

# Modélisation Transactionnelle des Systèmes sur Puces en SystemC Ensimag 3A — filière SLE Grenoble-INP Introduction du cours

Matthieu Moy  
(transparents originaux : Jérôme Cornet)

Matthieu.Moy@imag.fr

2011-2012



## Objectifs et place du cours dans ACE

### Cours lié

- Conception et exploration d'architectures, multi-cœurs, réseaux sur puces (F. Pétrot, S. Mancini)

### Objectifs

- Une vue sur le haut du flot de conception
- Différents niveaux d'abstractions
- Exemple concret : **modélisation transactionnelle**
- Pratique sur un outil utilisé par les industriels : SystemC



## Organisation concrète

- EnsiWiki :  
<http://ensiwiki.ensimag.fr/index.php/TLM>
- Supports de cours sur GitHub :  
<http://github.com/moy/cours-tlm>
  - ▶ `git clone`
  - ▶ `git pull`
  - ▶ Possibilité de travailler à plusieurs sur les squelettes (cf. EnsiWiki)



## Planning approximatif des séances

- 1 Introduction : **les systèmes sur puce**
- 2 Introduction : modélisation au niveau transactionnel (TLM)
- 3 Introduction au C++
- 4 Présentation de SystemC, éléments de base
- 5 Communications haut-niveau en SystemC
- 6 Modélisation TLM en SystemC
- 7 TP1 : Première plateforme SystemC/TLM
- 8 Utilisations des plateformes TLM
- 9 TP2 (1/2) : Utilisation de modules existants (affichage)
- 10 TP2 (2/2) : Utilisation de modules existants (affichage)
- 11 Notions Avancé en SystemC/TLM
- 12 TP3 (1/3) : Intégration du logiciel embarqué
- 13 TP3 (2/3) : Intégration du logiciel embarqué
- 14 TP3 (3/3) : Intégration du logiciel embarqué
- 15 Intervenant extérieur ?
- 16 Perspectives et conclusion

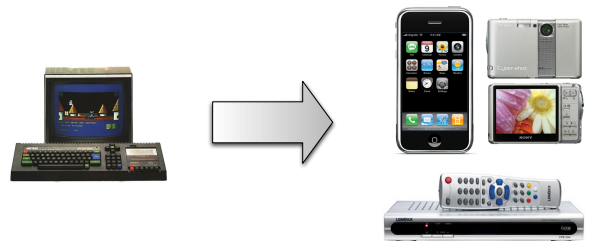


## Sommaire

- 1 Systèmes sur Puce (SoC)
- 2 Modélisation au niveau transactionnel



## Évolution des besoins du grand public



## Quelques exemples

- Lecteurs MP3



### Besoins techniques

- Mini ordinateur (interface utilisateur)
- Codecs (décodage/encodage) audio (MP3, etc.)
- Pilotage de disque dur
- Périphériques (USB, IEEE 1394...)
- Autonomie



## Quelques exemples

- Téléphones portables



### Besoins techniques

- Mini ordinateur (interface utilisateur, applications embarquées)
- Traitement de Signal (technologie de transmission)
- Périphériques (USB, Capteur CCD)
- Autonomie



## Quelques exemples

- Télévision numérique
  - ▶ Lecteurs/Enregistreurs DVD
  - ▶ Démodulateurs Satellite, Décodeurs TNT (*set-top boxes*)
  - ▶ Télévision Haute-définition (*HD-TV*)

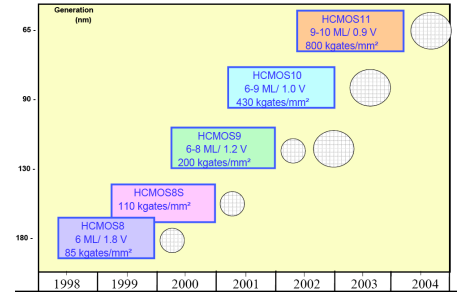
### Besoins techniques

- ★ Mini ordinateur (interface utilisateur)
- ★ Pilotage de disque dur, lecteur/graveur DVD
- ★ Fonctions graphiques (compositions, zoom, curseur...)
- ★ Encodage/décodage vidéo (MPEG2, MPEG2 HD, MPEG 4, H264...)
- ★ Encodage/décodage audio (PCM, AC3, AAC, MP3...)
- ★ Périphériques (IEEE 1394, S/PDIF, HDMI...)



## Évolutions techniques

- Miniaturisation des composants

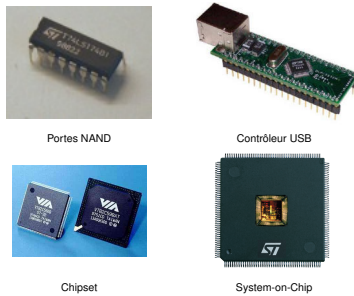


Source : Philippe Magarshak - STMicroelectronics Croiles (MPSoc'03)



## Évolutions techniques

- Évolution de la complexité des fonctions réalisables



## Définition

### System-on-Chip (Système sur puce)

*Puce regroupant tous les éléments électroniques (micro-processeur, composants spécifiques...) nécessaires à la réalisation d'un système (produit) complet.*

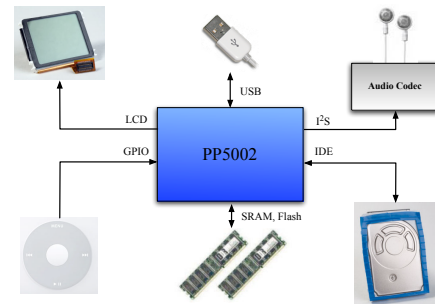


## Quelques utilisations des SoC



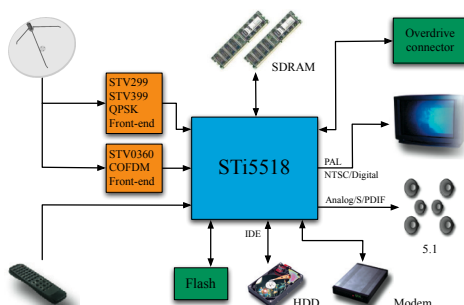
## iPod

- SoC : PP5002 (Portal Player)

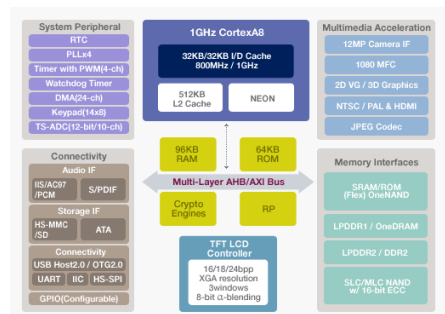


## Décodeur Satellite CDVB2300B

- SoC : STi5518 (STMicroelectronics)



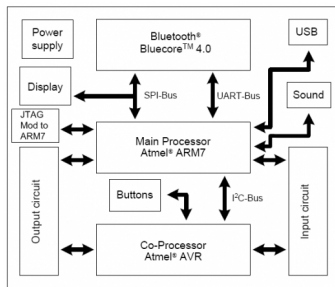
## S5PC110 (Samsung Galaxy S Smartphone)



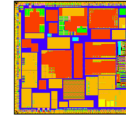
Source : [http://www.samsung.com/global/business/semiconductor/productInfo.do?family\\_id=834&partnum=S5PC110](http://www.samsung.com/global/business/semiconductor/productInfo.do?family_id=834&partnum=S5PC110)



## Brique lego NXT



Source : <http://iar.com/website/1.0.1.0/1518/1/>



## Composants des SoC (non exhaustif)



## Microprocesseurs

- CPU (Central Processing Unit) : processeur généraux
  - ▶ Fonctions : contrôle, interface utilisateur, traitements légers
  - ▶ Exemples : ARM9, ST20, SH4, Leon mais pas Intel Pentium 4 !...
- DSP (Digital Signal Processor)
  - ▶ Fonctions : Traitement du signal, calculs complexes
  - ▶ Exemples : Ti TMS320C55x, etc.
- Processeurs VLIW (Very Long Instruction Word)
  - ▶ Fonctions : traitement multimédia
  - ▶ Exemples : ST210...
- Supports de la partie logicielle du système



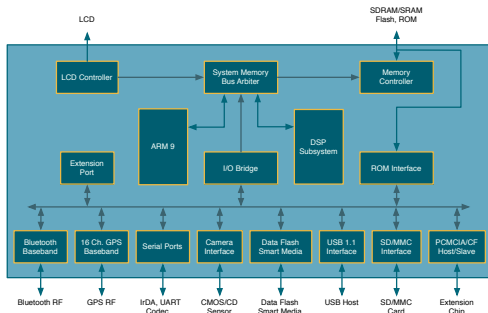
## Mémoires

- ROM, RAM, Flash...
  - ▶ Souvent hors du SoC
- Dans la puce :
  - ▶ Contrôleur(s) mémoires
  - ▶ Petites mémoires internes
  - ▶ Mémoires caches, « fifo » (tampons/files d'attente)
- Fonctions
  - ▶ Stockage temporaire (RAM)
  - ▶ Programme interne (ROM), possibilité de mise à jour (Flash)
  - ▶ Stockage (Flash)



## Exemple : PDA GPS

- SoC : Atlas-M (Centrality)

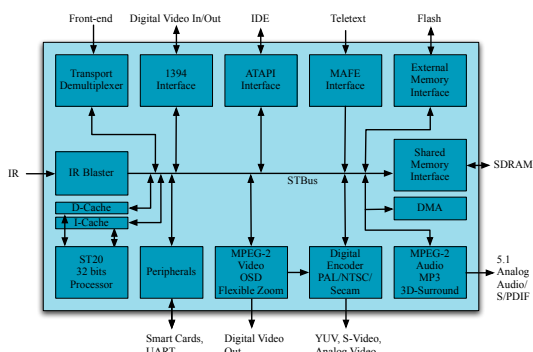


## Composants utilitaires

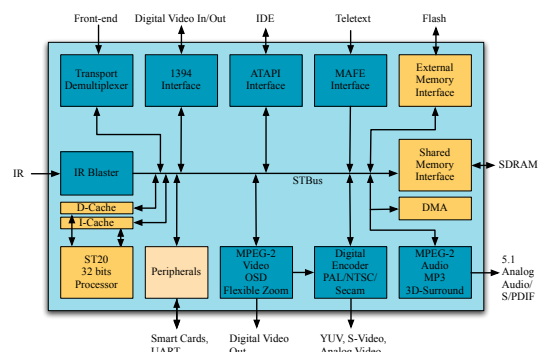
- DMAC (Direct Memory Access Controller)
  - ▶ Transferts mémoires/mémoires, mémoires/périphériques
  - ▶ Décharge le CPU (pas d'attente liée aux transferts)
- Timer, RTC (Real-Time Clock)
  - ▶ Mesure de l'écoulement du temps
  - ▶ Utilisations :
    - ★ Contrôle du nb d'images par secondes
    - ★ Programmation de délais d'expiration
    - ★ Utilisation par OS Temps Réel
- Contrôleur d'interruptions (ITC)
  - ▶ Centralisation de tous les signaux d'interruptions
  - ▶ Informations sur l'émetteur de l'interruption



## Exemple : STi5518



## Exemple : STi5518 - CPU, Mémoires, Utilitaires



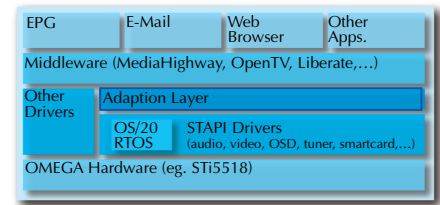


## Le logiciel (embarqué)

- Plusieurs logiciels sur architectures différentes
  - Mélange SH4, ARM, ST...
  - Endianess différentes
  - Problèmes techniques d'ordre de boot
  - Problèmes de synchronisations
- Communications de base : accès au bus et interruptions
- Partie génie logiciel
  - Utilisation d'OS multitâche/temps réel : Linux, OS/20, Windows CE...
  - Factorisation du code bas niveau dans des pilotes (drivers)
  - Couche supplémentaire au dessus de l'OS : le middleware



## Exemple de structure logicielle



Source : doc STi5518



## Problèmes de conception



## Complexité croissante de conception

- Nombre de transistors : + 50% par an (Moore)
- Productivité en conception : + 30% par an  
⇒ « Design Gap »
- Besoin incessant de nouvelles techniques de conception



## Complexité croissante de conception

- Circuits à assembler énormes
  - Multiples éléments complexes : processeurs, réseau d'interconnection...
  - Parfois plusieurs mini-SoC dans le SoC (sous-ensembles simples CPU+DMA+Mémoire)
  - Évolution possible vers le massivement multiprocesseur
- VHDL/Verilog suffisent difficilement...
- Pas de révolution type « niveau porte → RTL » pour l'instant



## Durée du cycle de développement

- Évolution rapide
- Dates à ne pas manquer (Noël, nouvel an chinois, ...)
- ⇒ Un produit prêt 6 mois trop tard est difficilement vendable !
- ⇒ Le « Time to market » est aussi important que la main d'œuvre totale.



## Coût d'une erreur

### Bug dans le logiciel

- Mise à jour du firmware
- Pas forcément acceptable partout... (difficulté pour l'utilisateur, systèmes critiques)

### Bug dans le matériel

- Fabrication de nouveaux masques
- Exemple de coût :

Finesse de gravure	0.25 $\mu m$	0.13 $\mu m$	65 nm
Coût masque 1 couche	10 000 \$	30 000 \$	75 000 \$
Nb de couches	12	25	40
Coût total	120 000 \$	750 000 \$	3 M\$

source EETimes



## Coût d'une erreur

### Bug dans le matériel (suite)

- Circuit déjà fabriqué : recherche d'un contournement (workaround)
- Valable en technologie ASIC
- SoC FPGA
  - ARM Excalibur : ARM 922 (200 MHz) + FPGA APEX 20KE
  - Xilinx Virtex 4 : PowerPC 405 (450 MHz) + FPGA + Ethernet MAC
  - Là encore, mise à jour limitée



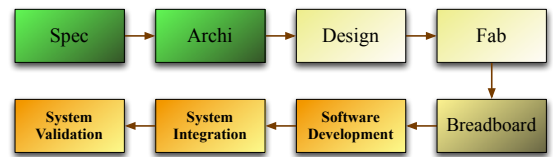
## Problèmes de conception

- Vitesse de simulation
  - ▶ Simulation du SoC niveau RTL : plusieurs heures, voire jours...
  - ▶ ex : Encodage et décodage d'une image en MPEG 4 = 1 h en simulation RTL
  - ▶ Impossibilité de tester le(s) logiciel(s) embarqué(s) à ce niveau
  - ▶ Moins de temps disponible pour valider le système...
  - ▶ Développement séparé des différents blocs
  - ▶ Quelques solutions couramment pratiquées :
    - ★ Cosimulation
    - ★ Émulation matérielle



## Problèmes de conception

- Intégration tardive du logiciel
  - ▶ Cycles de développements longs
  - ▶ Incompatible avec le **time to market**
  - ▶ Découverte tardive de bugs dans le matériel



## Problèmes d'intégration

- Fonctionnelle
  - ▶ Développement séparé des composants, réutilisation
  - ▶ Aucune garantie de fonctionnement
  - ▶ Problèmes de compatibilité plus complexes qu'électroniques
- Performances
  - ▶ Adéquation d'un ensemble de composants pour réaliser une tâche dans un temps donné
  - ▶ Dépendences non fonctionnelles complexes



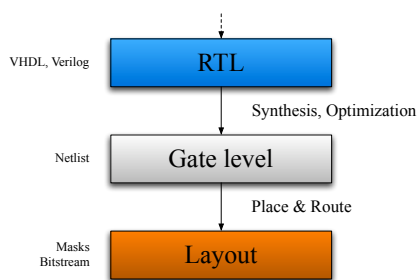
## Problèmes de validation

- Conformité du système à la spécification ?
- Spécification de plus en plus complexes
  - ▶ Normes MPEG x, H264, ...
  - ▶ Formats informatiques divers
  - ▶ Interprétation parfois erronée
- Format de la spécification ?



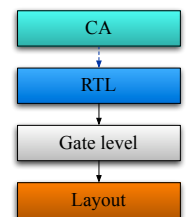
## Retour sur le flot de conception

- Bas du flot de conception (vu en 2A)
  - 3 principaux **niveaux d'abstraction**

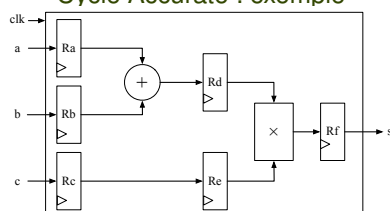


## Niveau supérieur : Cycle Accurate (1/2)

- Objectif : simulation rapide
- Caractéristiques
  - ▶ Précis au cycle d'horloge près
  - ▶ Précis au niveau données (**bit true**)
  - ▶ Écriture libre du modèle interne des composants (C, C++...)



## Cycle Accurate : exemple



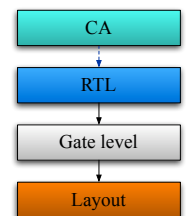
```

void my_function()
{
    Ra = a;
    Rb = b;
    Rc = c;
    wait(clk, 2);
    s = (Ra+Rb)*Rc;
}
    
```



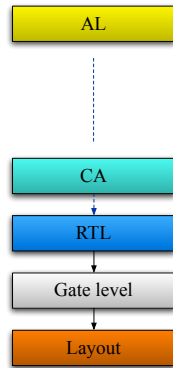
## Niveau supérieur : Cycle Accurate (2/2)

- Utilisation
  - ▶ Modèles de remplacement pour simulations RTL
  - ▶ Ex : modèles de processeurs...
- Inconvénients
  - ▶ Non synthétisable (référence = RTL)
  - ▶ Effort de modélisation important



## Niveau « Algorithmique »

- Programme décrivant la fonctionnalité (C, Matlab, etc.)
- Souvent séquentiel
- Ex : algorithme de référence décodage MPEG
- Écriture non ambiguë d'une portion de la spécification
- Pas de référence au matériel :
  - ▶ Pas de partitionnement hard/soft
  - ▶ Partie contrôle absente



## Exemple : décodage XVID

```

...
bound = 0;
for (y = 0; y < mb_height; y++) {
  cp_mb = st_mb = 0;
  for (x = 0; x < mb_width; x++) {
    MACROBLOCK *mb;

    /* skip stuffing */
    while (BitstreamShowBits(bs, 10) == 1)
      BitstreamSkip(bs, 10);

    if (check_resync_marker(bs, fcode == 1)) {
      bound = read_video_packet_header(bs, dec, fcode == 1,
        &quant, &fcode, NULL, &intra_dc_threshold);
      x = bound % mb_width;
      y = bound / mb_width;
    }
    mb = &dec->mbs[y * dec->mb_width + x];

    DPRINTF(XVID_DEBUG_MB, "macroblock (%i,%i) %08x\n", x, y, BitstreamShowBits(bs, 32));

    if (!BitstreamGetBit(bs)) { /* block _is_ coded */
      uint32_t mcbpc, cbpc, cbpy, cbp;
      uint32_t intra, acpred_flag = 0;
      int mcsel = 0; /* mcsel: '0'-local motion, '1'-GMC */

      cp_mb++;
      mcbpc = get_mcbpc_inter(bs);
      mb->mode = mcbpc & 7;
      cbpc = (mcbpc >> 4);

      DPRINTF(XVID_DEBUG_MB, "mode %i\n", mb->mode);
      RPRINTF(XVID_DEBUG_MB, "cbpc %i\n", cbpc);
      intra = (mb->mode == MODE_INTRA || mb->mode == MODE_INTRA_Q);

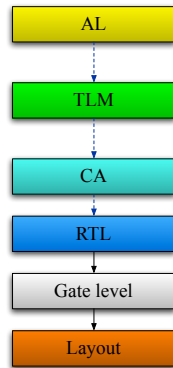
      if (gmc_warp && (mb->mode == MODE_INTER || mb->mode == MODE_INTER_Q))
        mcsel = BitstreamGetBit(bs);
      else if (intra)
        acpred_flag = BitstreamGetBit(bs);
    }
  }
}

```

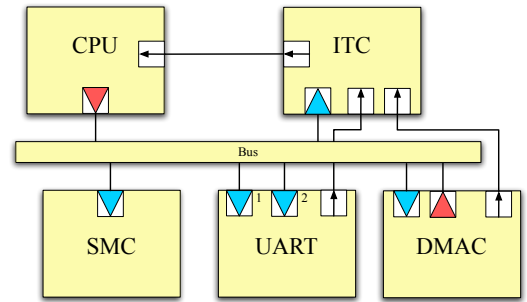


## Niveau transactionnel : TLM

- Transaction Level Modeling
- Plusieurs définitions...  
ici : TLM OSCI (standard)
- Objectif n°1 : développement du logiciel embarqué
- Caractéristiques
  - ▶ Pas d'horloge (cycle-less)
  - ▶ Bit true
  - ▶ Définition, modélisation précise des communications



## Un exemple



## Objectif développement du logiciel embarqué

### Contrat entre logiciel embarqué et matériel

- Logiciel correct en simulation TLM ⇒ logiciel correct en simulation RTL et sur puce
- Nécessite tous les éléments matériels « visibles » par le logiciel

⇒ Modéliser tout ce qui est nécessaire au logiciel embarqué, mais seulement cela



## Éléments nécessaires

- Possibilité d'actions lectures/écritures
- Plages d'adresses
- Mémoires
- Bancs de registres
- Interruptions



## Idée principale

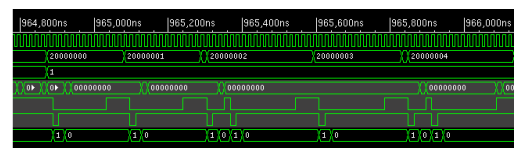
- Actions de lectures/écritures = **transactions** sur le réseau d'interconnexion
- Correspondance avec les opérations logiques effectuées sur le bus

### Informations minimales contenues dans une transaction

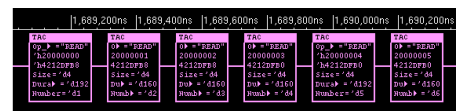
- Type : lecture ou écriture
- Adresse de base
- Données
- Taille (nb de données)
- État : OK, Erreur, Temps dépassé...
- Implicite : taille d'une donnée élémentaire, taille d'une adresse



## Ex : lectures successives sur bus AHB



RTL



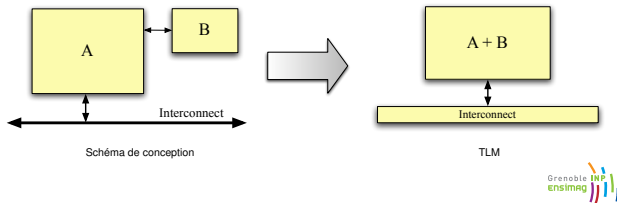
TLM

Abstraction des communications sur bus



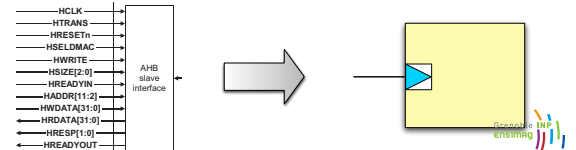
## Composants

- Séparation en fonction des communications
  - Fusion de composants communiquant en dehors du réseau d'interconnexion
  - Règle non absolue (fonction des besoins et des types de communications disponibles)
- Possibilité de hiérarchie
- Simulation indépendante de chaque composant :  
→ **concurrency**



## Ports de communication (transactions)

- Deux types de ports (de façon générale)
  - Initiateurs (**initiator**, **master**)
    - ★ « Émetteurs » des transactions
  - Cible (**target**, **slave**)
    - ★ « Répondent » aux transactions
    - ★ Partie passive d'un composant
    - ★ Association port cible ↔ plage d'adresse
- Analogie avec le réseau
  - initiateur ≈ client
  - cible ≈ serveur
  - ⚠ Rien à voir avec la distinction lecture/écriture !
- Exemple :



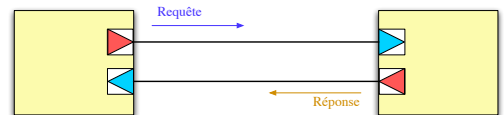
## Protocole (1/2)

- Ensemble des actions possibles au niveau des ports (lecture, écriture, ...)
- Protocoles **bloquants**



## Protocole (2/2)

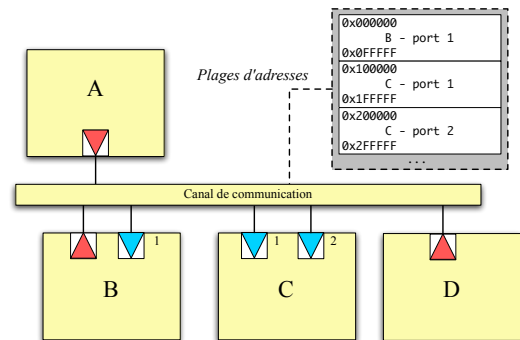
- Protocoles **non bloquants**
  - Ne bloquent pas l'**initiateur**
  - Deux canaux de communication : requête et réponse




## Canal de communication (1/2)

- Abstrait les différents éléments du bus
  - Décodage
  - Multiplexage
  - Signaux...
- Fonctionnalité principale : routage des transactions
- Connaissance des plages d'adresses des composants
- Ports de communications spéciaux (cible : plusieurs-vers-un, initiateur : un-vers-plusieurs)

## Canal de communication (2/2)

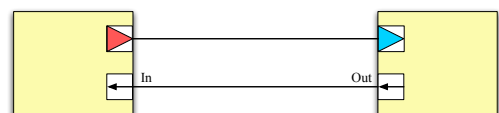


## Interruptions (1/2)

- Qu'est-ce que c'est ?
    - Connection directe entre composants par un fil
    - Unidirectionnel
- 
- À quoi ça sert ?
    - Processeurs
      - ★ Déclenchement immédiat de routines en cours d'exécution du code normal
      - ★ ex : Interruption Souris, traitement terminé par un autre composant...
      - ★ Économies d'énergies (HLT sur Pentium, ...)
    - Autres composants matériels
      - ★ Synchronisation
      - ★ « Arrêt d'urgence » d'un composant

## Interruptions (2/2)

- Modélisation non standardisée...
- Une solution : signaux booléens (RTL)
  - Sensibilité sur fronts (montants, descendants)
  - Entrées, sorties de signaux (idem RTL)
  - Connections point à point
  - Principal moyen de synchronisation pour les cibles





## Modélisation interne d'un composant

- Liberté de codage... mais respect du niveau d'abstraction !

### Partie « initiateur »

- Transactions générées réalistes
  - Mêmes adresses
  - Données précises

### Partie « cible »

- Mémoires
    - Respect de la taille des données
    - Endianess
  - Bancs de registres
    - Adresses relatives des différents registres
    - Actions associées
- Contrat d'utilisation du composant



## Exemple de bancs de registres (DMAC ARM)

Name	Address (base+)	Type	Reset value	Description
DMACIntStatus	0x000	RO	0x00	See Interrupt Status Register on page 3-10
DMACIntTCStatus	0x004	RO	0x00	See Interrupt Terminal Count Status Register on page 3-10
DMACIntTCClear	0x008	WO	-	See Interrupt Terminal Count Clear Register on page 3-11
DMACIntErrorStatus	0x00C	RO	0x00	See Interrupt Error Status Register on page 3-11
DMACIntErrClr	0x010	WO	-	See Interrupt Error Clear Register on page 3-12
DMACRawIntTCStatus	0x014	RO	-	See Raw Interrupt Terminal Count Status Register on page 3-13
DMACRawIntErrorStatus	0x018	RO	-	See Raw Error Interrupt Status Register on page 3-13
DMACEnblChns	0x01C	RO	0x00	See Enabled Channel Register on page 3-14
DMACSoftBReq	0x020	R/W	0x0000	See Software Burst Request Register on page 3-14
DMACSoftSReq	0x024	R/W	0x0000	See Software Single Request Register on page 3-15



## Exemple de mode d'emploi de registre (DMAC ARM)

### 3.4.10 Software Single Request Register

The read/write DMACSoftSReq Register, with address offset of 0x024, enables DMA single requests to be generated by software. You can generate a DMA request for each source by writing a 1 to the corresponding register bit. A register bit is cleared when the transaction has completed. Writing 0 to this register has no effect. Reading the register indicates the sources that are requesting single DMA transfers. You can generate a request from either a peripheral or the software request register. Figure 3-10 shows the bit assignments for this register.



Figure 3-10 DMACSoftSReq Register bit assignments

Table 3-11 lists the bit assignments for this register.

Table 3-11 DMACSoftSReq Register bit assignments

Bits	Name	Function
[31:16]	-	Read undefined. Write as zero.
[15:0]	SoftSReq	Software single request.

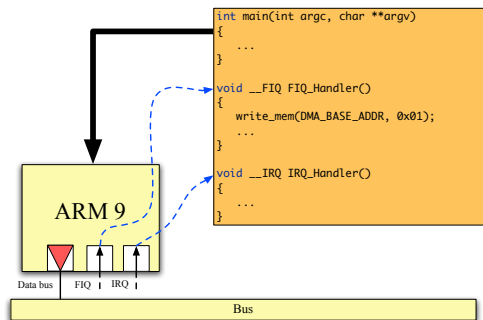


## Insertion du logiciel embarqué (1/2)

- « Emballage » autour du code (wrapper)
- Interface
  - Accès mémoire via un bus de données
    - port initiateur
  - Entrée(s) interruptions
    - Dépendant du modèle de processeur...
- Mise en correspondance interruptions ↔ code de traitement

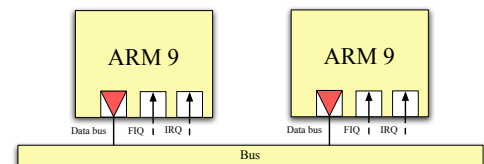


## Insertion du logiciel embarqué (2/2)



## Question

- Communication d'une valeur entière x entre les processeurs ?



### Question



Comment faire ? Faut-il ajouter des composants ?



## Comparaison avec les autres niveaux (1/2)

### TLM vs. Algorithmique

- Découpage de l'algorithme en blocs indépendants
- Validation du fonctionnement en parallèle
- Aspect composant
  - Réutilisation
  - Hierarchie de composants
- Partitionnement



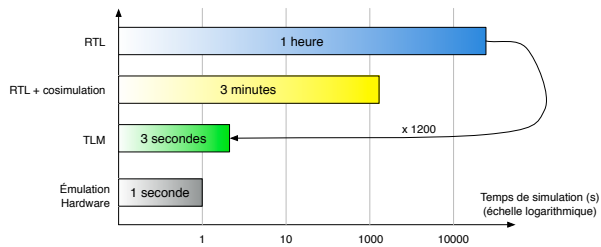
## Comparaison avec les autres niveaux (2/2)

### TLM vs. Cycle accurate

- Vitesse de simulation
  - Écriture libre de l'intérieur du composant
  - Communications abstraites
  - Comportement asynchrone
  - Dépend de l'implémentation !
- Précision des données
- Modélisation facile : réutilisation du code de niveau AL



## Vitesse de simulation



Encodage + décodage d'une image MPEG 4



## Apports de TLM

- Écriture du code embarqué possible en avance de phase !
  - Vitesse de simulation
  - Facilité de modélisation
- Debuggage de l'intégration des composants
- Nouveau niveau de référence
  - Moyen de communications entre monde du hard et monde du soft...
  - Référence disponible en avance de phase
- Analyse d'architecture



## Synthèse

- Synthèse comportementale : pas en industrie
  - Quelques essais : MathLab/Simulink → RTL, C → RTL, ...
  - Nécessité d'un niveau d'abstraction bien défini
- TLM
  - Précision des communications uniquement
  - Intérieur du composant : aucune règle (pointeurs, bibliothèque, etc.)
  - Synthèse éventuelle :
    - ★ Partie connection au bus
    - ★ Réseau d'interconnection

