Jean-José BERENGUER (Ingénieur R&D STMicroelectronics)
Nicolas COSTE (PhD STMicrolectronics – INRIA)
Iker DE POY ALONSO (Ingénieur R&D STMicroelectronics)
Giuseppe DESOLI (Architecte en chef STMicroelectronics)
Etienne LANTREIBECQ (Ingénieur R&D STMicroelectronics)
Julien LEGRIEL (PhD STMicroelectronics – VERIMAG)
Gilbert RICHARD (Ingénieur R&D STMicroelectronics)

xSTream : Architecture multi-cœur sur puce pour des applications multimédia

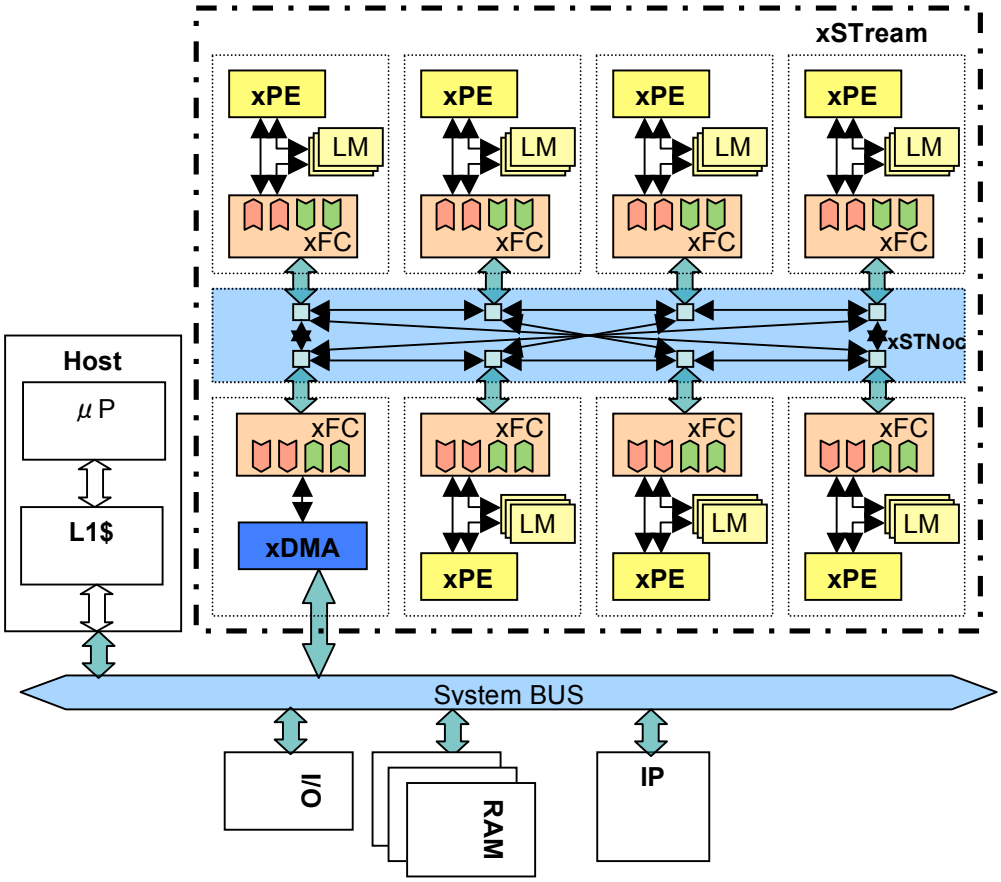
Les systèmes sur silicium développés dans le but d'accomplir des fonctions multimédia résultent, le plus souvent, de l'assemblage de sous systèmes hétérogènes et indépendants. Ces systèmes sont constitués d'ilots de calculs implémentés par un ensemble de processeurs, ayant chacun sa mémoire locale et sa solution logicielle. Ces ilots indépendants interagissent au travers de mémoires partagées et autres boites aux lettres. La complexité de ces systèmes et de leur programmation est un frein au développement de nouvelles applications ou simplement à l'évolution d'une application existante pour prendre en compte une nouvelle norme.

STMicroelectronics développe une nouvelle architecture multiprocesseurs sur puce destinée aux applications multimédia. Cette architecture régulière est basée sur un ensemble de processeurs identiques, interconnectés par un réseau sur puce.

Cette architecture est détaillée dans la première partie de ce document. Dans la seconde section, nous détaillons l'approche utilisée pour la vérification fonctionnelle et l'évaluation de performance de ce système. Les aspects relatifs au modèle de programmation et au portage d'applications sont présentés dans la troisième partie. Les seconde et troisième sections sont l'objet de recherches et développement dans le cadre des projets pôle de compétitivité ATHOLE et MULTIVAL de MINALOGIC.

1. L'architecture xSTream

L'architecture xSTream se présente comme une ferme de calcul connectée sur le bus principal du système. La ferme de calcul est composée de nœuds connectés par un réseau sur puce permettant une bande passante importante entre les nœuds. Grâce au caractère asynchrone des communications dans le réseau sur puce, il est possible d'envisager que les différents nœuds possèdent leur propre horloge et que le système se comporte d'une manière globalement asynchrone et localement synchrone. Cette caractéristique est particulièrement intéressante pour des aspects de fabricabilité et de consommation électrique du système.



Chaque nœud est composé d'un processeur (Processing Element), d'une mémoire et d'un contrôleur de flux (xFC). Même si la mémoire est surtout destinée à être utilisée localement, toute la mémoire du système est accessible depuis n'importe quel acteur (Non Uniform Memory Access time).

Le « processing element » est un VLIW dual issue, hyper–threaded (Multifils), dont le jeu d'instruction comporte des opérations vectorielles destinées à favoriser les calculs des applications multimédia. Avec une fréquence de 600 MHz, la puissance théorique de calcul de chaque processeur est de 1.2 GOPS.

Le contrôleur de flux est un élément important de l'architecture xSTream. Il contient des files d'attente matérielles (qui peuvent être étendues en mémoire) et la logique de contrôle associée. Chaque file peut être configurée en file d'émission de données ou en file de réception. Une file d'émission est associée logiquement à une file de réception située sur un autre nœud. Les contrôleurs de flux assurent, de manière transparente pour les processeurs, le déplacement des données entre les files d'émissions et celles de réception. Avec ce support matériel, les applications basées sur un flot de données se décomposent, naturellement, en fonctions s'exécutant chacune sur un nœud de calcul, avec leurs entrées/sorties limitées aux files présentes dans le contrôleur de flux.

Un nœud dédié permet de gérer les échanges entre la mémoire principale du système et la ferme de calcul. Ce nœud ne comporte pas de processeur mais un DMA performant à plusieurs canaux et permettant des échanges de schémas mémoire complexes.

2. Vérification

2.1 Vérification fonctionnelle

En plus des méthodes traditionnelles de vérification du matériel, nous avons décidé de vérifier l'architecture en utilisant des méthodes formelles au travers des outils CADP de l'équipe VASY de l'INRIA[1]. Ces outils se basent principalement sur des modèles LOTOS des comportements à vérifier. Ce travail porte principalement sur les communications au sein de l'architecture au travers des files d'attente. Les files d'attente de xSTream ne sont pas des files d'attente simples, et la première tâche a été de réaliser un modèle de référence afin d'exprimer des propriétés de fonctionnement correct. Ensuite, des modèles plus fins, plus proches de l'implémentation matérielle ont été réalisés afin de vérifier qu'ils comportaient toujours les propriétés énoncées précédemment.

Comme on pouvait s'y attendre dans le cadre de vérification énumérative, une approche brutale comportant des modèles précis de l'ensemble du système est vouée à l'échec. Nous avons du appliquer l'approche du « client identifié ». Au lieu de vérifier le comportement total du système, on identifie un comportement à vérifier (par exemple une communication point à point) et l'on vérifie que ce comportement reste correct quand le reste du système agit comme perturbateur.

Cette approche nous a permis de mettre en évidence plusieurs sous spécifications de l'architecture. En particulier, un cas où une application correcte d'un point de vue logiciel (bon nombre de consommations dans les files par rapport au nombre d'émissions) pouvait entrainer le blocage du système par suite d'inter-blocage dans des protocoles. Ce type de problème est particulièrement difficile à interpréter sur une plateforme de simulation. C'est dans ce contexte que l'approche formelle démontre son intérêt.

2.2 Evaluation de performance

A partir du modèle développé dans le cadre de la vérification fonctionnelle, l'approche IMC (Chaîne de Markov Interactive) intégrée à la boîte à outils CADP permet de générer un modèle pour l'évaluation de performance sous forme de chaîne de Markov à temps continu. Cette approche consiste à enrichir le modèle fonctionnel d'informations temporelles telles que des délais issus de la spécification du système. L'approche IMC impose une approximation des délais sous forme de distributions phase-type. Nous avons montré que malgré les résultats de performance atteignables, cette approche ne permettait pas d'obtenir une estimation de l'erreur imputable aux approximations phase-type. Cette approche ne pouvait être pleinement satisfaisante dans ces conditions.

La nature discrète des délais présents dans nos systèmes matériels nous a amenés à réfléchir à une approche similaire mais basée sur l'insertion de délais discrets modélisés par des distributions probabilistes discrètes. Cette nouvelle approche permet de générer des chaînes de Markov à temps discret. Nous avons intégré cette approche dans un flot de performance basé sur les outils CADP et de nouveau outils propres à la manipulation de modèles probabilistes. Cette approche nous permet d'obtenir des résultats de performance relatifs à des taux d'occupation, des latences, et des débits. En ce qui concerne les résultats de latence, nous sommes en mesure d'obtenir les distributions exactes, nous permettant ainsi de connaître les valeurs moyennes ainsi que les valeurs minimales et maximales. Cette approche permet de s'affranchir des méthodes brutales et classiques basées sur la simulation.

3. Aspects logiciels

3.1 Modèle de programmation

Le modèle de programmation permet de décorréler l'application de l'architecture sous-jacente tout en tirant partie des ressources matérielles des nœuds de calculs et des moyens de communication.

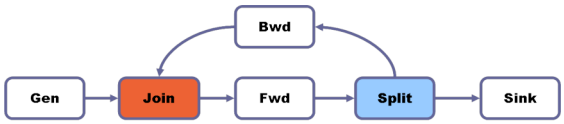
---

[1] http://www.inrialpes.fr/vasy/

Ce modèle de programmation permet la description d'applications sous la forme d'un graphe d'entités indépendantes (appelées « filtres ») communicant de manière asynchrone par des canaux logiques.

En adéquation avec l'architecture xSTream, il supporte le parallélisme de données comme le parallélisme de tâches, de façon à exécuter divers types d'applications et si nécessaire changer la topologie des données et l'organisation de l'algorithme.

Le langage retenu est le C, et les constructions proposées apparaissent comme une extension du langage. Dans un premier temps, les constructions de langage étaient de type simple entrée et simple sortie avec toutefois la possibilité de faire des opérations de type séparation et jointure (« split – join » en anglais). Des fonctionnalités additionnelles comme les boucles de rétroaction sont aussi disponibles.



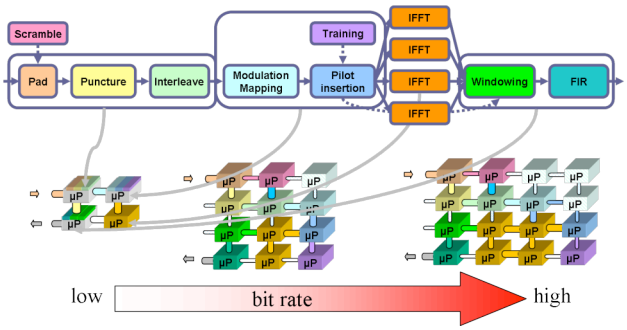

```
pipeline mypipe(int joinparam, int splitparam) {
    MyGen Gen();
    feedback MyJoiner Join(joinparam) {
        MyForward Fwd();
        Backward Bwd();
    } loop MySplitter Split(splitparam);
    MySink Sink();
}
```

Dans un deuxième temps, les constructeurs vont permettre de travailler sur des filtres avec plusieurs entrées et plusieurs sorties (MIMO en anglais), de façon à diminuer le nombre de filtres et faciliter la construction des algorithmes.

Une fois implémenté sous forme de filtres logiciels, l'algorithme est ensuite distribué sur l'architecture matérielle. Plusieurs filtres peuvent coexister sur un même nœud de calcul et communiquer en utilisant les ressources mémoires. Les communications entre nœuds sont assurées par le mécanisme matériel de contrôle de flux.

Sans changer ce graphe de filtres, il est possible de générer différents « mappings » en fonction du nombre de nœuds de calcul disponibles. Cette fonctionnalité permet d'adapter l'application à l'architecture matérielle. Il est ainsi possible d'augmenter le nombre de filtres par nœuds de calcul afin de réduire les ressources matérielles et donc la consommation électrique. Cette même approche permet également de résoudre le problème de la variabilité technologique par exemple pour les systèmes avec des nœuds défectueux.

Une des problématiques est de pouvoir découper et ordonnancer une application quelconque. Nous donnerons un exemple avec une application de type vidéo (ci-dessous). De façon à aider ce découpage et l'ordonnancement, il faut pouvoir calculer les poids des communications et des traitements, manuellement ou en utilisant des méthodes plus automatisées.



3.2 Logiciels applicatifs

Une application test pour l'architecture xSTream est un encodeur vidéo au standard H264 avec une résolution "full HD" (1080p) et supportant le "main profile".

Un des modèles de programmation xSTream permet de décrire l'application sous forme de graphe de filtres interconnectés ; la synchronisation des filtres est assurée par le flot de données (streaming). Chaque filtre sera exécuté sur un élément de calcul du réseau de processeurs et affecté à un processeur grâce à l'environnement de programmation.

L'encodeur H264 exploite le parallélisme de données ainsi que le parallélisme de tâche. Le parallélisme de tâche correspond à la décomposition de l'encodeur en filtre ; chaque filtre s'exécute de façon autonome et indépendante par rapport aux autres et se synchronisent sur des files de données d'entrées et de sorties bloquantes.

Le parallélisme de données intervient au niveau de l'écriture d'un filtre par l'utilisation du jeu d'instructions vectorielles du processeur, et par la multiple instanciation d'un filtre (ou ensemble de filtres) traitant des ensembles de données indépendant. Il est possible de découper l'image à encoder en un ensemble de tranches horizontales (slices) qui sont encodées en parallèle. L'enjeu de l'implémentation temps réel de cet encodeur dans un système embarqué est de satisfaire les contraintes de bande passante, de puissance de calcul ainsi que de mémorisation. En effet, la mémoire disponible sur chaque processeur étant limitée, seule une petite partie d'une image peut être stockée localement ; il faut donc construire un flot de données et effectuer les traitements « en continu ». La phase d'estimation de mouvement est donc réalisée sur une fenêtre glissante définissant un voisinage restreint autour du macro-block courant. Ce principe, associé à un algorithme propriétaire de sélection des vecteurs de mouvement, réduisent bande passante et puissance de calcul. Pour limiter les échanges de données entre filtres, les traitements de transformée, quantification (directes et inverses) ainsi que le décodage sont regroupés en un seul filtre. L'équilibrage de charge de calcul est assuré statiquement au chargement de l'application sur le réseau de processeurs, sachant qu'un processeur a la possibilité d'exécuter jusqu'à quatre filtres (4 contextes disponibles grâce au multithreading).

L'architecture xSTream permet l'implémentation flot de données d'un encodeur H264 full HD, au prix d'une réflexion sur la localité des données nécessaire pour satisfaire les contraintes de bande passante mémoire (5x format PAL), et de réécriture des traitements pour exploiter les capacités vectorielles des processeurs.