Introduction des systèmes Soc en DUT GEII

Pascal Aygalinc, S. Calvez

Introduction des systèmes Soc en DUT GEII

P. Aygalinc[a], S. Calvez[b]

[a] Dép. GEII, IUT d'Annecy et LISTIC, Université de Savoie-Mont Blanc, France
[b] LISTIC, Polytech Annecy Chambéry, Université de Savoie-Mont Blanc, France
Contact email : pascal.aygalinc@univ-smb.fr

Découvrir les systèmes embarqués basés sur des architectures ***Soc*** (*System On chip*) intégrant un système d'exploitation de type ***Linux*** est possible au niveau DUT GEII si l'on en restreint le champ des thèmes abordés. L'expérience menée ici sous la forme d'un projet réalisé au semestre 4 de cette formation s'intéresse uniquement à la conception de composants custom mappés en mémoire et au développement de pilotes de type caractère autorisant leur utilisation. L'objectif est de réaliser avec un système *Soc* la commande hybride d'un train miniature. Cet article traite des prérequis nécessaires ainsi que les moyens indispensables à mettre en œuvre par l'enseignant pour faciliter le prototypage rapide aussi bien au niveau matériel que logiciel.

I. Introduction

L'évolution des systèmes électroniques numériques nécessite de la part des développeurs d'avoir une vision complète d'un système complexe pour lequel l'association matériel-logiciel est de plus en plus imbriquée. Les systèmes basés sur des architectures *Soc* (***S**ystem **o**n **c**hip*) en sont une bonne illustration. Ils réclament d'adopter une approche «système» dans laquelle les notions de co-design matériel-logiciel sont nécessaires. A chaque étape de décomposition de l'application, le concepteur est alors amené à évaluer l'intérêt de choisir une solution reposant soit sur une logique programmable, soit sur une logique programmée, soit sur le concours des deux.

L'objet de cet article est de montrer que les formations de type *DUT* en ***G**énie **É**lectrique et **I**nformatique **I**ndustrielle* (*GEII*) peuvent constituer une excellente opportunité d'effectuer les premiers pas vers l'ingénierie des systèmes complexes utilisant comme solution technologique les systèmes *Soc*. En effet, au fil des trois premiers semestres de cette filière, les étudiants suivent des modules leur permettant d'aborder tous les fondamentaux nécessaires à la mise en œuvre de cette technologie (*langage de description matérielle, langage informatique structuré, programmation bas niveau, apports d'un système d'exploitation pour le développement d'applications en informatique industrielle*). Pour renforcer ces apprentissages et développer leur autonomie, des projets tuteurés ainsi que des travaux de réalisation sur ces thèmes sont aussi effectués durant ces semestres. Ces prérequis, décrits au paragraphe 2, autorisent ces futurs techniciens (*et pour certains futurs ingénieurs*) à effectuer une synthèse mettant en œuvre toutes ces compétences au semestre 4 lors d'un projet utilisant une structure *Soc* dotée d'un système d'exploitation de type *Linux.*

La réalisation d'un système embarqué répondant à un cahier des charges requiert une multitude de compétences et de connaissances ainsi que la maitrise d'un bon nombre d'outils informatiques de développement. Au niveau *DUT GEII*, l'enseignant est amené à opérer des choix sur les aspects qu'il souhaite aborder lors d'un projet et en masquer un

Article publié par EDP Sciences et disponible sur le site https://www.j3ea.org ou https://doi.org/10.1051/j3ea/20171007

certain nombre dont il assurera le développement. Il doit alors fournir des squelettes suffisamment documentés aussi bien pour la conception matérielle que logicielle. Par ailleurs, la station de travail de type *PC* doit offrir un certain nombre de services et d'outils de développement dont les installations sont à la charge de l'enseignant.

Au paragraphe 3, seront abordés les moyens à mettre en œuvre par l'enseignant pour arriver à rendre accessible la technologie *Soc* à des étudiants de *DUT GEII* pendant le projet de réalisation du semestre 4. La solution matérielle choisie repose sur une carte *DE0-Nano-SoC* du constructeur *Terasic*® (1) à base de *Cyclone V Altera*®. Elle s'inscrit dans le renouvellent du programme des *Services Nationaux du CNFM "Un étudiant - Une carte FPGA*" avec ce type de cartes.

Au paragraphe 4, le projet transversal sera décrit illustrant l'utilisation de ces moyens. Ce projet consiste à établir la commande hybride d'un train miniature du constructeur *Märklin*® (2) au format voie 1 (figure 1). Il est clair que ce projet pluridisciplinaire réclamera aussi des compétences en énergie et en automatique des procédés continus, autres aspects non développés ici mais qui constituent la facette génie électrique du *DUT GEII.*

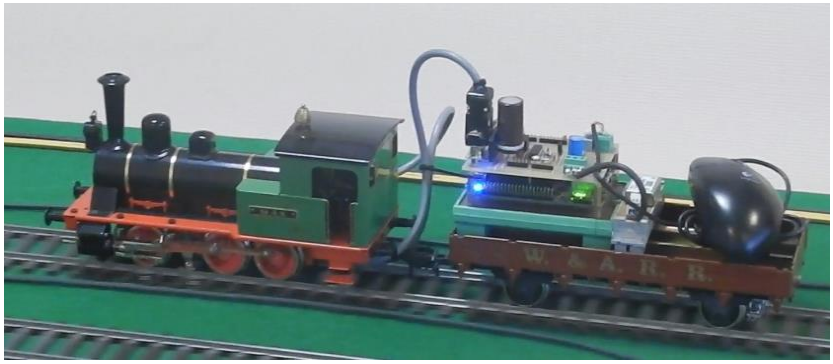

Fig.1. train miniature équipé d'un système embarqué de type Soc

Pour conclure, un bilan sera fait compte tenu de l'état d'avancement de ce projet, et de ce qu'il convient d'améliorer pour rendre plus autonome les étudiants dans cette introduction aux systèmes *Soc.*

II. Les prérequis

Les formations en *IUT* sont régies par des ***P**rogrammes **P**édagogiques **N**atio*naux (*PPN*) (3). Ils précisent les objectifs de la formation, les volumes horaires, les modalités pédagogiques et de contrôles de connaissances et des aptitudes. Pour la partie pratique, le département est libre de ses choix technologiques.

En *GEII* à *l'IUT d'Annecy*, au semestre 1, les étudiants découvrent l'électronique numérique et un langage de description matérielle, le langage *VHDL (module M1102)*. Pour la partie tp, les cartes sont des *DE2-Board* de l'intégrateur *Terasic*® : comme elles sont équipées de *FPGA* du constructeur *Altera*®, l'environnement de développement est *Quartus* et *ModelSim* pour la partie simulation. Parallèlement, les notions d'algorithmes, de représentation des données et les structures associées, la programmation classique sont enseignées au travers d'un langage structuré : *le langage C (module M1103)*. Les étudiants font leurs premières applications de type « *console* » dans l'environnement *Visual Studio*®.

Au semestre 2, l'initiation à la programmation bas-niveau est effectuée dans un module d'informatique industrielle utilisant des microcontrôleurs *(module M2103)*. Les problèmes de configuration des coupleurs, les mécanismes d'accès aux données de ces derniers

(mappage en mémoire, scrutation cyclique ou sur interruption) sont alors abordés sur des périphériques de base (*port d'entrée/sortie, timer, liaison série, bus i2c,…*). L'environnement de développement reste *Visual Studio*® qui permet d'effectuer de la cross-compilation pour des cibles de type *Arduino*®.

Depuis environ une dizaine d'années, l'apparition de modules complémentaires au choix dans les semestres 3 et 4 ont permis la notion de parcours afin que la formation soit en adéquation avec le projet personnel et professionnel des étudiants (*poursuites d'études longues, poursuites d'études professionnelles*) ainsi que de prendre en compte les attentes industrielles locales. Au département *GEII d'Annecy*, nous avons décidé d'établir un parcours orienté informatique au semestre 3 parmi les trois que nous proposons. Un module complémentaire dont j'ai la responsabilité permet de découvrir les objectifs d'un système d'exploitation et de ses propriétés avec pour l'illustrer le système *Linux* (4). Dans ce cadre, le développement d'applications en mode *user* et en *langage C* familiarise les étudiants à la virtualisation des périphériques et la communication synchrone avec ces derniers (5). C'est au travers des techniques de multiplexage qu'est abordée la communication asynchrone. Pour le mode *kernel*, les étudiants sont sensibilisés à la conception de pilotes de périphériques de type caractère (6). Les tps s'effectuent sur des cibles construites autour d'une carte de développement du constructeur *Axis*® (*Developper board LX*) (7). Elles proposent à l'aide d'une carte d'extension que j'ai développée, plusieurs périphériques *(écran lcd, deux afficheurs 7-segments, un capteur de température, liaison série, …)*. Elles sont dotées d'un système d'exploitation de type *Linux* et seront bientôt remplacées par de nouvelles cibles intégrant des cartes *DE1-Soc Board* (1).

A ce stade de la formation, les étudiants disposent de tous les fondamentaux nécessaires pour envisager la réalisation d'un projet de fin d'études basé sur un système embarqué de type *Soc*. Néanmoins, ces systèmes ne requérant pas uniquement des fondamentaux, l'enseignant est amené à limiter le domaine d'intervention des étudiants en effectuant des choix sur les aspects qu'il souhaite aborder avec eux et en masquant certains travaux qu'il prendra à sa charge. J'ai choisi de doter le système embarqué de type *Soc* d'un système d'exploitation plutôt qu'il soit de type *« bare metal »*. En effet, je considère que cette solution est plus enrichissante et de plus, elle est une application directe du module complémentaire vu précédemment.

Par ailleurs, afin d'éviter la découverte de nouveaux outils de développement pour la partie programmée (*FPGA*), le système *Soc* est basé sur une carte de développement de type *DE0-Nano-SoC* intégrant un *Cyclone V* du constructeur *Altera*®. On conserve ainsi les acquis sur l'environnement *Quartus* et les problèmes de configuration de la partie ***H****ard* ***P****rocessor* ***S****ystem* (*HPS*) peuvent être occultés au profit de la conception de composants « *custom* » mappés dans l'espace mémoire du *HPS*. Les étudiants devront pour cela découvrir le bus *Avalon* et l'outil *Qsys* intégré dans *Quartus*.

Comme le *HPS* comprend un *Dual-core ARM Cortex-A9* et que les composants « *custom* » ne sont pas découvrables par le système *Linux* (*au sens classique d'un bus PCI par exemple*), la notion de *Device Tree* devra être aussi abordée. En effet, cette technique est utilisée depuis quelques temps sur les architectures *ARM*. Elle consiste à ajouter au noyau une description de la configuration matérielle sous la forme d'un fichier binaire « ***d****evice* ***t****ree* ***b****lob* (*.dtb*) » produit à partir de différents fichiers « ***d****evice* ***t****ree* ***s****ource* (*.dts*) » écrits dans un langage spécifique.

Pour le développement des pilotes faisant appel aux notions de « *platform device, platform driver* » associés aux composants « *custom* », ainsi que pour la compilation du nouveau noyau les intégrant et les applications en mode *User*, c'est la suite *Altera's SoC* ***E****mbedded* ***D****esign* ***S****uite* (*EDS*) qui est utilisée. Notons que le système *sysfs*, de par les

attributs qu'il permet de créer pour chaque périphérique se révèle très efficace pour tester les composants « *custom* ». En effet, plutôt que de développer le service *ioctcl* et un programme en mode *user* qui l'utilise, quelques commandes de type *echo*, *cat* munies d'une indirection suffisent pour valider le fonctionnement.

Comme on peut le remarquer, de nouvelles compétences devront être enseignées et bon nombre d'outils sont nécessaires pour la réalisation d'un tel projet. L'enseignant doit fournir un poste de développement efficace afin que les étudiants focalisent leur attention sur les objectifs que l'on souhaite leur faire découvrir.

III. Le poste de développement

L'ensemble des machines de type PC au département fonctionnent sous *Windows 7 ou 10* ®. Quand une configuration particulière est requise, on a l'habitude de faire appel à des machines virtuelles à l'aide de *Oracle VM Virtual Box*® (8). En ce qui nous concerne, la machine hôte doit disposer de deux interfaces réseau et d'un port USB pour la liaison console avec le système. Elle doit être suffisamment puissante pour supporter un machine virtuelle de type ***Linux*** et la chaine *Quartus* qui est gourmande en temps *CPU*. Bien qu'il soit possible avec l'environnement fourni par *Altera* de travailler sous *Windows*®, les avantages de cette solution sont de pouvoir installer différents services facilitant la mise au point et le prototypage rapide. En effet, la seconde interface réseau est connectée physiquement à la carte *DE0-Nano-SoC* et est montée en pont pour la machine virtuelle sous *VirtualBox* afin de pouvoir établir les services suivants :

Un serveur *DHCP* dont l'objectif est de fournir l'adresse *IP* de la carte, qui, associé au service *TFTP*, autorise les différents téléchargements nécessaires au boot de la carte (*le noyau linux, le fichier binaire représentant la programmation de la partie FPGA, le fichier binaire associé au Device Tree décrivant l'ensemble des périphériques*). Le service *NFS* a pour objectif de rendre distant le système de fichiers *rootfs* qui contient, entre autres, les *modules* à tester et les *applications.* La configuration de ces services effectuée par l'enseignant ainsi que celle de « *u-boot* » permettent aux étudiants de pouvoir rapidement procéder à des tests sur tous les niveaux (*matériel, logiciel mode User et Kernel*). Pour obtenir une nouvelle version du projet, il suffit simplement sur la station de travail de copier/coller les nouveaux fichiers dans certains répertoires clés, ou d'actualiser des liens symboliques d'utilisation plus aisée pour la gestion de versions. Une modification sur le matériel ou sur le noyau nécessite un *reboot* de la carte pour être prise en compte, ce qui est peu coûteux en temps.

Nota
- ✓ *Si le réseau formé par la machine virtuelle et la carte DE0-Nano-SoC doit être spécifique à chaque poste de travail, il est possible de paramétrer la machine virtuelle en fonction du nom de la machine hôte par exemple. Il suffit de définir une loi de codage de la machine adresse associée à la carte bridgée et de l'utiliser via des scripts pour configurer la machine virtuelle (hostname, eth1, dhcp, tftp, …).*
- ✓ *L'auto-négociation entre les interfaces réseaux machine hôte sous Windows et celle de la DE0-Nano-SoC sous u-boot peut poser des problèmes de latence, voire d'échec pour une configuration en vitesse de 1Go/s. Pour ma part, il m'en a couté un switch pour résoudre ce problème.*

Bien que le site *http://rocketboards.org* fournisse toutes les indications pour mener à bien un développement sur une architecture *Soc* du constructeur *Altera*®, il convient de fournir aux étudiants des procédures ou scripts facilitant la génération de code (*makefile, script de menuconfig, …*).

- Au niveau matériel, les ressources proposées par *Terasic*® pour la carte *DE0-Nano-SoC* sont écrites en *Verilog*. Comme ce langage n'est pas enseigné au département, j'ai fourni un projet initial contenant la configuration du *HPS* et son instanciation en langage *VHDL*, ainsi qu'une procédure décrivant l'ajout de composants « *custom* » sous *Qsys*. Pour l'application visée, j'ai aussi joint une aide sur le bridge mappant les périphériques développés sur la partie *FPGA* dans l'espace mémoire du *HPS (the lightweidght HPS-to-FPGA AXI Master port)*.
- Au niveau logiciel, j'ai rappelé la procédure de construction du *Device Tree* en insistant sur ce qui fait le lien entre le matériel et la reconnaissance de celui-ci afin que le noyau puisse monter les bons pilotes de périphériques au démarrage du système (*mot clé « compatible » que l'on trouve dans le Device Tree et le pilote*). Dans la phase de test, les pilotes sont des modules qu'on installe avec la commande *insmod* évitant une compilation du noyau. Un exemple de pilote à trous a été mis à disposition pour expliquer les différentes structures nécessaires à mettre en œuvre quand on utilise le couple « *platform driver, platform device* », les différents services à écrire aussi bien pour l'exploration *(technique de probing)*, pour la création d'attributs pour *sysfs*, pour l'obtention d'un fichier de périphérique sous */dev*. De manière similaire, les scripts pour *menuconfig* nécessaires pour pouvoir intégrer les modules au noyau ont été fournis.
- Pour la programmation en mode *user*, le module complémentaire dispensé au semestre 3 est suffisant compte tenu des thèmes abordés.

Nota : L'outil Qsys lors de l'édition de composant « custom » ne permet pas de générer un fichier NomDuComposantCustom_hw.tcl suffisamment complet pour obtenir le champ compatible dans le Device Tree associé à ce composant. Bien qu'il soit déconseillé d'éditer ce fichier, je n'ai trouvé que cette solution pour inscrire à l'aide de la commande « set_module_assignment » les champs minimum nécessaires au pilote (embeddedsw.dts.vendor, embeddedsw.dts.group et embeddedsw.dts.compatible).

IV. Le projet

Afin de faire découvrir cette technologie *Soc* aux étudiants du parcours informatique, j'ai soumis un projet de fin d'études qui consiste à établir la commande hybride d'un train miniature (*hybride dans la mesure on cherche à faire une régulation de vitesse en prenant aussi en compte des informations binaires d'arrêt émises par les voies ou par la détection d'obstacles*). Les contraintes associées étaient : le système embarqué est basé sur une carte *DE0-Nano-Soc* et doté d'un système d'exploitation de type *Linux*. L'alimentation disponible au niveau de la voie est une tension alternative 24 V/50 Hz. La mesure de vitesse devra être faite à l'aide d'une souris au standard *PS2* dont les touches seront remplacées pour fournir les informations de détection d'obstacles et d'arrêt. Le pont en H nécessaire pour la commande du moteur a pour référence *A4973B* du constructeur *Allegro*™ (9).

L'application de cette méthodologie de travail a permis à quatre étudiants sur le projet initial demandé :

- ✓ de concevoir l'interfaçage de la carte *DE0-Nano-Soc* avec le train dont les principales fonctionnalités sont : *redressement, filtrage, conversion DC/DC, pont en H, interface PS2 (figure 2),*
- ✓ d'implanter sur la partie FPGA :

- un composant gérant la modulation de largeur d'impulsion (*PWM*), les signaux de phase et de brake du pont en H
- un composant assurant la mesure de vitesse avec une configuration particulière de la souris PS2.

✓ de développer deux pilotes de périphériques permettant de valider ces 2 composants

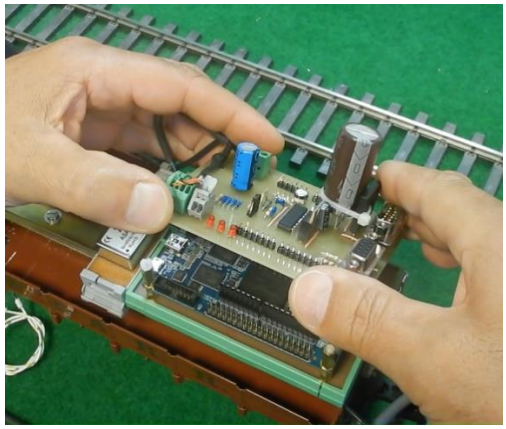

Fig.2. carte interfaçage *DE0-Nano-Soc* avec le train.

La vidéo de la figure 3 donne un aperçu plus démonstratif du travail réalisé.

TrainSoc.avi

Fig.3. Attention au départ !

V. Conclusion

L'avancement de ce projet est satisfaisant bien qu'il ne soit pas totalement terminé. Il est clair qu'il est réservé à de bons étudiants passionnés par le numérique et la programmation et peut-être nés *sous le signe du pingouin* !

Les problèmes rencontrés sont plus liés à des erreurs de jeunesse (gestion d'une voie bidirectionnelle pour le standard PS2, guerre des étoiles que l'on rencontre dans les prototypes des fonctions du noyau, …) qu'à la manipulation d'outils informatiques. La configuration de la station de développement proposée répond donc aux attentes.

Au niveau des outils de développement, proposer un environnement intégré afin de faciliter l'édition des sources C serait un plus (travailler avec un éditeur de texte d'un côté et de l'autre la compilation et l'édition de liens via des makefile semble d'une autre époque bien qu'étant très efficace).

Enfin cette introduction a masqué beaucoup de travaux tels que la configuration du preloader, du boot loader u-boot, la génération du système de fichier rootfs, …). J'en suis bien conscient pour les avoir réalisés.

Afin de persévérer dans cette voie, j'ai décidé d'utiliser comme nouvelle cible pour le module complémentaire une carte *DE1-Soc* sur laquelle j'ai ajoutée quelques périphériques de base réclamant pour certains des IPs, pour d'autres des composants « customs ». J'espère ainsi que les nouveaux tps permettront aux étudiants de gagner en autonomie sur ces systèmes *Soc* lors des projets.

Remerciements

Les auteurs souhaitent remercier vivement *Thierry Gil*, Ingénieur de recherche au CNRS, laboratoire LIRMM, Université de Montpellier pour sa disponibilité et ses encouragements, *Gilles Sicardi*, Technicien au département GEII d'Annecy pour le routage et la réalisation de la carte, *Alexis Artigues*, *Jean-Christophe Chudziak*, *Mathieu Collomb*, *Jules Simon*, Etudiants de 2ème année de DUT GEII : promotion 2014-2016 pour s'être beaucoup impliqués dans cette expérience. Que *Soizick Calvez*, trouve ici toute ma gratitude compte tenu de ses encouragements, ses multiples relectures et conseils donnés lors de ce travail. Les auteurs souhaitent également remercier l'ensemble des membres des Services Nationaux qui distribuent les logiciels ainsi que le GIP-CNFM et l'IDEFI FINMINA (ANR 2011-IDFI-0017) pour les co-financements apportés aux Services Nationaux pour développer ces nouvelles approches pédagogiques.

Références

1. TERASIC: *Website: http://www.terasic.com*
2. MARKLIN: *Website: http://www.marklin.com*
3. Programmes pédagogiques des DUTs: *Website: http://www.enseignementsup-recherche.gouv.fr/cid53575/programmes-pedagogiques-nationaux-d.u.t.html*
4. Linux Documentation: *Website: https://www.kernel.org*
5. C. Blaess, Title, *Développement système sous Linux, 4ème édition*, 964, Eyrolles (2016).
6. J. Corbet, A. Rubini, G. Kroah-Hartman, Title, *Linux Device Drivers*, 640, *O'Reilly Media* (2005).
7. AXIS *(Developper Board Lx)*: *Website: http://developer.axis.com/*
8. Oracle VM Virtual Box® machines virtuelles : *https://www.virtualbox.org*
9. ALLEGRO A4973B: *Website: http://www.allegromicro.com*