Mise en œuvre de Linux embarqué sur le processeur softcore Libre LEON

Ahmed Ben Atitallah, benatita@enseirb-matmeca.fr

Avec la participation de Patrice Kadionik, kadionik@enseirb-matmeca.fr

Sans nous en rendre compte, nous utilisons actuellement tous de nombreux systèmes embarqués. En effet, de nos jours, ces systèmes électroniques sont présents dans de nombreux domaines d'applications tels que les systèmes multimédia, la domotique, la voiture, l'avionique...

Le Libre est actuellement en train de prendre de plus en plus de place dans le développement des systèmes embarqués. Les initiatives Open Source sont nombreuses dans le domaine des systèmes embarqués, des processeurs *softcore* sous forme de blocs IP (*Intellectual Property*), des chaines de compilation croisées et de Linux embarqué.

Cet article décrit la mise en œuvre d'une plateforme de développement à base du processeur *softcore* Libre LEON et de Linux embarqué comme un système d'exploitation afin de gérer les tâches du système embarqué réalisé.

1. L'EMBARQUE ET L'OPEN SOURCE

Un système embarqué peut être défini comme « *un système électronique et informatique autonome ne possédant pas des Entrées/Sorties standards comme un clavier ou un écran d'ordinateur*».

1.1. Les contraintes des systèmes embarqués

Les contraintes liées à l'électronique embarquée sont :

- Les contraintes physiques : le système embarqué doit présenter un faible encombrement et un faible poids.
- La dissipation de puissance : la dissipation est directement reliée à l'autonomie du système embarqué.
- Le temps de développement : en termes de développement de l'architecture système et des fonctions logicielles associées.
- Le coût : le système ne doit pas être cher tout en tenant compte des performances à atteindre.

Dans le développement des systèmes embarqués, des choix doivent être réalisés pour satisfaire un ou plusieurs de ces critères. Par exemple, pour un ordinateur portable, l'autonomie est souvent négligée par rapport aux contraintes de performances et de taille. Par contre, pour un téléphone portable, on a intérêt à accroître l'autonomie.

1.2. Les SoC et l'embarqué

La conception des systèmes numériques embarqués est basée sur des SoC (*System on Chip*) ou SoPC (*System on Programmable Chip*).

Le développement des SoC et des SoPC se base sur l'usage d'un langage de description matérielle textuel tel que VHDL (*Very high speed integrated circuit Hardware Description Language*) ou Verilog avec la mise en œuvre de composants matériels déjà élaborés. Les blocs IP (*Intellectual Property*) sont généralement catalogués et échangés par Internet en respectant les standards VSIA (*Virtual Socket Interface Alliance*) et sont disponibles au libre téléchargement par exemple à partir du site www.opencore.org. Par contre, les modules IP commerciaux sont référencés sur le site www.us.design-reuse.com.

On peut définir un SoC ou un SoPC comme un ensemble de blocs IP intégrés dans un composant électronique avec au moins un processeur comme élément de traitement.

La figure ci-dessous illustre un exemple de SoC. Dans ces systèmes miniatures, on trouve :

- Des cœurs de processeurs (LEON, NIOS II, MicroBlaze, PowerPC…).
- Des bus de communication.
- Des mémoires.
- Des contrôleurs vidéo, audio...

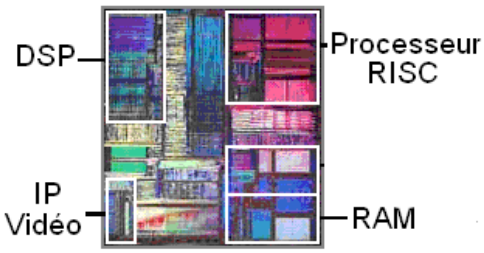

Figure 1. Exemple de système sur puce SoC

Dans le contexte de l'embarqué, on privilégie les processeurs qui sont écrits dans un langage de description matérielle (VHDL/Verilog) dont le code source peut être librement distribué et implanté dans n'importe quel circuit programmable FPGA (*Field Programmable Gate Array*), c'est-à-dire des processeurs sous forme d'un bloc IP Libre.

Pour les processeurs *softcore* Libres, on trouve principalement le processeur LEON, le processeur OpenRisc, le processeur F-CPU, le processeur LatticeMico32…
Ce sont généralement des processeurs 32 bits ayant une architecture de type Harvard avec un jeu d'instructions réduit RISC.

1.3. L'OS pour l'embarqué

Vue la complexité croissante des systèmes de traitement de l'information, on est arrivé à un stade où il devient impossible de gérer les ressources matérielles d'une architecture embarquée sans avoir recours à une logique programmée (logiciel) plus ou moins complexe. Cette interface logicielle est appelée système d'exploitation ou OS (*Operating System*).

Un système d'exploitation offre ainsi différents services pour mieux appréhender la complexité des systèmes embarqués :

- Apport du multitâche. Une application monolithique est divisée en une somme de tâches coopératives (système multitâche).
- Maîtrise des contraintes temporelles (système Temps Réel).
- Masquage des spécificités du matériel. On y accède de façon homogène et standard.
- Développement de pilotes de périphérique (*driver*) simplifié pour pouvoir avoir accès aux accélérateurs matériels.
- Apport d'un système de fichiers.
- Possibilité de communications réseau : pour un contrôle du système à distance par exemple.

Linux comme système d'exploitation devient de plus en plus prépondérant dans l'univers des systèmes embarqués.

On retrouve Linux dans l'embarqué pour les raisons suivantes :

- Logiciel Libre disponible gratuitement au niveau source.
- On a un système d'exploitation multitâche, un système de fichiers à disposition et une connectivité TCP/IP en standard.
- Fiabilité et stabilité reconnues du système.
- Il est possible d'avoir des versions Temps Réel.
- Support d'un grand nombre de processeurs autres que x86 : Power PC, ARM, MIPS, LEON, NIOS II, MicroBlaze...
- Taille du noyau modeste et compatible avec les tailles de mémoires utilisées dans un système embarqué.
- Différentes distributions proposées suivant l'usage : routeur IP, PDA, téléphone…

Linux pour l'embarqué existe pour 2 catégories de processeurs (32 bits minimum) :

- Processeur avec MMU (*Memory Management Unit*) : Linux embarqué, version Linux standard.
- Processeur sans MMU : µClinux, version Linux adaptée.

Par la suite, on s'intéresse à la mise en place d'un environnement embarqué à base de modules IP Libres. La plateforme matérielle s'articule autour d'une carte STRATIX II d'Altera (on peut aussi utiliser des cartes de Xilinx). Le cœur du système met en œuvre le module IP *softcore* LEON et Linux comme système d'exploitation pour gérer les différents périphériques et les tâches du système embarqué ainsi réalisé.

2. LE PROCESSEUR SOFTCORE LEON

2.1. Présentation du processeur LEON

LEON est un processeur RISC 32 bits. Il a été développé par l'ingénieur suédois Jiri GAISLER pour l'ESA (Agence Spatiale Européenne). Il a été conçu initialement comme un processeur compatible avec l'architecture SPARC V8 (*Scalable Processor ARChitecture*) avec une tolérance aux erreurs pour les applications spatiales (électronique durcie).

Il est actuellement disponible en deux versions destinées aux applications embarquées :

- Une version Libre sous licence LGPL pour LEON 2 et GPL pour LEON 3 disponibles pour l'éducation ou la recherche et téléchargeables à partir du site http://www.gaisler.com.
- La version commerciale nécessite de payer une licence.

2.2. Caractéristiques du softcore LEON 3

LEON 3 est un processeur *softcore* qui est écrit en langage VHDL et qui est fourni comme composant de la bibliothèque GRLIB contenant plusieurs blocs IP matériels (contrôleur Ethernet, unité de calcul en virgule flottante, contrôleur DMA…) et permettant de construire un système sur puce SoC ou SoPC.

LEON 3 est synthétisable sur des circuits numériques de type FPGA comme ceux de Xilinx ou Altera ou sous forme d'un circuit spécifique ASIC. Il fonctionne jusqu'à 125 MHz sur FPGA et 400 MHz sur un ASIC 0.13 µm.

Le softcore LEON 3 a les caractéristiques suivantes :

- Instructions aux normes SPARC V8.
- 7 étages de *pipeline*.
- Cache d'instructions et cache de données (Architecture Harvard).
- Cache configurable : 1 à 4 étages, 1 à 256 Ko par étage.
- Multiplication et division câblées ou non.
- Interface de bus AMBA 2.0.
- MMU.
- FPU.
- Utilisable en SMP (plusieurs processeurs).

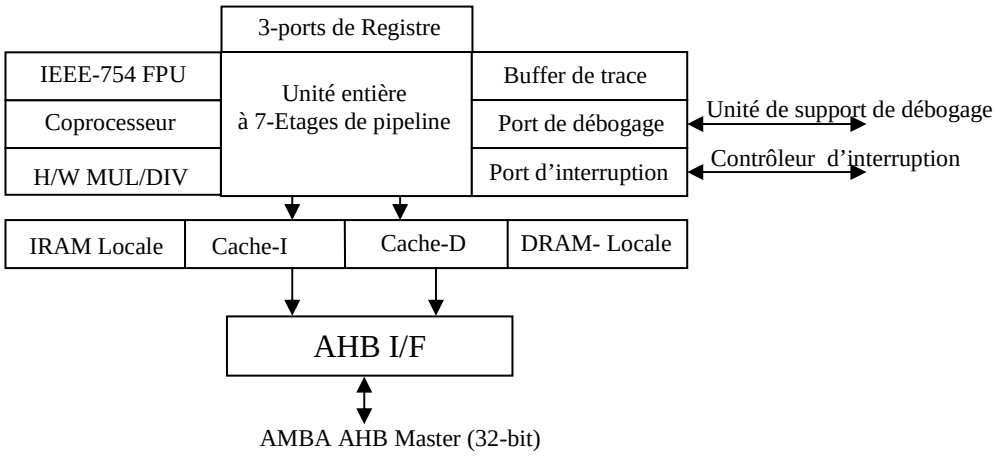

Figure 6. Schéma bloc du processeur LEON 3

La figure 6 illustre le schéma bloc du processeur LEON 3 avec toutes ses fonctionnalités. Certaines de ces fonctionnalités peuvent être enlevées soit parce que l'application visée ne les nécessite pas, soit pour générer des architectures plus petites en surface de silicium ou pour pouvoir atteindre des fréquences de fonctionnement plus élevées.

Du point de vue logiciel, on a à disposition les outils suivants :
- On peut choisir comme plateforme de développement Quartus II d'Altera.

- Simulateur ModelSim de Mentor Graphics.
- Chaine d'outils GNU (compilation croisée).
- Système d'exploitation supportés : SnapGear Linux, eCos, RTEMS.

2.3. Configuration et mise en place du processeur LEON 3

La mise en place du processeur LEON 3 sur une plateforme FPGA passe par trois étapes.

La première étape consiste à configurer le processeur LEON 3, ce qui revient à configurer la bibliothèque GRLIB qui est disponible au libre téléchargement à partir du site http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=156&Itemid=104
La configuration du processeur est effectuée sous Linux ou sous Windows en utilisant l'environnement Cygwin (http://www.cygwin.com) à travers la commande « *make xgrlib* » en mode console (figure 7) afin de choisir les paramètres de LEON 3 tels que la taille des caches (instructions et données), MUL/DIV câblées ou non, utilisation de MMU, FPU… ainsi que les blocs IP matériels pour la gestion des différents périphériques du système.

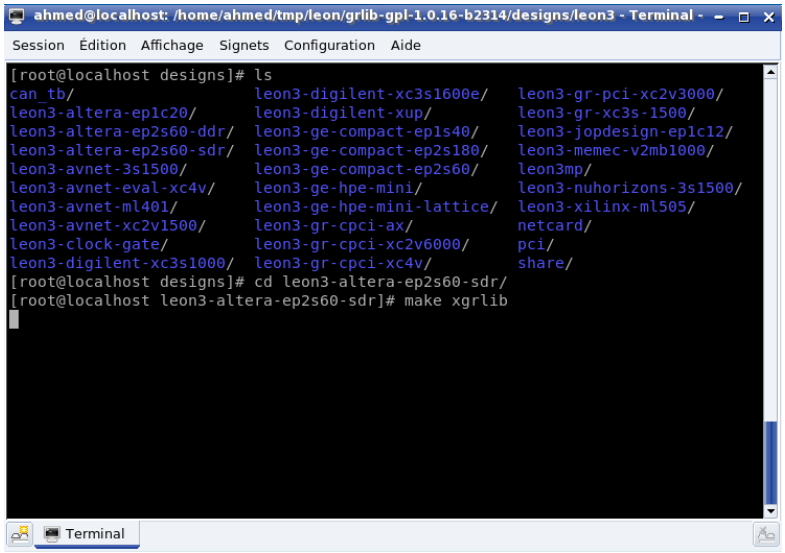

Figure 7. Trace de la commande « make xgrlib »

La commande « *make xgrlib* » donc permet de lancer la fenêtre suivante :

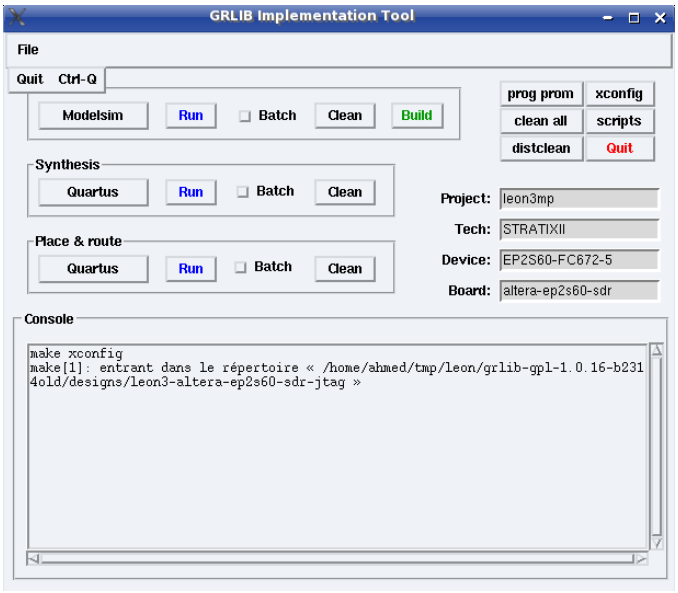

Figure 8. Fenêtre de choix

A travers cette fenêtre, on peut choisir l'outil de simulation (ModelSim) et l'outil de synthèse et de placement routage (Quartus II) pour la conception de notre système. Ainsi on peut configurer le processeur LEON 3 et ses périphériques en cliquant sur le bouton *« xconfig »* ou en mode console à travers la commande *« make xconfig »* (figures 9, 10 et 11).

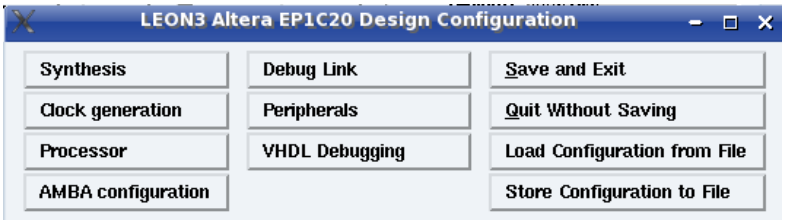

Figure 9. Fenêtre de configuration

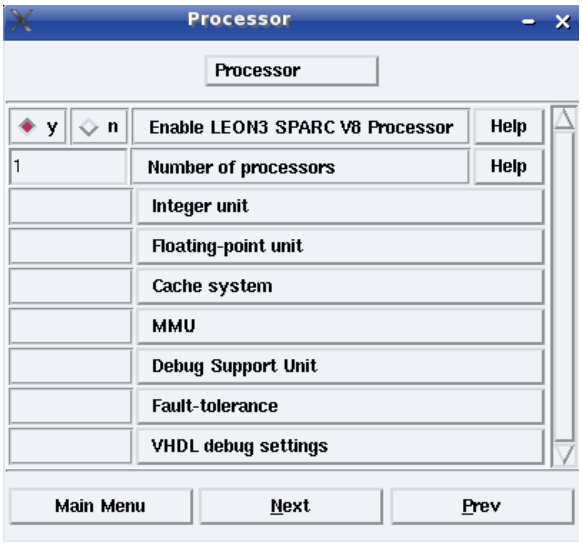

Figure 10. Choix des paramètres du processeur LEON 3

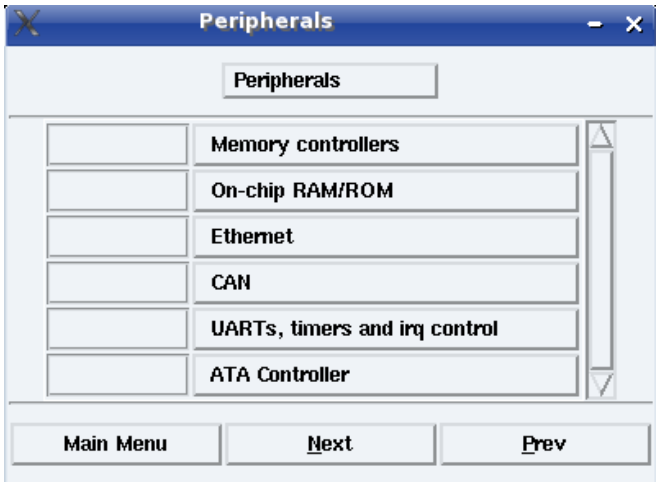

Figure 11. Choix des modules IP pour la carte cible

La deuxième étape consiste à adapter les Entrées/Sorties du processeur LEON 3 aux périphériques disponibles sur la carte : SDRAM, SRAM, ports série, port Ethernet, afficheur 7 segments, LED, boutons poussoirs...
Dans ce cas, si la carte est supportée par la bibliothèque GRLIB, il suffit de sélectionner la carte cible pendant la configuration du système (*Board Support Package* disponible). Si ce n'est pas le cas, on doit se référer aux documents techniques et à la schématique de la carte.

La troisième étape consiste à utiliser Quartus II (version Webpack) qu'on peut télécharger gratuitement à partir du site https://www.altera.com/support/software/download/altera_design/quartus_we/dnl-quartus_we.jsp afin de générer le projet en intégrant tous les modules IP. Après synthèse et routage, on a alors le fichier *.sof* de programmation du circuit FPGA correspondant au design SoPC. Il est possible aussi de simuler le design en utilisant ModelSim (version d'évaluation) qui est disponible à partir du site http://www.model.com/downloads/evaluations.asp.

Il reste maintenant à générer la partie logicielle afin de piloter les périphériques d'Entrées/Sorties du système en utilisant Linux et plus précisément la distribution Linux SnapGear qui est disponible sous licence GPL pour le processeur LEON 3.

3. MISE EN ŒUVRE DE LINUX SUR LE PROCESSEUR LEON 3

SnapGear Linux est une distribution Linux développée par la société CYBERGARD sous licence GPL pour les processeurs embarqués avec ou sans MMU afin d'offrir au développeur plus de souplesse dans le choix de l'architecture cible.
En fait, SnapGear Linux offre deux noyaux : le noyau µClinux pour les processeurs dépourvus de MMU et le noyau Linux pour les processeurs avec MMU.

Pour la mise en place de SnapGear Linux sur LEON 3, il convient de définir le design de référence qui servira après synthèse à programmer le composant FPGA de la carte cible, ici la carte Altera Stratix 2S60.

On a alors, après programmation du circuit FPGA Stratix II connecté à l'ensemble des périphériques de la carte cible, la configuration matérielle suivante :

- CPU LEON3 32 bits avec MMU et FPU.
- Interface SRAM (1Mo).
- Interface FLASH (16 Mo).
- Interface SDRAM (16 Mo).
- Timer.
- UART pour avoir un port série de communication.
- Interface CompactFlash (64 Mo).

La mise en place d'une plateforme de développement croisée nécessite le téléchargement de la version de SnapGear Linux portée pour LEON 3, le compilateur croisé *sparc-linux-gcc* ainsi que l'outil de débogage GRMON du site de Gaisler Research disponible à l'adresse http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=115&Itemid=103

La configuration de SnapGear Linux peut s'effectuer à travers la commande « *make xconfig* » sur le PC hôte sous Linux. Une série de fenêtres apparaît pour configurer le noyau SnapGear Linux et choisir ses applications (figures 12, 13, 14 et 15).

Figure 12. Menu principal

Figure 13. Choix du processeur LEON3

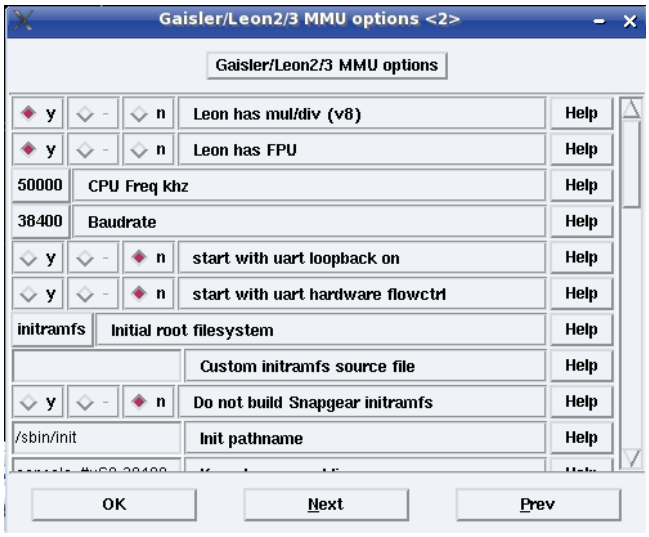

Figure 14. Choix des caractéristiques matérielles liées au processeur LEON 3

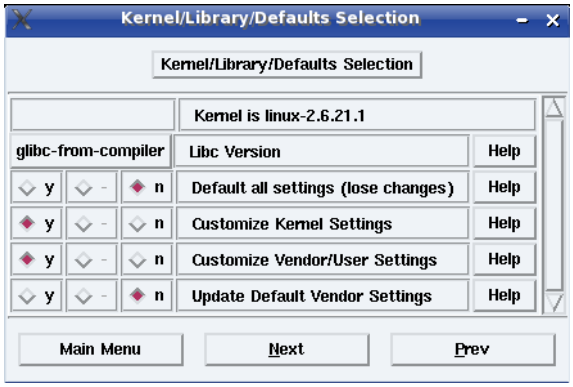

Figure 15. Configuration du noyau et de ses applications

Ensuite, à l'issue de la compilation croisée via la commande « *make* », on peut récupérer plusieurs images du système Linux contenant le noyau SnapGear Linux et le système de fichiers *root*, chacune de ces images étant destinée à une utilisation particulière :

- « image.dsu » : image destinée à être chargée via le DSU (*Debug Support Unit*).
- « image.tsim » : image destinée à être simulée.
- « image.dis » : c'est une version désassemblée du système.
- « image.flashbz » : image destinée à être flashée.

Enfin la dernière étape consiste à charger le fichier image *« image.dsu »* via le port *Debug Support Unit*. Cette étape suppose que le circuit FPGA est déjà programmé via le programmateur de Quartus II (figure 16) avec le fichier *.sof* contenant l'architecture matérielle compatible avec la partie logicielle.

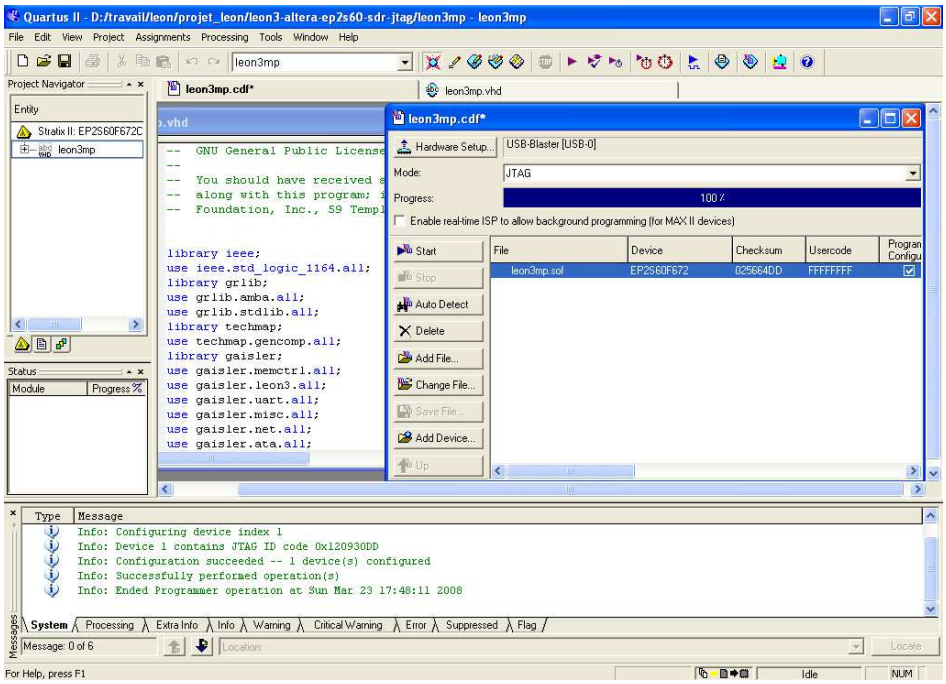

Figure 16. Programmation du circuit FPGA

Durant cette étape, on doit lancer l'outil de débogage nommé GRMON afin de pouvoir communiquer avec la carte cible en mode debug à travers le JTAG ou le port série (figure 17). A partir cette interface, on effectue le chargement de l'image du système « *image.dsu* » dans la mémoire SDRAM puis on exécute Linux via la commande « *run* ».

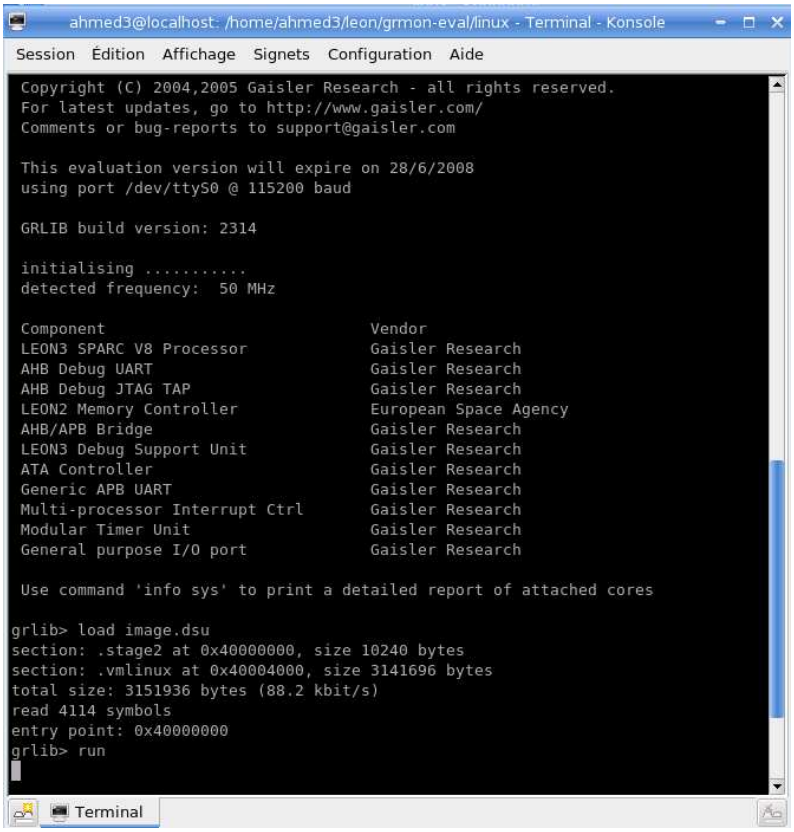

Figure 17. Chargement puis exécution de Linux à travers le port de débogage

L'accès à Linux se fera à travers l'outil *Minicom* sous Linux ou *Hyper Terminal* sous Windows. Ces outils nous permettent de communiquer avec la carte cible via le port série (38400, 8, N, 1).
La figure 18 montre le fonctionnement de SnapGear Linux sur le processeur LEON 3.

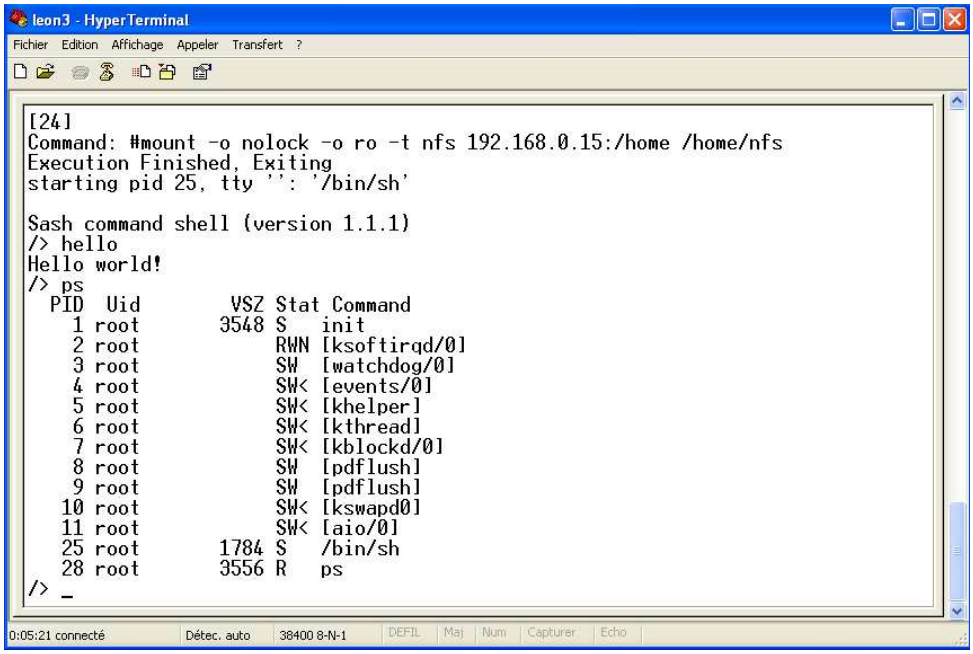

Figure 18. SnapGear Linux sur LEON 3 en action !

4. EXEMPLE D'APPLICATION SNAPGEAR LINUX : LE CODEUR VIDEO H.263 :

Un exemple d'application sous SnapGear Linux - LEON3 est la mise en place d'un codeur de traitement vidéo tel que le codeur H.263 disponible au libre téléchargement à l'adresse http://euler.slu.edu/~fritts/mediabench/mb2/mediabench2_video/h263enc/tmn-1.7.tar.gz

L'application a été ajoutée au menu « *Core Applications* » (figure 19) afin qu'on puisse la sélectionner lors de la configuration et le choix des paquetages à utiliser.

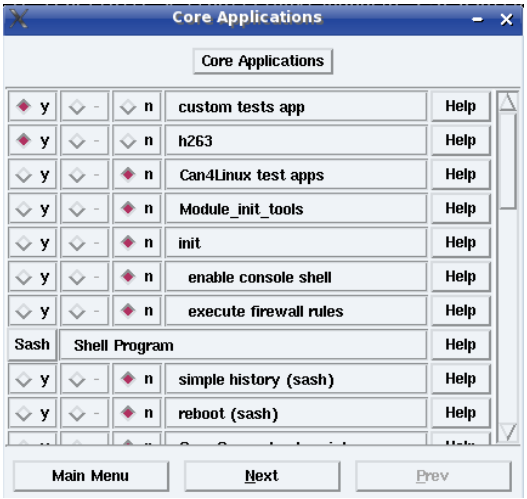

Figure 19. Menu « *Core Applications* » contenant le codeur H.263

Pour définir l'application vidéo dans le menu « *Core Applications* », on va associer un fichier *Makefile* au codeur H.263 comme le montre la figure ci-dessous :

```
EXEC = h263
OBJS = main.o io.o dct.o coder.o quant.o mot_est.o pred.o  snr.o \
       countbit.o putbits.o ratectrl.o sac.o putvlc.o
all: $(EXEC)
$(EXEC): $(OBJS)
        $(CC) $(LDFLAGS) –o $@ $(OBJS) $(LDLIBS$(LDLIBS_$@)) –lm
romfs:
        $(ROMFSINST) /bin/$(EXEC)
clean:
        -rm –f $(EXEC) *.elf *.gdb *.o
```

Figure 20. Makefile pour le codeur H.263

Les fichiers du codeur avec le fichier *Makefile* seront placés dans l'arborescence suivante : *~/linux/snapgear-pxx/user/h263*

Puis, on ajoute l'application vidéo au fichier *Makefile* (*~/linux/snapgear-pxx/user*) par l'addition de la ligne : *dir_$(CONFIG_USER_H263_H263) += h263*

Enfin, il reste à changer le fichier de configuration *~/linux/snapgear-pxx/config/config.in* par l'ajout de la ligne *bool 'h263' CONFIG_USER_H263_H263* afin que l'application choisie soit affichée dans le menu « *Core Applications* » lors de la configuration du système.

Les séquences vidéo utilisées pour tester le codeur sont disponibles à l'adresse http://ftp.osuosl.org/pub/xiph/websites/media.xiph.org/video/derf/ et sont stockées dans la carte mémoire *CompactFlash*. Pour ceci, on doit activer le contrôleur de la carte *CompactFlash* par la sélection de la ligne *Grlib ATA controller* à partir de l'interface de la configuration du noyau de linux et choisir le système de fichier *VFAT* comme montre les figures 21 et 22.

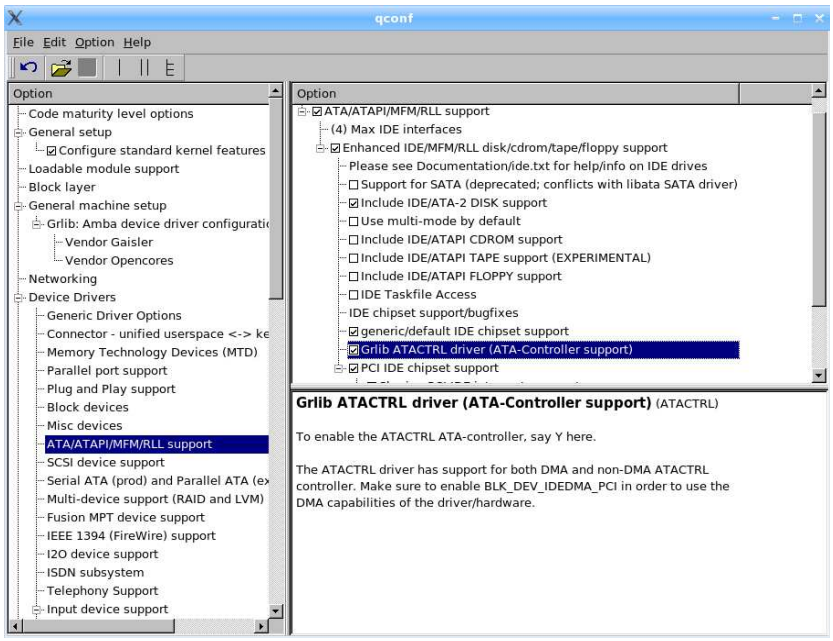

Figure 21. Activation du contrôleur pour l'accès à la carte *CompactFlash*

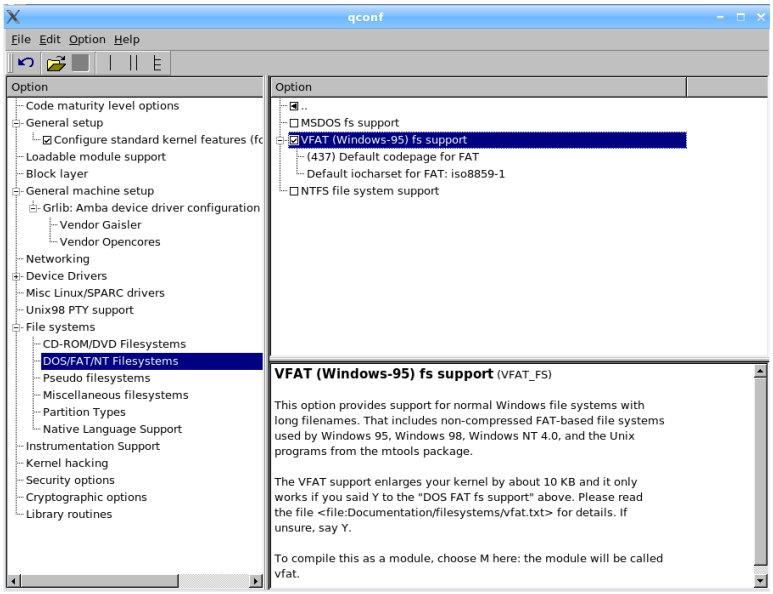

Figure 22. Activation du système de fichier *VFAT*

Après la génération et le chargement de l'image du système sur carte cible, le montage de la carte mémoire *CompactFlash* se fait au niveau du *shell* par la commande :
 # mount –t vfat /dev/hda1 /mnt.
Les figures 23 et 24 montrent les traces d'exécution du codeur H.263 sur la plateforme ainsi réalisée.

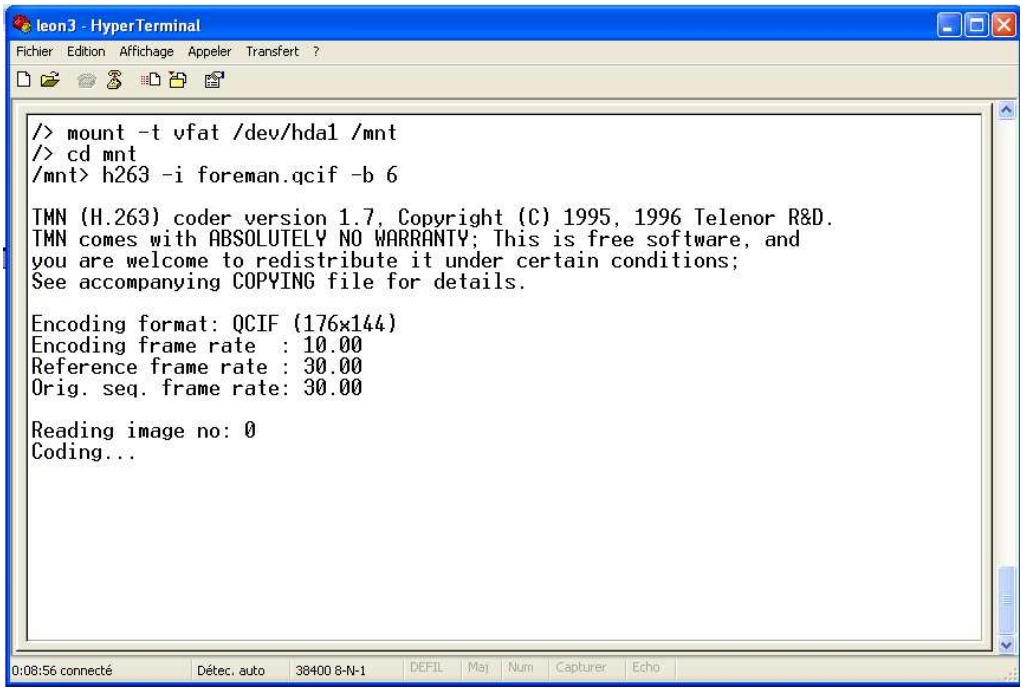

Figure 23. Exécution du codeur H.263 sur la séquence Foreman

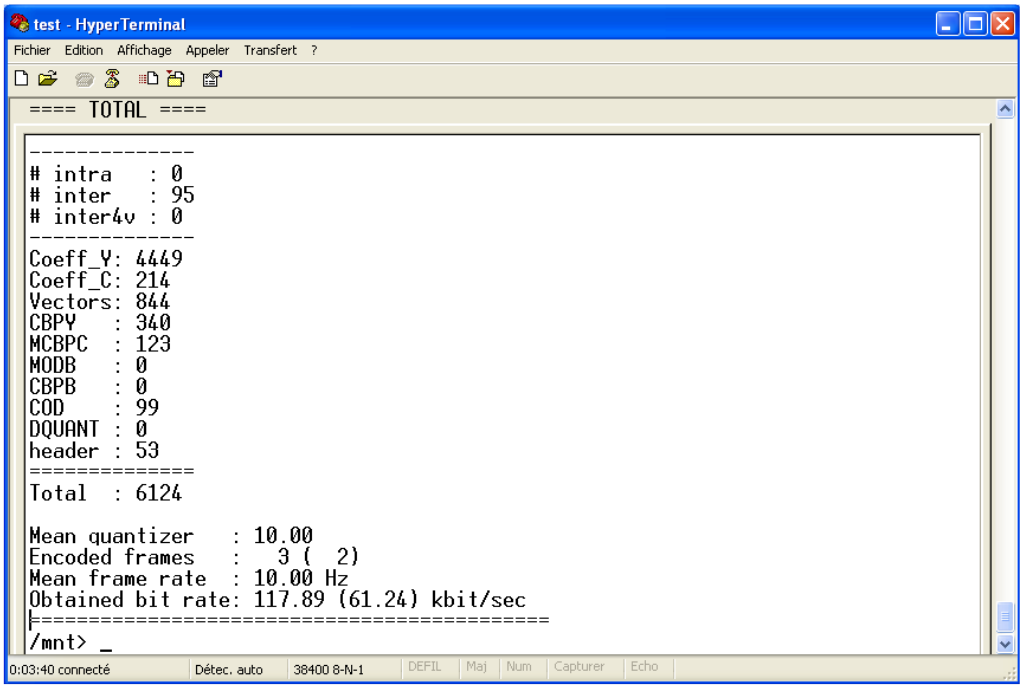

Figure 24. Résultats d'exécution du codeur H.263

5. CONCLUSION

Il est possible maintenant de concevoir un système embarqué permettant de gérer différents types d'applications embarquées en utilisant du Matériel Libre couplé à du Logiciel Libre.

Le choix d'un système d'exploitation comme Linux nous permet d'avoir un système embarqué incluant les fonctionnalités de base du noyau Linux mais avec la possibilité d'étendre les fonctionnalités de ce système en utilisant des briques logicielles issues du monde du Libre.

La conception des systèmes embarqués nécessite maintenant la mise en œuvre d'accélérateurs matériels sous forme de blocs IP qu'on peut développer en VHDL ou Verilog ou bien récupérer sous forme de blocs IP Libres.

La mise en œuvre de Linux embarqué (SnapGear) sur le processeur LEON 3 est véritablement une offre de *codesign* complète pour un développement conjoint matériel et logiciel !

Références :

1. Le site d'Altera http://www.altera.com
2. Le site de Xilinx http://www.xilinx.com
3. Le site de P. Kadionik http://www.enseirb.fr/~kadionik/
4. Le site de P. Nouel http://www.enseirb.fr/~nouel/
5. Le site des blocs IP libres http://www.opencores.org
6. Le processeur LEON http://www.gaisler.com/index.html
7. Le processeur OpenRisc http://www.opencores.org/projects.cgi/web/or1k/overview
8. Le processeur F-CPU http://www.f-cpu.org
9. Le processeur LatticeMico32 Open http://www.latticesemi.com/
10. Le forum du processeur LEON http://tech.groups.yahoo.com/group/leon_sparc/

11. Gaisler Research : GRMON User’s Manual
12. Gaisler Research : GRLIB IP Library User’s Manual
13. Gaisler Research : SnapGear Linux for LEON
14. N. Souissi, A. Ben Atitallah, F. Ghozzi, M. Ben Ayed, N. Masmoudi « Etude comparative de deux processeurs softcore NIOS II et LEON 3 », JTEA’06, Hammamet, Tunisie. 12-14 Mai 2006
15. P. Kadionik, A. Ben Atitallah, P.Nouel, H. Levi « De l’usage des processeurs softcore et de Linux pour la conception des systèmes embarqués », CETSIS '07, Bordeaux, France. 29-31 Octobre 2007