

Design Space Explorer : Un Outil de Co-Conception pour Plate-formes Multi-Processeurs sur Puce

Daniela Genius, Nicolas Pouillon, Alain Greiner
LIP6, Département SoC, Université Pierre et Marie Curie

Résumé

Nous montrons comment une application simple, mais représentative des applications multimédia et réseau réellement visées est utilisée en enseignement du niveau master pour former les étudiants à la conception au niveau système. L'application est déployée à l'aide de DSX, un outil de co-conception matérielle/logicielle de systèmes sur puce (SoC).

1 Introduction

L'outil DSX (Design Space Explorer) [2] permet la co-conception de plate-formes matérielles-logicielles destinées au traitement de flux basées sur des architectures multi-processeurs sur puce. Il est essentiellement introduit au niveau de la deuxième année du Master intitulé "*Systèmes embarqués : architecture, système et programmation*". Notre but est de former ces étudiants à l'utilisation d'un outil de conception au niveau système. Nous leur exposons en détail la description de l'application, la description de la plate-forme matérielle et le choix du système d'exploitation.

Nous présentons d'abord l'outil DSX y compris, de manière très concise, une partie de la bibliothèque SoCLib, enseignée dans plusieurs modules qui précèdent le nôtre. Nous introduisons ensuite l'application Motion JPEG, destinée à décoder des flux d'images au format JPEG.

2 L'Outil DSX

DSX assiste la conception d'applications logicielles décrites de manière statique sous forme d'un graphe de tâches et du matériel sous-jacent. Au niveau

d'études visé, comme pour le concepteur expérimenté, il est impératif d'avoir un contrôle maximum sur le système sur puce construit, tant au niveau de l'architecture, que du placement des objets physiques. Un objectif essentiel est de permettre au concepteur de l'application de contrôler de façon très fine non seulement le placement des tâches sur les processeurs, mais également le placement des différents objets logiques (piles d'exécution des tâches, canaux de communication, barrières de synchronisation, verrous, ...) sur les bancs mémoire physiques.

DSX décompose le développement d'un SoC en trois étapes : la conception d'une application décrite sous forme d'un graphe de tâches communicantes par des canaux de communication spécifiques appelés MWMR (*Multi Writer Multi Reader*, multi-écrivain multi-lecteur [1]), la conception d'une plate-forme matérielle pour héberger cette application et le déploiement de l'application logicielle sur la plateforme matérielle.

Le graphe des tâches et communications (TCG) est composé d'instances de tâches, de canaux de communication et d'autres ressources logicielles. Chaque tâche doit au préalable être définie. Cette définition comprend aussi bien la façon dont la tâche s'intégrera dans le graphe (entrées, sorties, ressources, ...) que la liste de toutes ses implémentations matérielles et logicielles. Ces implémentations se doivent d'être équivalentes, et seront ensuite considérées comme interchangeables. La description se fait à travers une API en Python [4].

La cible matérielle est une plate-forme construite à l'aide des IP-cores de la bibliothèque SoCLib [5]. SoCLib est une plate-forme ouverte conçue pour le prototypage virtuel des systèmes multi-processeurs sur puce. Il s'agit d'une bibliothèque de modèles de simulation SystemC des composants matériels. Les composants sont liés par un réseau d'interconnexion sur puce à interface VCI/OCP [6]. Les plateformes sont simulées soit au niveau CABA (*cycle accurate bit accurate*, cycle précis bit précis) soit au niveau transactionnel (TLM). Tous les composants sont intégrés sur une seule puce. L'architecture générique contient un nombre variable de processeurs RISC 32 bit programmables et de bancs mémoire embarquée. Elle contient du matériel pour l'affichage et pour la gestion des verrous, et des coprocesseurs spécifiques à l'application. Il y a deux types de composants : initiateurs (typiquement les processeurs et coprocesseurs) et cibles (typiquement, les bancs mémoires, terminaux, etc.). SoCLib est devenue une plate-forme reconnue au niveau national ; six partenaires industriels et onze laboratoires de recherche font partie du consortium.

3 L'Application MJPEG

Le traitement de flux, multimédia comme réseau, préconise un flux continu de données. Nous modélisons les applications de flux sous forme de tâches parallèles communiquant entre elles. Nous avons choisi l'application de décompression Motion JPEG car elle ne constitue pas en elle-même de difficulté particulière et nous permet de nous concentrer sur son développement avec DSX. L'intérêt particulier de cette application est qu'elle permet aux étudiants d'observer leurs progrès (L'image a-t-elle été decodée correctement ? Les blocs arrivent-ils dans le bon ordre ? Le flux d'images est-il decodé à vitesse suffisante ?)

La décompression d'une image JPEG [3] se fait en six étapes : (1) Demux, Démultiplexage du flux de données ; (2)VLD, Décompression du flux compressé issu d'un codage entropique ; (3) IQ, Application de facteurs de quantification ; (4) ZZ, Transformée ZigZag ; (5) IDCT, Transformée Cosinus inverse ; (6) LiBu, Construction de lignes complètes d'image. Les étapes IQ et ZZ ont été regroupées en une seule tâche. La tâche TG représente la chaîne de démodulation d'un flôt

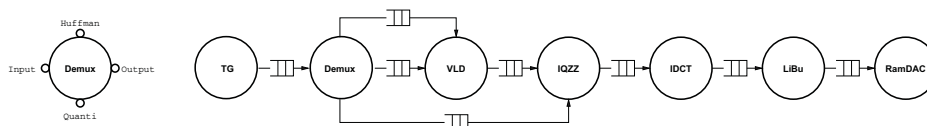


FIG. 1 – Tâche DEMUX et TCG de l'application Motion JPEG

d'images, la tâche RAMDAC son affichage. Ces tâches seront sur le SoC des coprocesseurs spécialisés.

Nous expliquons dans la suite comment l'outil DSX est utilisé dans notre enseignement de Master deuxième année.

4 Le Module MJPEG

Nous avons mis en place un module constitué d'une demi-journée de cours qui couvre les notions présentées ci-dessus, suivi de plusieurs demi-journées de travaux sur machine encadrés. L'application est d'abord validée en version logicielle, puis déployée sur plate-forme mono-processeur et multi-processeur. La partie finale du module comprend l'exploration de l'espace de conception afin de détecter les goulets d'étranglement et d'améliorer la performance.

4.1 Exécution sur Station de Travail POSIX

Il est souhaitable de pouvoir valider une application avant son déploiement sur un système matériel donné. Dans ce but, DSX prévoit la possibilité d'exécuter l'application sur une station de travail type Unix tout au long de sa conception.

Au début, une simple application composée d'une tâche productrice et une tâche consommatrice est à faire fonctionner. Dans tout le reste du TP, on s'intéresse à l'application MJPEG. La plupart des sources est fournie, il faut les faire fonctionner ensemble. Les tâches les plus simples, IQZZ et LIBU, sont à écrire. L'application est alors prête pour une mise au point sur station de travail. On peut l'exécuter, la déboguer, la profiler, ... avec tous les outils classiques disponibles sur la station de travail. Ceci minimisera le temps de débogage sur la plateforme embarquée ultérieure, qui n'est pas toujours aussi aisé (observabilité, temps d'exécution).

4.2 Exécution sur SoC Mono-Processeur

Ce TP vise à décrire une architecture matérielle en exploitant les composants matériels de la bibliothèque SoCLib et le déploiement de l'application. Nous simulons l'exécution du code binaire de l'application logicielle sur le modèle SystemC de l'architecture matérielle. Voici un exemple :

```
mapper.map( mapper.tcg["fifo"],          mapper.map(mapper.tcg["idct"],
  buffer = "cram1",                      run = "processor0",
  status = "cram1",                      stack  = "cram0",
  desc   = "cram1")                      desc   = "cram0",
                                                status = "uram0")
```

On se limitera à une architecture ne contenant qu'un seul processeur programmable et son cache, deux contrôleurs mémoire et un contrôleur de terminal TTY, organisée autour d'un micro-réseau générique à interface VCI. Les coprocesseurs TG et RAMDAC sont spécifiques, ils doivent faire l'objet d'un déploiement en tant que coprocesseurs rattachés par un contrôleur MWMR à l'interconnect VCI, et assignés à une adresse. Un déploiement valide pour TG est par exemple :

```
mapper.map( 'tg0', vci = 'vgmn0', addr = 0x94000000 )
```

4.3 Exécution sur SoC Multi-Processeurs

On vise la description d'une architecture matérielle multiprocesseurs (figure 2), générique dans le sens qu'on peut faire varier par un simple paramètre le nombre de processeurs et le nombre de bancs mémoire, ainsi que les caractéristiques des caches rattachés aux processeurs. On déploie l'application MJPEG sur cette plateforme générique en faisant varier les paramètres. Cette exploration architecturale est le but réel de l'outil DSX. Typiquement, on varie le nombre de processeurs, de bancs mémoire, le nombre de lignes des caches et de mots par ligne des caches.

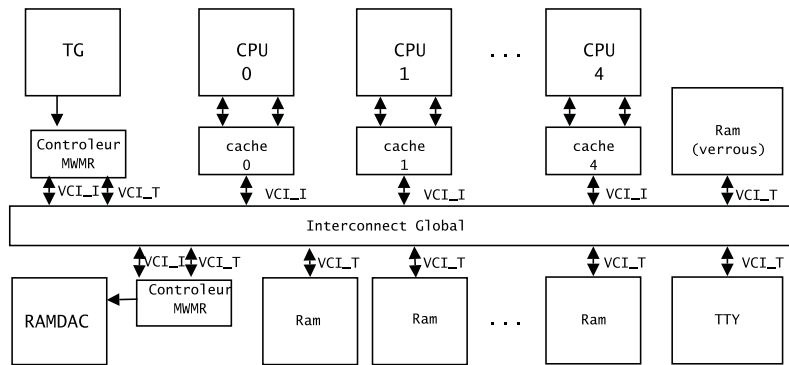


FIG. 2 – Architecture multi-processor du décodeur Motion JPEG

4.4 Exécution sur SoC Multi-Clusters

Afin d'augmenter encore le débit, on peut faire varier la structure logicielle de l'application, par exemple en dupliquant l'ensemble des tâches logicielles réalisant le traitement comme le montre la figure 4.4. Le déploiement est plus délicat :

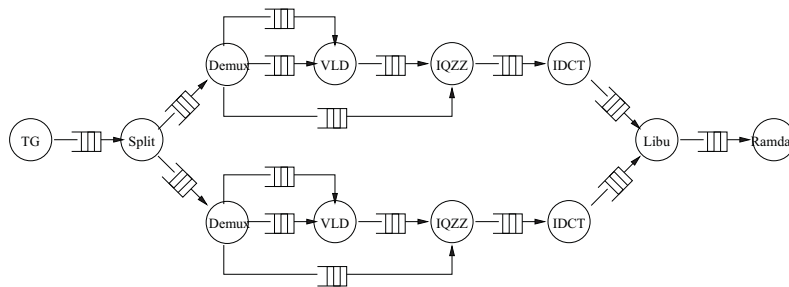


FIG. 3 – Variante de l'application avec deux images traitées en parallèle

dans ce type d'architecture multi-clusters, les temps d'accès à la mémoire sont très différents suivant qu'un processeur adresse la mémoire locale à son sous-système ou à un autre sous-système (architecture NUMA, *Non Uniform Memory Access*).

4.5 Coprocesseurs Spécialisés

L'objectif est de montrer comment introduire un coprocesseur matériel spécialisé dans une architecture comportant principalement des processeurs programmables. La tâche IDCT étant la plus gourmande en temps de calcul, nous allons analyser les gains de performance apportés par son implantation comme coprocesseur.

4.6 Exploration Architecturale Libre

DSX n'impose rien sur la structure même des fichiers de description Python. L'ensemble du langage est accessible (modules externes, structures de contrôle, ...). Le développeur peut donc semi-automatiser l'exploration architecturale par une description réalisant le placement explicite des tâches et ressources en fonction de paramètres externes. Dans ce TP, on propose une exploration architecturale avec plusieurs critères : la surface de circuit, la fréquence de travail et la consommation.

5 Conclusion et Perspectives

DSX a déjà permis la conception de plusieurs applications multimédia (décompresseurs Motion-JPEG, H-264, Theora), réseau (application de classification de paquets IP) ou autre (application de détection d'obstacles routiers basée sur une capture vidéo stéréoscopique).

L'enseignement décrit ici est organisé avec succès depuis deux ans et fera partie de la nouvelle proposition soumise au Ministère. A partir de 2009, il verra son volume horaire augmenté et comportera un volet TLM supplémentaire alors qu'actuellement il se limite au niveau CABA.

Références

- [1] E. Faure, A. Greiner, and D. Genius. A generic hardware/software communication mechanism for Multi-Processor System on Chip, targeting telecommunication applications. In *Proceedings of the 2006 conference on Reconfigurable Communication-centric SoCs*, 2006.
- [2] N. Pouillon and A. Greiner. DSX. URL=<https://www-asim.lip6.fr/trac/dsx/>, 2006-2008.
- [3] W. B. Pennebaker and J. L. Mitchell. *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, New York, NY, USA, 1993. The JPEG CCITT Recommendation.
- [4] Python Software Foundation. Python programming language - official website. URL=<http://www.python.org>, 1990-2008.
- [5] SoCLib Consortium. Projet SoCLib : Plate-forme de modélisation et de simulation de systèmes intégrés sur puce. Technical report, CNRS, 2003. <http://www.soclib.fr>.
- [6] VSI Alliance. Virtual Component Interface Standard (OCB 2 2.0). Technical report, VSI Alliance, Aug. 2000.