TaskMapper

Outil de placement de tâches pour systèmes multiprocesseurs sur puce

Gestion de projet : Réalisation

P. Combier, V. Comiti, **M. Hubert**, R. Jamet, M. Le Du, **P. Lelouette**, J. L'Hermitte, **A. Morvan**, N. Premillieu, **L. Ren**, C. Souti, F. Tesniere, Y. Zhao, **S. Mellah**

Encadrés par S. Derrien

15 mai 2008

Plan

- Introduction
- Présentation
- Vue Utilisateur
- Réalisation
- Gestion de projet
- Conclusion

Introduction

Un client veut construire la maison de ses rêves, mais ne sait pas comment faire. Il fait donc appel à des ouvriers.

Introduction

- Deux types d'ouvriers :
  - Spécialisés
  - Polyvalents, moins performants
- Comment organiser les travaux ?
- Mauvais choix = conséquences néfastes.
- Idéal : réaliser toutes les maisons, choisir la meilleure

Introduction

- Comparaison avec création d'un circuit de décodage de lecteur MP3 :
  - Ouvriers = ressources
  - Travaux = tâches
  - Allocation des tâches aux ressources, de plusieurs façons, plus ou moins optimales.

Plan

- Introduction
- Présentation
- Vue Utilisateur
- Réalisation
- Gestion de projet
- Conclusion

Rappels

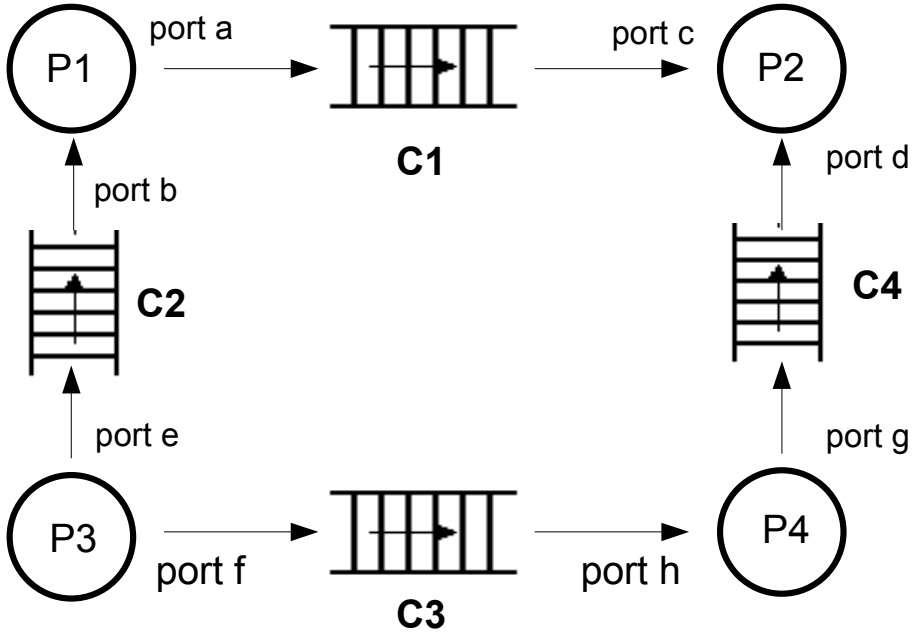
Qu'est-ce qu'un système embarqué ?

- Système autonome électronique et informatique
- Dédié à une tâche
- Ressources limitées
- Omniprésent
- Contraintes de conception (coût, performance, consommation)
- Implantation sur des *SoC : System on a Chip*

Rappels

Qu'est-ce qu'un Réseau de processus de Kahn?

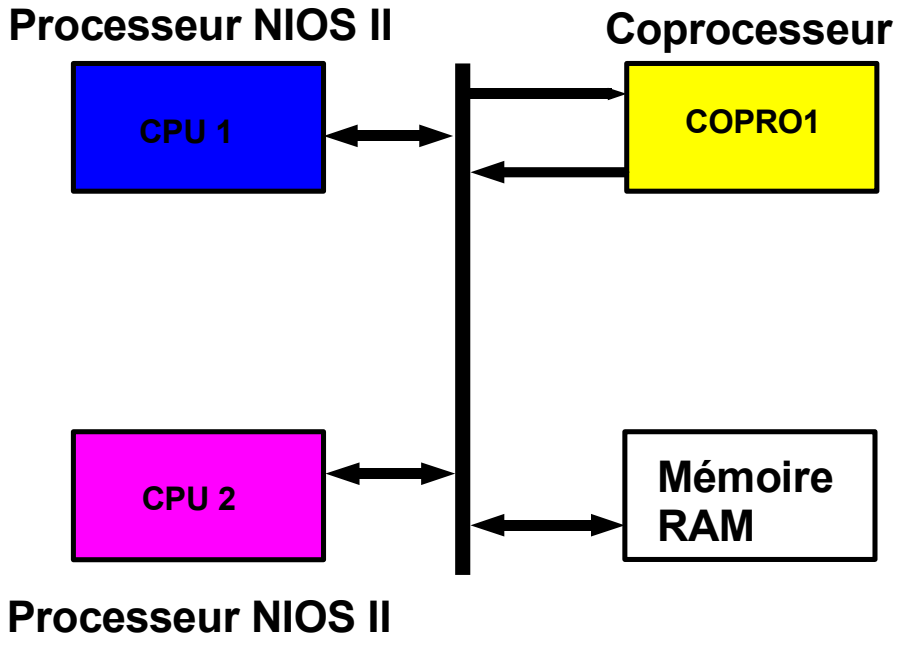
- Ensemble de tâches
- Notion de parallélisme
- Communication via des FIFOs
  - taille infinie
  - lecture bloquante
- Exécution déterministe
- Adapté à la conception de systèmes embarqués



Rappels

Qu'est-ce qu'un *mpSoC*?

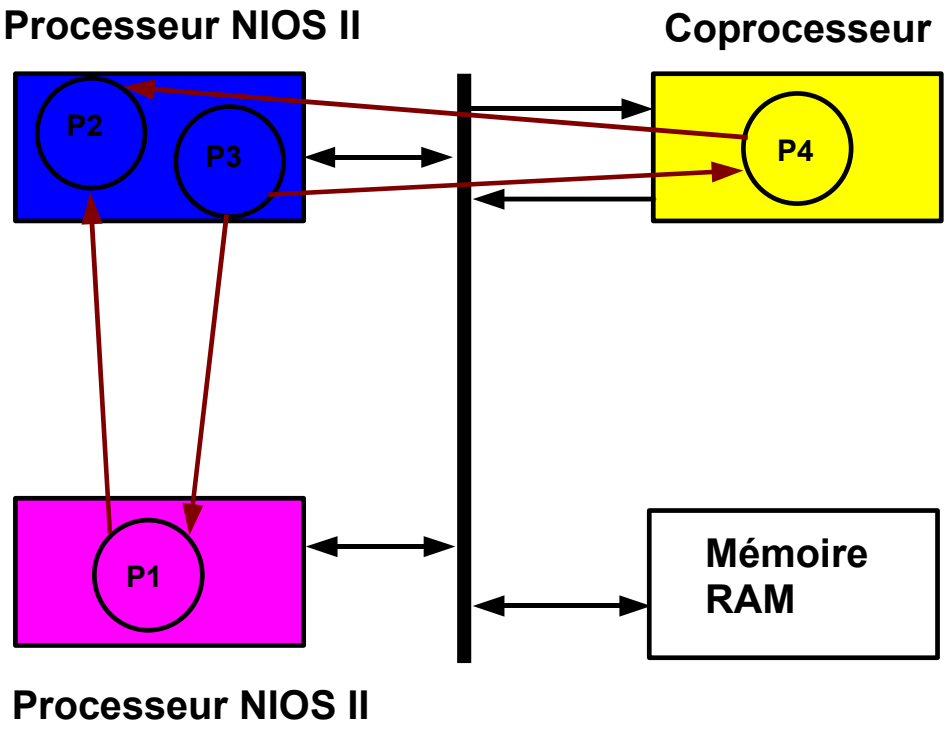
- *Multiprocessor System on a Chip*
- Différentes familles de processeurs
- Utilisation de coprocesseurs



Objectifs

1. Outil d'édition
   - Modèle d'architecture *mpSoC*
   - Modèle d'application sous forme de RPK
   - Allocation des tâches sur les ressources

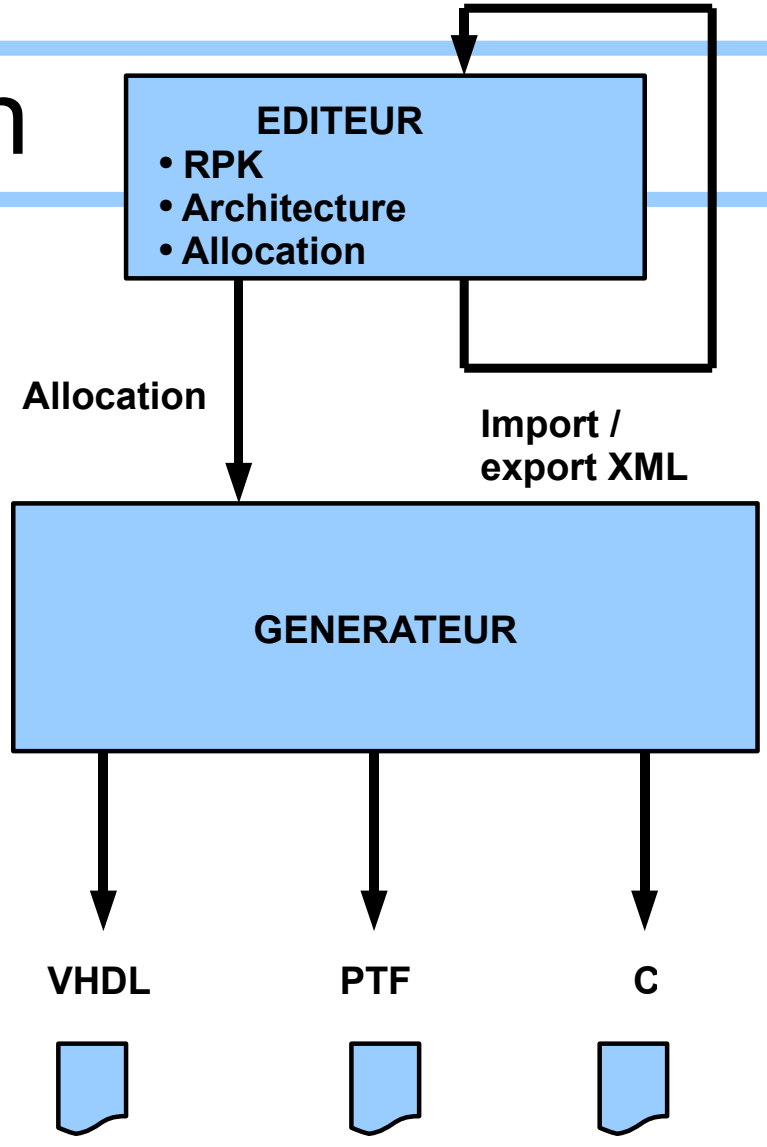
2. Outil de génération de code



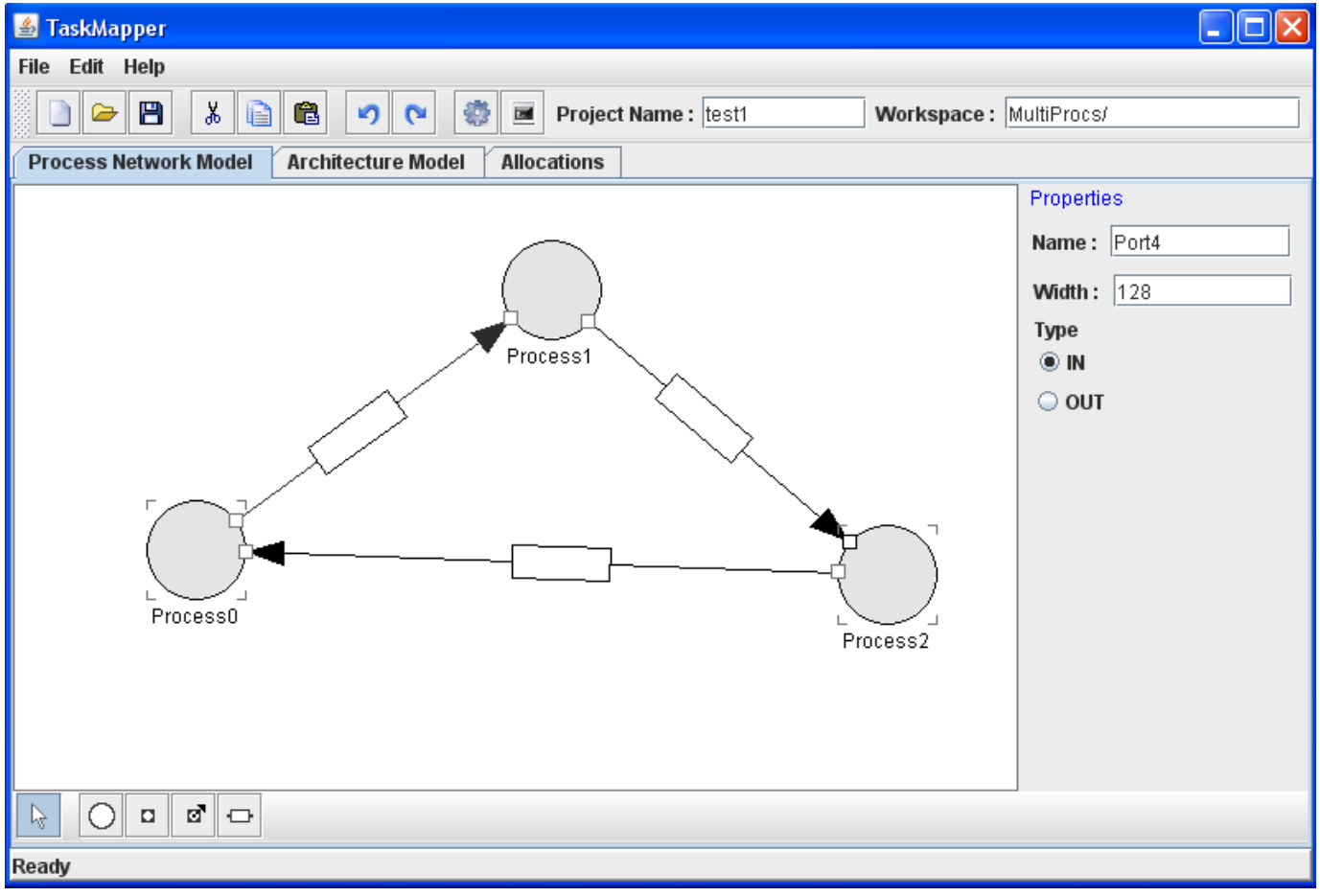
Plan

- Introduction
- Présentation
- Vue Utilisateur
- Réalisation
- Gestion de projet
- Conclusion

Présentation

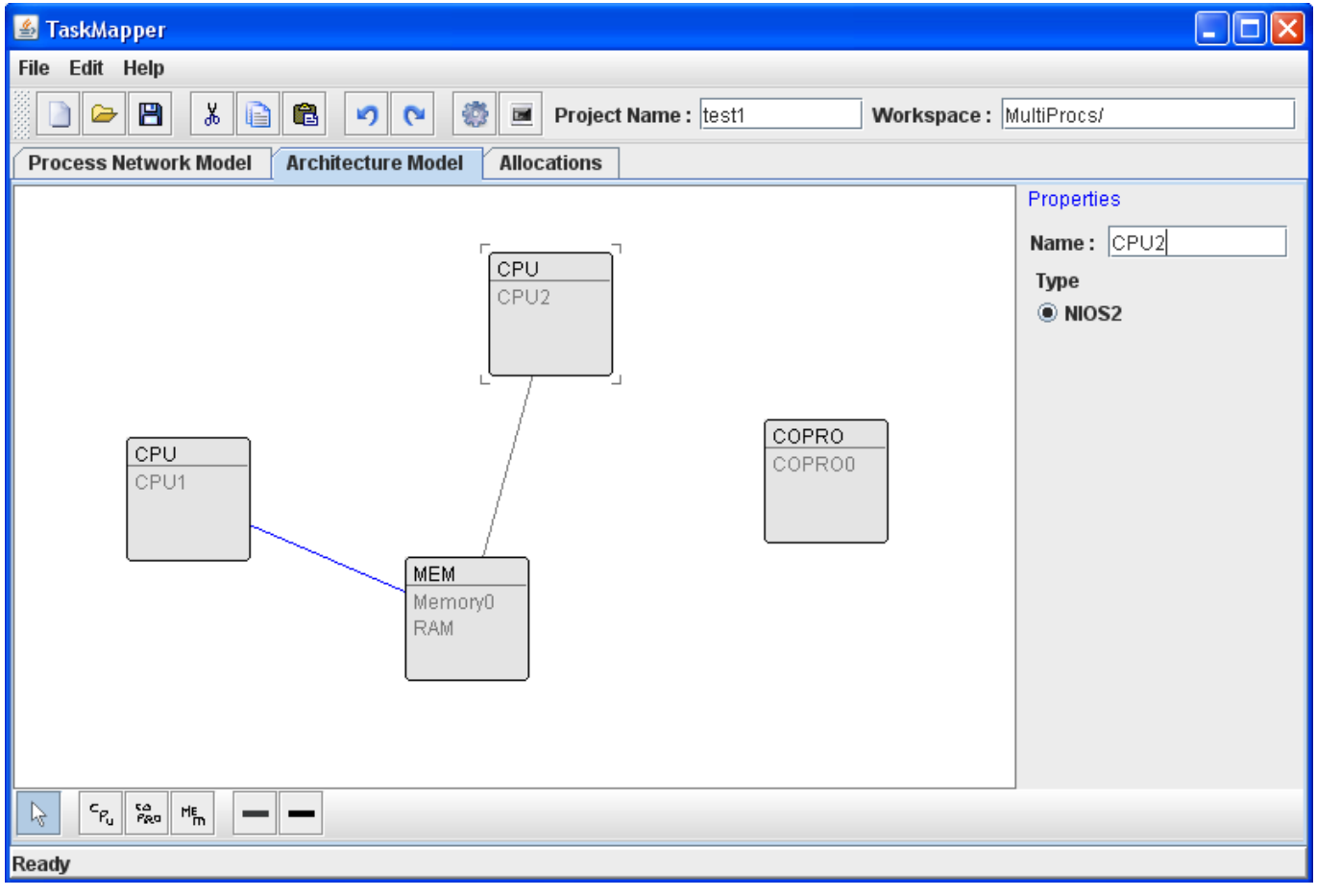


Editeur du RPK



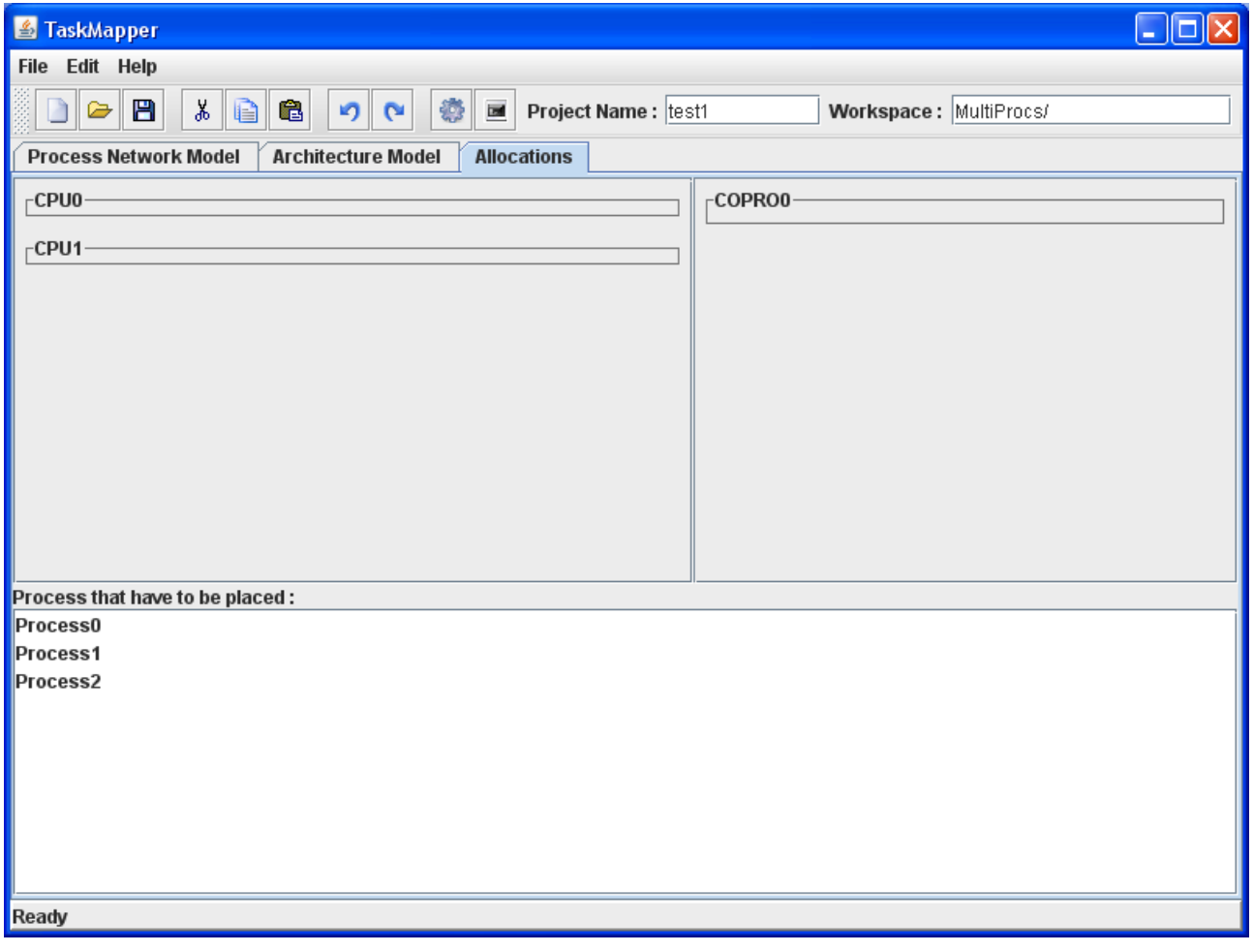
- Gestion des tâches par glisser-déposer
- Champ propriétés
  - Tâches
  - FIFOs
  - Ports
- Import / Export de modèles de RPK

Editeur du modèle d'architecture



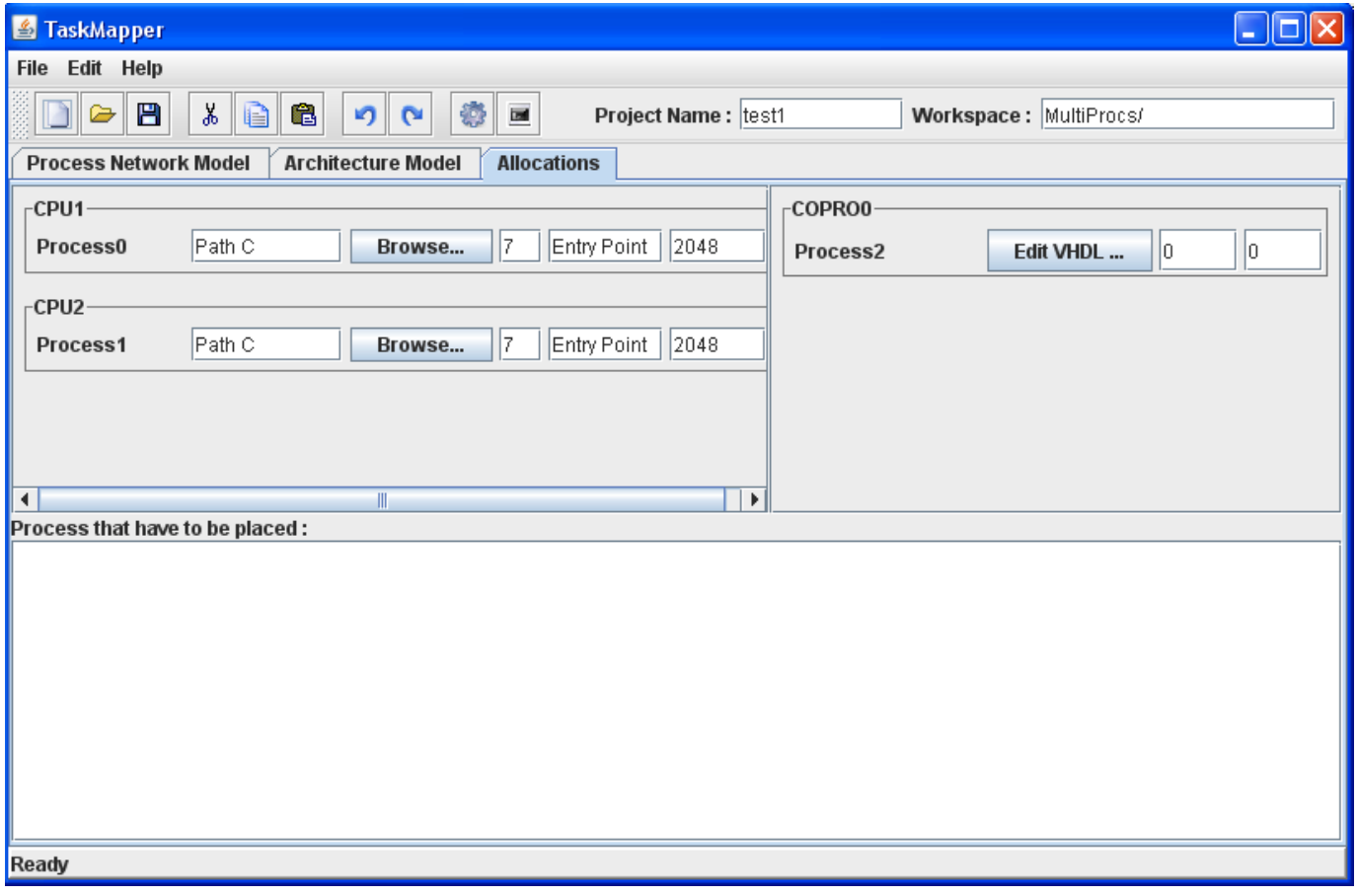
- Gestion des composants par glisser-déposer
- Champ propriétés pour tous les composants
- Import / Export de modèles d'architecture

Editeur d'allocation



- Listes des différents tâches, processeurs, coprocesseurs

Editeur d'allocation



- Allocation par glisser-déposer

Plan

- Introduction
- Présentation
- Vue Utilisateur
- Réalisation
- Gestion de projet
- Conclusion

Plan

- Réalisation
  - Modèle-Vue-Contrôleur
  - Générateur de code
    - C
    - VHDL
    - PTF
  - Implantation

Utilisation du Modèle-Vue-Contrôleur

- Adapté aux développements complexes
- Architecture logicielle connue et documentée

- Séparation en 3 parties
  - modèle = données utilisées
  - vue = interface graphique
  - contrôleur = synchronisation entre la vue et le modèle

➔ Développement indépendant des différentes parties de TaskMapper

Modèle du MVC

Modèle de graphe : modèle entité-relation
- ➔ « moule » du graphe d'objets

Gestionnaire du modèle
- Gère les données (tâches, processeurs, mémoires, ...)
- Assure l'intégrité du graphe d'objets lors des opérations
  - Insertion
  - Modification
  - Suppression

Vue du MVC

- Choix de conception
  - API standard de Java
  - Bibliothèque Swing
  - Une bibliothèque extensible : composants personnalisés

- Interface simple et sobre
  - Éléments courants (barre d'outils, menus, glisser-déplacer)
  - Composants personnalisés pour TaskMapper
    - Éditeur de modèle de RPK
    - Éditeur de modèle d'architecture

Contrôleur du MVC

- Gestionnaire d'événements qui assure la synchronisation entre :
  - Le modèle entité-relation
  - Le modèle de données propre à la vue

- Architecture du contrôleur :
  - Implémentation du patron de conception Command
  - Patron de conception connu car étudié en ADT
  - Relativement simple à mettre en place

Import / Export

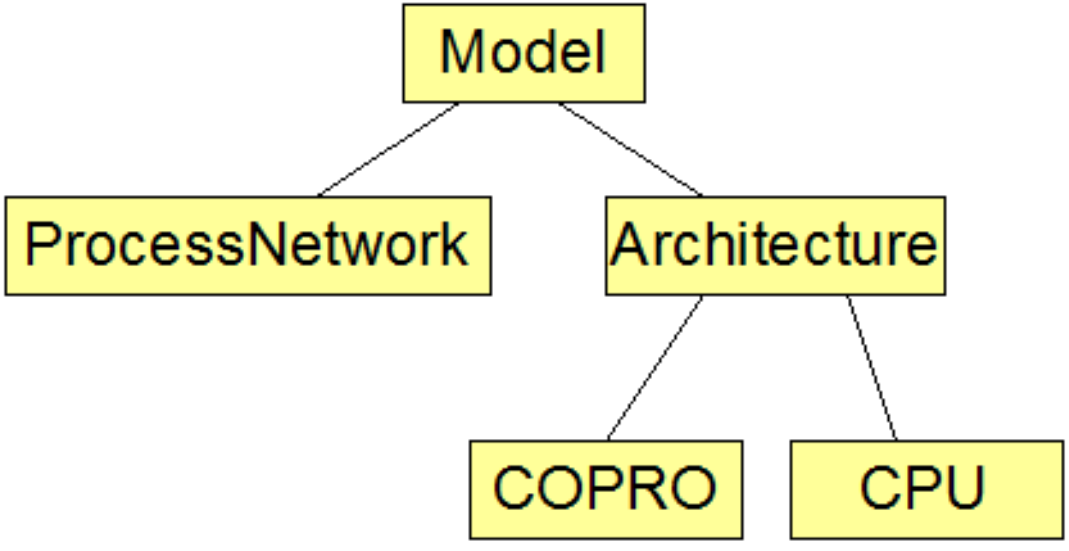
- Trois types d'import/export :
  - Modèle de RPK
  - Modèle d'architecture
  - Allocation

- Utilisation de la bibliothèque Xstream
  - gratuite et libre
  - simple d'emploi
  - sérialise / désérialise des graphes

Plan

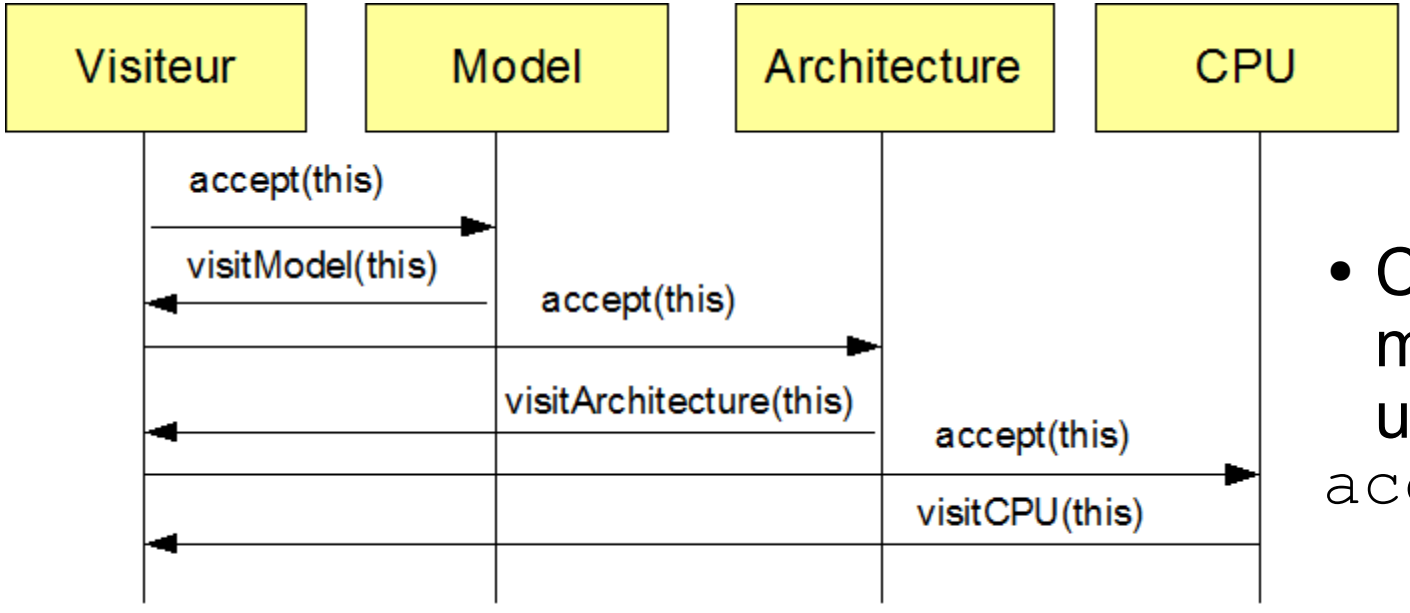
- Réalisation
  - Modèle-Vue-Contrôleur
  - Générateur de code
    - C
    - VHDL
    - PTF
  - Implantation

Utilisation du motif de conception Visiteur



- Multitude de traitements différents sur les éléments du modèle entité-relation
- Utilisation du motif de conception *Visiteur*

Utilisation du motif de conception Visiteur



- Chaque élément du modèle implémente une méthode `accept(Visiteur v)`

Les Visiteurs du générateur de code

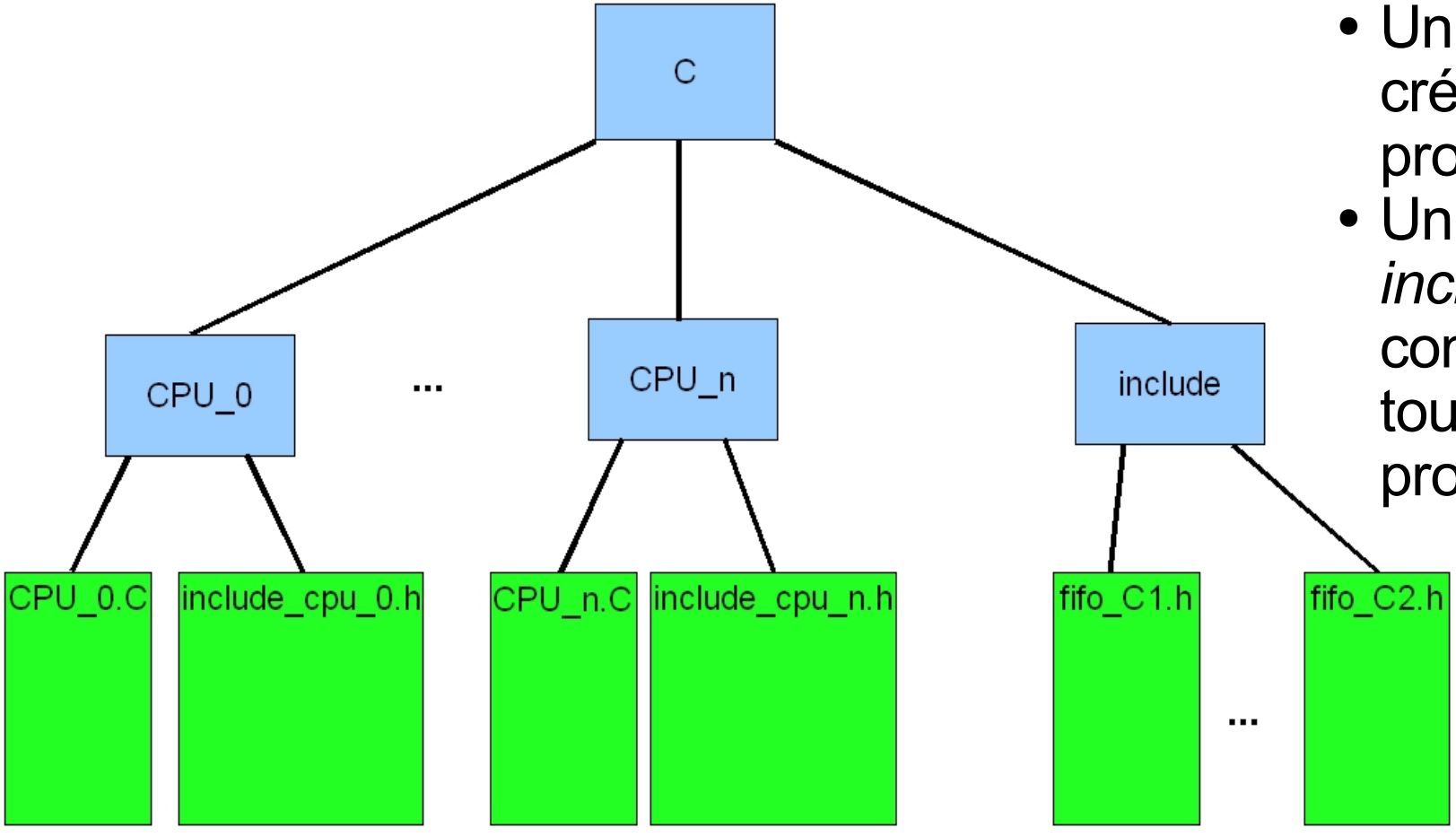
4 *Visiteurs* différents

- Visiteur qui génère les fichiers C
- Visiteur qui génère les fichiers VHDL
- Visiteur qui génère le fichier PTF
- Visiteur qui implante le projet dans le FPGA

Plan

- Réalisation
  - Modèle-Vue-Contrôleur
  - Générateur de code
    - C
    - VHDL
    - PTF
  - Implantation

Fichiers générés



- Un répertoire créé par processeur
- Un répertoire *include* commun à tous les processeurs

Fichier d'initialisation

- Un fichier d'initialisation par processeur (fichier *cpu_x.h*)
- Initialisation des données nécessaires aux FIFOs
- Initialisation et lancement des tâches, à l'aide de *µC-OS II*

Déclaration des FIFOs

- Un fichier généré pour chaque FIFO :
  - Constantes : *largeur* et *profondeur*
  - Indices *iPlein* et *iVide,* et le *nombre d'éléments*
  - *Mutex* et *sémaphores*

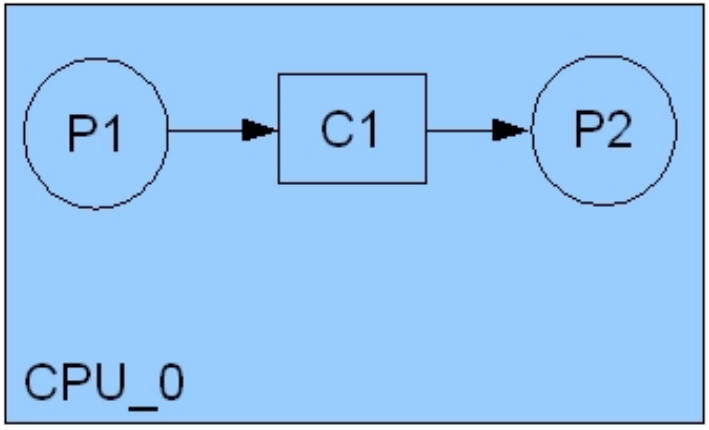
Fonctions de manipulation des FIFOs

- Un fichier par processeur
  - Définition des fonctions de communication

```
Write_In_c1(void* pdata)
Read_From_c1(void* pdata)
```

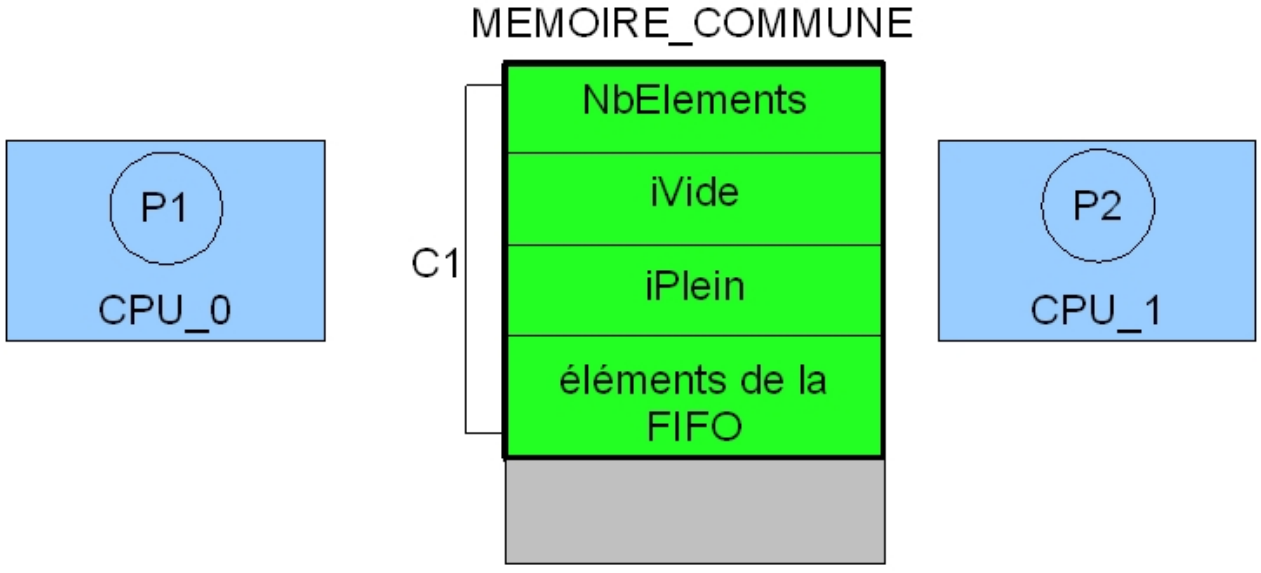
- 3 cas :
  - communications intra-processeur
  - communications inter-processeurs
  - communications processeur-coprocesseur

Cas 1 : communications intra-processeur



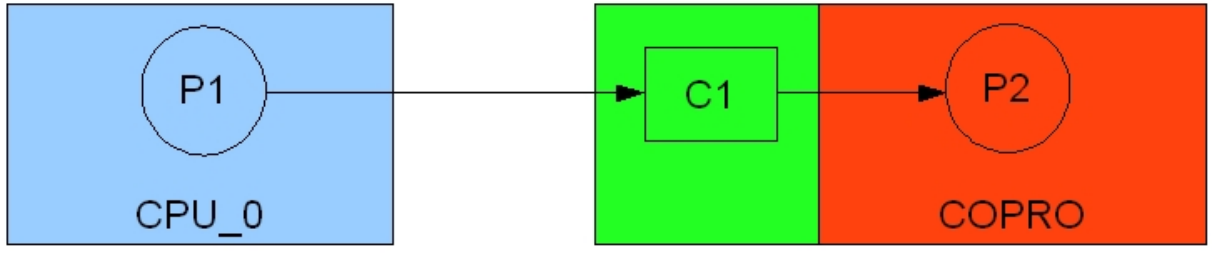
- Deux tâches sur le même processeur
- FIFO : tableau géré circulairement et protégé par un sémaphore logiciel

Cas 2 : communications inter-processeurs



- FIFO : dans une mémoire commune aux deux processeurs, protégée par un mutex matériel

Cas 3 : communications processeur-coprocesseur



- Nécessité d'utiliser une FIFO matérielle, manipulée à l'aide d'un pilote logiciel (*driver*)

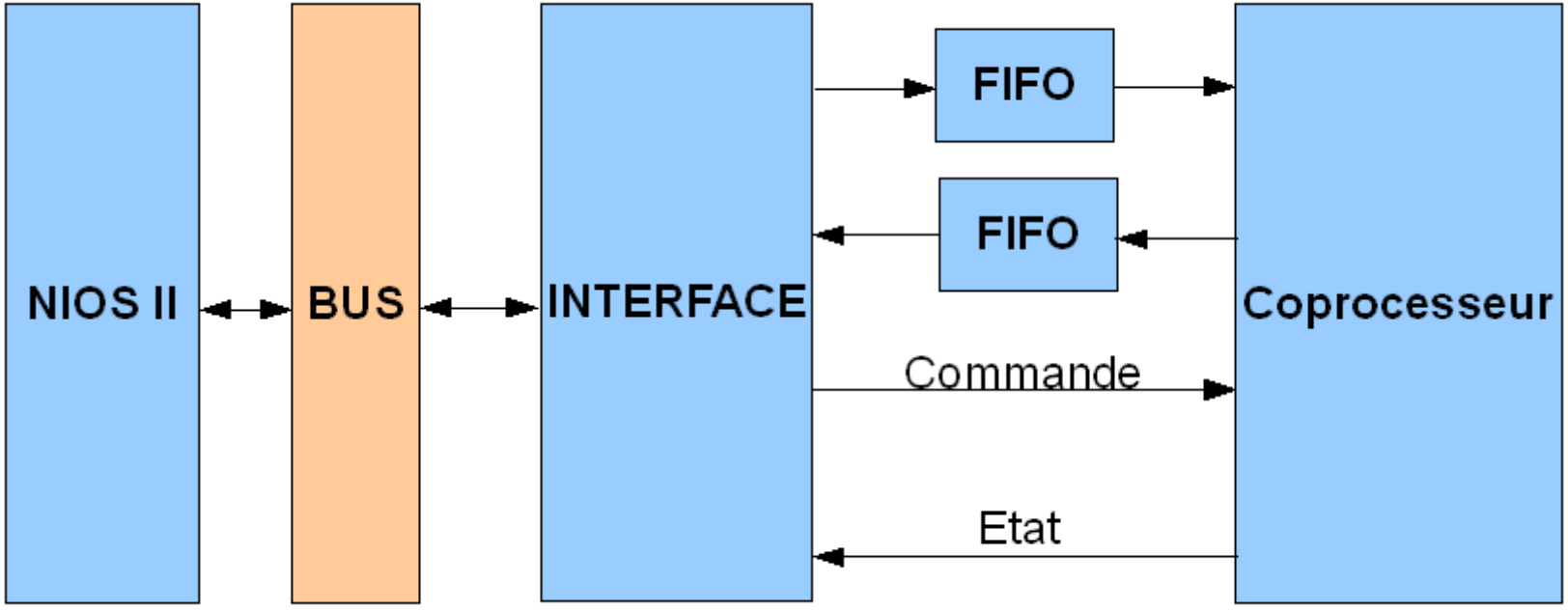
Plan

- Réalisation
  - Modèle-Vue-Contrôleur
  - Générateur de code
    - C
    - VHDL
    - PTF
  - Implantation

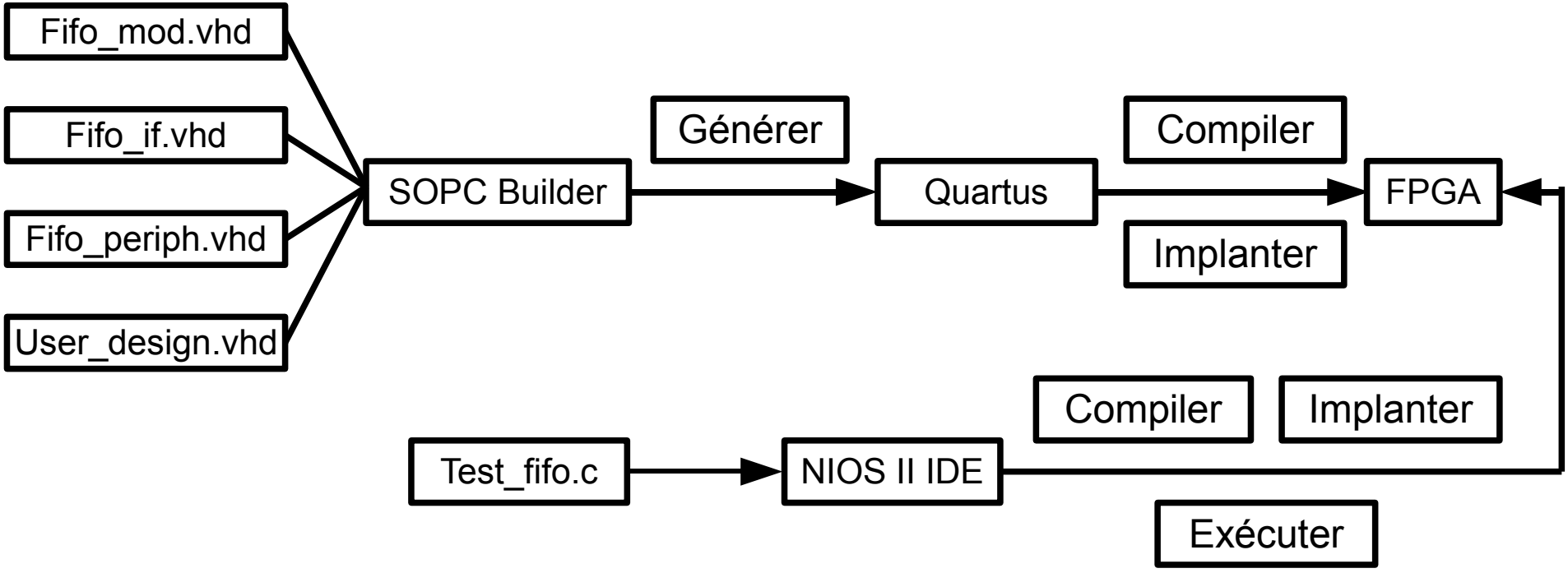
Nécessité d'une FIFO matérielle

- Coprocesseur : composant esclave
- Utilisation du langage VHDL

  ➔ Recherche d'un modèle valide

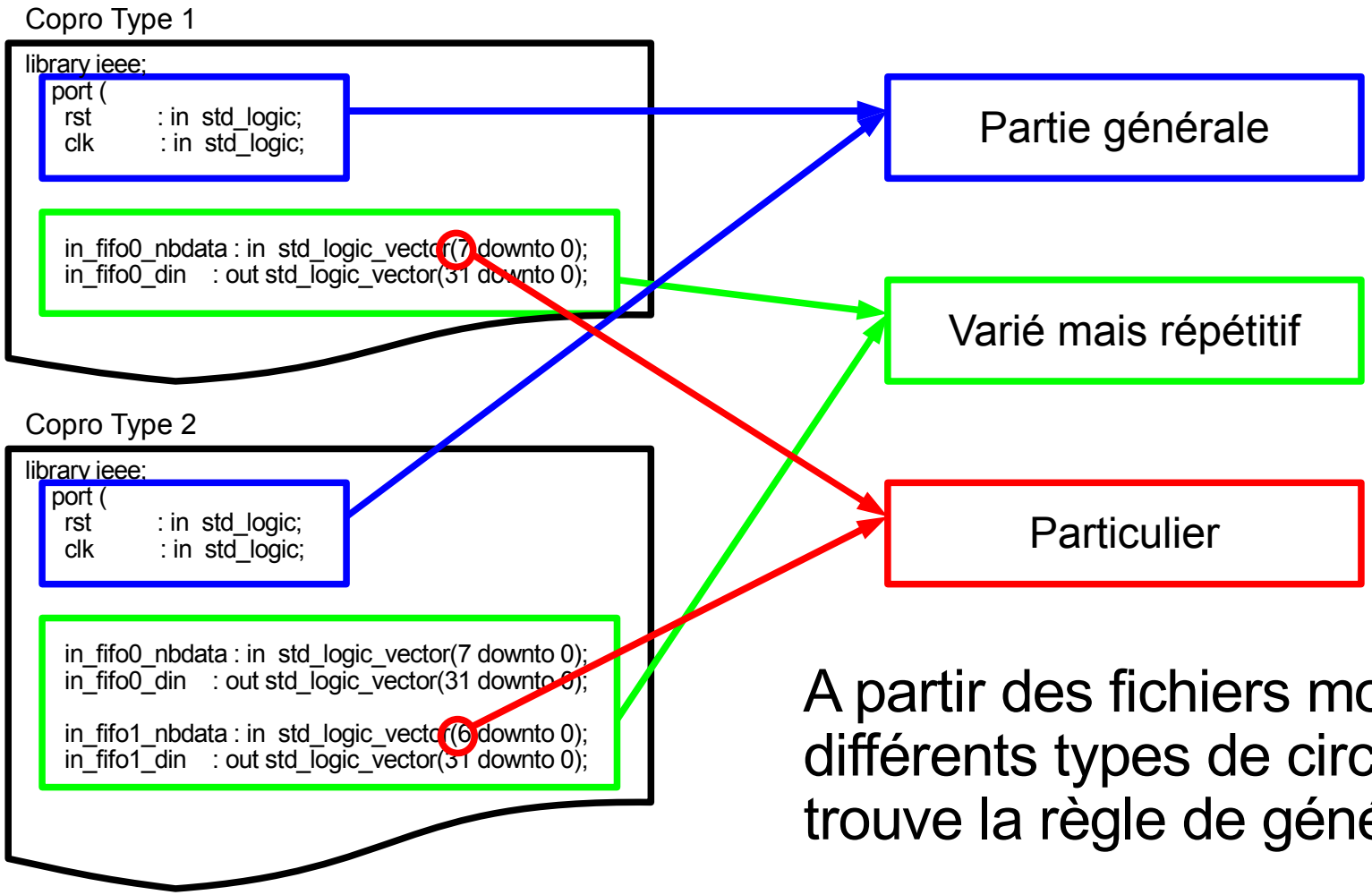


Recherche d'un modèle valide



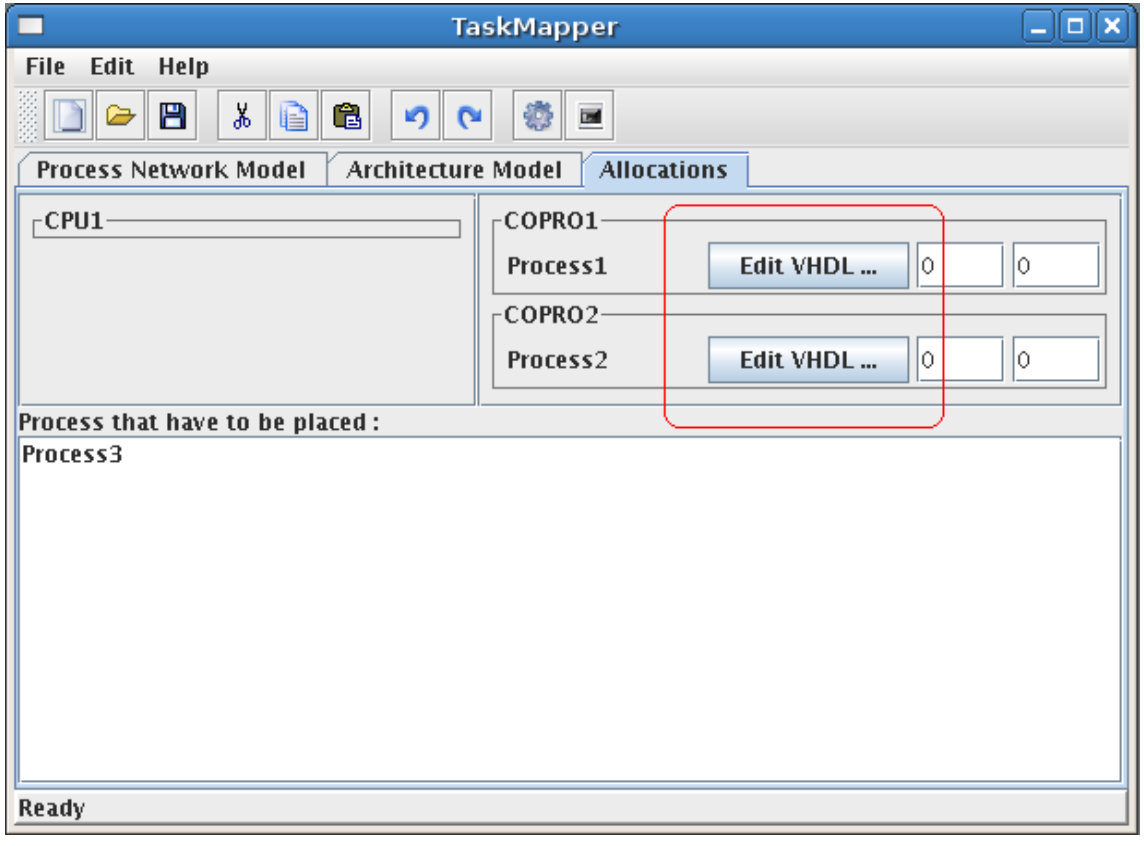
Procédure générale de mise en œuvre et de mise au point d'un périphérique personnalisé

Etude des fichiers



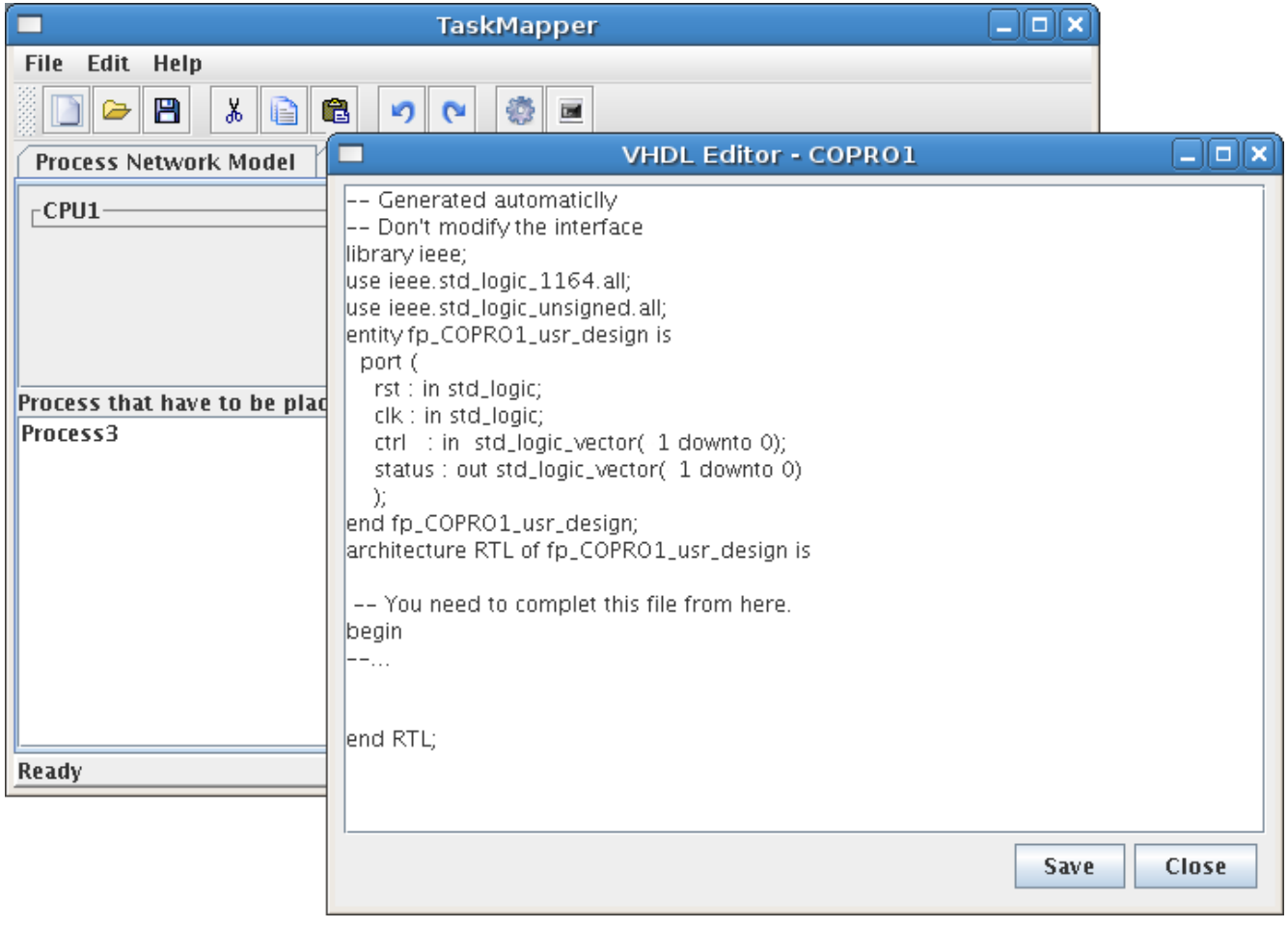
A partir des fichiers modèles de différents types de circuits, on trouve la règle de génération.

Création du squelette VHDL



- Création des squelettes VHDL lors de l'allocation
- Squelette éditable
- Évite des erreurs de nommage

Remplissage du squelette VHDL



- Création des squelettes VHDL lors de l'allocation
- Squelette éditable
- Évite des erreurs de nommage

Plan

- Réalisation
  - Modèle-Vue-Contrôleur
  - Générateur de code
    - C
    - VHDL
    - PTF
  - Implantation

Structure d'un fichier PTF

- Plusieurs imbrications de sections
- Composant = section MODULE
- Propriétés :
  - MASTER
  - SLAVE (définition des interconnexions)
- Bibliothèque de composants Altera
- Composants personnalisés : HDL_INFO

Génération du PTF de l'architecture

- Utilisation du motif Visiteur
- Fichiers PTF génériques avec mots clés (balises)
- Ajout de composants pour maintenir la cohérence (mutex)
- Inclusion dans un fichier PTF global
- Calcul des adresses de base

Plan

- Réalisation
  - Modèle-Vue-Contrôleur
  - Générateur de code
    - C
    - VHDL
    - PTF
  - Implantation

Partie architecture

1. SOPC_Builder

```
executeCmd("sopc_builder --no_splash -s -generate <chemin du
projet architecture>/<nom du projet
architecture>/system_0.ptf",false);
```

2. Quartus

```
executeCmd("quartus_cmd <chemin du projet architecture>/<nom du
projet architecture>.qpf -c <chemin du projet architecture>/<nom du
projet architecture>.qsf",false);
```

Partie logicielle

- Création d'un projet NIOS II IDE par processeur

  `nios2-create-application-project`

- Modification des fichiers de configurations
- Intégration des fichiers C générés et fournis par l'utilisateur
- Compilation via un compilateur Altera

  `nios2-build-project`

Communication avec la carte

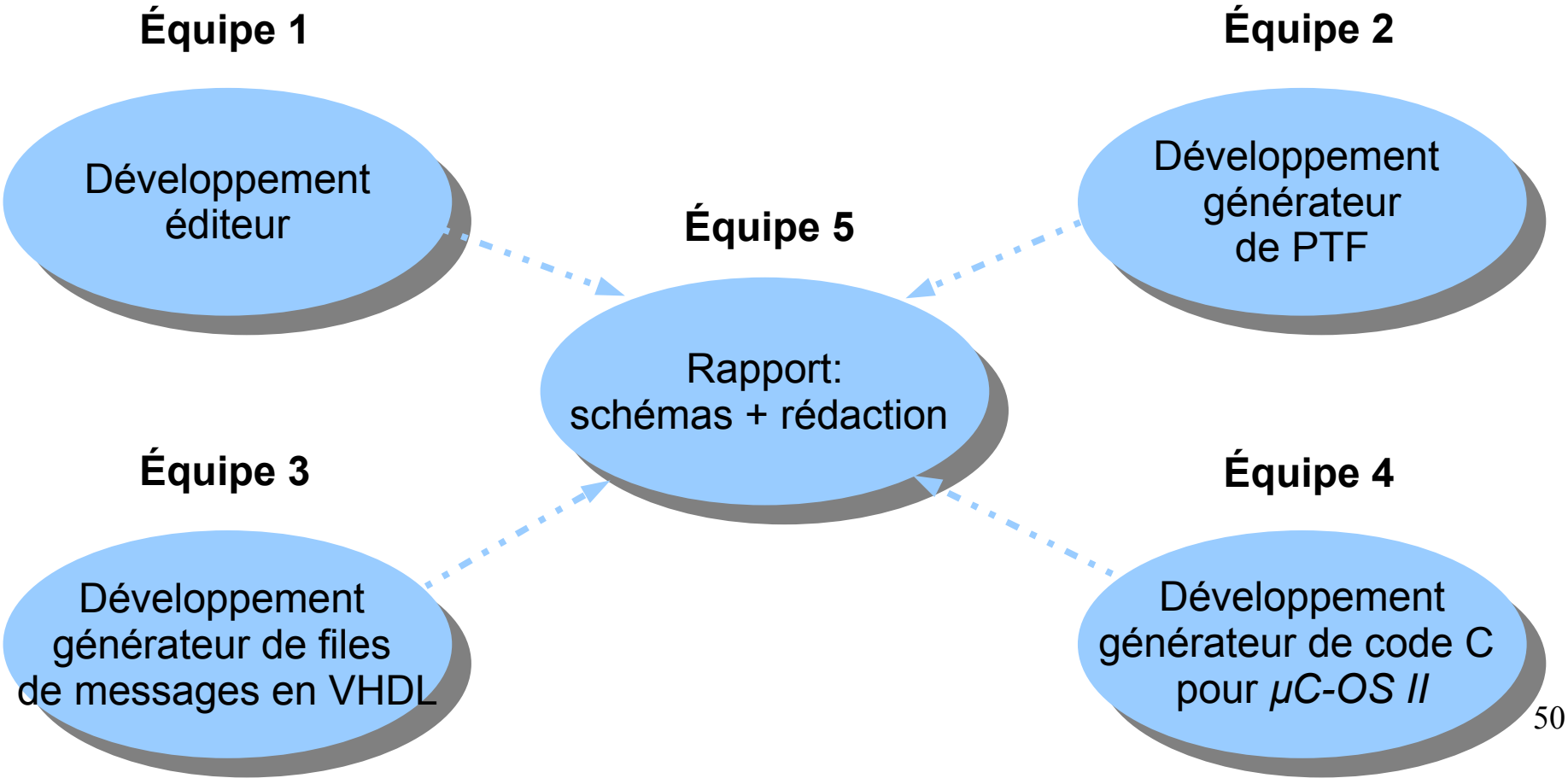
- Configuration de la carte :
  `nios2-configure-sof`
- Chargement du code sur la carte :
  `nios2-download`
- Communication avec la carte :
  `nios2-terminal`

Plan

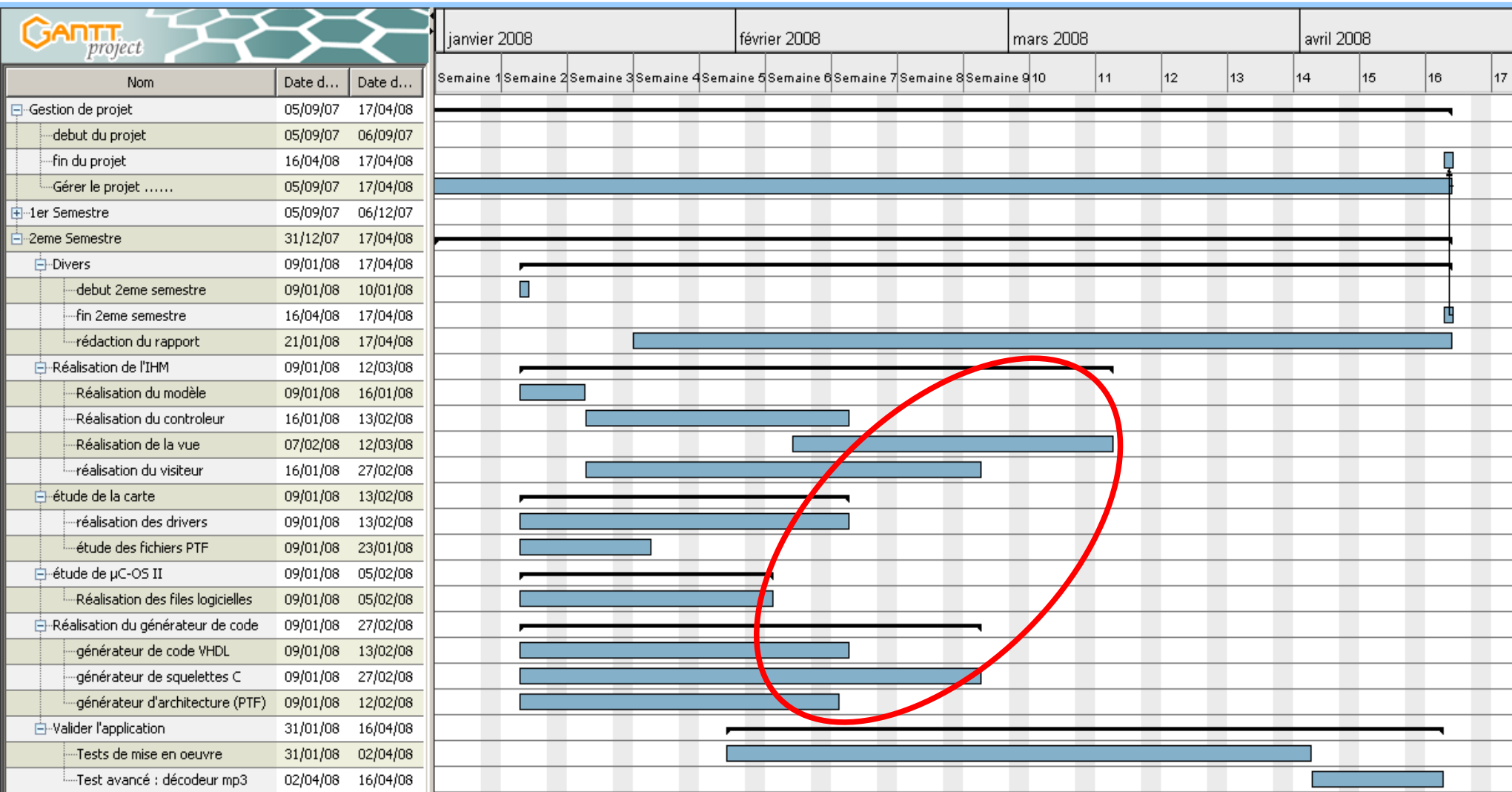
- Introduction
- Présentation
- Vue Utilisateur
- Réalisation
- Gestion de projet
- Conclusion

Organisation

5 équipes de travail :

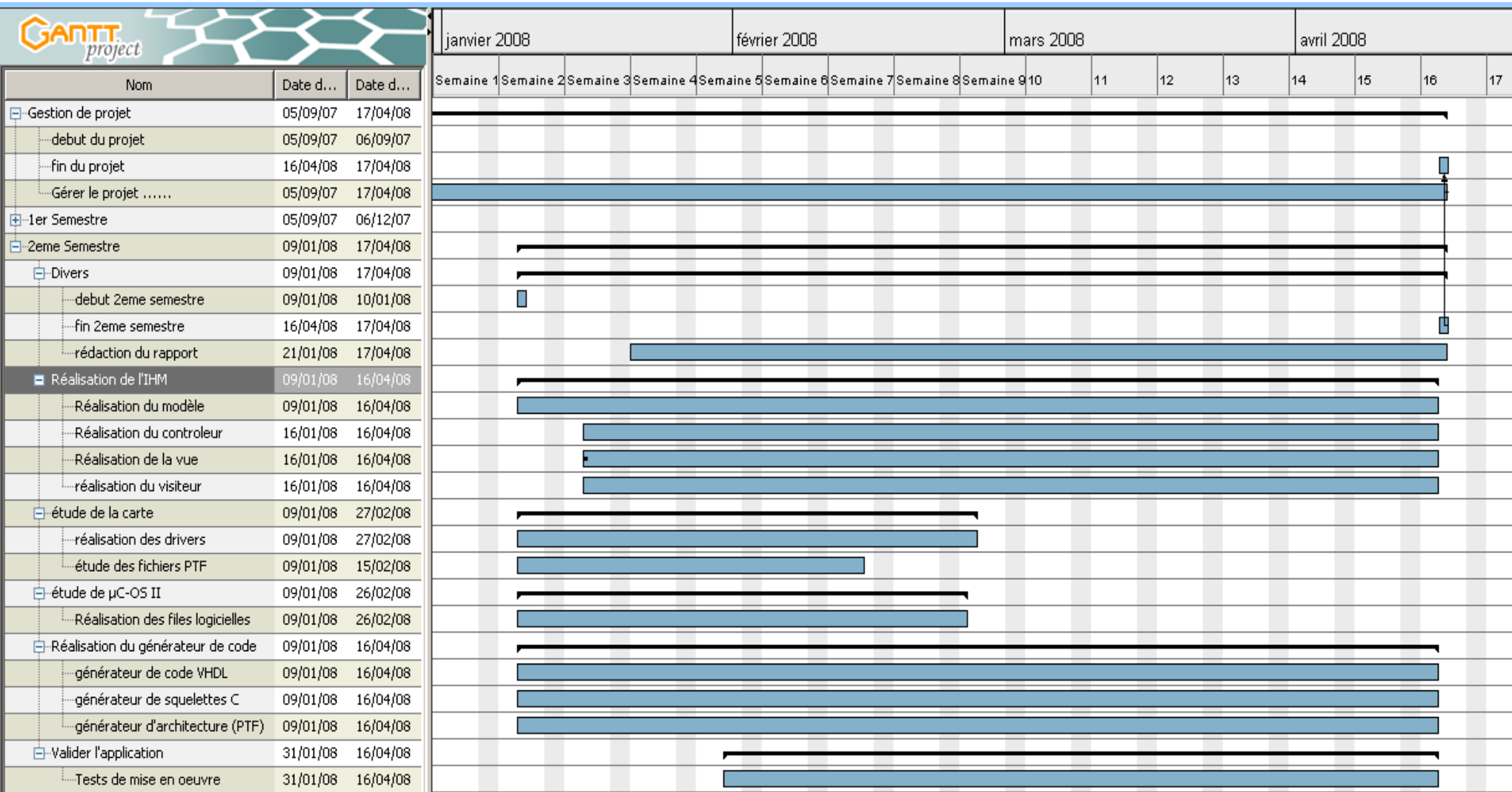


Planning prévisionnel



Nom	Date d...	Date d...
Gestion de projet	05/09/07	17/04/08
debut du projet	05/09/07	06/09/07
fin du projet	16/04/08	17/04/08
Gérer le projet	05/09/07	17/04/08
1er Semestre	05/09/07	06/12/07
2eme Semestre	31/12/07	17/04/08
Divers	09/01/08	17/04/08
debut 2eme semestre	09/01/08	10/01/08
fin 2eme semestre	16/04/08	17/04/08
rédaction du rapport	21/01/08	17/04/08
Réalisation de l'IHM	09/01/08	12/03/08
Réalisation du modèle	09/01/08	16/01/08
Réalisation du controleur	16/01/08	13/02/08
Réalisation de la vue	07/02/08	12/03/08
réalisation du visiteur	16/01/08	27/02/08
étude de la carte	09/01/08	13/02/08
réalisation des drivers	09/01/08	13/02/08
étude des fichiers PTF	09/01/08	23/01/08
étude de µC-OS II	09/01/08	05/02/08
Réalisation des files logicielles	09/01/08	05/02/08
Réalisation du générateur de code	09/01/08	27/02/08
générateur de code VHDL	09/01/08	13/02/08
générateur de squelettes C	09/01/08	27/02/08
générateur d'architecture (PTF)	09/01/08	12/02/08
Valider l'application	31/01/08	16/04/08
Tests de mise en oeuvre	31/01/08	02/04/08
Test avancé : décodeur mp3	02/04/08	16/04/08

Planning effectif



Nom	Date d...	Date d...
Gestion de projet	05/09/07	17/04/08
debut du projet	05/09/07	06/09/07
fin du projet	16/04/08	17/04/08
Gérer le projet	05/09/07	17/04/08
1er Semestre	05/09/07	06/12/07
2eme Semestre	09/01/08	17/04/08
Divers	09/01/08	17/04/08
debut 2eme semestre	09/01/08	10/01/08
fin 2eme semestre	16/04/08	17/04/08
rédaction du rapport	21/01/08	17/04/08
Réalisation de l'IHM	09/01/08	16/04/08
Réalisation du modèle	09/01/08	16/04/08
Réalisation du controleur	16/01/08	16/04/08
Réalisation de la vue	16/01/08	16/04/08
réalisation du visiteur	16/01/08	16/04/08
étude de la carte	09/01/08	27/02/08
réalisation des drivers	09/01/08	27/02/08
étude des fichiers PTF	09/01/08	15/02/08
étude de µC-OS II	09/01/08	26/02/08
Réalisation des files logicielles	09/01/08	26/02/08
Réalisation du générateur de code	09/01/08	16/04/08
générateur de code VHDL	09/01/08	16/04/08
générateur de squelettes C	09/01/08	16/04/08
générateur d'architecture (PTF)	09/01/08	16/04/08
Valider l'application	31/01/08	16/04/08
Tests de mise en oeuvre	31/01/08	16/04/08

Plan

- Introduction
- Présentation
- Vue Utilisateur
- Réalisation
- Gestion de projet
- Conclusion

Conclusion

Quelques problèmes :

- L'instabilité et la lenteur des logiciels
- L'analyse des fichiers PTF , VHDL

Connaissances acquises :

- Langage de description matériel VHDL
- Travailler avec des logiciels austères
- Fonctionnement de la carte Altera DE2 avec son FPGA

Améliorations futures possibles :

- Développement d'une aide au placement des tâches.
- Utiliser d'autres architectures
- Parser C pour déterminer le RPK automatiquement
- Utiliser un ordonnanceur statique (trouver une alternative à *µC-OS II*)

Merci, des questions?

Merci de nous avoir écouté.
Avez-vous des questions ?