

## Porter Linux sur la DE-1

### Objectif

Vous apprendrez comment partitionner une carte SD, compiler et démarrer Linux.

### Génération de la carte SD

Le firmware du HPS est capable de démarrer depuis une carte SD. Il s'attend à trouver le SPL sur une partition contenant l'identificateur 0xA2. Cette partition peut se trouver n'importe où. Le reste peut être partitionné n'importe comment. L'exemple suivant comprend une partition 0xA2 contenant le SPL et U-Boot. Une partition FAT32 contient le kernel Linux (zImage), le device tree (dtb) et le bitstream (rbf). La partition ext4 contient le rootfs utilisé par Linux.

```
# fdisk -lu /dev/sdX
```

```
Disk /dev/sdX: 7948 MB, 7948206080 bytes
```

```
245 heads, 62 sectors/track, 1021 cylinders, total 15523840 sectors
```

```
Units = sectors of 1 * 512 = 512 bytes
```

Device Boot	Start	End	Blocks	Id	System
/dev/sdX1	62	167089	83514	b	W95 FAT32
/dev/sdX2	167090	182279	7595	a2	Unknown
/dev/sdX3	182280	15508989	7663355	83	Linux

Il existe de la documentation et un reference design pour la carte DE-1. Nous allons y récupérer une image de carte SD. Ceci nous fournira une base de travail stable et nous épargnera la création des partitions avec fdisk, la compilation de U-Boot et la création du rootfs avec Buildroot. Ce reference design étant vieux (Linux 3.12), nous allons recompiler et mettre à jour une version récente du kernel.

L'outil dd permet de copier une image binaire dans un autre fichier ou sur un périphérique. **Soyez extrêmement prudent avec cet outil.** Une mauvaise manipulation peut engendrer la destruction logique de n'importe quel disque de la machine. Utilisez l'outil lsblk pour afficher les périphériques de stockage disponibles. Une fois que vous êtes certains du nom de votre carte sd (/dev/sdX), effectuez la copie binaire.

<https://www.raspberrypi.org/documentation/installation/installing-images/linux.md>

- Téléchargez l'image SD « Linux console » de Terasic pour la DE1-SoC et copiez-la sur la carte SD avec dd. **ATTENTION : N'ECRASEZ PAS LE DISQUE DUR PAR ERREUR!**

- Utilisez picocom et vérifiez que votre carte démarre correctement jusqu'au login (le baudrate est 115200).

**NB : il nous faut 1 à 4 heures pour réinstaller une machine de laboratoire !**

## Compilation de Linux

Depuis 2018, les FPGA-SoC sont supporté dans le linux mainline, contrairement à avant où Altera maintenait sa propre version pour ses FPGA-SoC. Vous pouvez trouver les sources sur Github : <https://github.com/torvalds/linux/tree/master>. Vous pouvez rester sur le master.

Il existe une configuration du noyau pour tous les SoC FPGA. Le kernel n'est donc pas dépendant de la board utilisée ou de la famille du SoC (ceci est géré dans le device tree). La compilation est donc simple.

- Compilez le noyau pour ARM. Une toolchain ARM est présente sur vos machines dans /opt.
- Copiez le fichier zImage sur la carte SD, démarrez. Le boot doit échouer pendant l'initialisation du kernel.

```
make ARCH=arm CROSS_COMPILE=<TOOLCHAIN_DIR>/bin/arm-linux-gnueabi- socfpga_defconfig  
make ARCH=arm CROSS_COMPILE=<TOOLCHAIN_DIR>/bin/arm-linux-gnueabi- -j8
```

*Note: si vous obtenez une erreur lors de cette étape, installez le package libssl-dev.*

## Création du Device Tree

Le device tree décrit les composants présents dans le SoC et sur la carte. Le kernel met à disposition des dtsi à inclure dans votre device tree. Ces dtsi décrivent les principales familles de SoC (Cyclone V, Aria V, Aria 10, Stratix 10, etc.). Il nous reste qu'à activer et configurer les principaux composants présents sur la DE-1.

Un conseil que l'on rencontre souvent lorsque l'on désire porter une carte sur Linux ou U-Boot est de partir d'une carte similaire déjà supportée. Un device tree pour la DE-0 est déjà présent dans les sources du kernel, nous allons l'utiliser comme point de départ. Le device tree doit être compilé.

- Trouvez le device tree (dts) de la de0 et copiez-le sous un nouveau nom.
- Editez-le et analysez-le. Regardez également le contenu du dtsi inclus.
- Enlevez les nœuds gpio0, gpio1, gpio2 que nous n'utiliserons pas. Le nœud i2c0 est spécifique à la DE-0 et doit être enlevé également.
- L'adresse de base du bridge lightweight est fausse dans les sources, vous devez également la corriger
- Compilez le device tree. Copiez le dtb sur la carte SD dans la partition FAT32. La carte doit maintenant démarrer correctement jusqu'au login.

```
make ARCH=arm CROSS_COMPILE=<TOOLCHAIN_DIR>/bin/arm-linux-gnueabi- <DEVICE_TREE>.dtb
```

## Test du bridge HPS <-> FPGA

Les logs de démarrage du kernel (affichable avec la commande `dmesg`) donnent une bonne indication sur les éventuels problèmes de portage. Entre autre, les bridges doivent être correctement initialisés.

```
[ 1.019502] fpga_manager fpga0: Altera SOCFPGA FPGA Manager registered
[ 1.026590] altera_hps2fpga_bridge ff400000.fpga_bridge: fpga bridge [lwhps2fpga] registered
[ 1.035320] altera_hps2fpga_bridge ff500000.fpga_bridge: fpga bridge [hps2fpga] registered
```

Comme test final, vous allez tester le bridge `lwhps2fpga`. Un mapping de la mémoire physique est accessible par le fichier `/dev/mem`. Il existe un utilitaire nommé `devmem2` qui permet de lire et d'écrire dans ce fichier facilement. Le code source de cet utilitaire est disponible sur Github : <https://github.com/hackndev/tools/blob/master/devmem2.c>

- Chargez le bitstream que nous vous avons fourni à l'aide de Quartus. Deux des 7-segments doivent s'allumer.
- Cross-compilez l'utilitaire `devmem2` et transférez-le dans le rootfs de votre carte SD (`/home/root/` est un bon endroit).
- Utilisez `devmem2` pour lire l'état des boutons poussoirs et des switches. Les boutons sont mappés sur le `lwhps2fpga` bridge à l'offset `0x30` et les switches à l'offset `0x20`.

## Documents à rendre

Vous devez rendre un rapport à l'issu de ce laboratoire contenant les explications sur les différentes étapes de la réalisation de votre système.

Vous devez également rendre votre device tree final (dts) ainsi qu'un fichier `kernel.log` contenant les logs du kernel (`dmesg`).

Les fichiers sont à rendre sur Moodle.