Mise en place d'un environnement de développement linux pour ZedBoard

Auteur : Corentin Nouviale

29/07/2015

Version : 1.1

Sommaire

1. Préparation de l'environnement

Sources :

Configuration de yocto :

http://picozed.org/content/building-zedboard-images

Git de meta-xilinx :

https://github.com/Xilinx/meta-xilinx

Wiki Xilinx, création d'un boot.bin :

http://www.wiki.xilinx.com/Prepare+Boot+Image

Tuto yocto : http://wiki.elphel.com/index.php?

title=Yocto_tests

1.1. Téléchargement des sources à partir des dépôts git

Fido correspond à la dernière version de yocto qui ne contient pas une version 3.10 du noyau.

Pour obtenir une version 3.10, utiliser la version Dizzy.

Télécharger Yocto meta layer:

```
$ git clone -b fido git://git.yoctoproject.org/poky.git
```

Télécharger Xilinx meta layer:

```
$ git clone -b fido git://github.com/Xilinx/meta-xilinx
```

Télécharger meta-oe layer:

```
$ git clone -b fido git://github.com/openembedded/meta-oe
```

1.2. Modification des fichiers de configuration (.conf)

Dans le répertoire poky/

```
$ source oe-init-build-env
```

Puis modifier le fichier

poky/build/conf/bblayers.conf

Ajouter les 2 lignes :

BBLAYERS ?= " \

<path to layer>/meta-xilinx \

<path to layer>/meta-oe/meta-oe \

"

Configurer la machine dans 'poky/build/conf/local.conf' modifier la ligne :
MACHINE ?= "xxxx"

Par :

MACHINE ?= "zedboard-zynq7 "

Ainsi que la ligne :

PACKAGE_CLASSES ?= "package_rpm"

Par :

PACKAGE_CLASSES ?= "package_deb"

Juste après, ajouter la ligne :

IMAGE_FEATURES += "package-management"

Vous pouvez limiter le nombre de threads en ajoutant les lignes :

BB_NUMBER_THREADS = "nb"

PARALLEL_MAKE = "-j nb"

nb = nombre de cœurs x 2

Exemple pour un processeur 8 cœurs :

BB_NUMBER_THREADS = "16"

PARALLEL_MAKE = "-j 16"

Puis ajouter la ligne :

DISTRO_HOSTNAME = "zynq"

1.3. Compilation de poky

Se mettre dans le répertoire 'poky/'

```
$ source oe-init-build-env
```

```
$ bitbake core-image-minimal
```

Les résultats se trouvent dans le répertoire ‘poky/built/tmp/deploy/images/ <nom de la machine>

(En cas de problème, supprimer le répertoire 'tmp' dans 'poky/build’)

1.4. Installation du SDK Xilinx

```
$ sudo apt-get install ia32-libs
```

```
$ export CROSS_COMPILE=arm-xilinx-linux-gnueabi-
```

```
$ source <Xilinx Tools installation directory>/ISE_DS/settings64.sh
```

(Utiliser settings32.sh sur un système 32-bit)

2. Création du chargeur de démarrage (bootloader)

2.1. Téléchargement des sources:

Source:

http://www.wiki.xilinx.com/Fetch+Sources

```
$ git clone git://github.com/Xilinx/u-boot-xlnx.git
```

Repository Name	**Content**
linux-xlnx.git	The Linux kernel with Xilinx patches and drivers
u-boot-xlnx.git	The u-boot bootloader with Xilinx patches and drivers
device-tree.git	Device Tree generator plugin for xsdk

(sudo ln -s /usr/bin/make /usr/bin/gmake)

2.2. Utilisation de Vivado pour créer une description du matériel

Télécharcher Xilinx Vivado (il est nécessaire d'activer une licence) :

http://www.xilinx.com/support/download.html

Utiliser Vivado afin de créer un .hdf en suivant le tuto suivant (joint en pdf également)

http://zedboard.org/zh-hant/node/1454

2.3. Construction du First Stage Boot Loader (FSBL)

Source:

www.wiki.xilinx.com/Build+FSBL

Utiliser la commande hsi livrée avec Vivado :

```
$ hsi
```

```
hsi% open_hw_design <hardware.hdf>
```

```
hsi% generate_app -hw <hw_design> -os standalone -proc
ps7_cortexa9_0 -app zynq_fsbl -compile -sw fsbl -dir
<dir_for_new_app(attention, le chemin ne doit pas être trop long)>
```

```
(hw_design = zynq_design_1_imp)
```

2.4. U-boot

Sources:

http://www.wiki.xilinx.com/Build+U-Boot

Aller dans le répertoire 'u-boot-xlnx'

(apt-get install libssl-dev)

```
$ export CROSS_COMPILE=arm-xilinx-linux-gnueabi-

$ make zynq_zed_config
$ make

$ cd tools
$ export PATH=`pwd`:$PATH)
```

Le résultat de la compilation se trouve dans le dossier 'u-boot-xlnx'

3. Mise en place de l'environnement

Sources:

Doc sur la ZedBoard :

http://architechboards-zedboard.readthedocs.org/en/latest/board.html
http://zedboard.org/sites/default/files/documentations/GS-AES-Z7EV-7Z020-G-V7.pdf
http://zedboard.org/sites/default/files/GS-AES-Z7EV-7Z020-G-14.1-V5.pdf
http://zedboard.org/sites/default/files/ZedBoard_HW_UG_v1_1.pdf

3.1. Création du boot.bin

http://www.wiki.xilinx.com/Prepare+Boot+Image

Dans un dossier, mettre les fichiers :

- executable.elf (qui se trouve dans le dossier donné après le '–dir' de generate_app (hsi))

- u-boot (qui se trouve dans le dossier u-boot-xlnx et à renommer en u-boot.elf)

Le reste des fichiers ce trouvent dans 'poky/build/tmp/deploy/images/zedboard-zynq7'

- zedboard-zynq7.dtb

- core-image-minimal-zedboard-zynq7-<nombre>.rootfs.cpio.gz.u-boot (que je conseil de renommer en 'core-image-minimal-zedboard-zynq7.rootfs.cpio.gz.u-boot')

- uImage—3.14-xilinx+git0+2b48a8aeea-r0-zedboard-zynq7-<nombre>.bin (que je conseil de renommer en 'uImage.bin')

Créé un fichier boot.bif avec :

```
image: {
	[bootloader]executable.elf
	u-boot.elf
	[load=0x2a00000]zedboard-zynq7.dtb
	[load=0x2000000]core-image-minimal-zedboard-zynq7.rootfs.cpio.gz.u-boot
	[load=0x3000000]uImage.bin
 // currently bootgen requires a file extension. This is just a renamed uImage
}
```

Exécuter la commande:

```
$ bootgen -image boot.bif -o i boot.bin
```

Créé un fichier uEnv.txt avec :

bootcmd=fatload mmc 0 0x3000000 uImage.bin; fatload mmc 0 0x2A00000 zedboard-zynq7.dtb; fatload mmc 0 0x2000000 core-image-minimal-zedboard-zynq7.rootfs.cpio.gz.u-boot; bootm 0x3000000 0x2000000 0x2A00000

uenvcmd=boot

3.2. Préparation de la carte SD

Sur la carte SD mettre les fichiers :

-boot.bin

- core-image-minimal-zedboard-zynq7.rootfs.cpio.gz.u-boot

-uImage.bin

-zedboard-zynq7.dtb

-uEnv.txt

3.3. Préparation de la carte ZedBoard

Avant d'allumer la carte, la préparée pour un boot sur la carte SD :

By default, the ZedBoard uses the SD Card configuration mode. The boot mode pins are MIO[8:2] and are used as follows:

- MIO[2]/Boot_Mode[3] sets the JTAG mode
- MIO[5:3]/Boot_Mode[2:0] select the boot mode
- MIO[6]/Boot_Mode[4] enables the internal PLL
- MIO[8:7]/Vmode[1:0] are used to configure the I/O bank voltages, however these are fixed on ZedBoard and not configurable

Table 18 – ZedBoard Configuration Modes

Xilinx TRM→	MIO[6] Boot_Mode[4]	MIO[5] Boot_Mode[2]	MIO[4] Boot_Mode[1]	MIO[3] Boot_Mode[0]	MIO[2] Boot_Mode[3]
JTAG Mode					
Cascaded JTAG					0
Independent JTAG					1
Boot Devices					
JTAG		0	0	0	
Quad-SPI		1	0	0	
SD Card		1	1	0	
PLL Mode					
PLL Used	0				
PLL Bypassed	1				
Bank Voltages					
MIO Bank 500			3.3V		
MIO Bank 501			1.8V		

(Image provenant du document de zedboard.org page 29)

MIO 6: set to GND
MIO 5: set to 3V3
MIO 4: set to 3V3
MIO 3: set to GND
MIO 2: set to GND
jp6

3.4. Préparation de minicom

Installer minicom puis le configurer :

```
$ sudo apt-get install minicom

$ sudo minicom -ws
```

Aller dans « Configuration du port série » le configurer :

```
+-----------------------------------------------------------------------+
| A - Serial Device : /dev/ttyACM0                                      |
| B - Lockfile Location : /var/lock                                     |
| C - Callin Program :                                                  |
| D - Callout Program :                                                 |
| E - Bps/Par/Bits : 115200 8N1                                         |
| F - Hardware Flow Control : No                                        |
| G - Software Flow Control : No                                        |
|                                                                       |
| Change which setting?                                                 |
+-----------------------------------------------------------------------+
     | Screen and keyboard |
     | Save setup as dfl   |
     | Save setup as..     |
     | Exit                |
     | Exit from Minicom   |
     +---------------------+
```

3.5. Lancement de la carte

Allumer la carte puis lancer minicom :

```
$ sudo minicom
```

Une fois cela fait, démarrer le system :

```
u-boot>boot
```

En cas d'erreur, essayer les commandes suivantes :

```
u-boot> fatload mmc 0 0x3000000 uImage.bin
u-boot> fatload mmc 0 0x2A00000 zedboard-zynq7.dtb
u-boot> fatload mmc 0 0x2000000 core-image-minimal-zedboard-
zynq7.rootfs.cpio.gz.u-boot
u-boot> bootm 0x3000000 0x2000000 0x2A00000
```

4 . Configuration du noyau

AV : avant bitbake core-image-minimal

AP : après bitbake core-image-minimal

4.1. Choisir la version du noyau (AV)

Editer le fichier:

meta-xilins/conf/machine/include/machine-xilinx-default

Modifier ensuite la ligne :

PREFERRED_PROVIDER_virtual/kernel ?= "<nom linux >"

Le nom linux peut être :

- Linux-xlnx
- Linux yocto

Choisir ensuite la version du noyau au modifiant la ligne :

PREFERRED_VERSION_<nom linux> ?= "<numéro de version>"

Pour connaitre les versions possibles il suffit d'aller dans le

dossier : meta-xilins/recipes-kernel/linux/<nom linux>

Le nom des fichiers correspond aux numéros des versions disponibles.

4.2. Sélectionner les sources du noyau (AP)

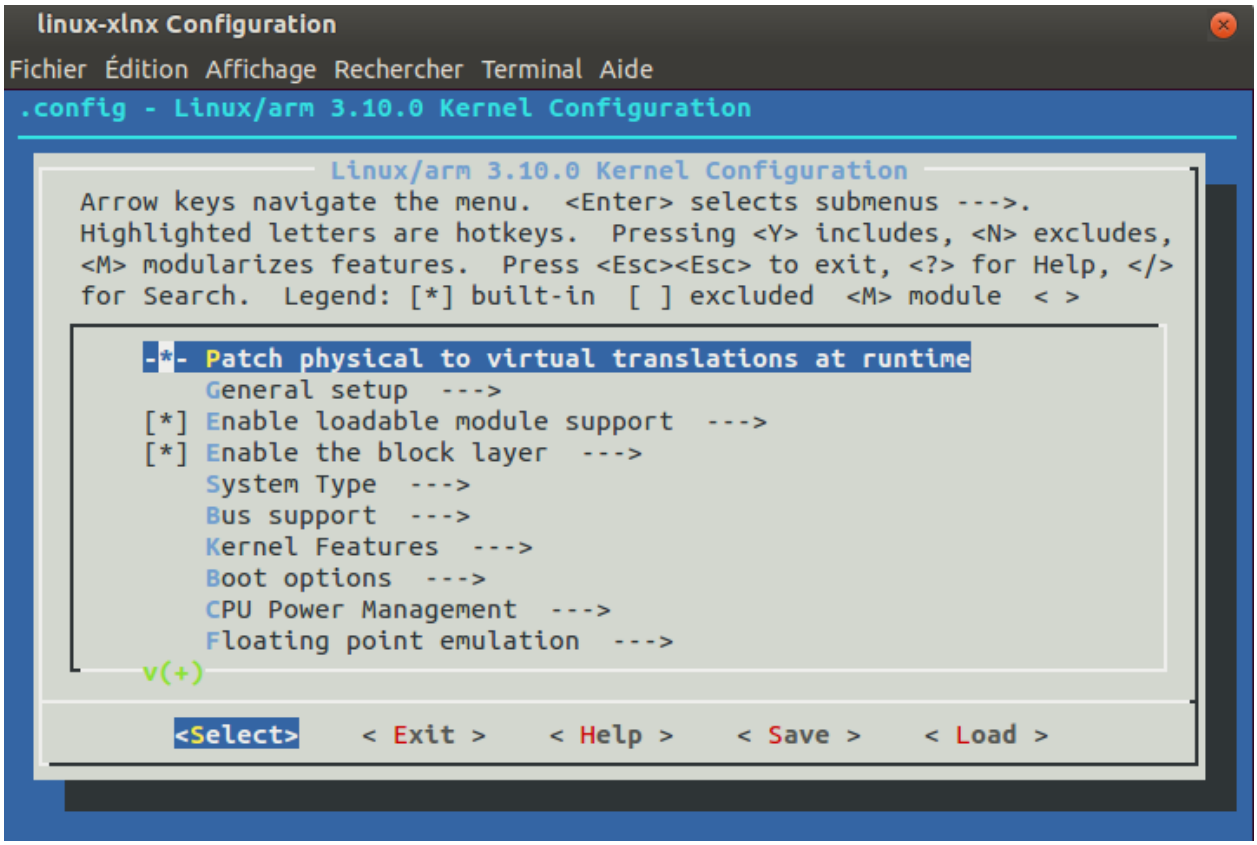
Les sources ce trouvent dans le fichier :

poky/build/tmp/work/zedboard_zynq7-poky-linux-gnueabi/linux-xlnx/3.14-xilinx+gitAUTOINC+2b48a8aeea-r0/linux-zedboard_zynq7-standard-build/source

4.3. Configurer le noyau (AP)

```
$ bitbake linux-xlnx -c menuconfig
```

Cette fenêtre apparaît :



Vous pouvez modifier le la configuration du noyau comme vous le voulez, après cela, il ne vous reste plus qu'à sauvegarder la configuration et compiler le noyau:

```
$ bitbake linux-xlnx -c compile -f
```

Puis le déployer :

```
$ bitbake linux-xlnx -c deploy
```

Un nouveau fichier "uImage" est alors créé dans le répertoire :

Poky/build/tmp/deploy/images/<nom de la machine>

4.4. Ajouter les options de debug au noyau (AP)

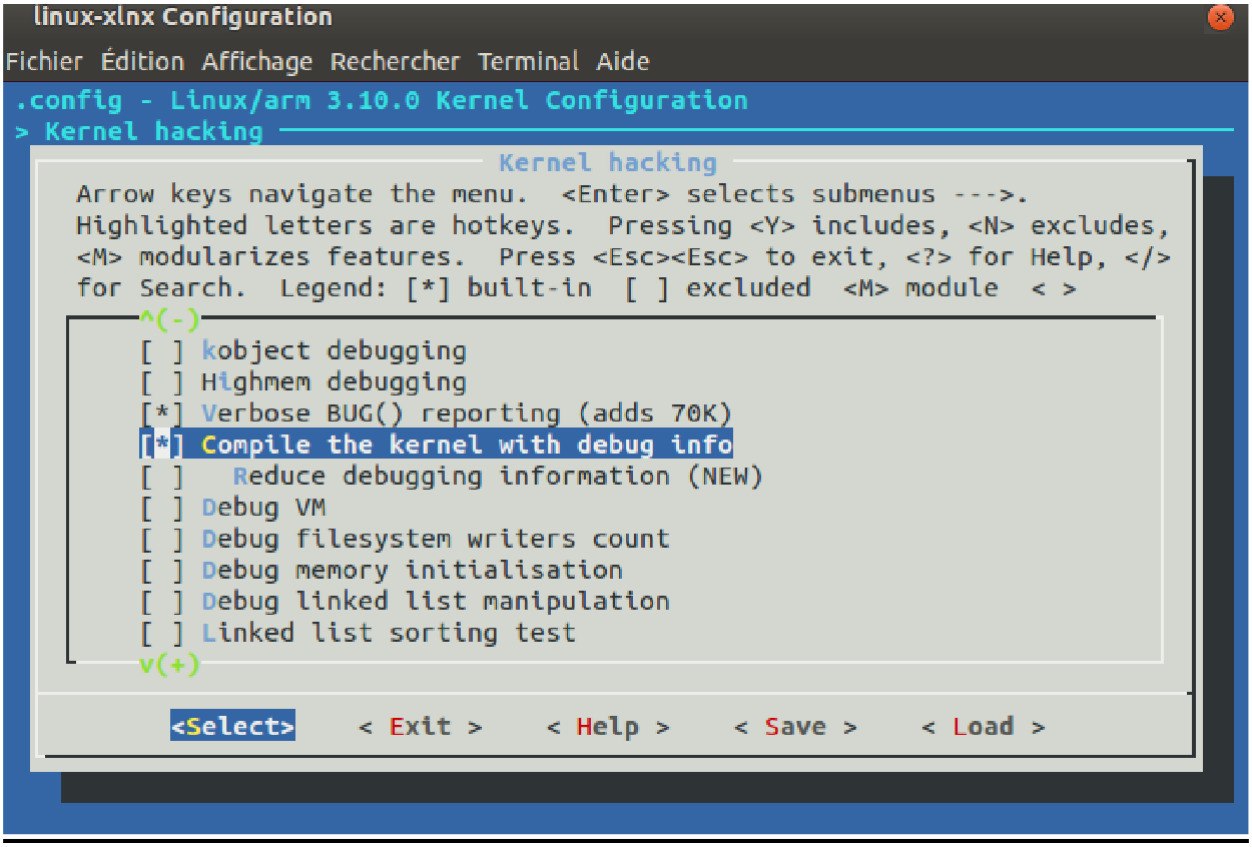
```
$ bitbake linux-xlnx -c menuconfig
```

Dans:

Kernel hacking --->

Activer:

[*] Compile the kernel with debug info



Sauvegarder, compiler et déployer :

```
$ bitbake linux-xlnx -c compile -f
```

```
$ bitbake linux-xlnx -c deploy
```

Un nouveau fichier "uImage" est alors créé dans le répertoire :

Poky/build/tmp/deploye/images/<nom de la machine>

4.5. Ajouter les modules gdb facilitant le debug du noyau (AP)

Aller dans le dossier contenant les sources du noyau :

build/tmp/work/zedboard_zynq7-poky-linux-gnueabi/linux-xlnx/3.10-xilinx+gitAUTOINC+efc2750571-r1/linux-zedboard_zynq7-standard-build/source

Extraire l'archive "patch.tar.gz" joint avec le

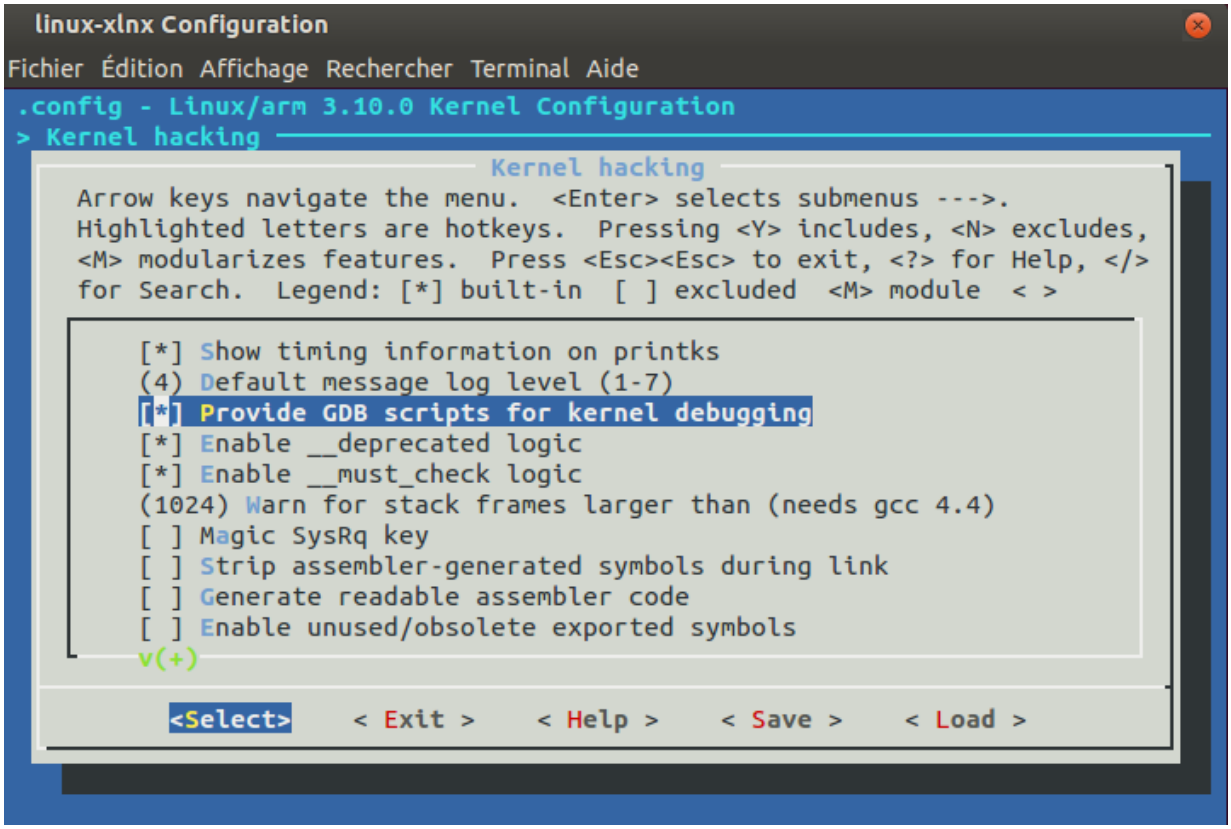
tuto Puis configurer le noyau :

```
$ bitbake linux-xlnx -c menuconfig
```

Kernel hacking --->

Activer:

[*] Provide GDB scripts for kernel debugging



Sauvegarder, compiler et déployer :

```
$ bitbake linux-xlnx -c compile -f
```

```
$ bitbake linux-xlnx -c deploy
```

Un nouveau fichier "uImage" est alors créé dans le répertoire :

Poky/build/tmp/deploye/images/<nom de la machine>

4.6. Ajouter les options du noyau pour pouvoir monter la carte SD (AP)

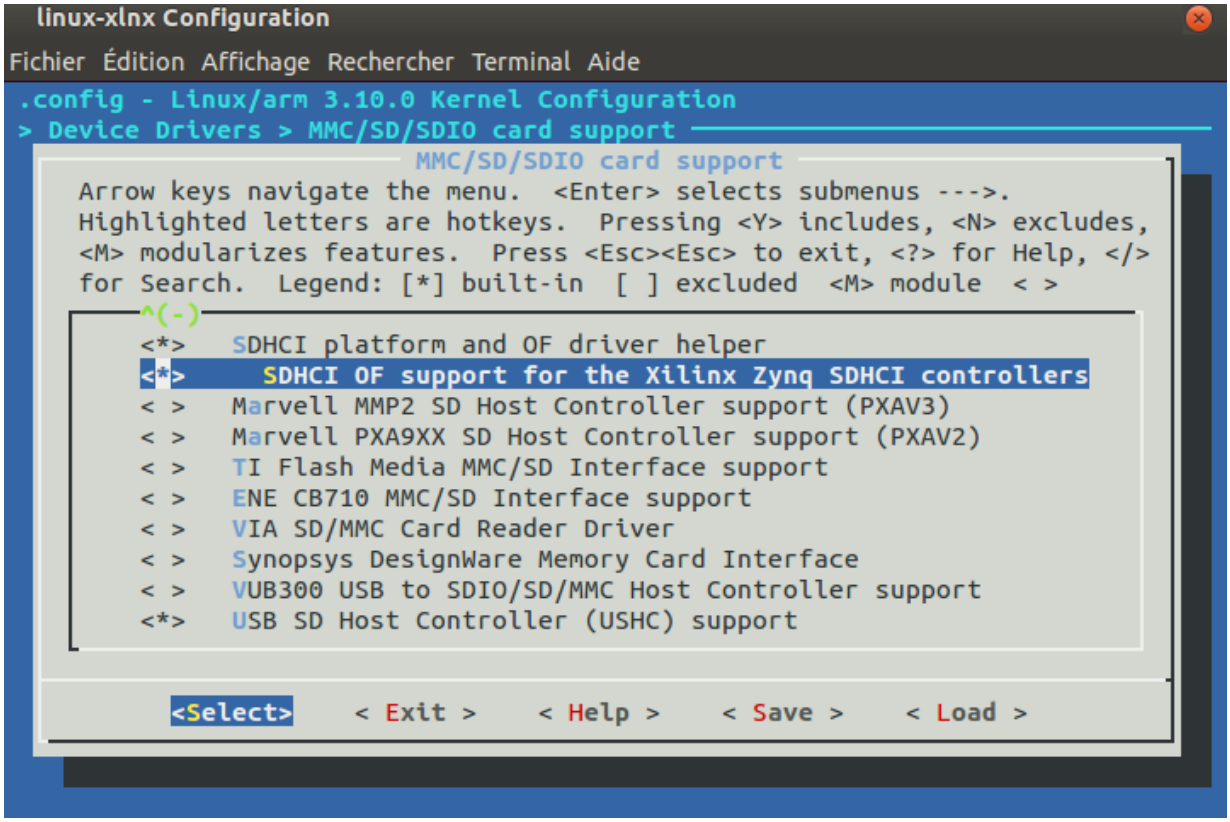
```
$ bitbake linux-xlnx -c menuconfig
```

Dans:

→ Device Drivers
→ MMC/SD/SDIO card support

Activer:

<*> SDHCI support on PCI bus
<*> SDHCI platform and OF driver helper
<*> SDHCI OF support for the Xilinx Zynq SDHCI controllers
<*> USB SD Host Controller (USHC) support



Sauvegarder, compiler et déployer :

```
$ bitbake linux-xlnx -c compile -f
```

```
$ bitbake linux-xlnx -c deploy
```

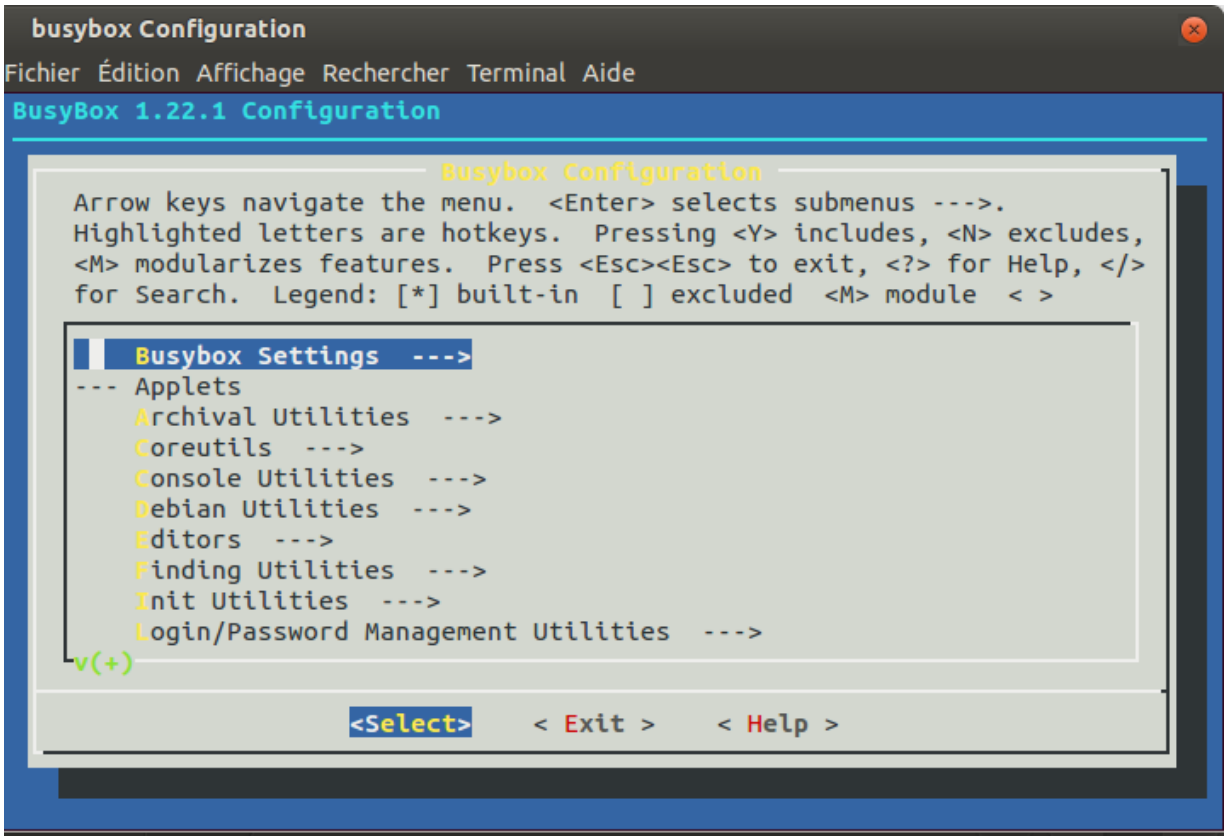
Un nouveau fichier "uImage" est alors créé dans le répertoire :

Poky/build/tmp/deploye/images/<nom de la machine>

4.7. Configurer Busybox (AV-AP)

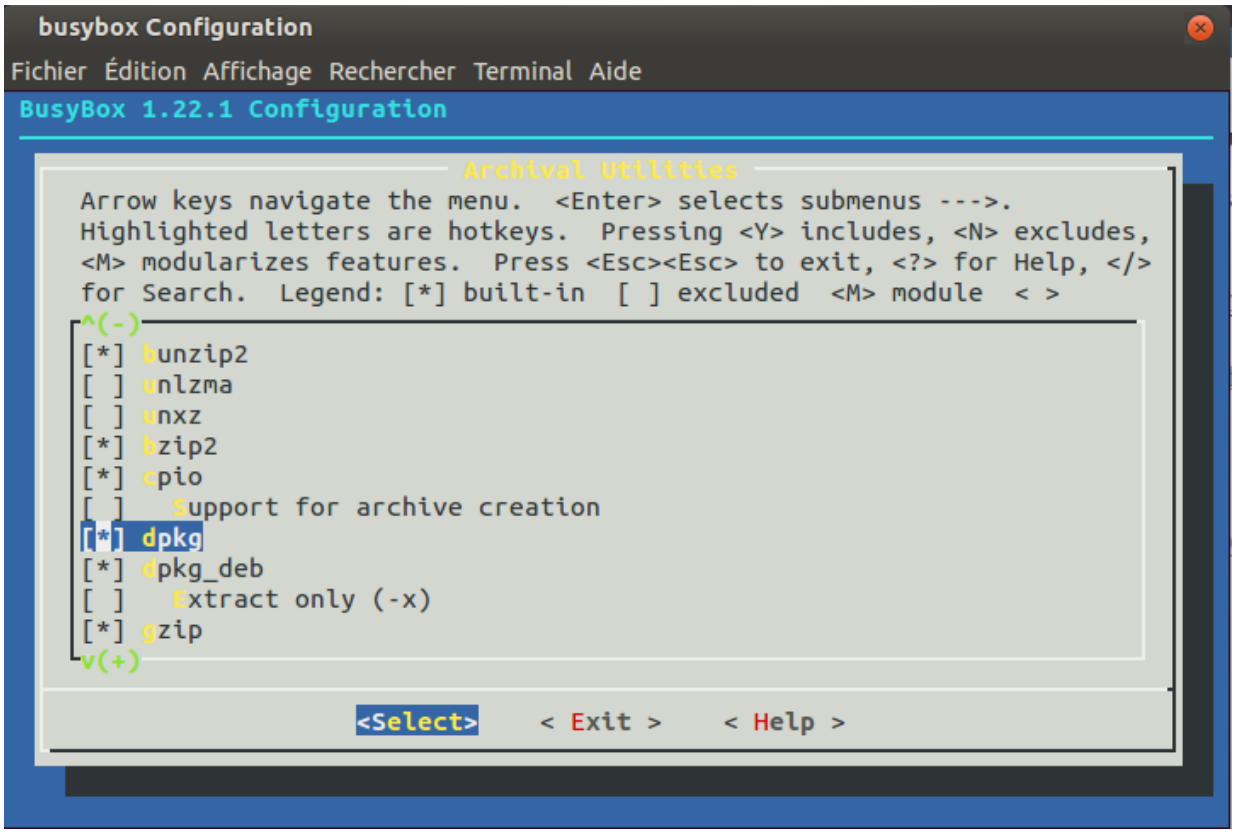
```
$ bitbake busybox -c menuconfig
```

La fenêtre de configuration apparaît :



Info utile :

Vous pouvez activer la commande « dpkg » dans le menu « Archival Utilities » :



Il reste à compiler busybox :

```
$ bitbake busybox -c compile -f
```

Puis à créer son image :

```
$ bitbake core-image-minimal
```

Un nouveau fichier " core-image-minimal-zedboard-zynq7.rootfs.cpio.gz.u-boot " est alors créé dans le répertoire :

Poky/build/tmp/deploye/images/<nom de la machine>

5. Compilation croisée d'un programme pour la ZedBoard

```
$ arm-linux-gnueabi-gcc -fno-stack-protector -O0 -pthread <nom-du-programme.c> -o <nom-de-l'exécutable> -march=armv7-a -mfpu=neon -mcpu=cortex-a9
```

6. Exécuter l'image avec qemu

```
$ qemu-system-arm -M xilinx-zynq-a9 -m 1024 -serial null -serial mon:stdio -dtb <zedboard.dtb> -smp 1 -nographic -kernel <uImage> -initrd <core-image-minimal-zedboard-zynq7.rootfs.cpio.gz.u-boot>
```

7. Exécuter l'image avec qemu en debug

Exécuter tout d'abord qemu :

```
$ qemu-system-arm -M xilinx-zynq-a9 -m 1024 -serial null -serial mon:stdio -dtb <zedboard.dtb> -smp 1 -nographic -kernel <uImage> -initrd <core-image-minimal-zedboard-zynq7.rootfs.cpio.gz.u-boot> -net nic,model=cadence_gem -net user -tftp ~/ -redir tcp:10023::23 -redir tcp:10080::80 -redir tcp:10022::22 -redir tcp:10021::21 -s -S
```

Puis GDB:

```
$  arm-none-eabi-gdb <vmlinux>
```

```
$  arm-none-eabi-gdb <dizzy/poky/build/tmp/work/zedboard_zynq7-poky-linux-gnueabi/linux-xlnx/3.10-xilinx+gitAUTOINC+efc2750571-r1/linux-zedboard_zynq7-standard-build/vmlinux>
```

8. Debug de la ZedBoard

(Il faut tout d'abord avoir ajouté les options de debug et le module gdb au noyau)

Lien utile : https://www.kernel.org/doc/Documentation/gdb-kernel-debugging.txt

8.1. Lancement de GDB

1er terminal:

```
$ xmd

XMD% connect arm hw -debugdevice
```

2éme terminal:

```
$ arm-none-eabi-gdb <vmlinux>

$ arm-none-eabi-gdb '/usr/src/hardblare/test/dizzy/poky/build/tmp/work/zedboard_zynq7- poky-linux-gnueabi/linux-xlnx/3.10-xilinx+gitAUTOINC+efc2750571- r1/linux-zedboard_zynq7-standard-build/vmlinux'

(gdb) target remote : 1234
```

8.2. Connaître le PID

```
96 static inline struct thread_info *current_thread_info(void)
 97 {
 98         register unsigned long sp asm ("sp");
 99         return (struct thread_info *)(sp & ~(THREAD_SIZE - 1));
100 }
```

cf : arch/arm/include/asm/thread_info.h ligne 96

```
THREAD_SIZE = 8192
```

cf : arch/arm/include/asm/thread_info.h ligne 19

Calcul:

```
(gdb) p ((((struct thread_info *)((((unsigned int)$sp)&~(8192-1))))->task)->pid)
```

8.3. Éteindre un cœur

```
$ cd /sys/devices/system/cpu/cpu<numéro>
```

echo 0 > online (éteindre)
echo 1 > online (allumer)

8.4. Désactiver l'ASLR

echo 0 >/proc/sys/kernel/randomize_va_space

8.5. Installer GDB dans l'image de la zedboard

Dans le fichier de configuration local.conf à « EXTRA_IMAGE_FEATURES = » ajouter 'tools-debug'

Exemple pour gdbserveur :

Sur la board :

```
$ gdbserveur localhost:2000 <programme>
```

Sur le pc :

```
$  arm-none-eabi-gdb  <programme>
```

```
(gdb) target remote 192.168.0.52:2000
```

9. Ajout d'un package

Pour recherche le nom du package :

```
$ bitbake -e <nom du package> | grep ^PACKAGES=
```

Exemple pour python :

```
$ bitbake -e python | grep ^PACKAGES=
```

Puis ajouter le nom du (ou des) package(s) dans le local.conf avec la ligne :

IMAGE_INSTALL += "<nom package 1> <nom package 2> "

La ligne :

EXTRA_IMAGE_FEATURES = "<nom> "

Permet également d'ajouter certains packages (pour connaitre "nom", cf. local.conf)