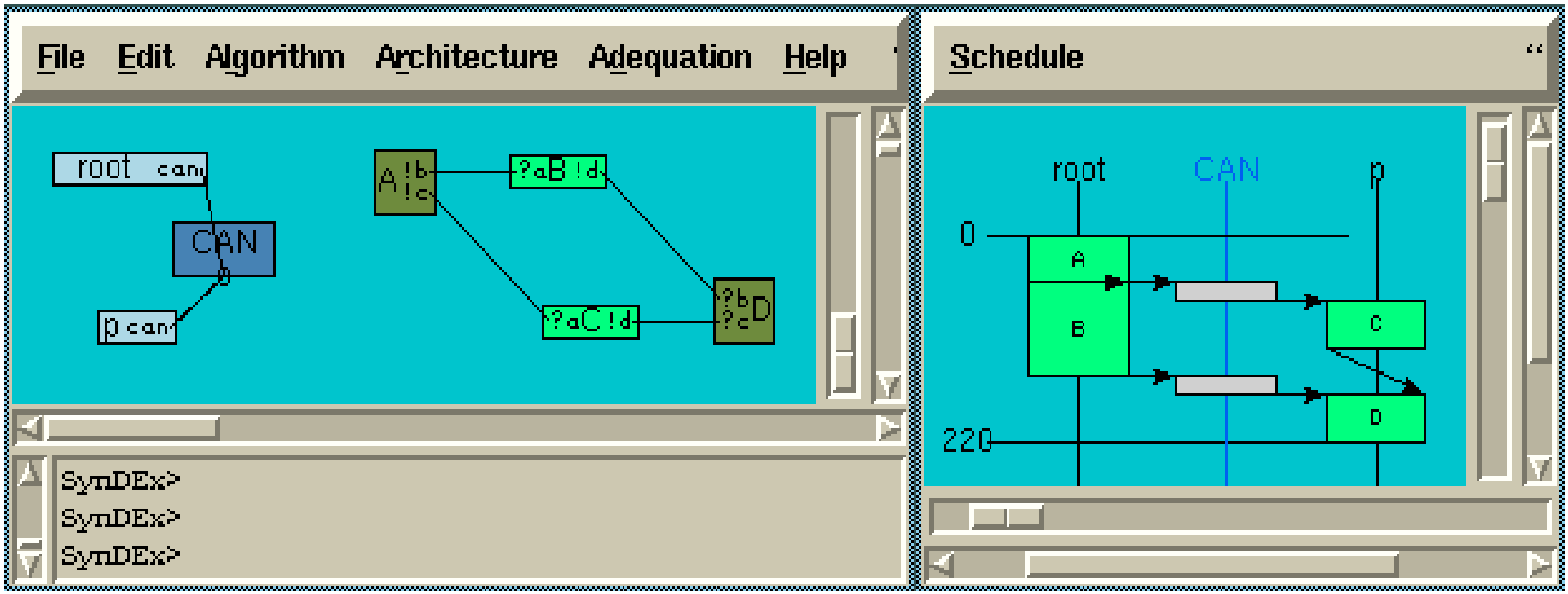

```
processor_( opr1, .)

semaphores_(s_empty...)
  alloc_(type_in/calc...)
  thread_(comr1a)
    pre0 (s_empty)
      loop_
        sucF(s_full)
        send(in/calc)
        preE(s_empty)
      endloop_
  endthread_
  main_
    in_ini( )
    spawn_thread(com1ra)
    loop_
      sucE(s_empty)
      in(in/calc)
      preF(s_full)
    endloop_
    in_end ( )
    endmain_
endprocessor_
```

# Compilation du macro-code



# Génération d'exécutif distribué temps réel : Application simple



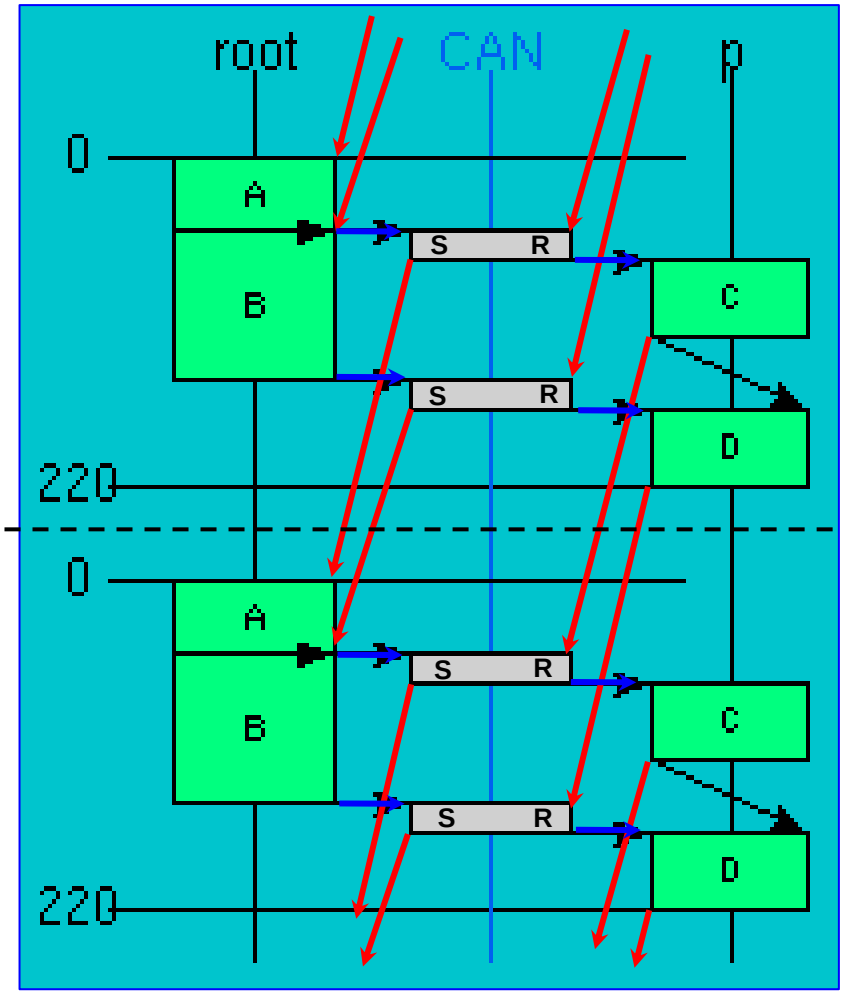
# Génération d'exécutif : Synchro calc/com

Tampon

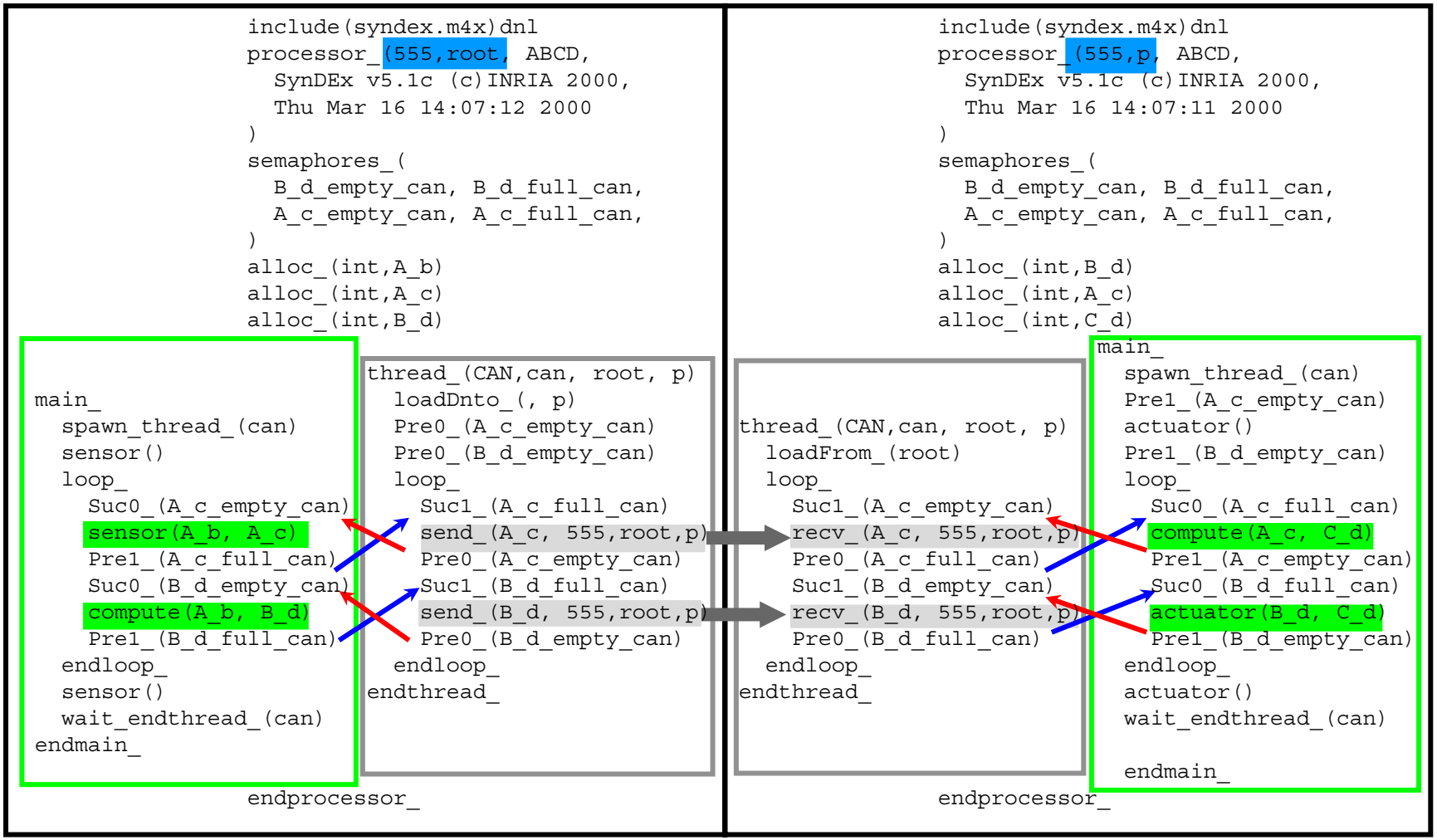
Synchro intra-répétition

Synchro inter- répétition

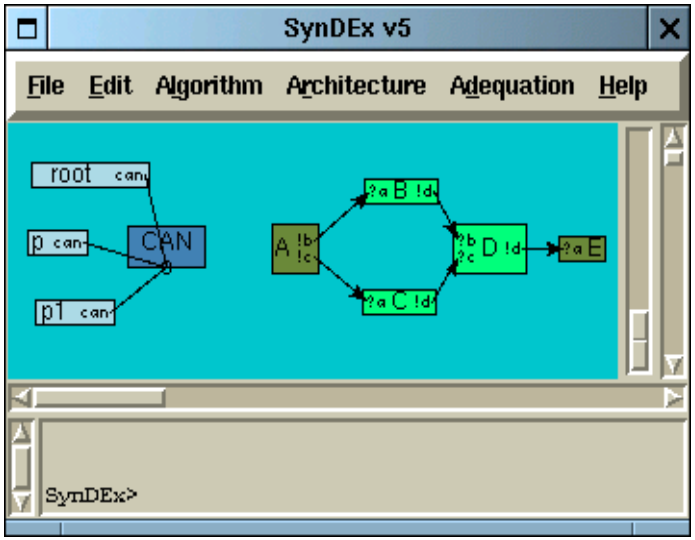
Frontière inter- répétition

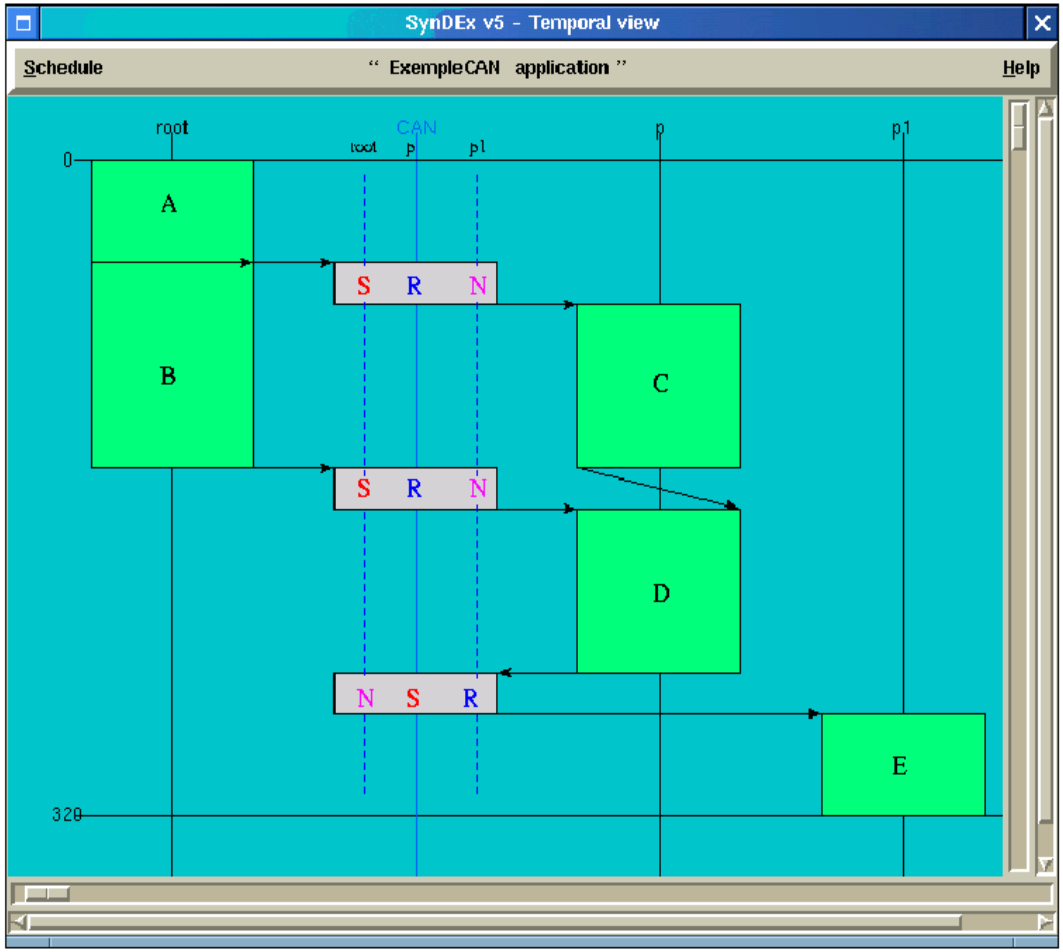


# Génération d'exécutif : Code



# Génération d'exécutif : Synchro. comm.



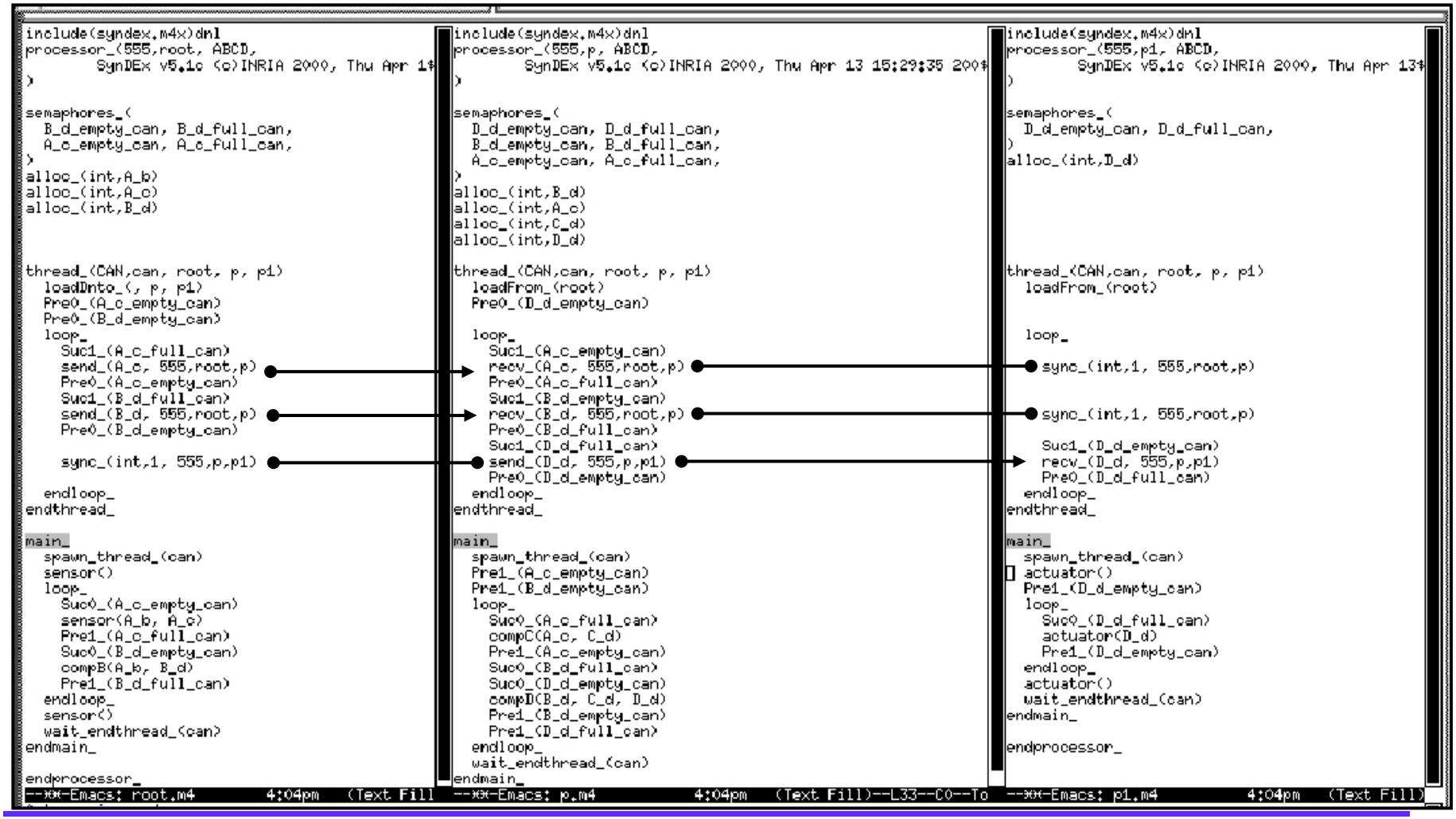




**S : Send**

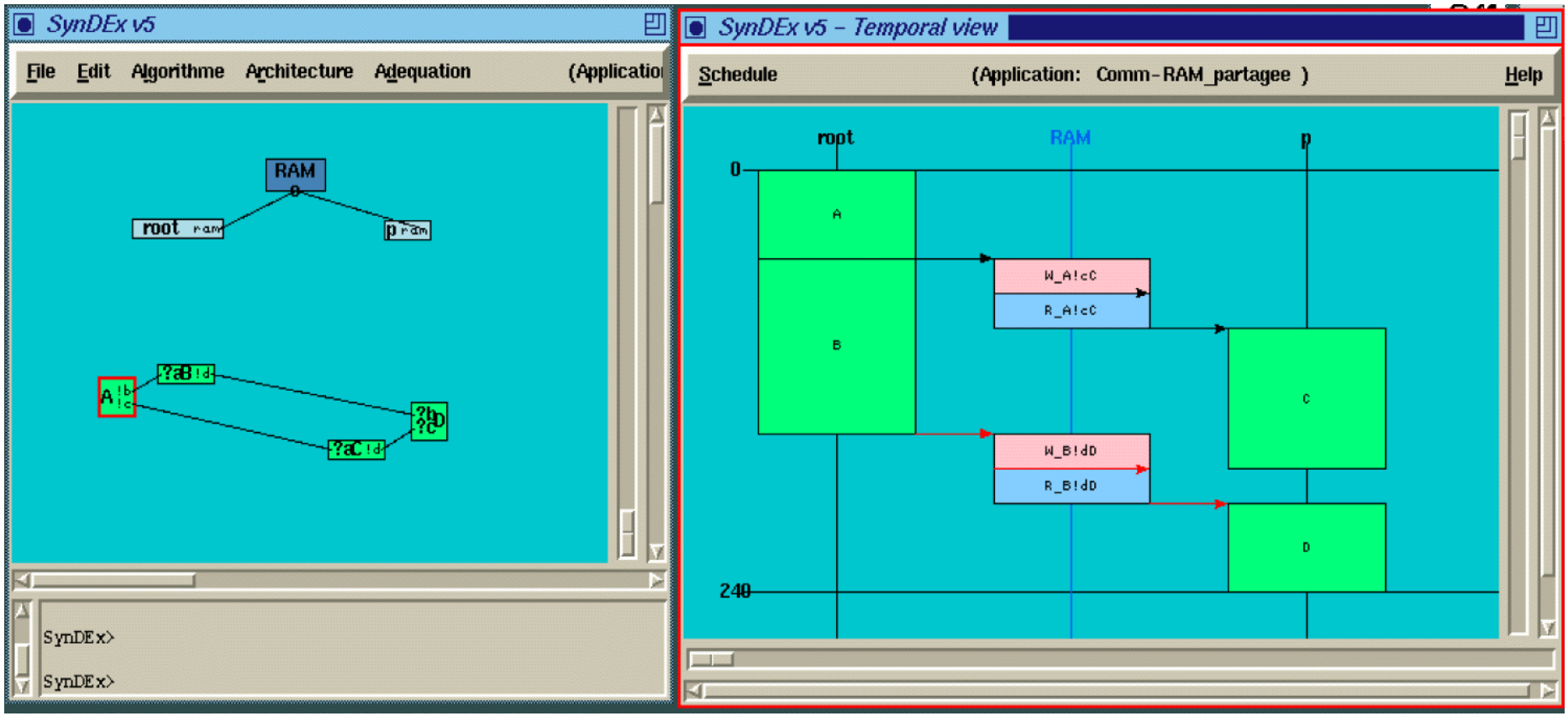
**R :Receive**

**N : Sync**

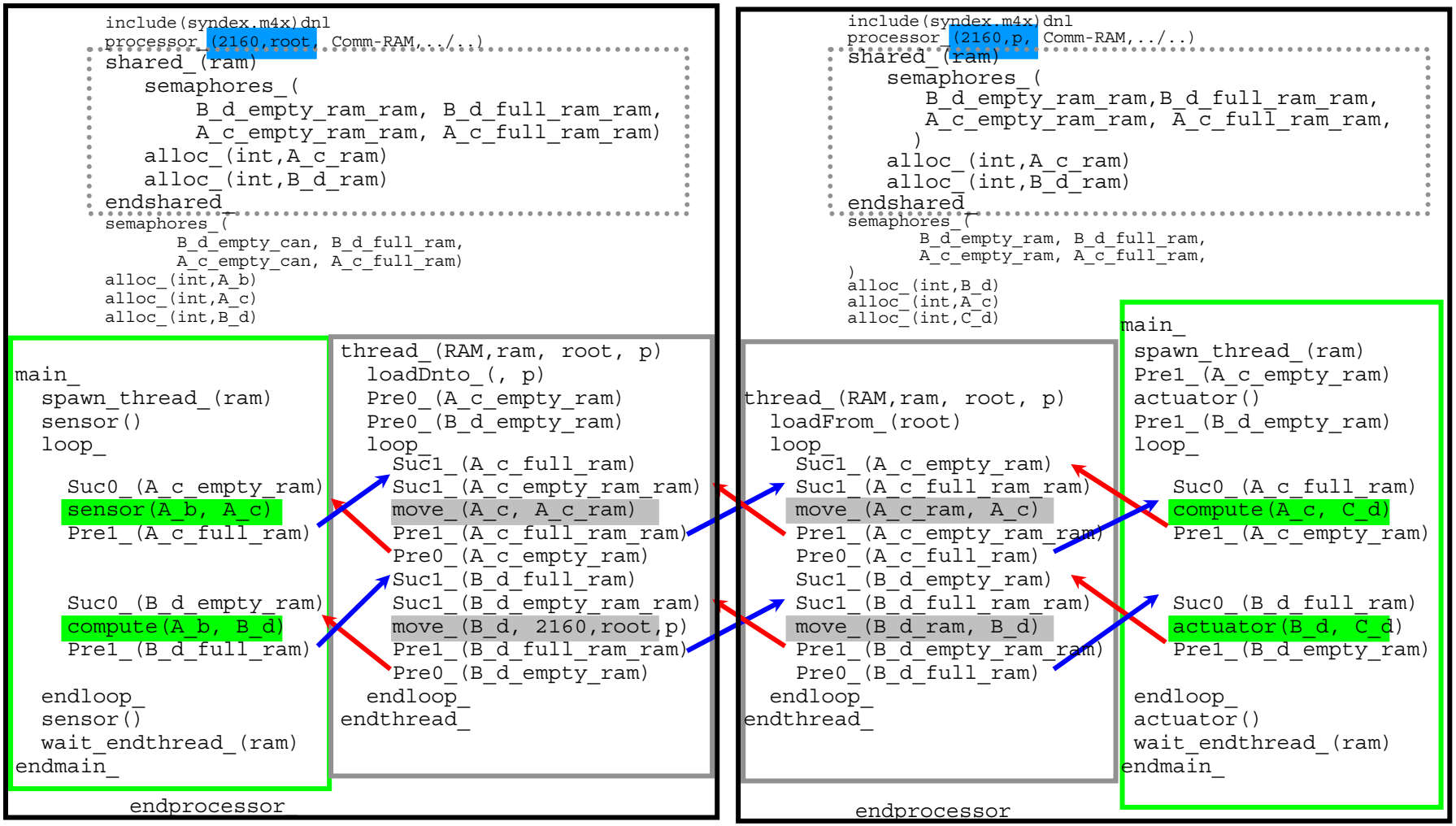
# Génération d'exécutif : Code



# Génération d'exécutif : Comm. Mém. Partagée



# Génération d'exécutif : Comm. Mém. Partagée



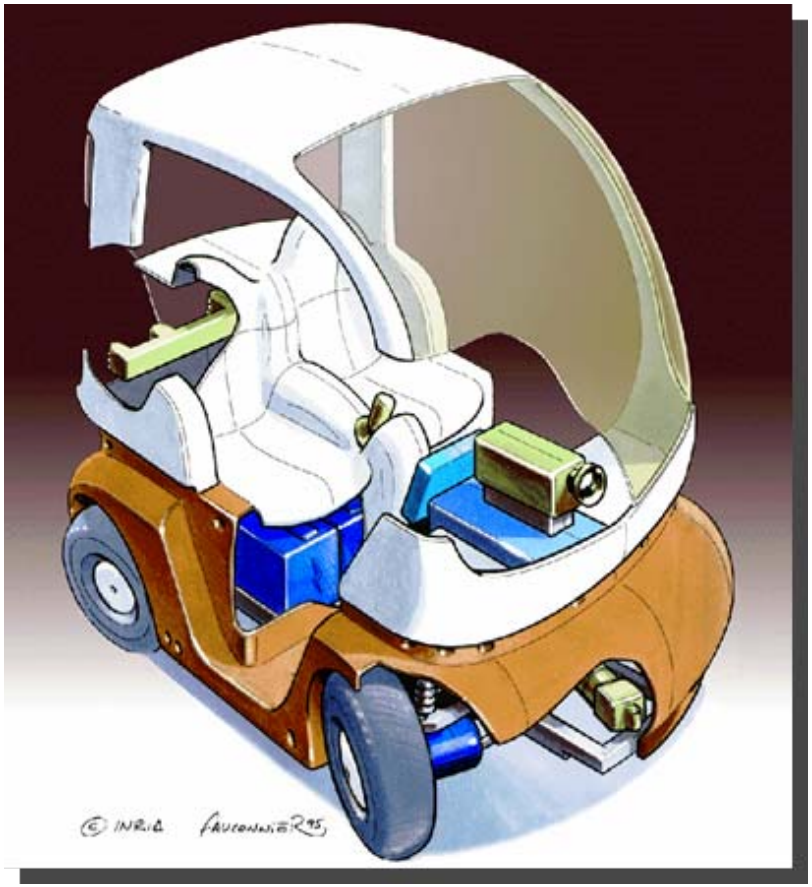
# Configuration d'exécutifs standards

- **Sans interblocage**
- **Traduction du macro-code en fichier de configuration d'exécutif standard**
- **Exécutif standards**
  - RT-Linux, RT-AI
  - Osek
  - etc

# Logiciel CAO système : SynDEx

- **Concrétise la méthodologie AAA**
- **Interface avec langages orientés métiers**
- **Graphique interactif Unix X11 ou Windows**
- **Edition graphes algorithme et architecture**
- **Distribution et ordonnancement statiques**
- **Visualisation diagramme temps réel prédit**
- **Mesures performances temps réel**
- **Génération exécutifs temps réel distribués**

# Exemple CyCab

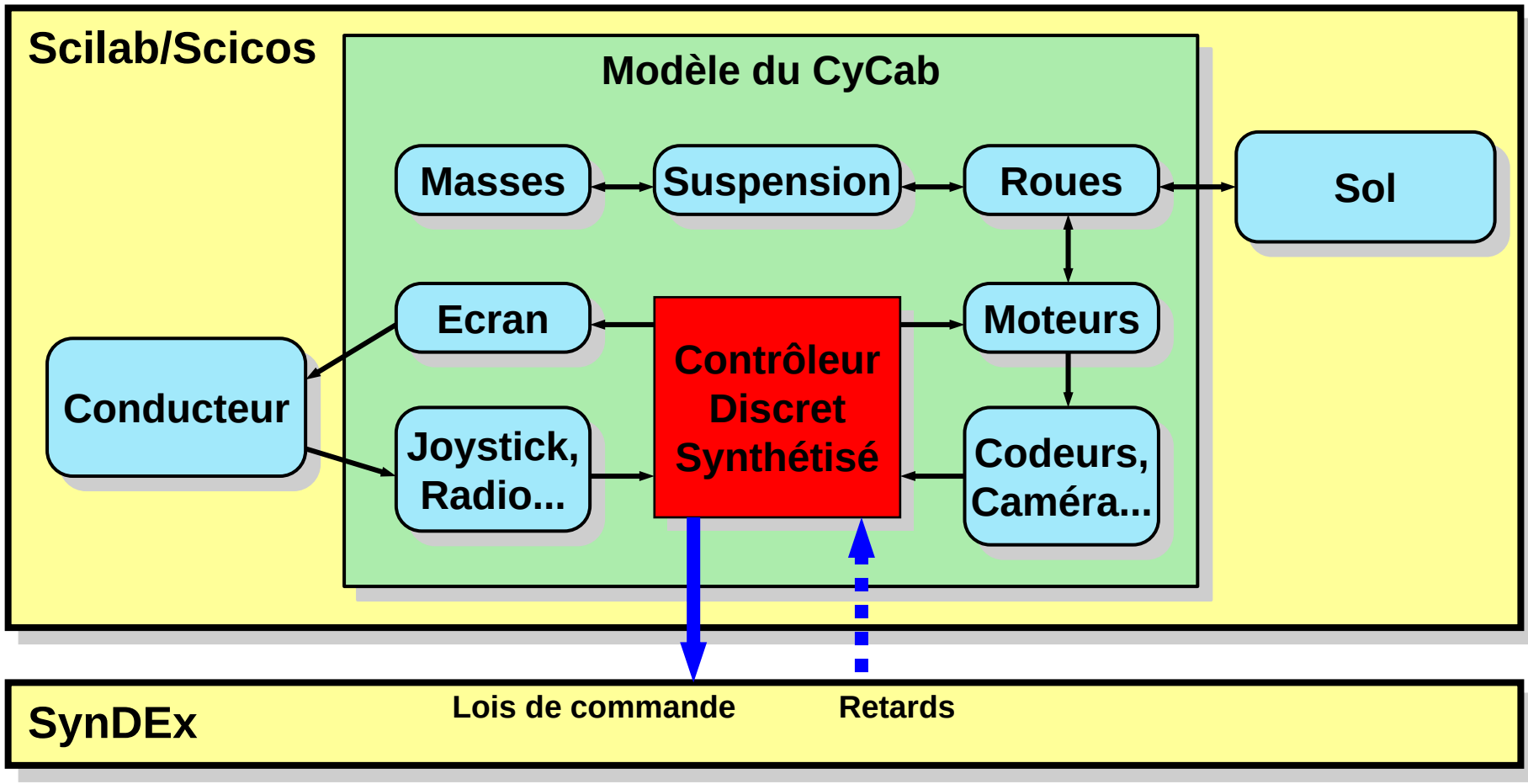


- Vitesse 30km/h
- Moteurs électriques
- 4 roues motrices
- 2 directions AV, AR
- Multi-processeur MPC555 + un Pentium
- Bus Can

Industrialisé par Robosoft
www.robosoft.fr

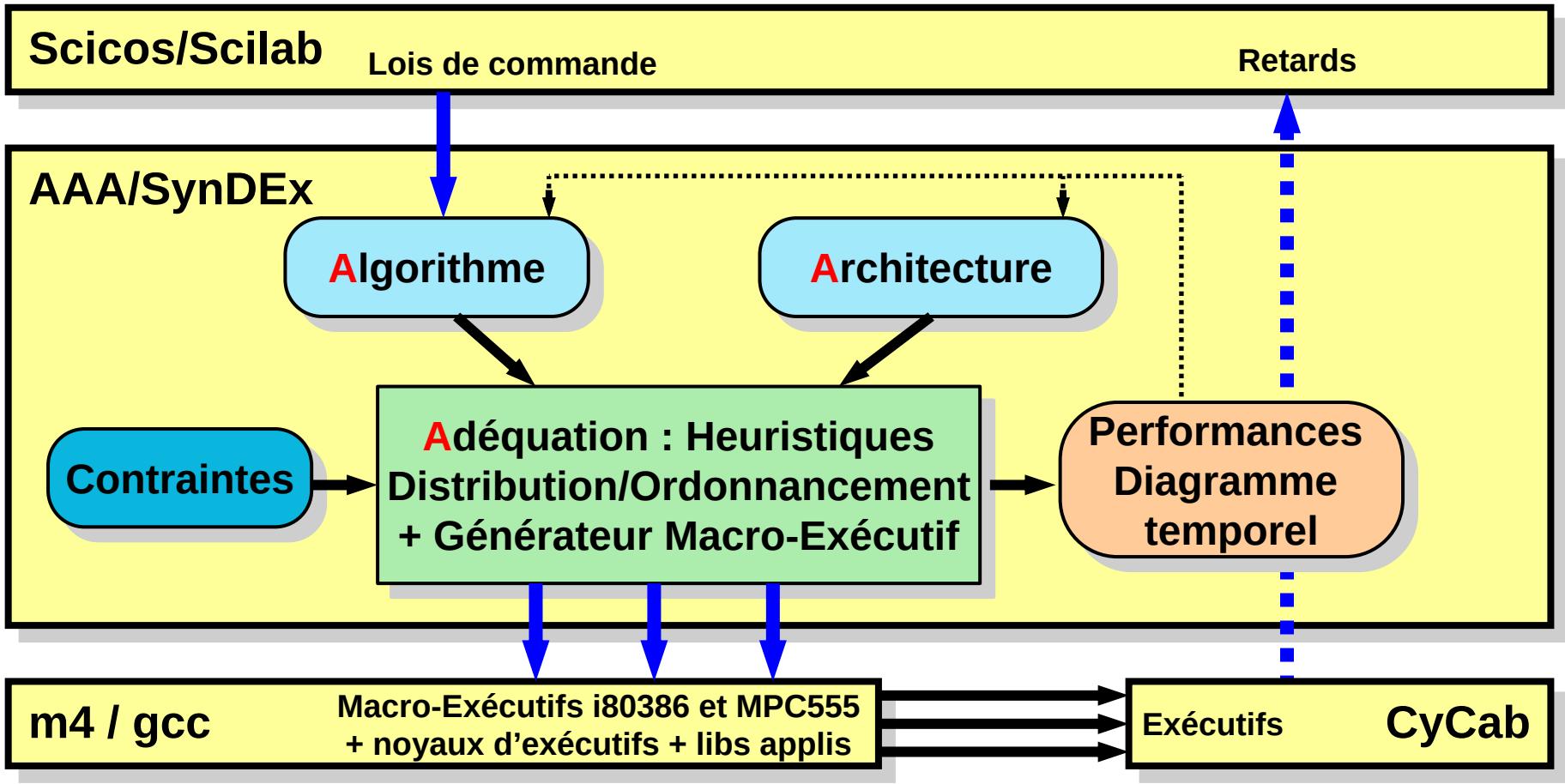
# Modélisation/Simulation Scilab/Scicos

**Gratuit sur : www.scicos.org**

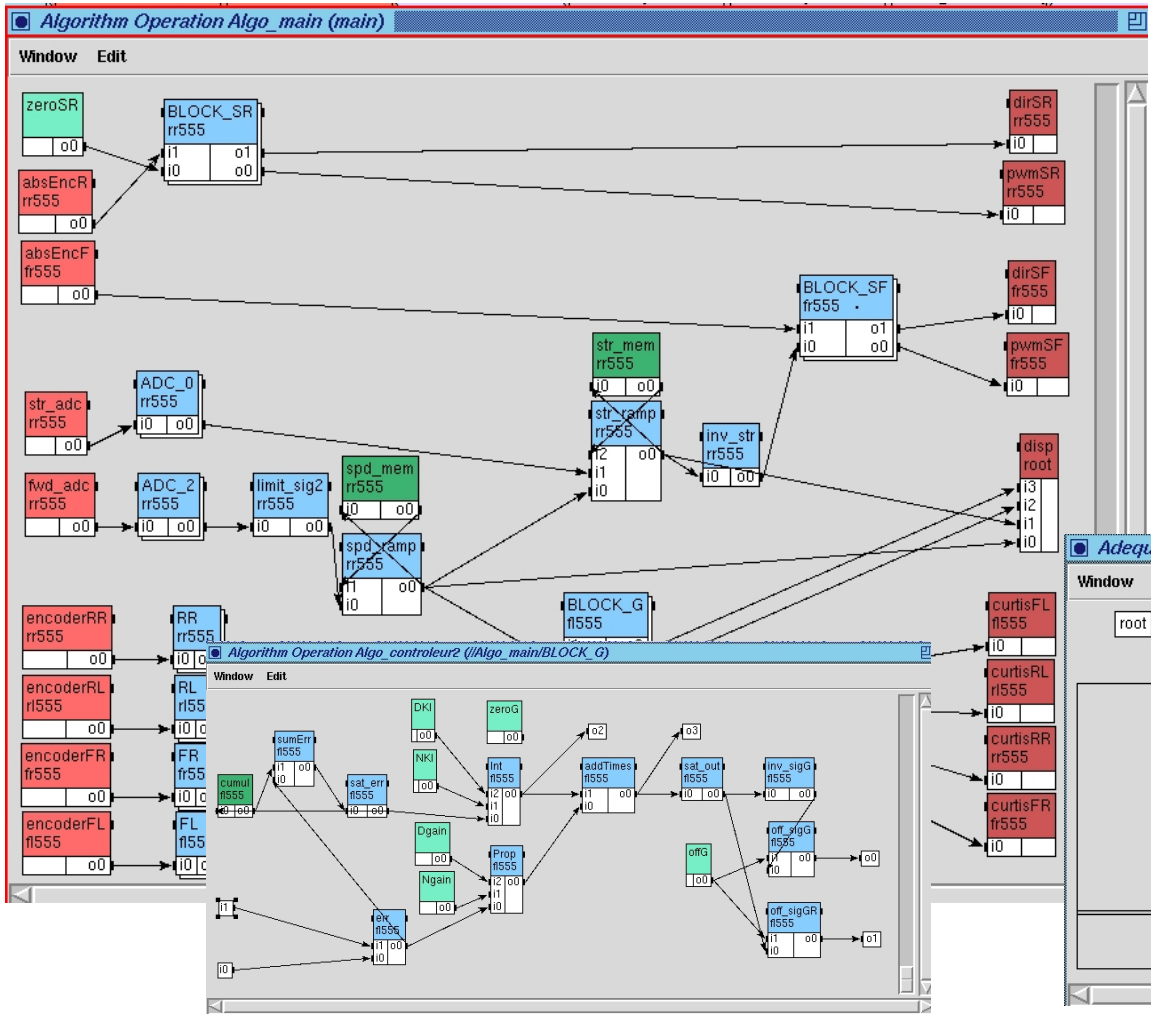


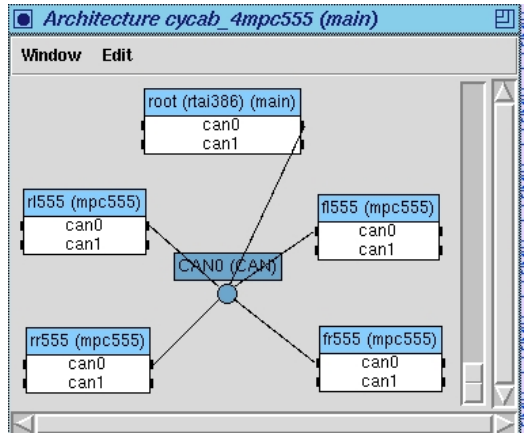
# Implantation AAA/SynDEx

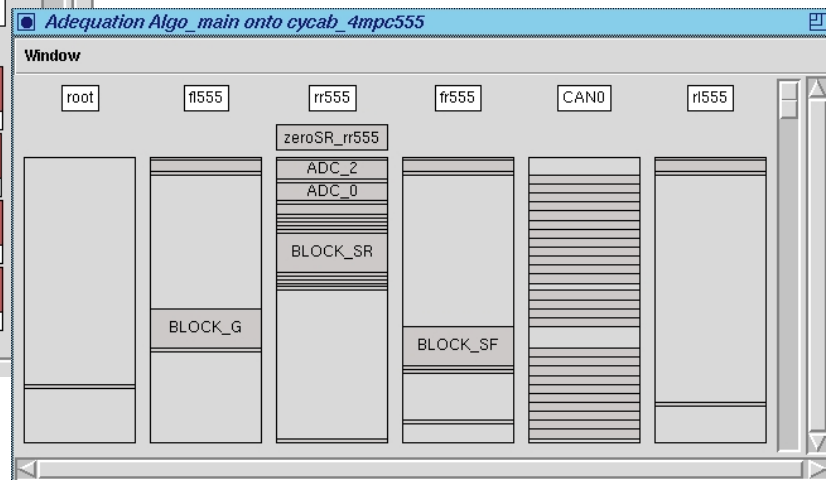
Gratuit sur : www.syndex.org



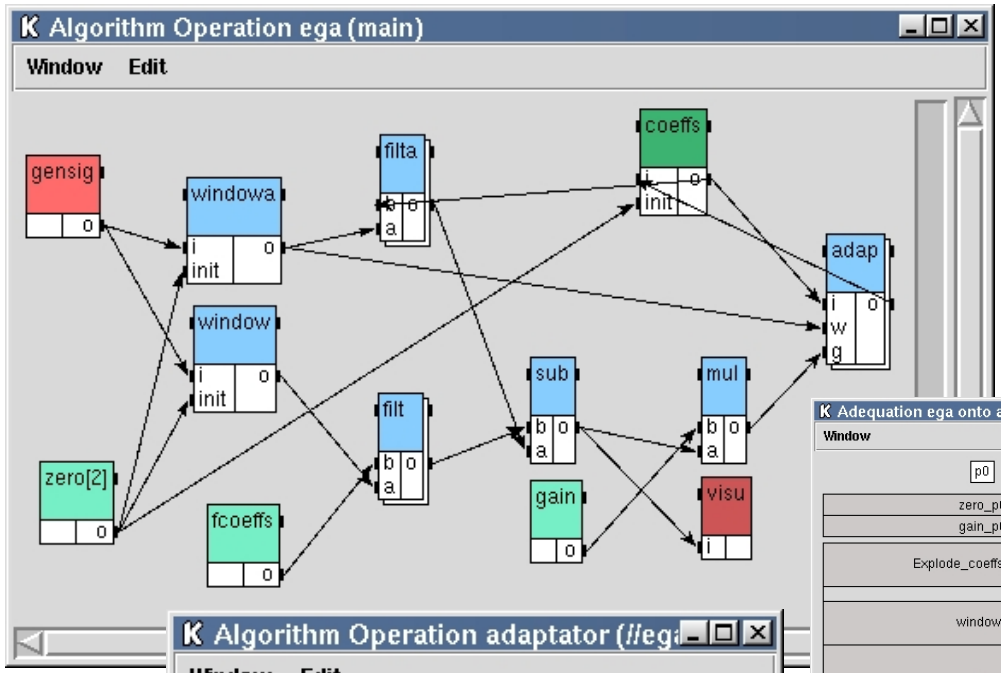
# Application de conduite Manuelle CyCab

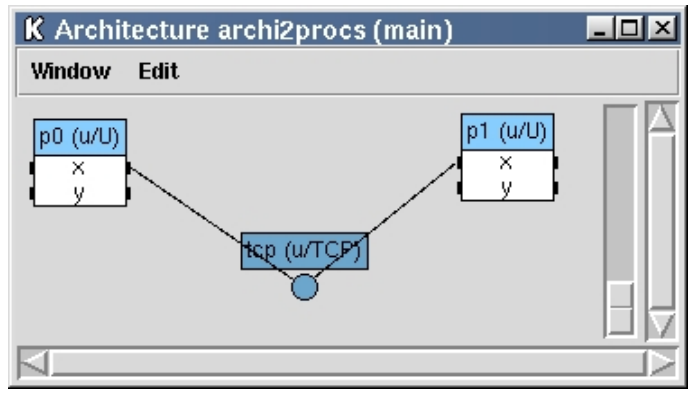


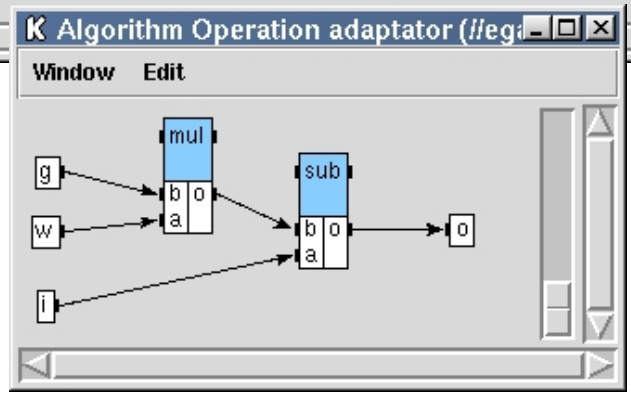



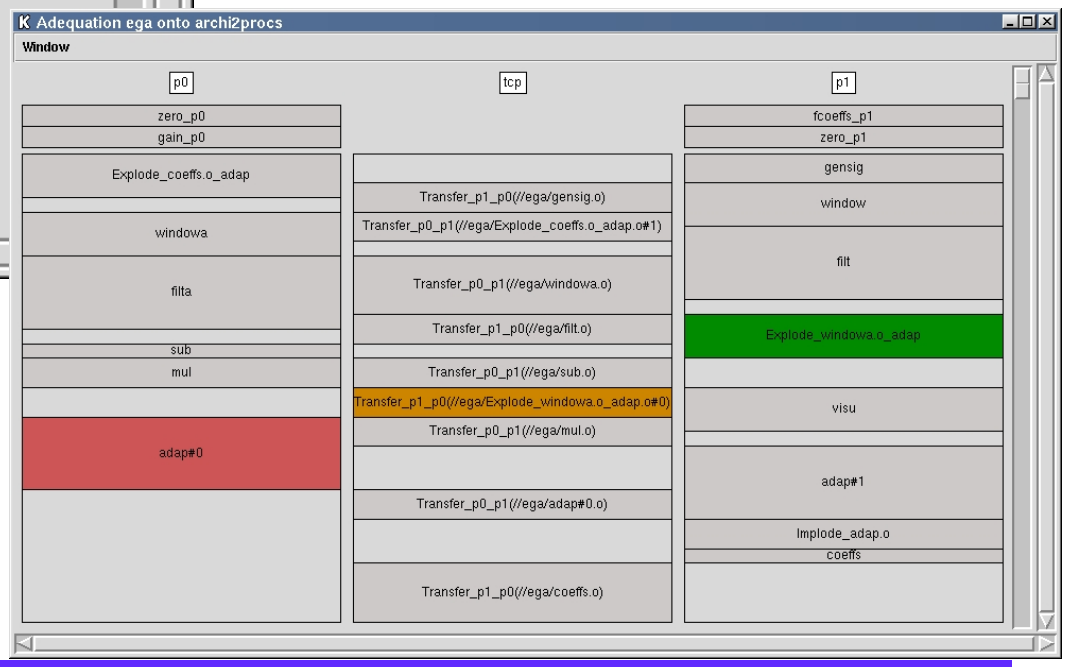



# Application d'égaliseur adaptatif

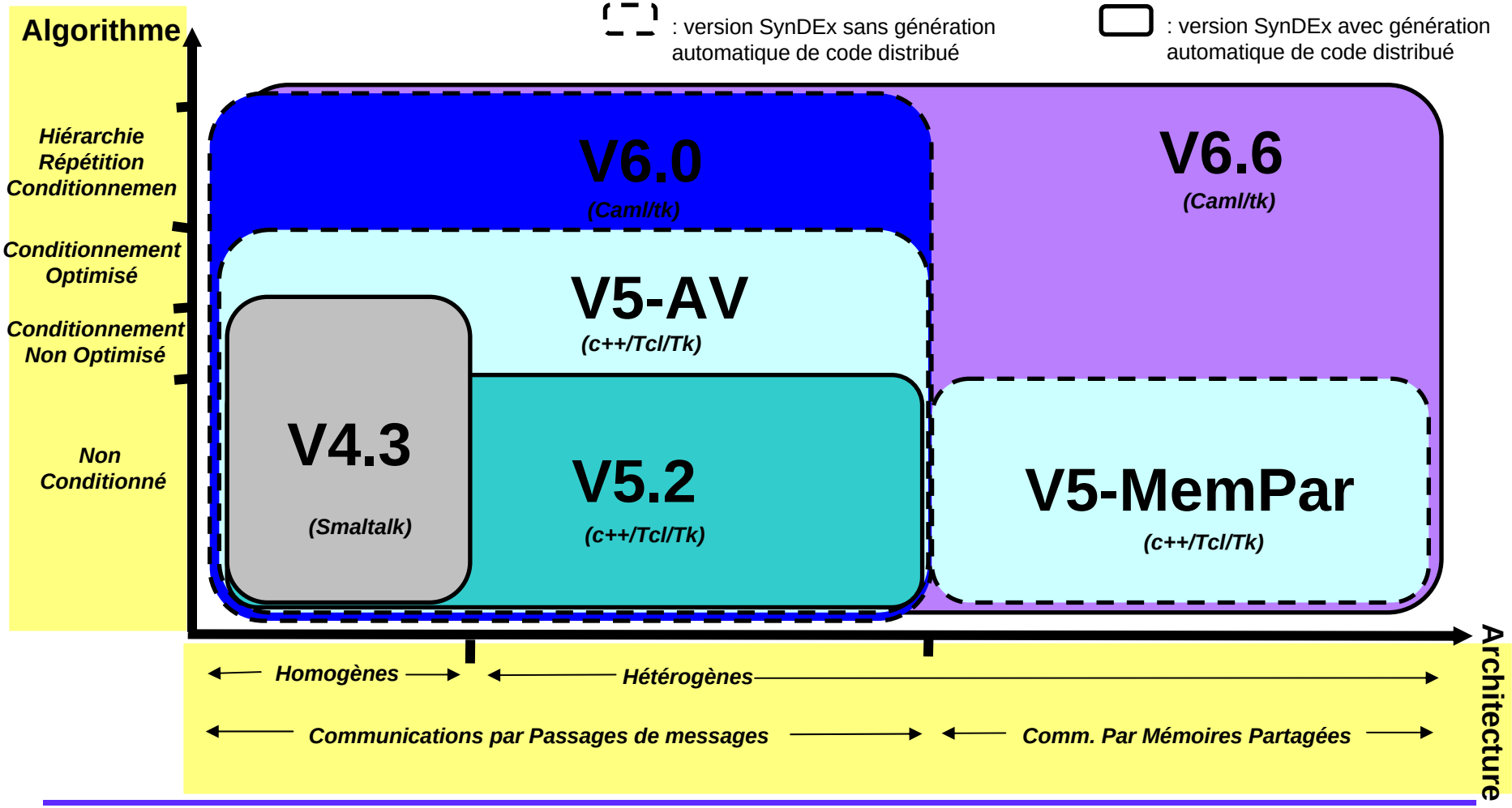


# SynDEx évolution des versions



# Synthèse de circuit optimisée

- **Graphe algorithme transformé en graphe implantation**
  - Sommet factorisé = composant VHDL exécuté répétitivement sur données différentes
  - Hyperarc factorisé = signal VHDL
  - Composant/signaux contrôlés par compteurs/mux/registres
  - Synthèse des chemins de données et de contrôle
- **Optimisation**
  - Compromis surface/(latence=cadence)
  - Défactorisation : data parallélisme, pipe-line, retiming
  - Jusqu'à contraintes temps réel satisfaites
- **SynDEx-IC collaboration A2SI ESIEE**

# Conclusion

- **Réduction temps de développement**
  - Sécurité de conception
  - Prototypage rapide / implantation optimisée
  - Exécutifs sans deadlock, uniquement noyaux à dégoguer
- **Perspectives**
  - Multi-contrainte temps réel : latences et cadences (périodes)
    - Hors-ligne avec et sans préemption, périodicité stricte ou gigue
    - En-ligne pour apériodique : slot shifting, adaptive scheduling
  - GALS : systèmes globalement. async. localement sync.
  - Couplage SynDEx/SynDEx-IC environ. co-développement
  - Partitionnement log./mat. automatique
  - Tolérance aux fautes : SynDEx-Tol collaboration POP-ART