



Développez vos systèmes embarqués sur SoC FPGA

Comment embarquer Linux et développer en VHDL vos applicatifs dédiés au traitement d'image sans pénaliser le CPU.

Jean-Marie CODOL
Développeur

Submarine Open Technologies
Montpellier

Avril 2022



Développez vos systèmes embarqués sur SoC FPGA

Comment embarquer Linux et développer en VHDL vos applicatifs dédiés au traitement d'image sans pénaliser le CPU.

Partie 3

Plan

Plan

JOUR 1

§001 Faire communiquer Linux avec un FPGA sur le SoC intel DE10-nano

- **distribution fournie par Altera**
- créer une image SD (en partie sur un serveur distant)
- compilation croisée (eclipse CDT embedded)
- activer les bridges HPS <-> FPGA par un device tree
- utiliser le bridge HPS2FPGA

J 2

§002 Partager une zone de RAM entre Linux et un FPGA sur le SoC intel DE10-nano

- projet de lecture/écriture en RAM sur FPGA
- projet de lecture/écriture en RAM sur CPU

J 3

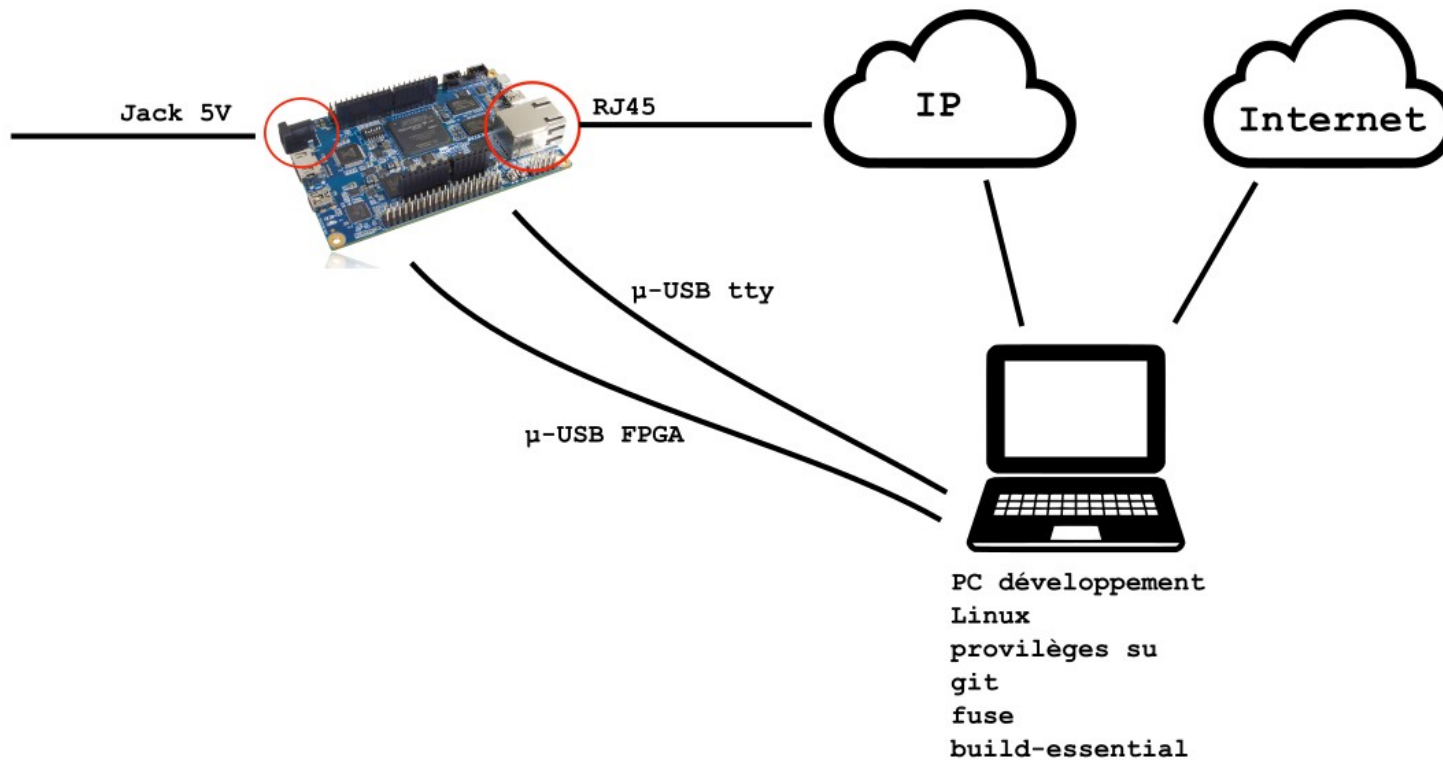
§003 Cas pratique avec seuillage d'image

- projet openCV
- échange sur mémoire RAM

§001 Faire communiquer Linux avec un FPGA

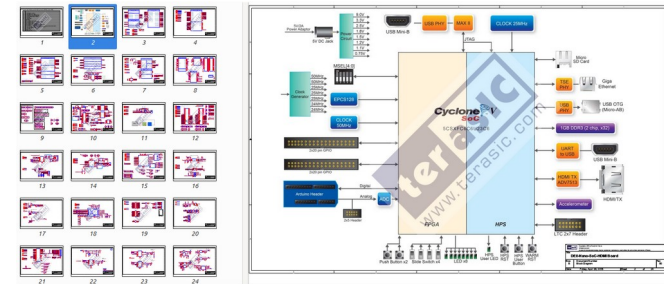
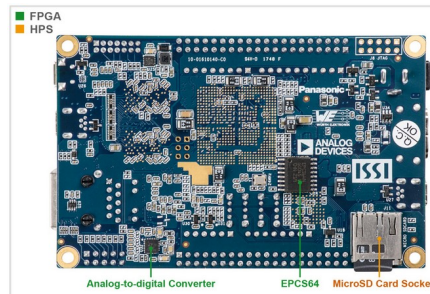
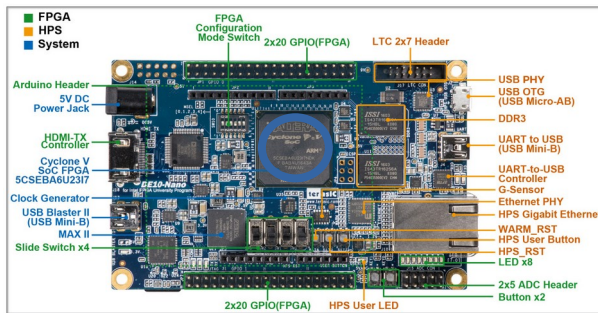
Récupérer l'image de la carte SD fournie par Terasic

Setup matériel



§001 Faire communiquer Linux avec un FPGA

Récupérer l'image de la carte SD fournie par Terasic



terasic.com.tw

- > Carte d'évaluation
- > Vues
- > Schémas
- > Images Linux
- > codes sources exemples

```
<captronic_formation_fpga_img_proc>/  
datasheets/de10-nano/  
DE10-Nano_User_manual.pdf  
DE10-Nano_Schematic.zip
```

Mettre en œuvre la distribution Linux fournie

Récupérer l'image de la carte SD fournie par Terasic

Les scripts sont dans le dossier git :

```
<captronic_formation_fpga_img_proc> /scripts/
```

Mettre en œuvre la distribution Linux fournie

Récupérer l'image de la carte SD fournie par Terasic

The first screenshot shows the Terasic website with the following elements:

- 1. Terasic logo
- 2. SoC Platform >
- 3. DE10-Nano Kit
- 4. DE10-Nano Angstrom Image
- 5. Sidebar menu with 'Resources' selected

The second screenshot shows the download table for the Linux BSP package:

Title	Version	Size	Date	Download
Linux Console (kernel 4.5)	1.3		2018-03-15	Download
Linux Xfce Desktop (kernel 4.1.33-ltsi-altera)	1.0		2017-04-11	Download
Linux LXDE Desktop (kernel 4.5)	1.1		2017-04-10	Download

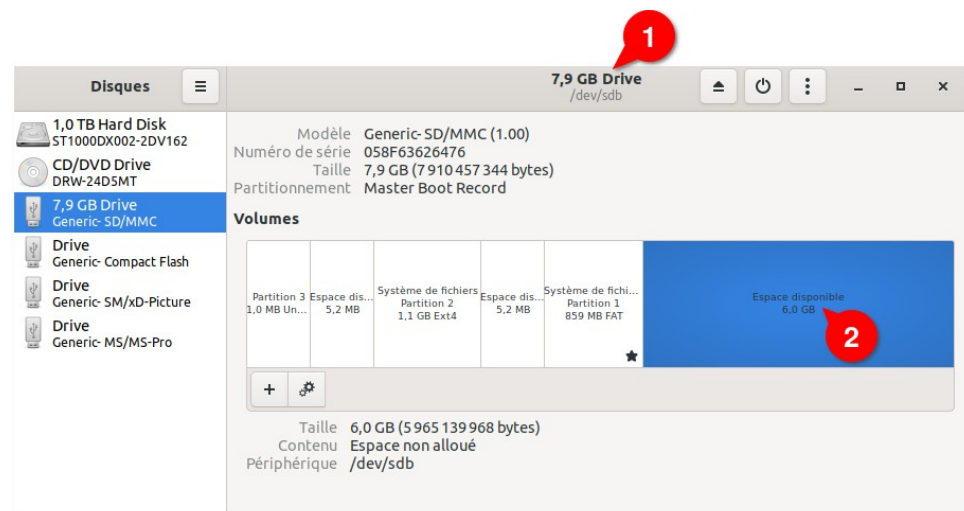
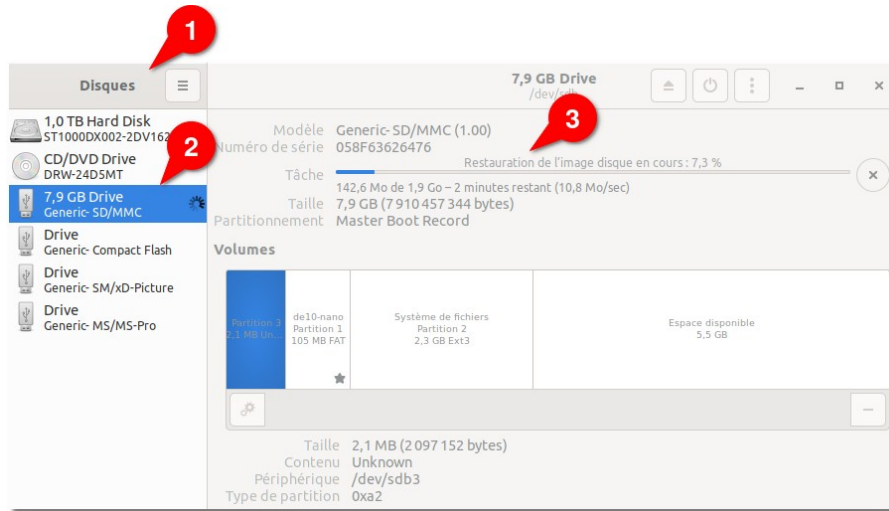
```
## ssh user@151.80.152.63
export DEWD=$HOME/de10nano-wd
mkdir -p $DEWD/official_sd
cd $DEWD/official_sd
wget http://download.terasic.com/downloads/cd-rom/de10-nano/linux_BSP/de10_nano_linux_console.zip
unzip de10_nano_linux_console.zip
ll -h
-rw-rw-r-- 1 ubuntu ubuntu 1.9G Sep 15 2017 de10_nano_linux_console.img
```

Graver l'image (2Go) de10_nano_linux_console.img sur une carte SD de 4Go minimum

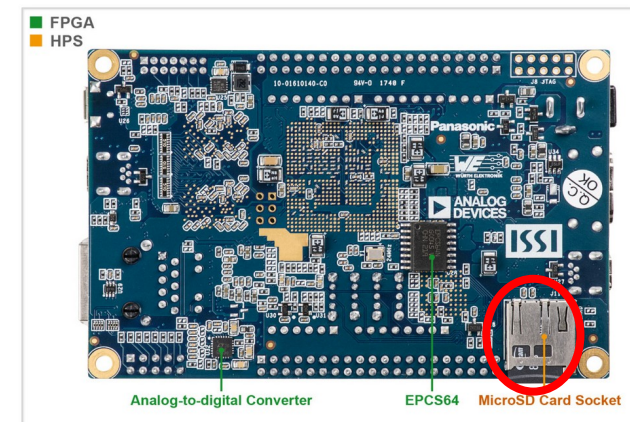
Mettre en œuvre la distribution Linux fournie

Récupérer l'image de la carte SD fournie par Terasic

Graver l'image `de10_nano_linux_console.img` sur une carte SD de 4Go minimum.



Retirer la carte SD et l'insérer dans le slot du DE10-nano.



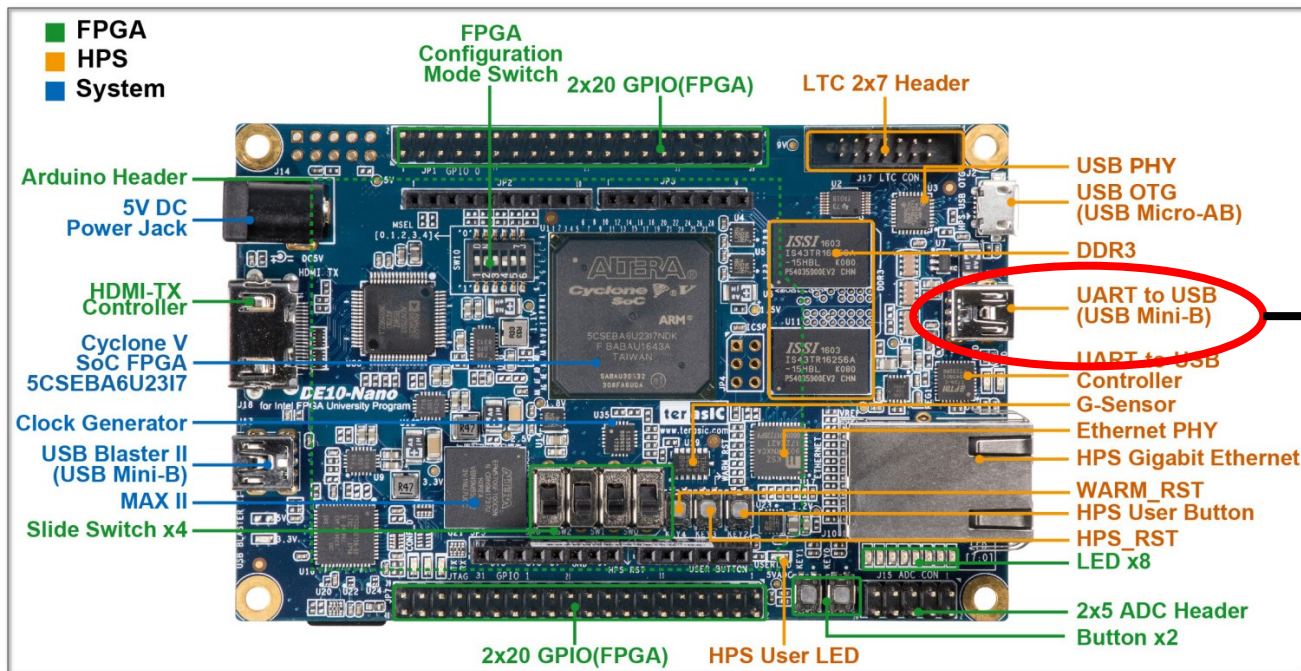
Mettre en œuvre la distribution Linux fournie

Démarrer la carte de prototypage et prendre le contrôle du SoC en UART

Connecter le PC de développement en USB au port UART de développement :

- Le chip FTDI est alimenté par le câble
- Le SoC FPGA n'est pas alimenté par le câble

On peut donc connecter l'USB alors que le DE10-nano n'est pas sous tension.



USB

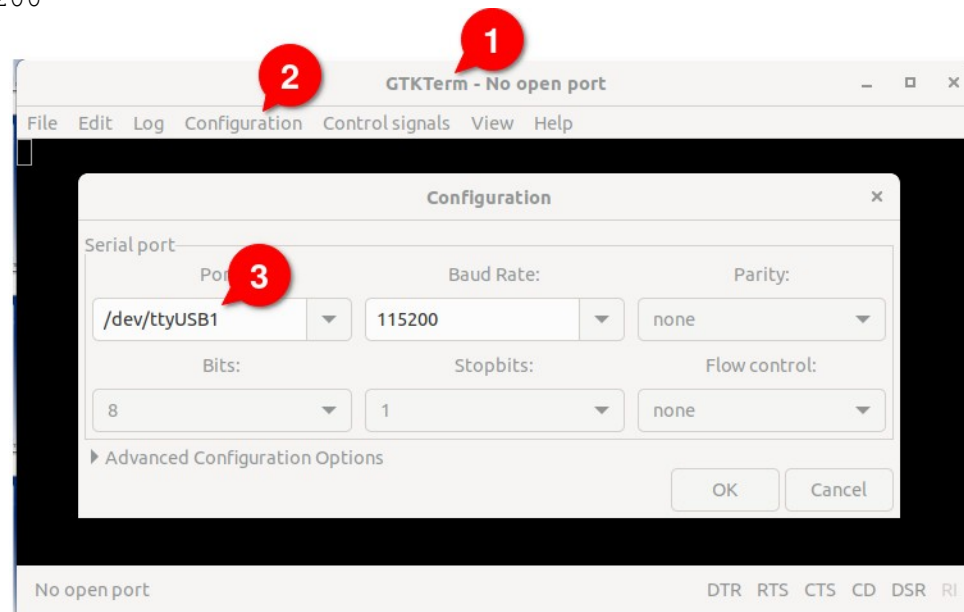
Mettre en œuvre la distribution Linux fournie

Démarrer la carte de prototypage et prendre le contrôle du SoC en UART

```
sudo dmesg
```

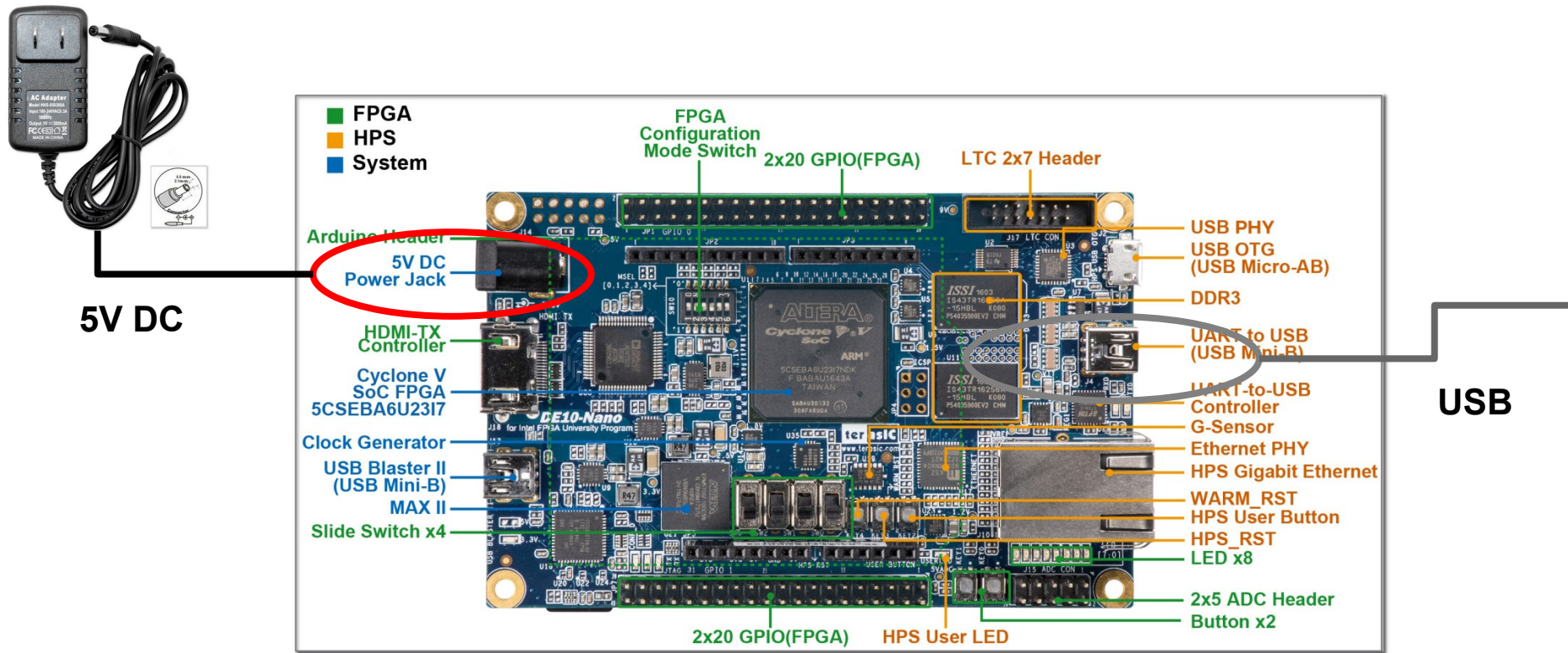
```
[12790.975212] usb 1-10.3.4: new full-speed USB device number 8 using xhci_hcd
[12791.201590] usb 1-10.3.4: New USB device found, idVendor=0403, idProduct=6001, bcdDevice= 6.00
[12791.201605] usb 1-10.3.4: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[12791.201612] usb 1-10.3.4: Product: FT232R USB UART
[12791.201616] usb 1-10.3.4: Manufacturer: FTDI
[12791.201620] usb 1-10.3.4: SerialNumber: AU00L0TH
[12791.215078] ftdi_sio 1-10.3.4:1.0: FTDI USB Serial Device converter detected
[12791.215143] usb 1-10.3.4: Detected FT232RL
[12791.217716] usb 1-10.3.4: FTDI USB Serial Device converter now attached to tttyUSB1
```

```
sudo apt-get install gtkterm
gtkterm
# ou
screen /dev/ttyUSB1 115200
```



Mettre en œuvre la distribution Linux fournie

Démarrer la carte de prototypage et prendre le contrôle du SoC en UART



Mettre en œuvre la distribution Linux fournie

Démarrer la carte de prototypage et prendre le contrôle du SoC en UART

Le démarrage prends 15 secondes.
Identifiant : root
Mot de passe : <enter>

```
File Edit Log Configuration Controlsignals View Help
[ OK ] Started Getty on tty1.
[ OK ] Started Serial Getty on ttyS0.
[ OK ] Reached target Login Prompts.
[ OK ] Reached target Multi-User System.
[ 6.559706] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
Starting Update UTMP about System Runlevel Changes...
[ OK ] Started Update UTMP about System Runlevel Changes.

--0--
[Angstrom]
o o

The Angstrom Distribution socfpga ttyS0

Angstrom v2014.12 - Kernel

socfpga login: root
Last login: Mon Mar 13 03:57:25 UTC 2017 on ttyS0
root@socfpga:~#
root@socfpga:~#
```

/dev/ttyUSB1 115200-8-N-1 DTR RTS CTS CD DSR RI

Mettre en œuvre la distribution Linux fournie

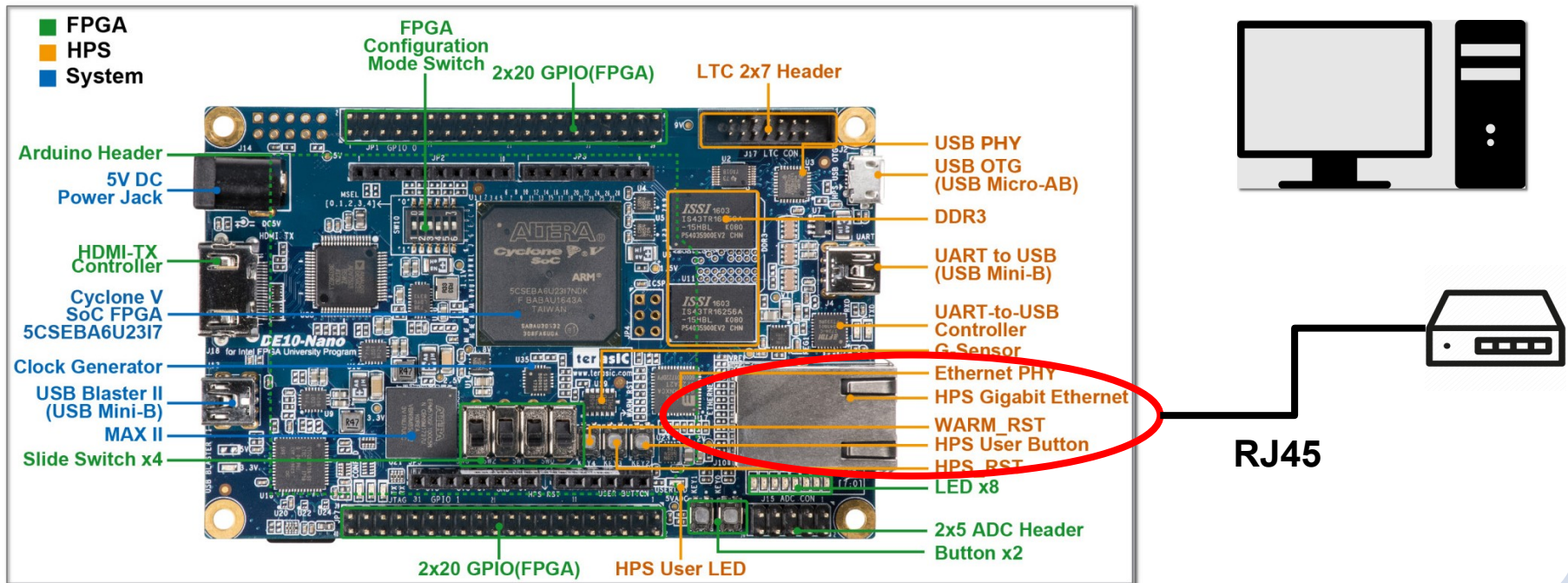
Démarrer la carte de prototypage et
prendre le contrôle du SoC en UART

```
## Quelle distribution est utilisée ?  
  
# lsb_release -a  
Distributor ID: Angstrom  
Description:    Angstrom GNU/Linux v2014.12 (Core edition)  
Release:        v2014.12  
Codename:       Core edition
```

Mettre en œuvre la distribution Linux fournie

Démarrer la carte de prototypage et prendre le contrôle du SoC en UART

Connecter un câble RJ45 pour se connecter en ssh.



Mettre en œuvre la distribution Linux fournie

Démarrer la carte de prototypage et prendre le contrôle du SoC en UART

Connecter un câble RJ45 pour se connecter en ssh.

Sur le SoC :

```
$ ifconfig eth0 down
```

```
$ ifconfig eth0 up
```

```
$ ifconfig
```

```
eth0      Link encap:Ethernet  HWaddr 76:fe:5e:f9:40:53  
          inet addr:192.168.1.15  Bcast:192.168.1.255  Mask:255.255.255.0  
          inet6 addr: fe80::74fe:5eff:fef9:4053/64  Scope:Link  
          UP BROADCAST RUNNING MULTICAST DYNAMIC  MTU:1500  Metric:1
```

```
[...]
```

```
ping www.google.com
```

```
PING www.google.com (216.58.205.196): 56 data bytes
```

```
64 bytes from 216.58.205.196: seq=0 ttl=115 time=27.816 ms
```

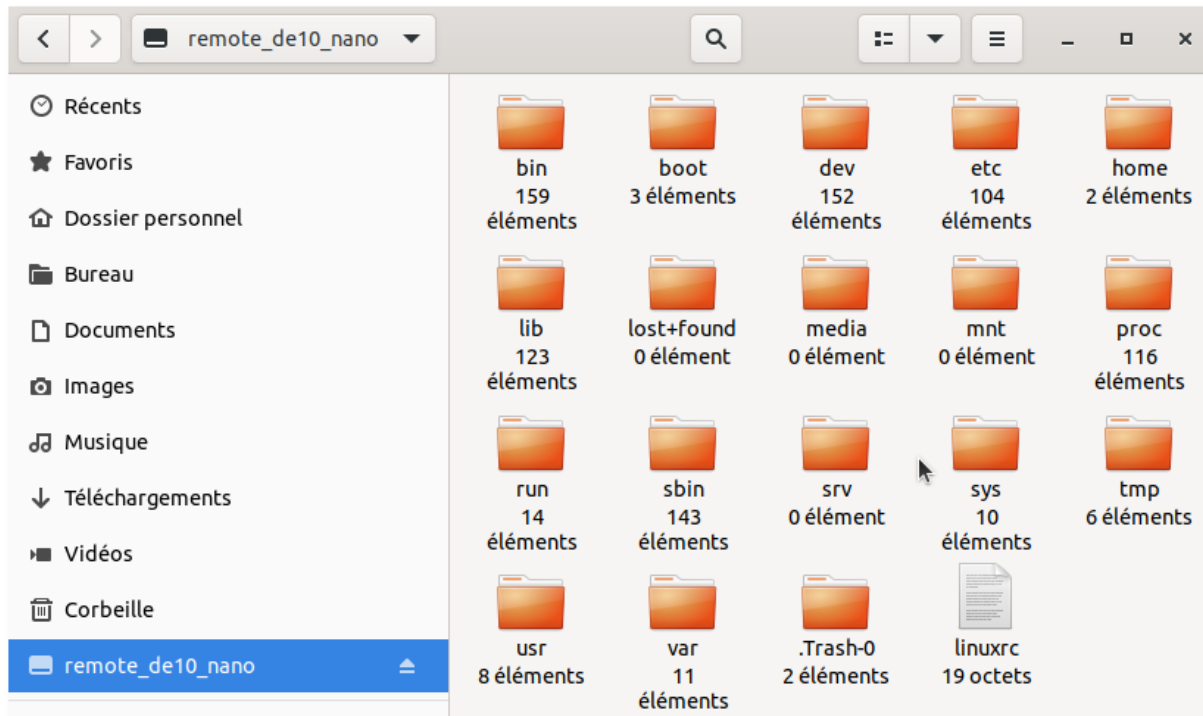
```
[...]
```

Mettre en œuvre la distribution Linux fournie

Démarrer la carte de prototypage et prendre le contrôle du SoC en UART

Monter un dossier réseau pour le DE10-nano

```
DE10NA_ADDR=192.168.1.38
DEWD=$HOME/de10nano-wd
mkdir -p $DEWD/remote_de10_nano/
ssh-keygen -f "$HOME/.ssh/known_hosts" -R $DE10NA_ADDR
sshfs -o reconnect,ServerAliveInterval=15,ServerAliveCountMax=3 \
root@$DE10NA_ADDR: $DEWD/remote_de10_nano/
```



Plan

Plan

JOUR 1

§001 Faire communiquer Linux avec un FPGA sur le SoC intel DE10-nano

- distribution fournie par Altera
- **créer une image SD (en partie sur un serveur distant)**
- compilation croisée (eclipse CDT embedded)
- activer les bridges HPS <-> FPGA par un device tree
- utiliser le bridge HPS2FPGA

J 2

§002 Partager une zone de RAM entre Linux et un FPGA sur le SoC intel DE10-nano

- projet de lecture/écriture en RAM sur FPGA
- projet de lecture/écriture en RAM sur CPU

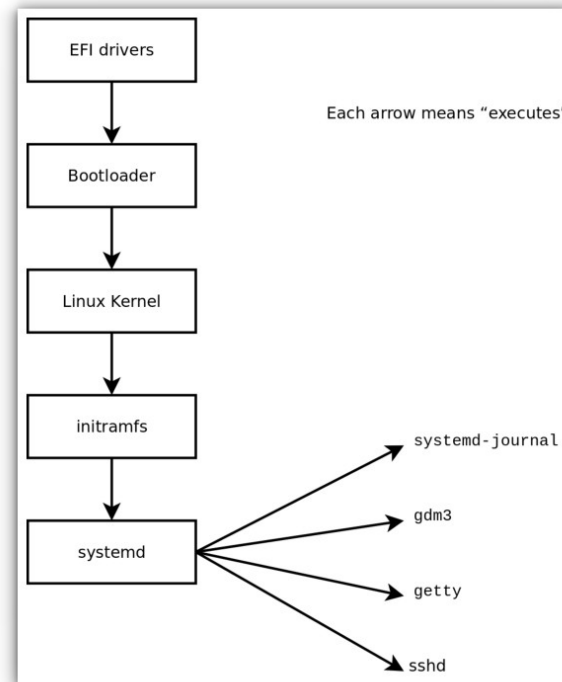
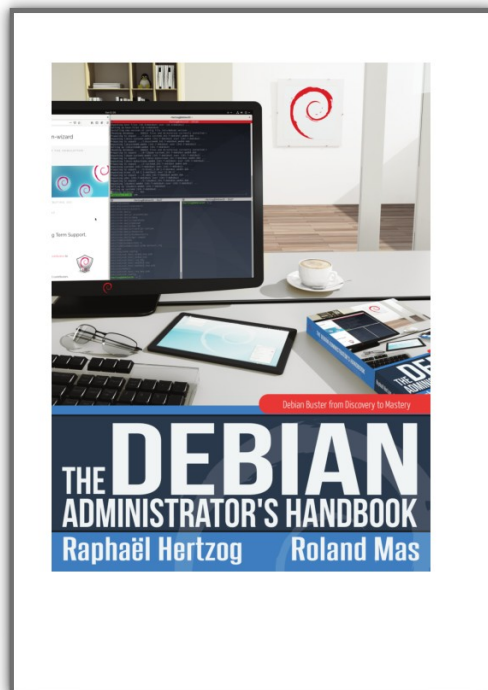
J 3

§003 Cas pratique avec seuillage d'image

- projet openCV
- échange sur mémoire RAM

[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

Le bootloader, le Noyau Linux et la distribution Linux sur un SoC FPGA Intel



[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

Le bootloader, le Noyau Linux et la distribution Linux sur un SoC FPGA Intel

Séquence de démarrage :

1. ROM
2. Pre-loader
3. BootLoader : **u-boot** (charge le bitstream dans le FPGA), charge le noyau Linux
4. Kernel : **Noyau Linux**, activer les devices-trees (pour la comm. Linux/FPGA)
5. RootFS : Distribution **debian**

Ressources :

- > rocketboards.org/foswiki/Documentation/BuildingBootloaderCycloneVAndArria10
- > github.com/zangman/de10-nano

[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

1/3 : Configurer et compiler u-boot

Un serveur de calcul est à votre disposition : 151.80.152.63

Suivre les instructions de <captronic_formation_fpga_img_proc> /scripts/2-creer_distrib.sh

- U-boot :

- . Récupérer la version de u-boot "officielle" sur github
- . patch #1 : détecter la présence d'un fichier bitstream pour le FPGA et auquel cas configurer le FPGA avec

Le patch ajoute les commandes u-boot suivantes dans la variable **distro_bootcmd** :

```
-----
"distro_bootcmd= " \
    "if test -e mmc 0:1 u-boot.scr; then " \
    "echo --- Found u-boot.scr ---; " \
    "fatload mmc 0:1 0x2000000 u-boot.scr; " \
    "source 0x2000000; " \
    "elif test -e mmc 0:1 soc_system.rbf; then " \
    "echo --- Programming FPGA ---; " \
    "fatload mmc 0:1 0x2000000 soc_system.rbf; " \
    "fpga load 0 0x2000000 0x700000; " \
    "else " \
    "echo u-boot.scr and soc_system.rbf not found in fat.; " \
    "fi; " \
    BOOTENV_SET_NVME_NEED_INIT
-----
```


[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

1/3 : Configurer et compiler u-boot

Un serveur de calcul est à votre disposition : 151.80.152.63

Suivre les instructions de <captronic_formation_fpga_img_proc> /scripts/2-2-creer_distrib.sh

- U-boot :

- . Récupérer la version de u-boot "officielle" sur github
- . patch #1 : détecter la présence d'un fichier bitstream pour le FPGA et auquel cas configurer le FPGA avec
- . patch #2 : forcer l'adresse ethernet pour ne pas qu'elle change à chaque démarrage

Ce patch ajoute une adresse ethernet fixe pour eth0 sur la variable CONFIG_EXTRA_ENV_SETTINGS :

```
-----
#define CONFIG_EXTRA_ENV_SETTINGS \
    "fdtfile=" CONFIG_DEFAULT_FDT_FILE "\0" \
    "bootm_size=0xa000000\0" \
    "kernel_addr_r="__stringify(CONFIG_SYS_LOAD_ADDR) "\0" \
    "fdt_addr_r=0x02000000\0" \
    "scriptaddr=0x02100000\0" \
    "pxefile_addr_r=0x02200000\0" \
    "ramdisk_addr_r=0x02300000\0" \
    "socfpga_legacy_reset_compat=1\0" \
    "ethaddr=56:42:1f:e6:6f:68\0" \
    BOOTENV
-----
```

[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

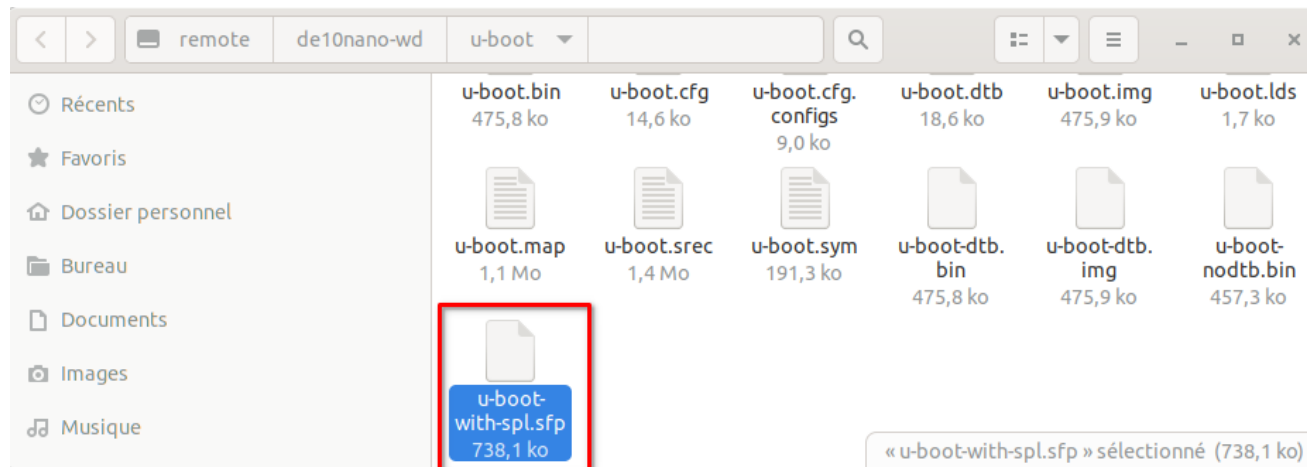
1/3 : Configurer et compiler u-boot

Un serveur de calcul est à votre disposition : 151.80.152.63

Suivre les instructions de `<captronic_formation_fpga_img_proc> /scripts/2-2-creer_distrib.sh`

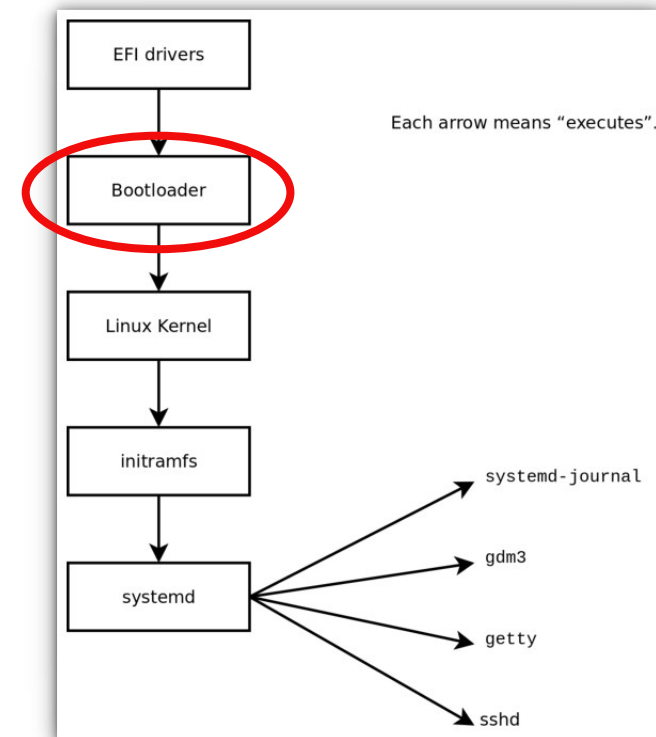
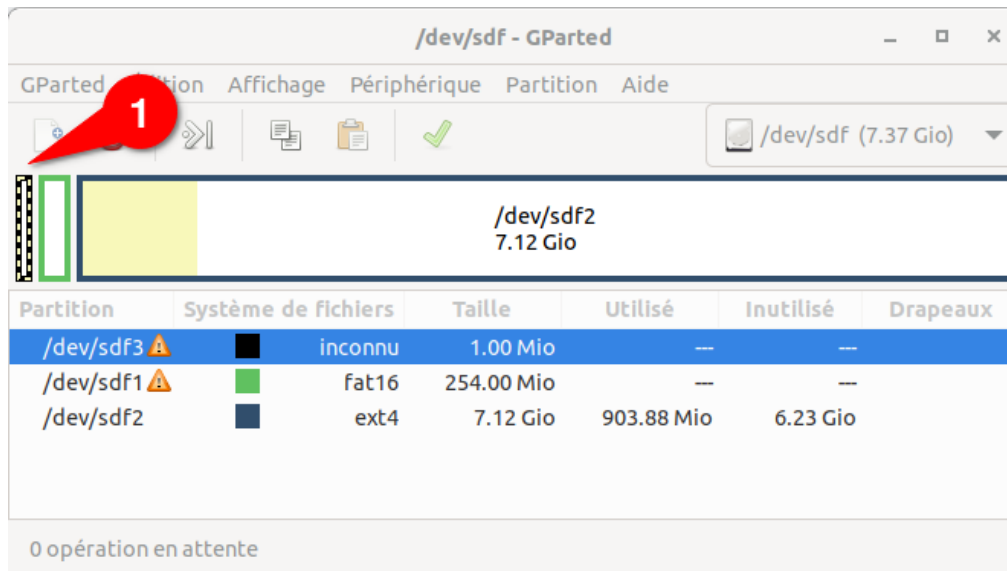
- U-boot :

- . Récupérer la version de u-boot "officielle" sur github
- . patch #1 : détecter la présence d'un fichier bitstream pour le FPGA et auquel cas configurer le FPGA avec
- . patch #2 : forcer l'adresse ethernet pour ne pas qu'elle change à chaque démarrage
- . installer Quartus et un compilateur croisé : linaro
- . compiler u-boot : le resultat se trouve dans `$DEWD/u-boot/u-boot-with-spl.sfp`



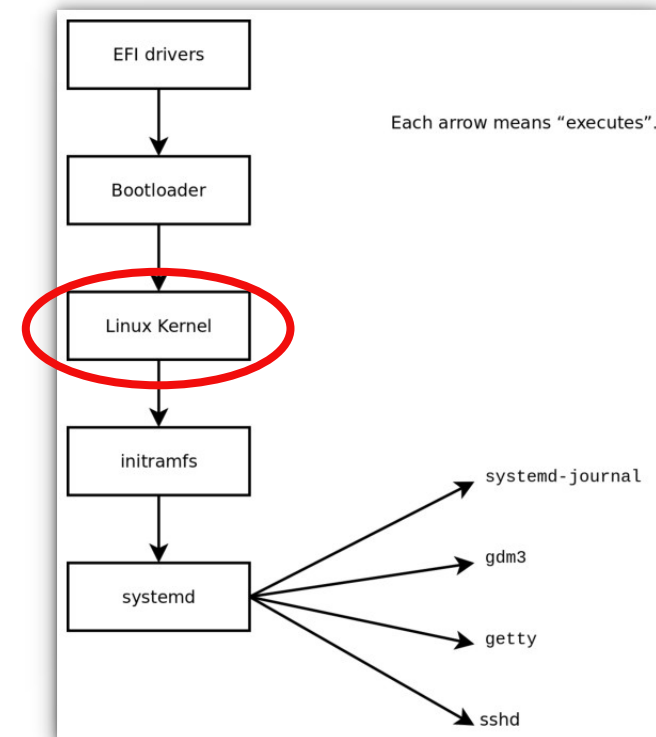
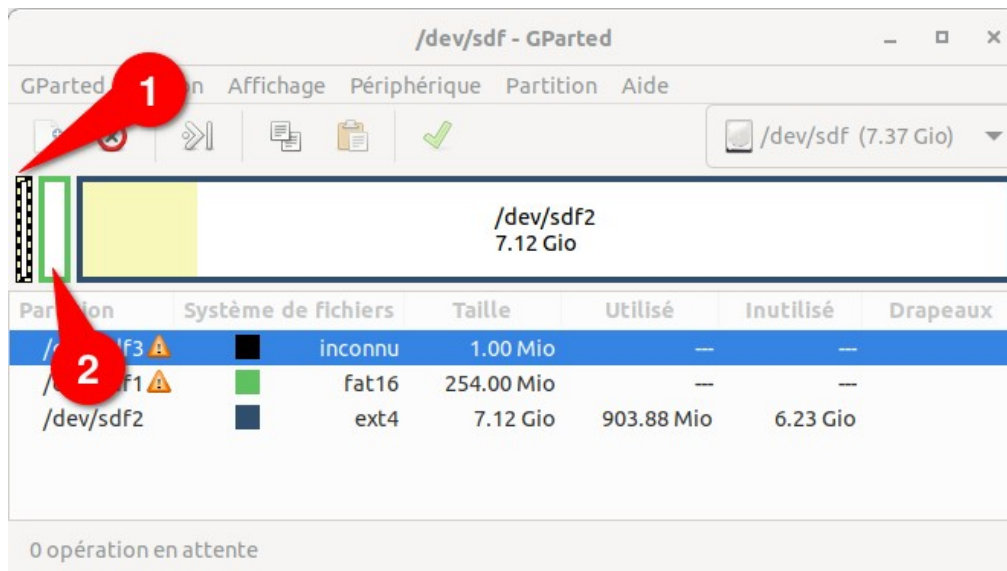
[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

1/3 : Configurer et compiler u-boot



[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

2/3 : Récupérer et compiler les sources du noyau Linux fourni par Altera



[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

2/3 : Récupérer et compiler les sources du noyau Linux fourni par Altera

Continuer à suivre les instructions de <captronic_formation_fpga_img_proc> /scripts/2-2-creer_distrib.sh

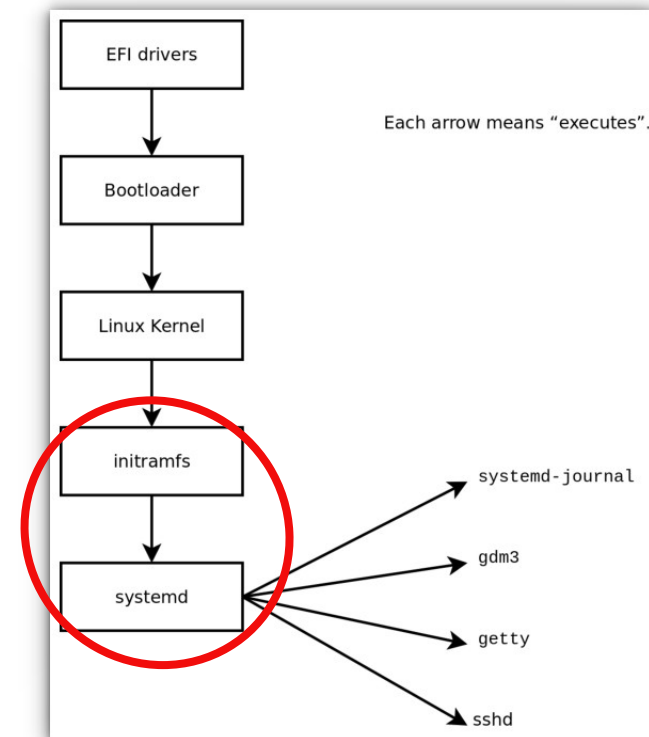
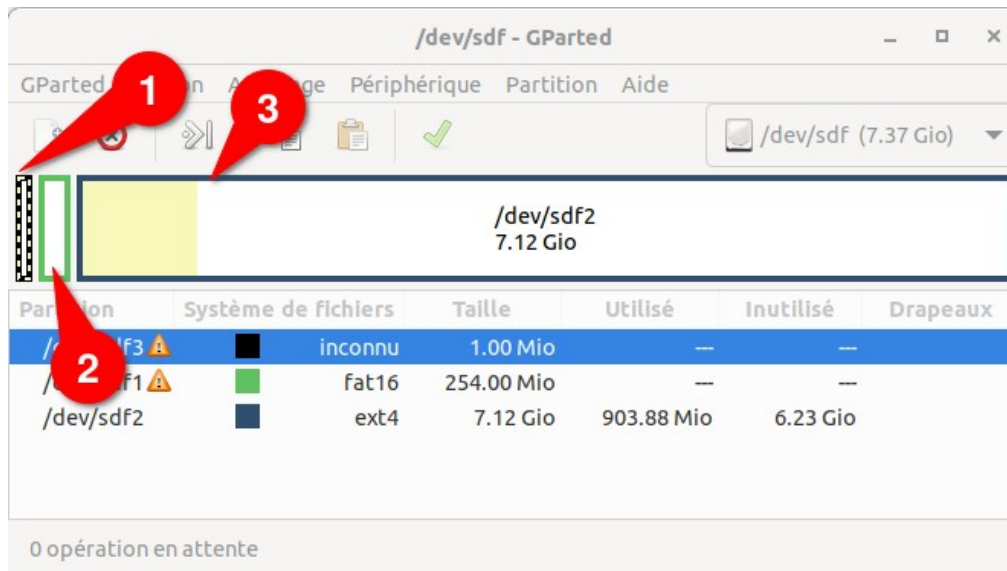
- Noyau Linux :

- . Récupérer la version des sources du noyau Linux fourni par Altera sur github
- . Paramétrer la compilation avec "menuconfig" : la description des paramètres modifiés se trouve dans :
https://bitlog.it/20170820_building_embedded_linux_for_the_terasic_de10-nano.html
<https://github.com/zangman/de10-nano/blob/master/docs/Building-the-Kernel.md>
- . Compiler le noyau Linux : le résultat se trouve dans
\$DEWD/linux-socfpga-socfpga-5.16/arch/arm/boot/zImage

```
11 $DEWD/linux-socfpga-socfpga-5.16/arch/arm/boot/zImage  
-rwxrwxr-x 1 formateur formateur 5591464 Nov 13 00:05 /home/formateur/de10nano-wd/linux-socfpga-socfpga-5.16/arch/arm/boot/zImage*
```

[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

3/3 : Configurer une distribution Linux



[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

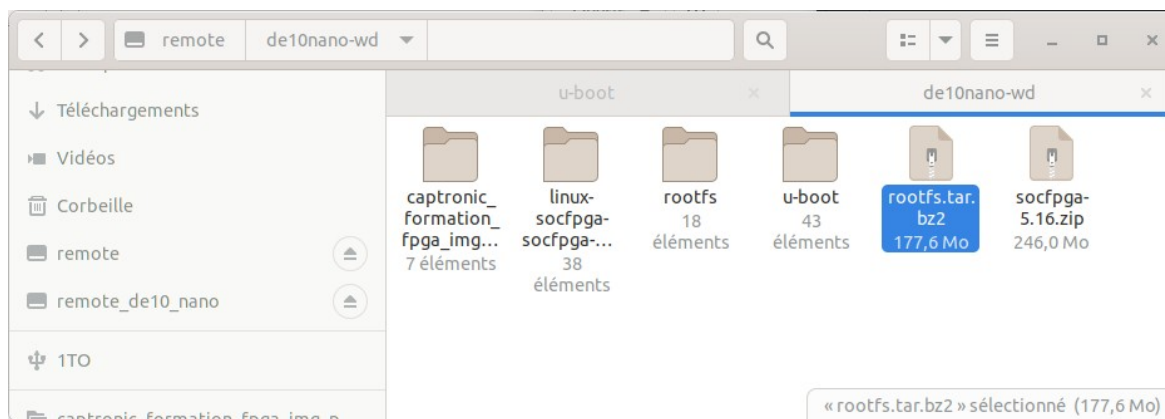
3/3 : Configurer une distribution Linux (3/3)

Continuer à suivre les instructions de <captronic_formation_fpga_img_proc> /scripts/2-2-creer_distrib.sh

- Distribution *Debian* :

- . configurer qemu
- . se connecter à l'émulateur qemu avec "chroot"
- . configurer la distribution dans qemu (dépôts logiciels, activer le réseau, définir le mot de passe root etc.)
- . archiver la distribution dans un fichier tar.bz2

```
ll -h .. |grep "debianRootFS.tar.bz2"
-rw-r--r--  1 root      root      172M Nov 15 11:11 debianRootFS.tar.bz2
```



[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

Bilan / Questions ?

1. Configurer u-boot : OK
2. Configurer un noyau Linux : OK
3. Configurer debian : OK

Prochaines étapes :

Créer une carte SD et booter sur cette carte

Faire communiquer Linux et le FPGA :

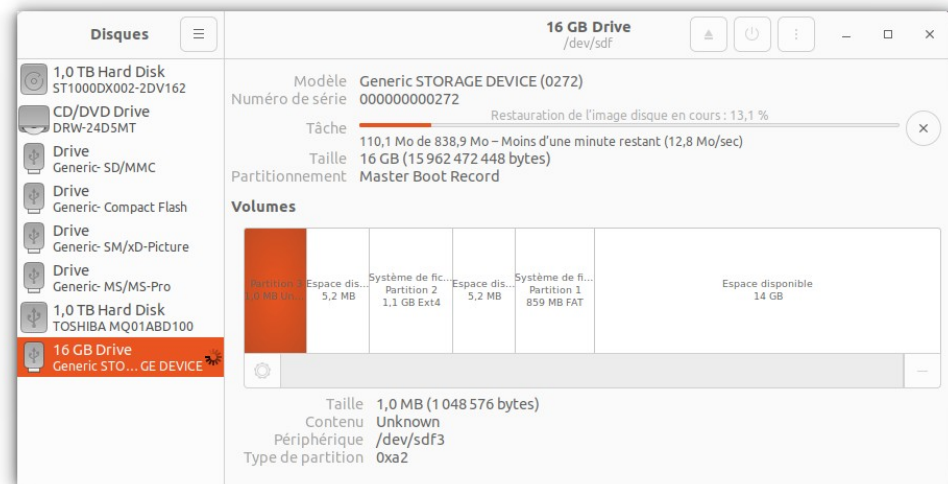
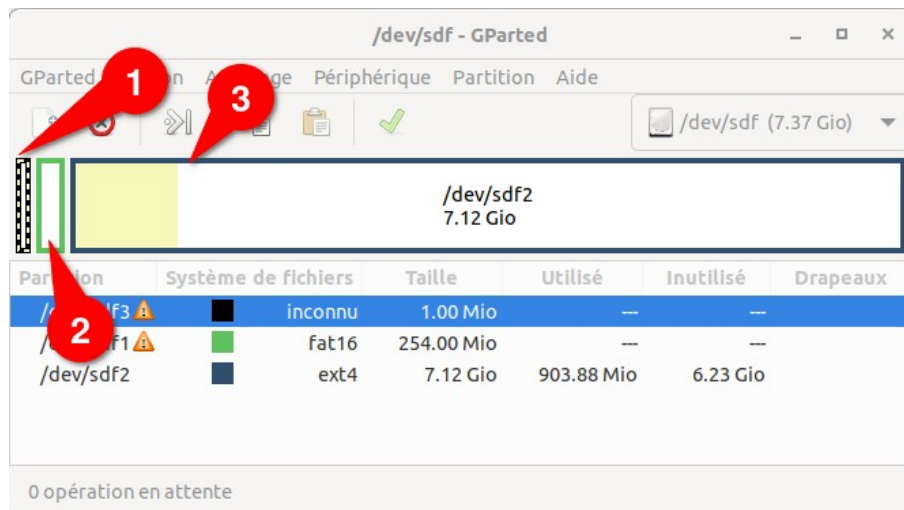
- Avalon
- RAM

[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

Utiliser notre Linux à façon

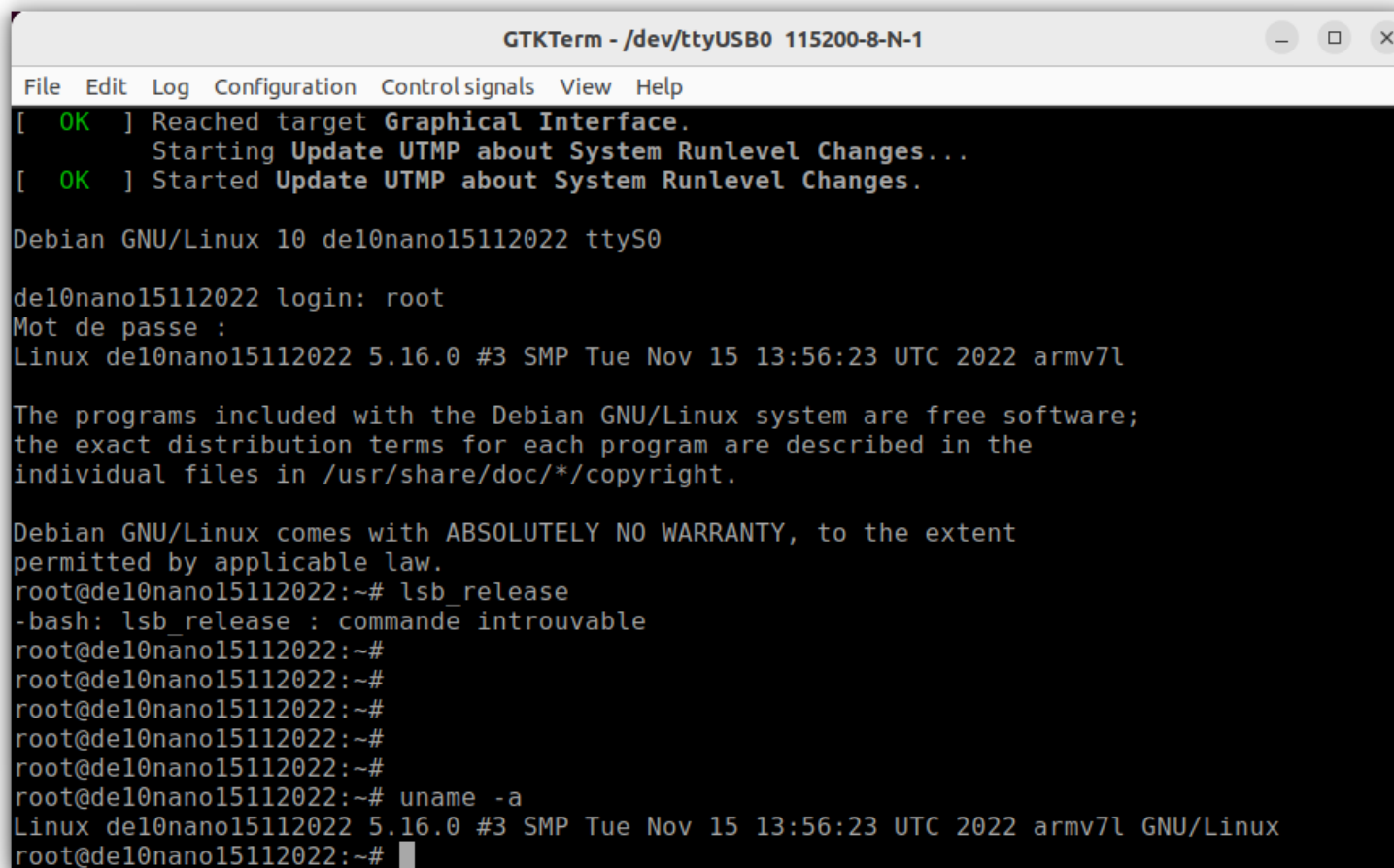
Suivre les instructions de `<captronic_formation_fpga_img_proc> /scripts/3-creer_sd.sh`

- Créer une image disque avec 3 partitions :
 - partition 1 : tel quel u-boot récemment paramétré et compilé
 - partition 1 : l'image du noyau Linux et un fichier texte donnant le chemin vers le device tree
 - partition 3 : la distribution Debian
- Récupérer l'image disque, graver une carte SD
- démarrer le DE10-nano avec la nouvelle distribution
- changer le device tree pour activer les fonctions ethernet



[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

Utiliser notre Linux à façon



```
GTKTerm - /dev/ttyUSB0 115200-8-N-1
File Edit Log Configuration Controlsignals View Help
[ OK ] Reached target Graphical Interface.
        Starting Update UTMP about System Runlevel Changes...
[ OK ] Started Update UTMP about System Runlevel Changes.

Debian GNU/Linux 10 del0nano15112022 ttyS0

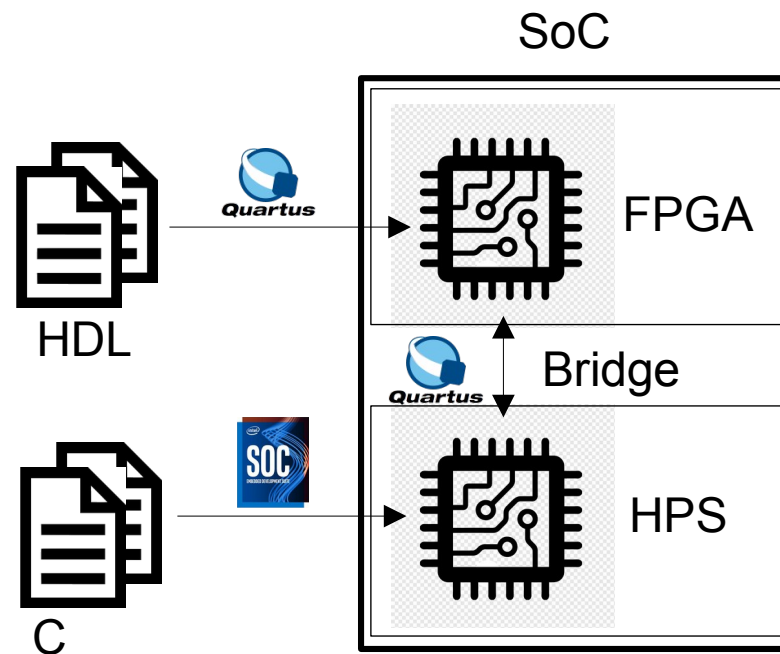
del0nano15112022 login: root
Mot de passe :
Linux del0nano15112022 5.16.0 #3 SMP Tue Nov 15 13:56:23 UTC 2022 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@del0nano15112022:~# lsb_release
-bash: lsb_release : commande introuvable
root@del0nano15112022:~#
root@del0nano15112022:~#
root@del0nano15112022:~#
root@del0nano15112022:~#
root@del0nano15112022:~#
root@del0nano15112022:~#
root@del0nano15112022:~# uname -a
Linux del0nano15112022 5.16.0 #3 SMP Tue Nov 15 13:56:23 UTC 2022 armv7l GNU/Linux
root@del0nano15112022:~#
```

[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

Activer les bridges FPGA / HPS

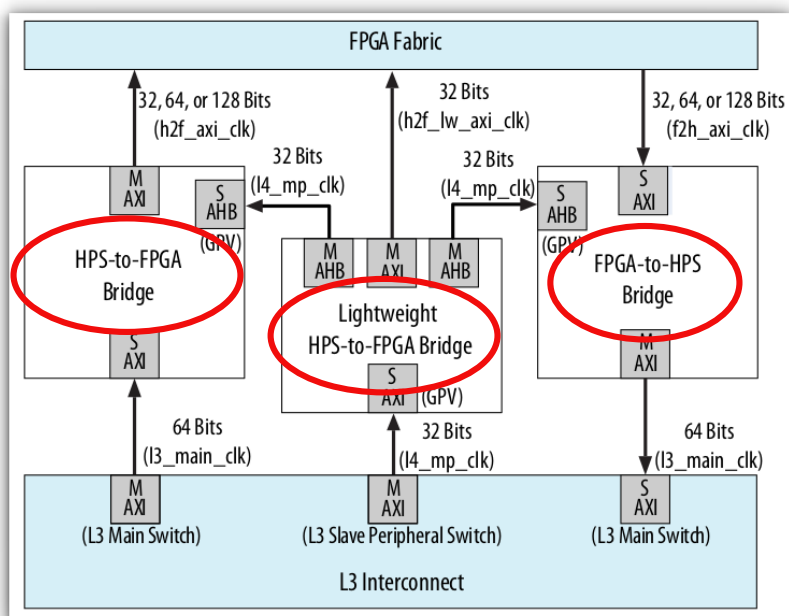


[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

Activer les bridges FPGA / HPS

Suivre les instructions de `<captronic_formation_fpga_img_proc> /scripts/4_activer_bridges.sh`

- Créer un nouveau device tree avec les bridges activés
- Compiler et transférer ce device tree
- Configurer le fichier de démarrage du noyau Linux pour utiliser ce nouveau device tree
- Vérifier que les bridges sont activés



Cyclone V Hard Processor System
Technical Reference Manual

Subscribe
Send Feedback

Last updated for Quartus Prime Design Suite 18.0
10-2018
2018.07.17

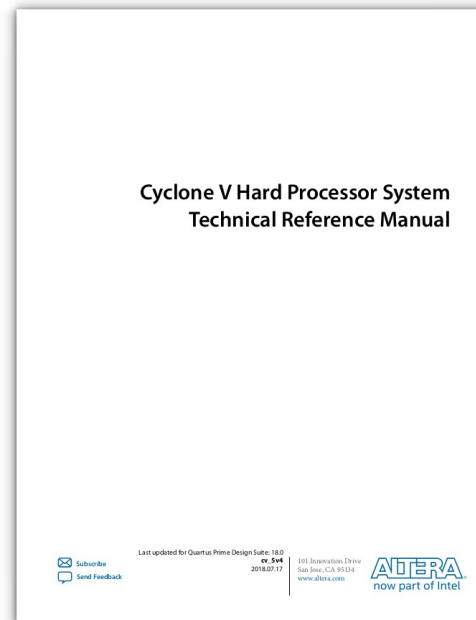
100 Subscription Drive
San Jose, CA 95134
www.altera.com

ALTERA
now part of Intel

activer les bridges HPS <-> FPGA par un device tree

HPS Block Diagram and System Integration

Figure 2-2: HPS Block Diagram



fpga_bridge0	lwhps2fpga-bridge	0xff400000
fpga_bridge1	hps2fpga-bridge	0xff500000
fpga_bridge2	fpga2hps-bridge	0xff600000
fpga_bridge3	fpga2sdram-bridge	0xffc25080

[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

activer les bridges HPS <-> FPGA par un device tree

my_socfpga_cyclone5_de0_nano_soc <----- socfpga_cyclone5.dtsi <----- socfpga.dtsi

less socfpga.dtsi

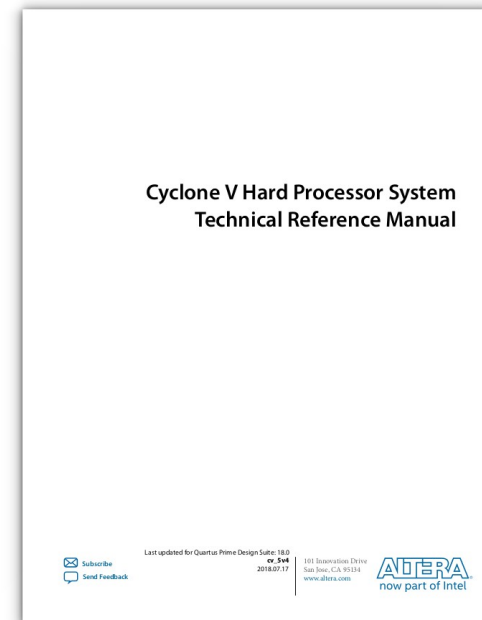
```
fpga_bridge0: fpga_bridge@ff400000 {
    compatible = "altr,socfpga-lwhps2fpga-bridge";
    reg = <0xff400000 0x100000>;
    resets = <&rst LWHPS2FPGA_RESET>;
    clocks = <&l4_main_clk>;
    status = "disabled";
};

fpga_bridge1: fpga_bridge@ff500000 {
    compatible = "altr,socfpga-hps2fpga-bridge";
    reg = <0xff500000 0x100000>;
    resets = <&rst HPS2FPGA_RESET>;
    clocks = <&l4_main_clk>;
    status = "disabled";
};

fpga_bridge2: fpga-bridge@ff600000 {
    compatible = "altr,socfpga-fpga2hps-bridge";
    reg = <0xff600000 0x100000>;
    resets = <&rst FPGA2HPS_RESET>;
    clocks = <&l4_main_clk>;
    status = "disabled";
};

fpga_bridge3: fpga-bridge@ffc25080 {
    compatible = "altr,socfpga-fpga2sdram-bridge";
    reg = <0xffc25080 0x4>;
    status = "disabled";
};
```

fpga_bridge0	lwhps2fpga-bridge	0xff400000
fpga_bridge1	hps2fpga-bridge	0xff500000
fpga_bridge2	fpga2hps-bridge	0xff600000
fpga_bridge3	fpga2sdram-bridge	0xffc25080

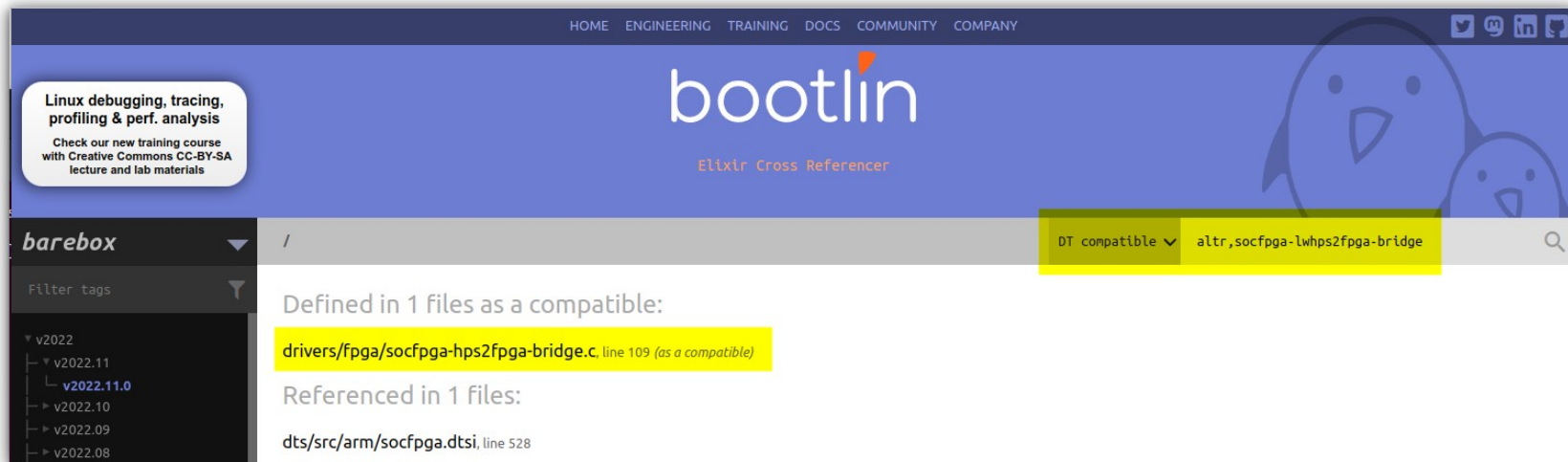


[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

activer les bridges HPS <-> FPGA par un device tree

```
my_socfpga_cyclone5_de0_nano_soc <----- socfpga_cyclone5.dtsi <----- socfpga.dtsi
```

```
less socfpga.dtsi
    fpga_bridge0: fpga_bridge@ff400000 {
        compatible = "altr,socfpga-lwhps2fpga-bridge";
        reg = <0xff400000 0x100000>;
        resets = <&rst LWHPS2FPGA_RESET>;
        clocks = <&l4_main_clk>;
        status = "disabled";
    };
```



<https://elixir.bootlin.com/barebox/v2022.11.0/B/ident/altr%2Csocfpga-lwhps2fpga-bridge>

[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

activer les bridges HPS <-> FPGA par un device tree

```
116 static int alt_fpga_bridge_probe(struct device_d *dev)
117 {
118     struct altera_hps2fpga_data *priv;
119     const struct of_device_id *of_id;
120     u32 enable;
121     int ret;
122
123     of_id = of_match_device(altera_fpga_of_match, dev);
124     priv = (struct altera_hps2fpga_data *)of_id->data;
125
126     priv->bridge_reset = of_reset_control_get(dev->device_node, NULL);
127     if (IS_ERR(priv->bridge_reset)) {
128         dev_err(dev, "Could not get %s reset control\n", priv->name);
129         return PTR_ERR(priv->bridge_reset);
130     }
131
132     priv->clk = clk_get(dev, NULL);
133     if (IS_ERR(priv->clk)) {
134         dev_err(dev, "no clock specified\n");
135         return PTR_ERR(priv->clk);
136     }
137
138     ret = clk_enable(priv->clk);
139     if (ret) {
140         dev_err(dev, "could not enable clock\n");
141         return -EBUSY;
142     }
143
144     priv->dev = dev;
145
146     if (!of_property_read_u32(dev->device_node, "bridge-enable", &enable)) {
147         if (enable > 1) {
148             dev_warn(dev, "invalid bridge-enable %u > 1\n", enable);
149         } else {
150             dev_info(dev, "%s bridge\n",
151                     (enable ? "enabling" : "disabling"));
152             ret = _alt_hps2fpga_enable_set(priv, enable);
153             if (ret)
154                 return ret;
155         }
156     }
157
158     return fpga_bridge_register(dev, priv->name, &altera_hps2fpga_br_ops,
159                                priv);
160 }
161 }
```

```
52 static int _alt_hps2fpga_enable_set(struct altera_hps2fpga_data *priv,
53                                     bool enable)
54 {
55     int ret;
56
57     /* bring bridge out of reset */
58     if (enable)
59         ret = reset_control_deassert(priv->bridge_reset);
60     else
61         ret = reset_control_assert(priv->bridge_reset);
62     if (ret)
63         return ret;
64
65     /* Allow bridge to be visible to L3 masters or not */
66     if (priv->remap_mask) {
67         l3_remap_shadow |= ALT_L3_REMAP_MPUZERO_MSK;
68
69         if (enable)
70             l3_remap_shadow |= priv->remap_mask;
71         else
72             l3_remap_shadow &= ~priv->remap_mask;
73
74         dev_dbg(priv->dev, "setting L3 visibility to 0x%08x\n",
75                 l3_remap_shadow);
76
77         writel(l3_remap_shadow, SOCFPGA_L3_ADDR + ALT_L3_REMAP_OFST);
78     }
79
80     return ret;
81 }
```

Le prise en compte du paramètre "bridge-enable" est faite dans le driver :

/drivers/fpga/socfpga-hps2fpga-bridge.c

Adresse :

```
#define SOCFPGA_L3_ADDR 0xff800000
#define ALT_L3_REMAP_OFST 0x0
```

Masque (reset ou activation) :

```
#define ALT_L3_REMAP_LWH2F_MSK 0x00000010
0x10 =====> binaire : 00010000 => masque sur le bit 4
```

[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

activer les bridges HPS <-> FPGA par un device tree

Module Instance	Base Address	Register Address
13regs	0xFF800000	0xFF800000

Offset: 0x0

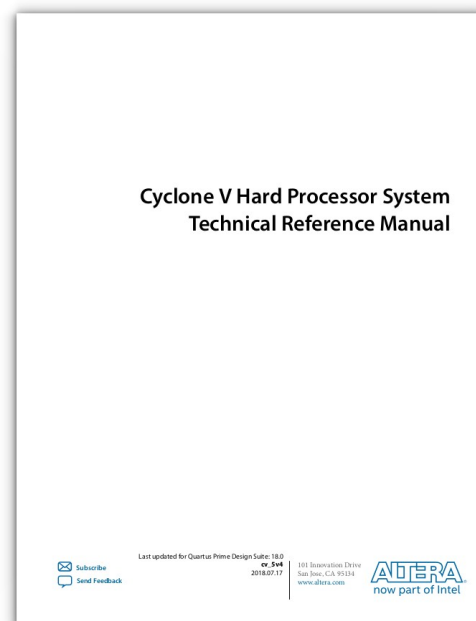
Access: WO

Important: To prevent indeterminate system behavior, reserved areas of memory must not be accessed by software or hardware. Any area of the memory map that is not explicitly defined as a register space or accessible memory is considered reserved.

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											lwhps2fpga	hps2fpga	Reserved	nonmpuzero	mpuzero
											WO 0x0	WO 0x0		WO 0x0	WO 0x0

remap Fields

Bit	Name	Description	Access	Reset						
4	lwhps2fpga	Controls whether the Lightweight HPS2FPGA AXI Bridge is visible to L3 masters or not. <table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>The LWHPS2FPGA AXI Bridge is not visible to L3 masters. Accesses to the associated address range return an AXI decode error to the master.</td></tr><tr><td>0x1</td><td>The LWHPS2FPGA AXI Bridge is visible to L3 masters.</td></tr></table>	Value	Description	0x0	The LWHPS2FPGA AXI Bridge is not visible to L3 masters. Accesses to the associated address range return an AXI decode error to the master.	0x1	The LWHPS2FPGA AXI Bridge is visible to L3 masters.	WO	0x0
Value	Description									
0x0	The LWHPS2FPGA AXI Bridge is not visible to L3 masters. Accesses to the associated address range return an AXI decode error to the master.									
0x1	The LWHPS2FPGA AXI Bridge is visible to L3 masters.									



[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

activer les bridges HPS <-> FPGA par un device tree

Créer le fichier device tree :

arch/arm/boot/dts/**my_socfpga_cyclone5_de0_nano_soc.dts**

Avec le contenu (en fin de fichier) :

```
&fpga_bridge0 {
    status = "okay";
    bridge-enable = <1>;
};

&fpga_bridge1 {
    status = "okay";
    bridge-enable = <1>;
};

&fpga_bridge2 {
    status = "okay";
    bridge-enable = <1>;
};

&fpga_bridge3 {
    status = "okay";
    bridge-enable = <1>;
};
```

[Pour ceux qui sont en avance] Mettre en œuvre sa propre distribution Linux pour SoC Intel

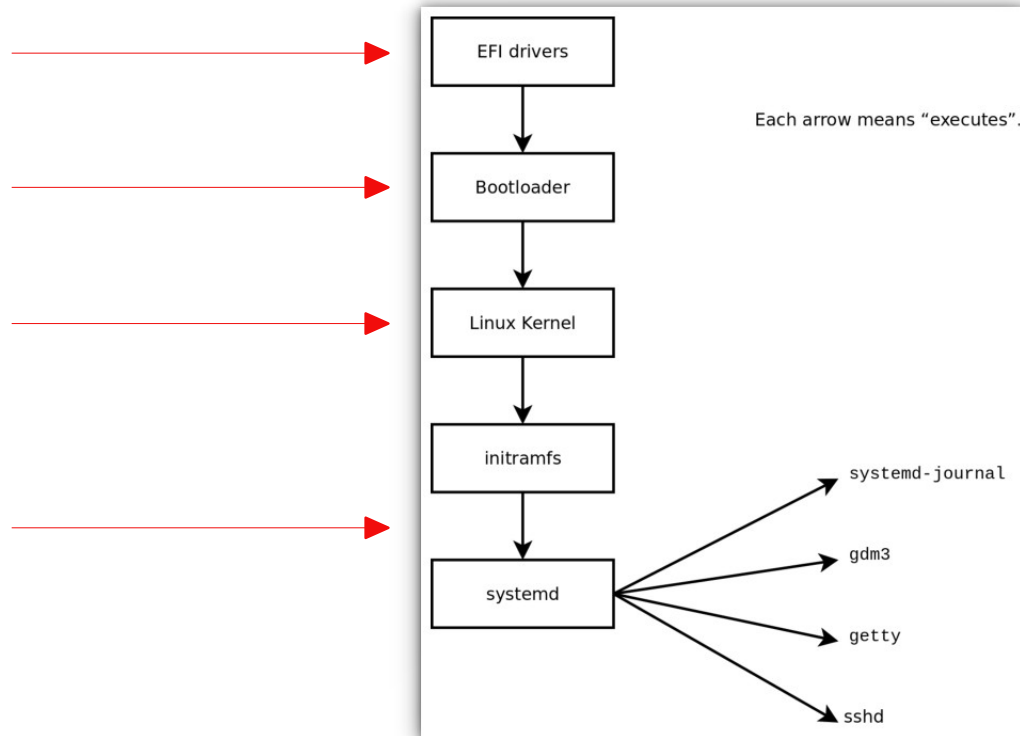
Phases de démarrage d'un SoC FPGA

Choix du périphérique de démarrage (NAND Flash, SD/MMC ou Quad SPI flash) - ici une carte SD - entre autres paramètres

Paramétrer l'adresse MAC
Charger un bitstream dans le FPGA

Choix et chargement du device tree
Activation de modules (filesystem ...)

Applicatifs (ls, vi ...)
Mounter des systèmes de fichiers,
Paramétrer le réseau ethernet ...



Plan

Plan

JOUR 1

§001 Faire communiquer Linux avec un FPGA sur le SoC intel DE10-nano

- distribution fournie par Altera
- créer une image SD (en partie sur un serveur distant)
- **compilation croisée (eclipse CDT embedded)**
- activer les bridges HPS <-> FPGA par un device tree
- utiliser le bridge HPS2FPGA

J 2

§002 Partager une zone de RAM entre Linux et un FPGA sur le SoC intel DE10-nano

- projet de lecture/écriture en RAM sur FPGA
- projet de lecture/écriture en RAM sur CPU

J 3

§003 Cas pratique avec seuillage d'image

- projet openCV
- échange sur mémoire RAM