Mise en place d’un

environnement de développement linux pour ZedBoard

*Auteur : Corentin Nouviale*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *Date : 29/07/2015* |
|  |  |  |  |  | *Version : 1.1* |

Sommaire

1. [Préparation de l’environnement 3](#_bookmark0)
   1. [Téléchargement des sources à partir des dépôts git 3](#_bookmark1)
   2. [Modification des fichiers de configuration (.conf) 4](#_bookmark2)
   3. [Compilation de poky 5](#_bookmark3)
   4. [Installation du SDK Xilinx 5](#_bookmark4)
2. [Création du chargeur de démarrage (bootloader) 6](#_bookmark5)
   1. [Téléchargement des sources: 6](#_bookmark6)
   2. [Utilisation de Vivado pour créer une description du matériel 6](#_bookmark7)
   3. [Construction du First Stage Boot Loader (FSBL) 6](#_bookmark8)
   4. [U-boot 7](#_bookmark9)
3. [Mise en place de l’environnement 8](#_bookmark10)
   1. [Création du boot.bin 8](#_bookmark11)
   2. [Préparation de la carte SD 9](#_bookmark12)
   3. [Préparation de la carte ZedBoard 9](#_bookmark13)
   4. [Préparation de minicom 10](#_bookmark14)
   5. [Lancement de la carte 10](#_bookmark15)
4. [Configuration du noyau 11](#_bookmark16)
   1. [Choisir la version du noyau (AV) 11](#_bookmark17)
   2. [Sélectionner les sources du noyau (AP) 11](#_bookmark18)
   3. [Configurer le noyau (AP) 12](#_bookmark19)
   4. [Ajouter les options de debug au noyau (AP) 13](#_bookmark20)
   5. [Ajouter les modules gdb facilitant le debug du noyau (AP) 14](#_bookmark21)
   6. [Ajouter les options du noyau pour pouvoir monter la carte SD (AP) 15](#_bookmark22)
   7. [Configurer Busybox (AV-AP) 16](#_bookmark23)
5. [Compilation croisée d’un programme pour la ZedBoard 18](#_bookmark24)
6. [Exécuter l'image avec qemu 18](#_bookmark25)
7. [Exécuter l'image avec qemu en debug 18](#_bookmark26)
8. [Debug de la ZedBoard 19](#_bookmark27)
   1. [Lancement de GDB 19](#_bookmark28)
   2. [Connaître le PID 19](#_bookmark29)
   3. [Éteindre un cœur 19](#_bookmark30)
   4. [Désactiver l’ASLR 20](#_bookmark31)
   5. [Installer GDB dans l’image de la zedboard 20](#_bookmark32)
9. [Ajout d’un package 20](#_bookmark33)

# 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](http://www.wiki.xilinx.com/Prepare%2BBoot%2BImage) Tuto yocto : <http://wiki.elphel.com/index.php?title=Yocto_tests>

## 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

## 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"

## 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’)

## 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)

# Création du chargeur de démarrage (bootloader)

## Téléchargement des sources:

Source: [http://www.wiki.xilinx.com/Fetch+Sources](http://www.wiki.xilinx.com/Fetch%2BSources)

$ 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)

* 1. 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>

## Construction du First Stage Boot Loader (FSBL)

Source: [www.wiki.xilinx.com/Build+FSBL](http://www.wiki.xilinx.com/Build%2BFSBL)

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)

## U-boot

Sources: [http://www.wiki.xilinx.com/Build+U-Boot](http://www.wiki.xilinx.com/Build%2BU-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'

# 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>

## Création du boot.bin

[http://www.wiki.xilinx.com/Prepare+Boot+Image](http://www.wiki.xilinx.com/Prepare%2BBoot%2BImage)

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

## Préparation de la carte SD

Sur la carte SD mettre les fichiers :

-boot.bin

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

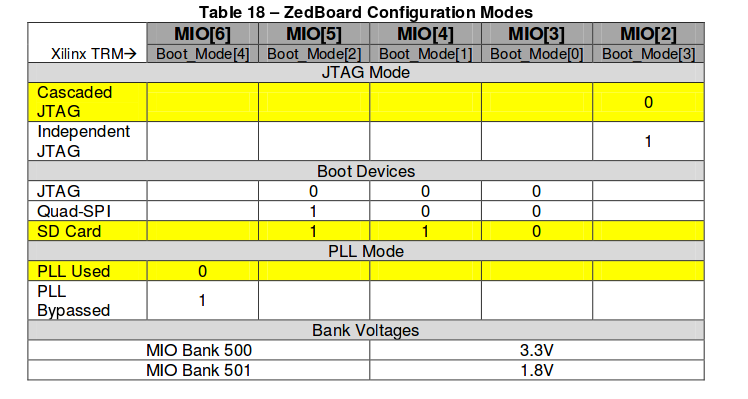
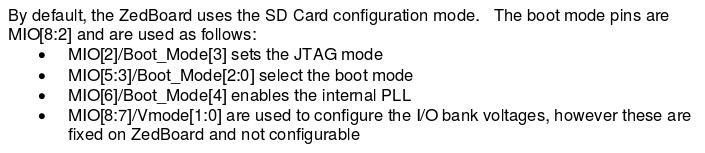
-uImage.bin

-zedboard-zynq7.dtb

-uEnv.txt

## Préparation de la carte ZedBoard

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



(Image provenant du [document](http://zedboard.org/sites/default/files/ZedBoard_HW_UG_v1_1.pdf) de zedboard.org page 29)

MIO 6: set to GND MIO 3: set to GND

MIO 5: set to 3V3 MIO 2: set to GND

MIO 4: set to 3V3 jp6

## 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 |

+--------------------------+

## 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

# Configuration du noyau

AV : avant bitbake core-image-minimal AP : après bitbake core-image-minimal

## 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.

## Sélectionner les sources du noyau (AP)

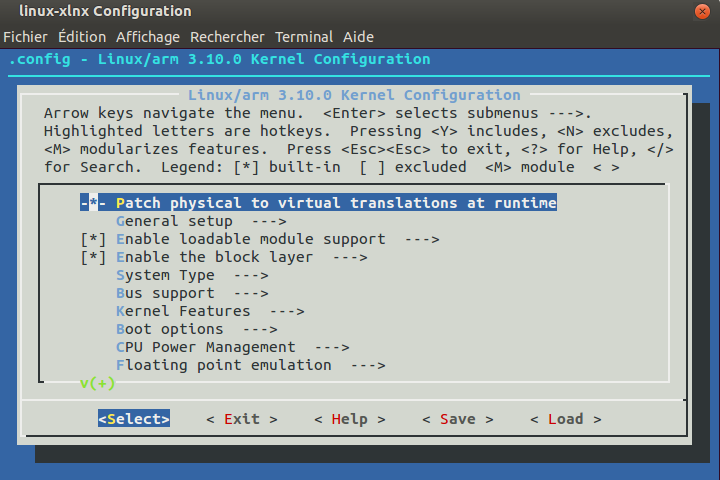
Les sources ce trouvent dans le fichier :

poky/build/tmp/work/zedboard\_zynq7-poky-linux-gnueabi/linux-xlnx/3.14- xilinx+gitAUTOINC+2b48a8aeea-r0/linux-zedboard\_zynq7-standard-build/source

## 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>

## Ajouter les options de debug au noyau (AP)

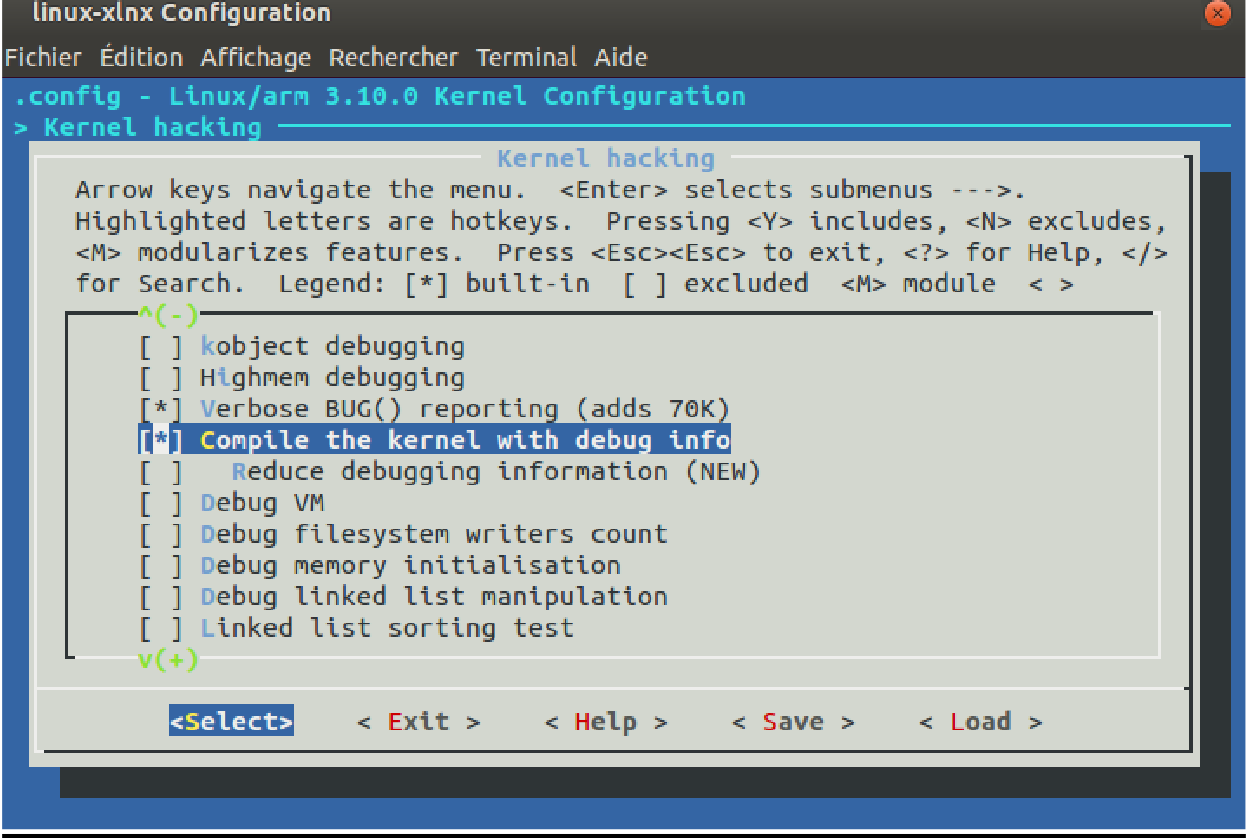
$ bitbake linux-xlnx -c menuconfig

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>

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

Aller dans le dossier contenant les sources du noyau :

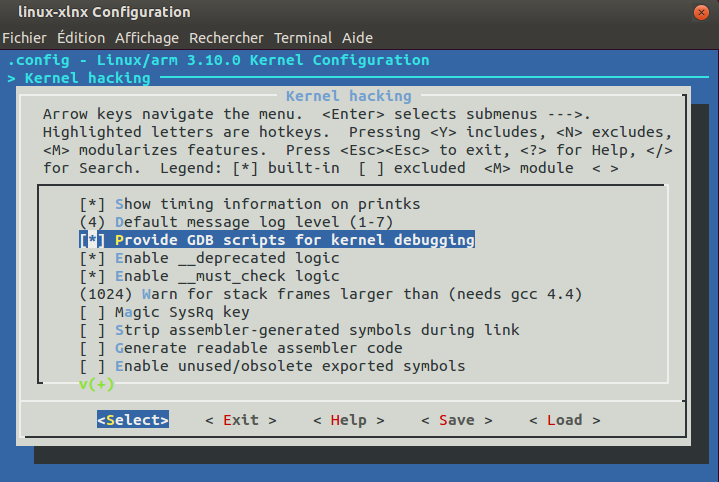
build/tmp/work/zedboard\_zynq7-poky-linux-gnueabi/linux-xlnx/3.10- xilinx+gitAUTOINC+efc2750571-r1/linux-zedboard\_zynq7-standard-build/source

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



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

$ bitbake linux-xlnx -c menuconfig

Dans:

→ Device Drivers

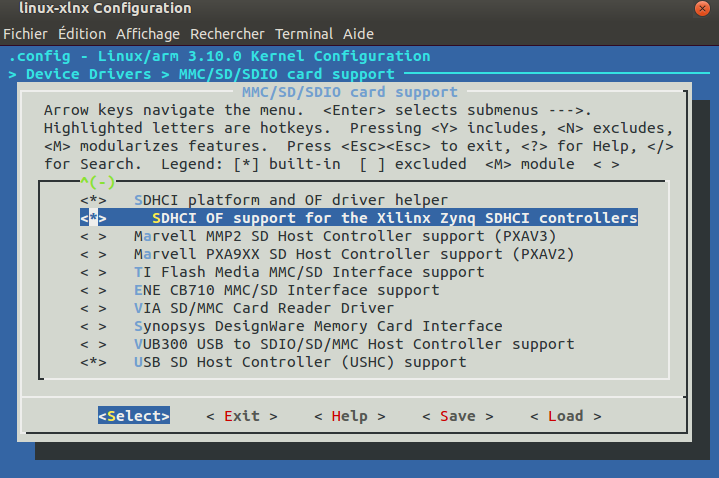
→ MMC/SD/SDIO card support Activer:

<\*> 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



Dans :

File systems --->

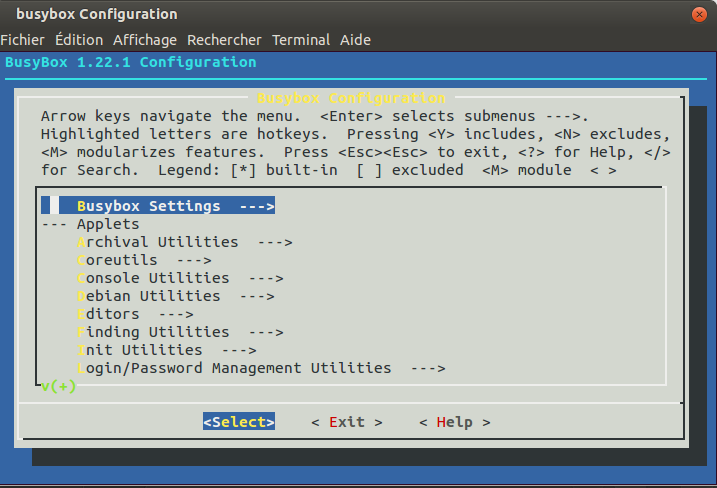
-\*- Native language support ---> Activer :

<\*> NLS UTF-8

## Configurer Busybox (AV-AP)

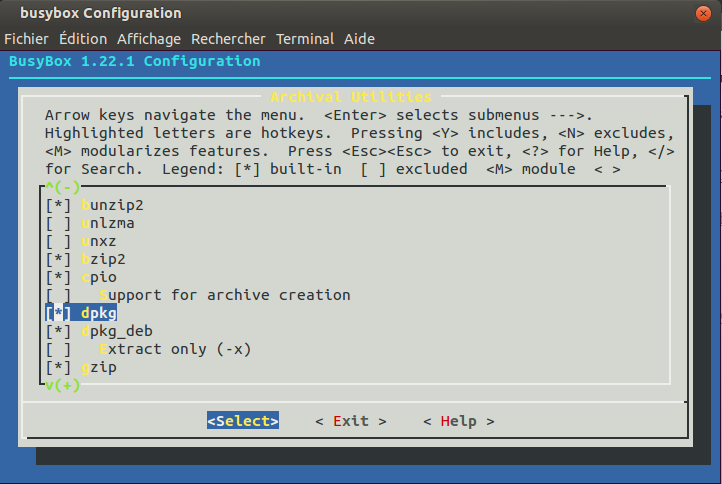
$ bitbake busybox -c menuconfig

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>

# 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

# 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>

# 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>

# 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>

### 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

### Connaître le PID

[96](http://lxr.free-electrons.com/source/arch/arm/include/asm/thread_info.h?v=3.10&amp;L96) static inline struct [thread\_info](http://lxr.free-electrons.com/ident?v=3.10%3Bi%3Dthread_info) \*[current\_thread\_info](http://lxr.free-electrons.com/ident?v=3.10%3Bi%3Dcurrent_thread_info)(void) [97](http://lxr.free-electrons.com/source/arch/arm/include/asm/thread_info.h?v=3.10&amp;L97) {

1. register unsigned long [sp](http://lxr.free-electrons.com/ident?v=3.10%3Bi%3Dsp) asm (*"sp"*);
2. return (struct [thread\_info](http://lxr.free-electrons.com/ident?v=3.10%3Bi%3Dthread_info) \*)([sp](http://lxr.free-electrons.com/ident?v=3.10%3Bi%3Dsp) & ~([THREAD\_SIZE](http://lxr.free-electrons.com/ident?v=3.10%3Bi%3DTHREAD_SIZE) - 1));

[100](http://lxr.free-electrons.com/source/arch/arm/include/asm/thread_info.h?v=3.10&amp;L100) }

THREAD\_SIZE = 8192

cf : arch/arm/include/asm/thread\_info.h ligne 96

cf : arch/arm/include/asm/thread\_info.h ligne 19 Calcul:

(gdb) p ((((struct thread\_info \*)((((unsigned int)$sp)&~(8192-

1))))->task)->pid)

### Éteindre un cœur

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

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

### Désactiver l’ASLR

echo 0 >/proc/sys/kernel/randomize\_va\_space

### 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

# 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)