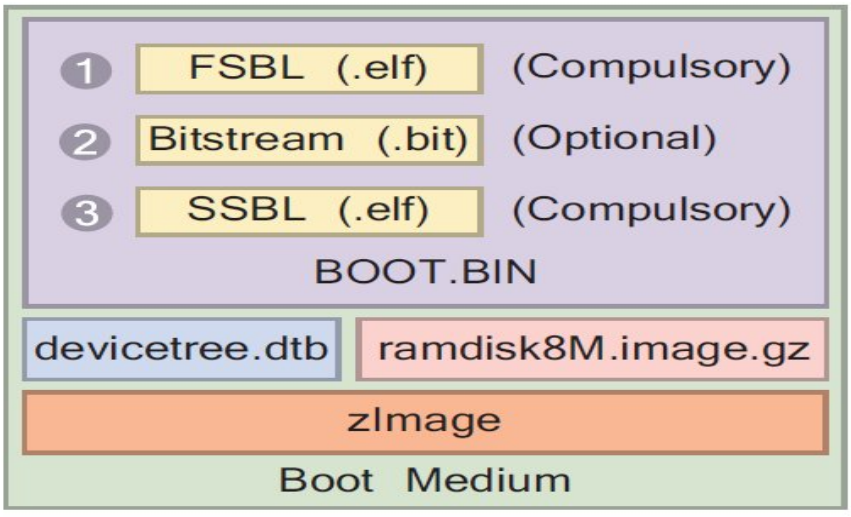
LINUX Digilent sur la carte ZYBO

La société Digilent fournit une distribution LINUX que l'on peut implémenter sur les cartes ZYNQ ZEDBOARD ou ZYNQ. Le noyau LINUX fourni par XILINX a été patché par la société Digilent afin de prendre en compte les éléments matériels de la carte ZYBO.
Un environnement de cross compilation type UBUNTU est nécessaire afin de compiler le chargeur de noyau U-BOOT le noyau LINUX et d'implanter la distribution LINUX.
La distribution LINUX de la carte ZYBO sera chargée dans la mémoire RAM (ramdisk)

1) La Phase de démarrage

Pour pourvoir démarrer la distribution LINUX, plusieurs fichiers sont nécessaires sur la SDCARD :

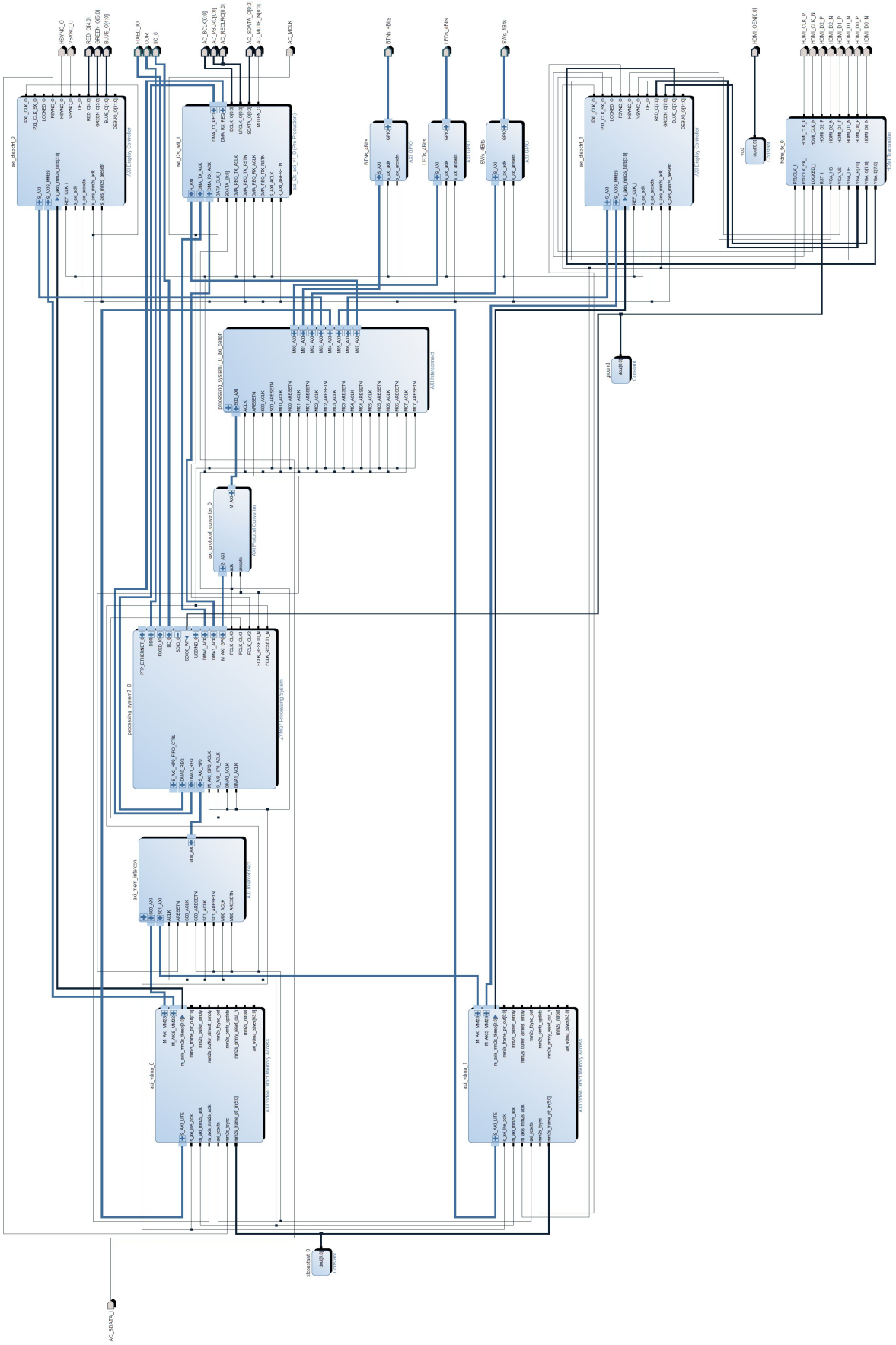
- ✓ Le fichier boot.bin qui intègre le premier chargeur de la société XILINX, la configuration matérielle du FPGA et le programme permettant de lancer le chargeur de boot U-BOOT
- ✓ Le fichier u-boot.elf qui prend en compte les informations de la carte et prépare le système de traitement ARM pour booter LINUX. Une fois la configuration terminée u-boot laissera la main au noyau LINUX pour l'ARM
- ✓ Le fichier ARM Linux (uImage) qui contient le noyau LINUX pour la carte ZYBO.
- ✓ ARM RAMDisk (uramdisk.image.gz) le RAMDisk qui est monté par le noyau et qui contient le système de fichier racine.



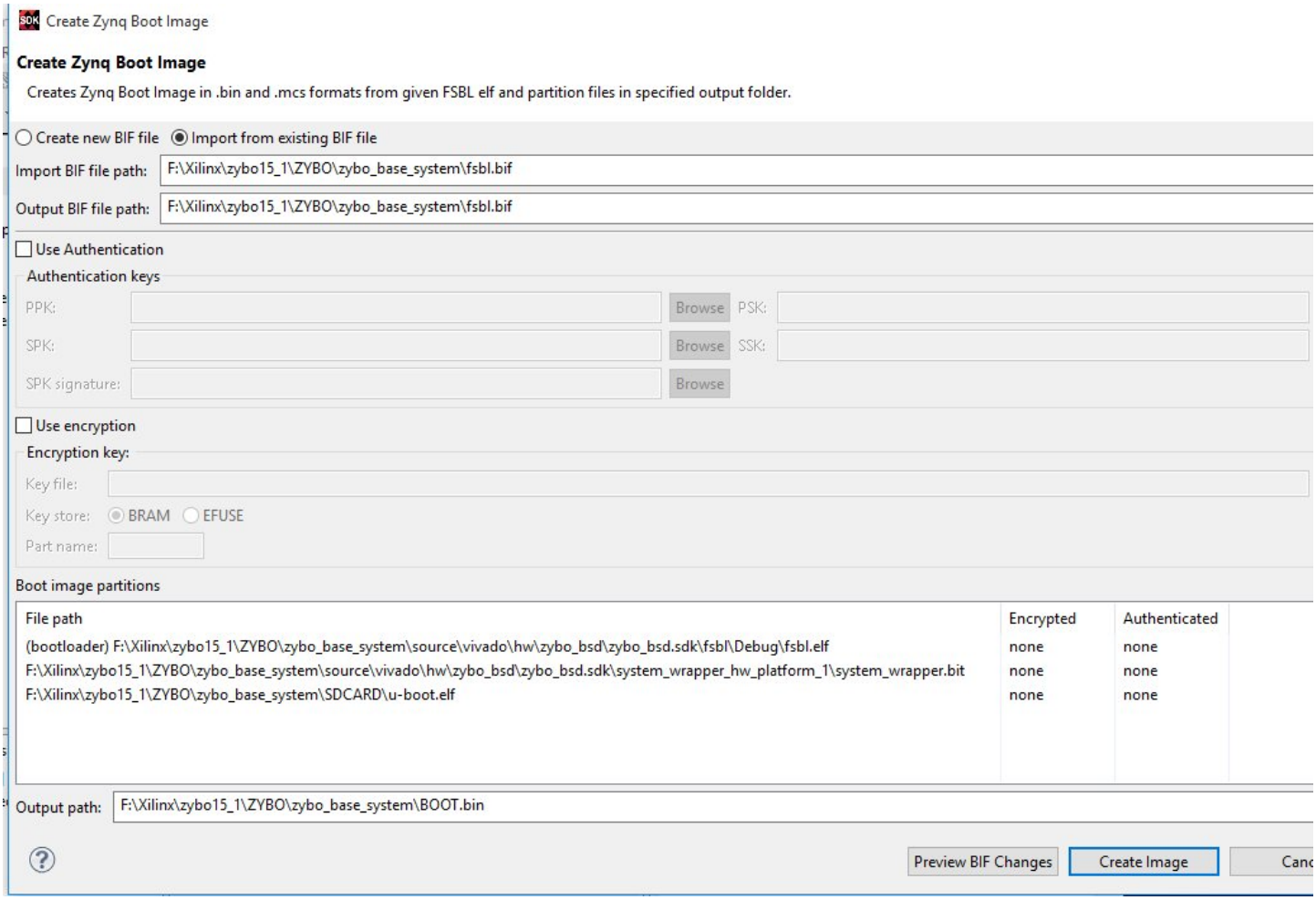
2) Configuration du matériel

La société Digilent met à notre disposition un package qui nous permettra de générer le fichier boot.bin. Ce package est nommé Zybo base design system.

Décompressez le fichier zybo_base_system.zip.
Ouvrir le projet sous Vivado, le board design est donné page suivante.
Générez par la suite le fichier bitstream et lancez le logiciel SDK



Sous SDK créez un fichier fsbl puis lancez la création d'un zynq boot image.
Les fichiers doivent être inclus dans l'ordre suivant :



Avant de pouvoir implémenter la distribution LINUX, il nous faut disposer d'un cross compilateur du type ubuntu 64 bits (une version 14.04 LTS suffit). Cette distribution ubuntu peut être lancée d'une machine virtuelle vmware par exemple. Sur Ubuntu on doit installer la suite logicielle vivado design et disposer d'une connexion internet

3) Compilation du chargeur de boot

La première étape consiste à installer les librairies qui permettent de lancer des exécutables 32 bits sur un OS 64 bits.

Se mettre en mode root

sudo apt-get install lib32z1 lib32ncurses5 lib32bz2-1.0

chargez les variables d'environnement afin de fournir le chemin du compilateur ARM pour Xilinx . Ce compilateur est installé avec la suite vivado

```
source /opt/Xilinx/Vivado/2014.4/settings64.sh
export PATH=/opt/Xilinx/SDK/2014.4/gnu/arm/lin/bin/:$PATH
export CROSS_COMPILE=arm-xilinx-linux-gnueabi-
export ARCH=arm
```

Chargez le package pour u-boot

```
git clone -b master-next https://github.com/DigilentInc/u-boot-Digilent-Dev.git
```
se placez dans le répertoire u-boot-Digilent-Dev

Lancez la compilation

```
make CROSS_COMPILE=arm-xilinx-linux-gnueabi- zynq_zybo_config
make CROSS_COMPILE=arm-xilinx-linux-gnueabi-
```

Le résultat de la compilation nous permettra d'exploiter le fichier u-boot, il faudra le renommer en u-boot.elf afin de l'exploiter et le copier sur la SDCARD.

4) Compilation du noyau LINUX

Restez en mode root
Téléchargez le noyau ici la version 3.18.0-xillinx
git clone -b master-next https://github.com/DigilentInc/Linux-Digilent-Dev.git

fournir le chemin de l'exécutable mkimage

```
PATH=$PATH:/usr/src/u-boot-Digilent-Dev/tools/
```

Lancez la compilation du noyau

```
make ARCH=arm CROSS_COMPILE=arm-xilinx-linux-gnueabi- xilinx_zynq_defconfig
make ARCH=arm CROSS_COMPILE=arm-xilinx-linux-gnueabi- UIMAGE_LOADADDR=0x8000 uImage
make ARCH=arm CROSS_COMPILE=arm-xilinx-linux-gnueabi- modules_install
```
Le fichier uImage qui se trouve dans arch/arm/boot sera chargé dans la SDCARD

5) Intégration du ramdisk

Téléchargez le fichier arm-ramdiskM.image.gz
se placez dans le répertoire Linux-Digilent-Dev
mkimage -A arm -T ramdisk -c gzip -d ./arm-ramdiskM.image.gz uramdisk.image.gz

L'image du système de fichier racine uramdisk.image.gz sera chargée par la suite sur la SDCARD

Il nous faut générer un fichier source du système de racine dont l'extension est .dtb et se trouve dans arch/arm/boot/dts/zynq-zybo.dts
Ce fichier contient des informations sur les éléments matériels qui doivent être bootés

Modifiez ce fichier de la façon suivante afin d'intégrer un système de fichier du type ramdisk

```
chosen {
49 /* bootargs = "console=ttyPS0,115200 root=/dev/mmcblk0p2 rw earlyprintk rootfstype=ext4 rootwait
devtmpfs.mount=1"; */
50 bootargs = "console=ttyPS0,115200 root=/dev/ram rw initrd=0x800000,8M init=/init earlyprintk
rootwait devtmpfs.mount=1";
51 linux,stdout-path = "/axi@0/serial@e0001000";
52 };
```

Générez le fichier devicetree.dtb

./scripts/dtc/dtc -I dts -O dtb -o ./devicetree.dtb arch/arm/boot/dts/zynq-zybo.dts

6) Modification du système de fichier racine

Le principal inconvénient d'utiliser un système de fichier du type ramdisk est l'impossibilité de modifier ce fichier une fois implantée sur la SDCARD. Seul l'environnement de cross compilation nous permettra de modifier ce fichier par les commandes suivantes :

Modifiez le nom du système de fichier
cp arm_ramdisk.image.gz ramdisk.image.gz

Décompressez le ramdisk
gunzip ramdisk.image.gz

Créez un répertoire temporaire pour y loger le contenu du RAMDISK
mkdir file_tmp

Montez le ramdisk
mount ramdisk.image –o loop ./file_tmp/

Faire les modifications du système de fichier dans le repertoire file_tmp. On pourra copier les modules générés lors de la compilation du noyau dans le répertoire /lib/modules

sudo umount file_tmp/
gzip ramdisk.image

mkimage –A arm –T ramdisk –C gzip –d ramdisk.image.gz uramdisk.image.gz

Copiez les fichiers suivants sur la SDCARD :

uImage
Boot.bin
Uramdisk.image.gz
Devicetree.dtb