

# Instaliranje linux operativnog sistema na zybo razvojnoj ploči

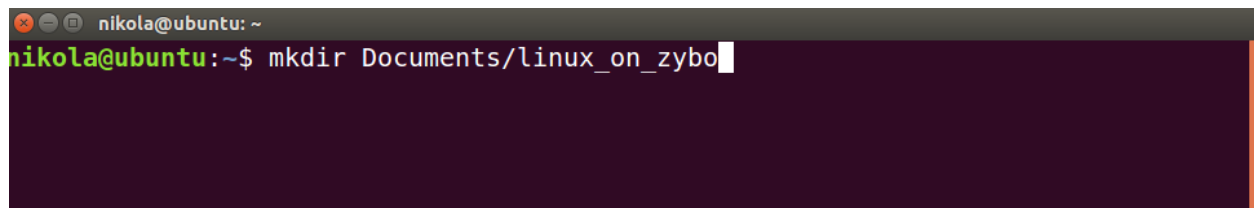
Ova skripta daje objašnjenje kako da se na zybo razvojnoj ploči podigne linux operativni sistem. I taj proces se sastoji iz određenog broja koraka:

## 1) Neophodno je da se poseduje personalni računar sa sledećim karakteristikama:

- Računar sa Ubuntu operativnim sistemom (ili nekim drugim linux operativnim sistemom)
- Računar sa instaliranim Xilinx Vivado i SDK alatima (prilikom pisanja ove skripte korišćena verzija Xilinx Vivada i SDK-a bila je 2018.3).
- MicroSD kartica veličine minum 4GB.

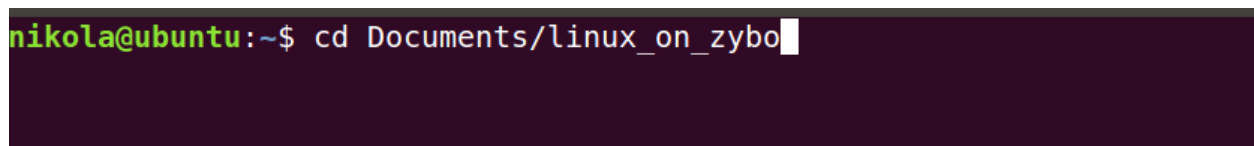
## 2) Dobavljanje neophodnih fajlova:

Prvo što je neophodno uraditi jeste da se na personalnom računaru napravi direktorijum u koji će se smeštati svi fajlovi potrebni za podizanje linux operativnog sistema na zybo-u (imati u vidu da će većina komandi biti pokrenuta iz terminala).

A terminal window with a dark background. The prompt is 'nikola@ubuntu: ~'. The command 'mkdir Documents/linux\_on\_zybo' has been entered, and the cursor is at the end of the line.

```
nikola@ubuntu: ~  
nikola@ubuntu:~$ mkdir Documents/linux_on_zybo
```

Nakon što je direktorijum napravljen potrebno je ući u taj direktorijum sledećom naredbom:

A terminal window with a dark background. The prompt is 'nikola@ubuntu:~\$'. The command 'cd Documents/linux\_on\_zybo' has been entered, and the cursor is at the end of the line.

```
nikola@ubuntu:~$ cd Documents/linux_on_zybo
```

Nakon ulaska u direktorijum, neophodno je dobiti neophodne podatke sa interneta, i to:

- Direktorijum potreban za generisanje U-boot fajla.
- Dodatak neophodan SDK programu kako bi se mogao generisati Device Tree fajl
- Xilinx Linux Kernel 4.10

To se radi kucanjem sledećih komandi u linux terminalu:

- Ova komanda će svući sa github-a **u-boot-xlnx** direktorijum .

```
nikola@ubuntu:~/Documents/linux_on_zybo$ git clone https://github.com/SDU-Embedded/u-boot-xlnx.git
Cloning into 'u-boot-xlnx'...
remote: Enumerating objects: 2, done.
remote: Counting objects: 100% (2/2), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 456798 (delta 0), reused 0 (delta 0), pack-reused 456796
Receiving objects: 100% (456798/456798), 117.84 MiB | 6.81 MiB/s, done.
Resolving deltas: 100% (365043/365043), done.
Checking connectivity... done.
nikola@ubuntu:~/Documents/linux_on_zybo$
```

- Sledeća komanda dobavlja dodatak neophodan SDK programu kako bi mogao da generiše Device Tree fajl.

```
nikola@ubuntu:~/Documents/linux_on_zybo$ git clone https://github.com/SDU-Embedded/device-tree-xlnx.git
Cloning into 'device-tree-xlnx'...
remote: Enumerating objects: 2, done.
remote: Counting objects: 100% (2/2), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3699 (delta 0), reused 0 (delta 0), pack-reused 3697
Receiving objects: 100% (3699/3699), 566.65 KiB | 0 bytes/s, done.
Resolving deltas: 100% (1586/1586), done.
Checking connectivity... done.
nikola@ubuntu:~/Documents/linux_on_zybo$
```

- A sa sledećeg linka je potrebno skinuti verziju Xilinx Linux Kernela 4.10:

<https://github.com/SDU-Embedded/linux-xlnx/releases/tag/zynmp-dt-fixes-for-4.10>

Skinuti fajl sa tar.gz ekstenzijom, i prebaciti ga u direktorijum koji smo napravili (u skripti to je folder sa imenom linux\_on\_zybo).

### 3) Instaliranje neophodnih alata i aplikacija:

Ukoliko se na personalnom računaru nalazi „sveža“ verzija linux-a, neophodno je instalirati određene pakete sledećom komandom u linux terminalu:

- *apt install libncurses5-dev libncursesw5-dev u-boot-tools device-tree-compiler flex libssl-dev*

```
nikola@ubuntu:~/Documents/linux_on_zybo$ apt install libncurses5-dev libncursesw5-dev u-boot-tools device-tree-compiler flex libssl-dev
```

Kako bi se moglo izvršiti kompajliranje za Zybo razvojnu ploču, neophodno je postaviti CROSS\_COMPILE promenljivu tako da se odnosi na Zynq-7000 sistem na čipu i to na sledeći način:

```
export CROSS_COMPILE=arm-linux-gnueabihf-
```

```
nikola@ubuntu:~/Documents/linux_on_zybo$ export CROSS_COMPILE=arm-linux-gnueabihf-
```

#### **NAPOMENA:**

*Ukoliko se koristi verzija Vivada 2017.2 i niže potrebno je ukucati:*

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

Takođe EXPORT komanda važi samo za trenutnu instancu linux terminala, pa u slučaju da se isti zatvori potrebno je uneti komandu ponovo. Nakon toga potrebno je ukucati sledeće:

```
source <Xilinx instalacioni direktorijum>/Vivado/<version>/settings64.sh
```

```
nikola@ubuntu:~/Documents/linux_on_zybo$ source /tools/Xilinx/Vivado/2018.3/settings64.sh
```

#### **4) Instalacija DTC (Device tree kompajlera):**

U slučaju da računar već poseduje DTC, ovaj korak se može preskočiti. Način na koji može da se proverí postojanje DTC-a jeste da se u terminalu ukuca sledeća komanda:

```
dtc -v
```

A da bi se proverio direktorijum u kome je instaliran dtc neophodno je u terminalu ukucati:

```
which dtc
```

Većina operativnih sistema ima instaliran dtc, ali ukoliko to nije slučaj, on se može skinuti sa sledećeg linka:

<https://git.kernel.org/pub/scm/utils/dtc/dtc.git>

Kada se taj direktorijum skine, potrebno je ući u njega i u terminalu ukucati sledeću komandu:

```
make
```

Takođe Xilinx preporučuje da se putanja na kojoj je instaliran DTC ubaci u PATH promenljivu na sledeći način:

```
export PATH=`pwd`:.$PATH
```

## 5) Kreiranje U-boot fajla

U direktorijumu koji smo napravili na početku bi trebalo da se nalaze tri direktorijuma, kao što je to prikazano na sledećoj slici:

```
nikola@ubuntu:~/Documents/linux_on_zybo$ ls
device-tree-xlnx  linux-xlnx-zynmp-dt-fixes-for-4.10  u-boot-xlnx
nikola@ubuntu:~/Documents/linux_on_zybo$
```

### NAPOMENA:

*Ukoliko se koristi ubuntu 18.04, potrebno je proveriti da li je verzija openssl-a veća od 1.0, i ukoliko jeste potrebno ju je spustiti na nižu verziju (1.0), kako bi mogao da se generiše u-boot fajl. To se radi sledećom komandom:*

```
sudo apt-get install libssl1.0-dev
```

Da bi se kreirao U-boot fajl, neohpodno je prvo ući u *u-boot-xlnx* folder pomoću sledeće komande:

```
cd u-boot-xlnx
```

U tom direktorijumu potrebno je ukucati sledeće:

```
make zynq_zybo_config
```

```
nikola@ubuntu:~/Documents/linux_on_zybo/u-boot-xlnx$ make zynq_zybo_config
HOSTCC  scripts/basic/fixdep
HOSTCC  scripts/kconfig/conf.o
SHIPPED scripts/kconfig/zconf.tab.c
SHIPPED scripts/kconfig/zconf.lex.c
SHIPPED scripts/kconfig/zconf.hash.c
HOSTCC  scripts/kconfig/zconf.tab.o
HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#
nikola@ubuntu:~/Documents/linux_on_zybo/u-boot-xlnx$
```

Nakon toga potrebno je jos ukucati sledeću komandu:

```
make
```

Nakon ovoga generisao se U-boot fajl u trenutnom direktorijumu, dok se u direktorijumu *u-boot-xilinx/tools* generisao *mkimage* fajl koji mora biti dostupan prilikom build-ovanja kernela (što je sledeći korak). Da bi se *mkimage* fajl omogućio dostupnim prilikom build-ovanja kernela, neophodno je uraditi sledeće dve komande:

```
cd tools
```

```
export PATH=`pwd`: $PATH
```

I poslednja stvar koju je neophodno uraditi u ovom koraku jeste da se preimenuje *u-boot* fajl koji je generisan u *u-boot.elf*, i to se radi se sledeće dve komande:

```
cd .. (komanda za izlazak iz trenutnog direktorijuma)
```

```
mv u-boot u-boot.elf (komanda za preimenovanje fajla)
```

## 6) Kreiranje linux kernela

Potrebno je prvo preći u direktorijum *linux-xlnx-zynmp-dt-fixes-for-4.10*. Nakon toga je potrebno skinuti sa sledećeg linka jedan konfiguracioni fajl:

[https://github.com/SDU-Embedded/linux\\_zynq/blob/master/linux\\_zybo/xilinx\\_zynq\\_defconfig](https://github.com/SDU-Embedded/linux_zynq/blob/master/linux_zybo/xilinx_zynq_defconfig)

Kada se to uradi, potrebno je kopirati taj fajl na sledeću putanju **arch/arm/configs** (**napomena, ime tog fajla mora biti xilinx\_zynq\_defconfig!**). Posle toga potrebno je pokrenuti sledeće komande kako bi se generisao *ulmage* fajl:

```
make ARCH=arm xilinx_zynq_defconfig
```

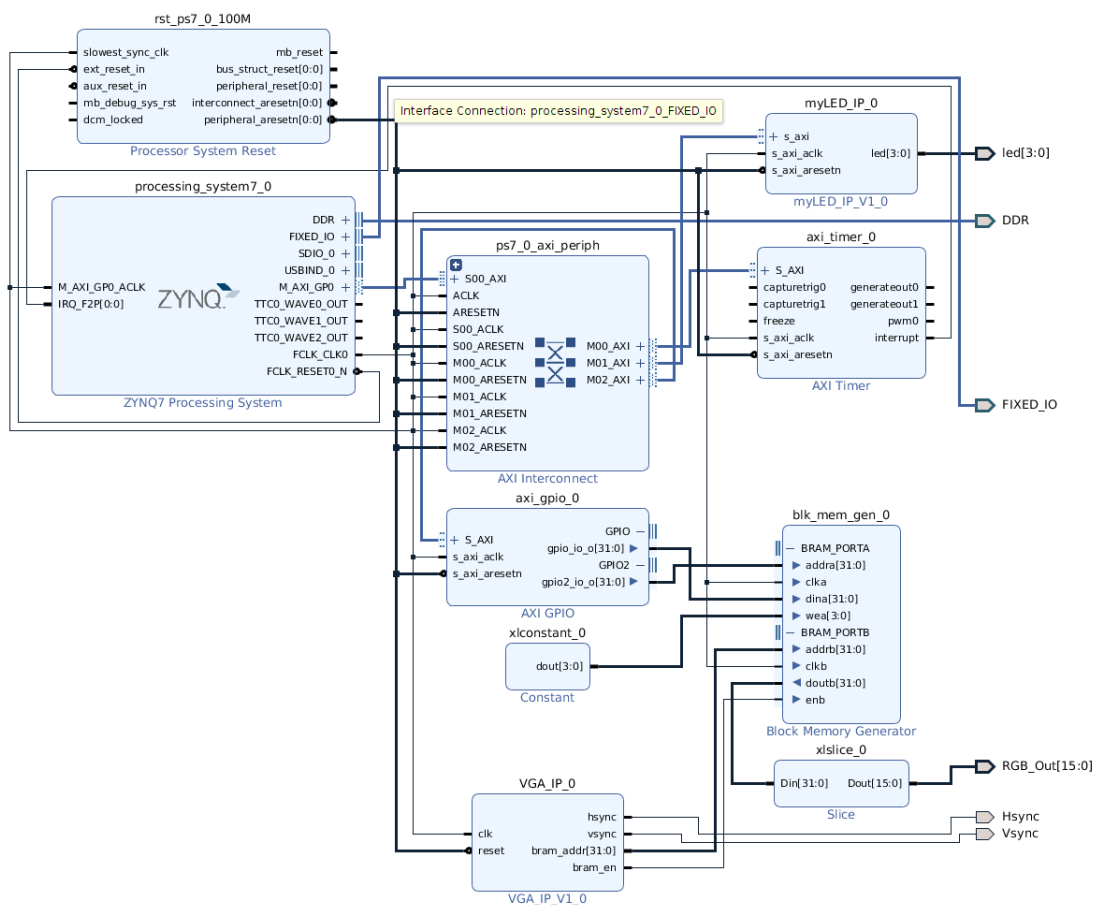
```
make ARCH=arm menuconfig
```

Nakon ove komande otvoriće se kernel konfiguracioni meni, njega je potrebno samo zatvoriti ukoliko nešto ne želite da menjate. Nakon sledeće komande kreće proces generisanja *ulmage* fajla koji traje nekoliko minuta, i kada se on završi *ulmage* fajl će se nalaziti na putanji **arch/arm/boot/**.

```
make ARCH=arm UIMAGE_LOADADDR=0x8000 uimage
```

## 7) Kreiranje BOOT image fajla u Vivado SDK-u

Kada se nešto planira staviti u programabilnu logiku zybo razvojne ploče, neohpodno je napraviti BOOT.bin fajl (pravi se u ovom koraku), i devicetree.dtb fajl (pravi se u sledećem koraku). Za primer pravljanja ova dva fajla koristićemo projekat prikazan na sledećoj slici:



Ovaj projekat je napravljen u vivado IP integratoru, i njegovo detaljno objašnjenje nalazi se u drugoj skripti. Kako bi se projekat automatski generisao, potrebno je skinuti sledeće sa github-a:

[https://github.com/DjordjeMiseljic/VGA\\_BRAM\\_Controller.git](https://github.com/DjordjeMiseljic/VGA_BRAM_Controller.git)

Prvo je potrebno pokrenuti vivado, u njemu pritisnuti *tools->Run Tcl Script*, navesti putanju do skinutog direktorijuma i u njemu dvoklikom pokrenuti Master.tcl fajl. Vivado na osnovu tog fajla generiše, sintetizuje, implementira projekat, i na osnovu toga generiše bitstream fajl(za sve ovo je potrebno 10-tak minuta).

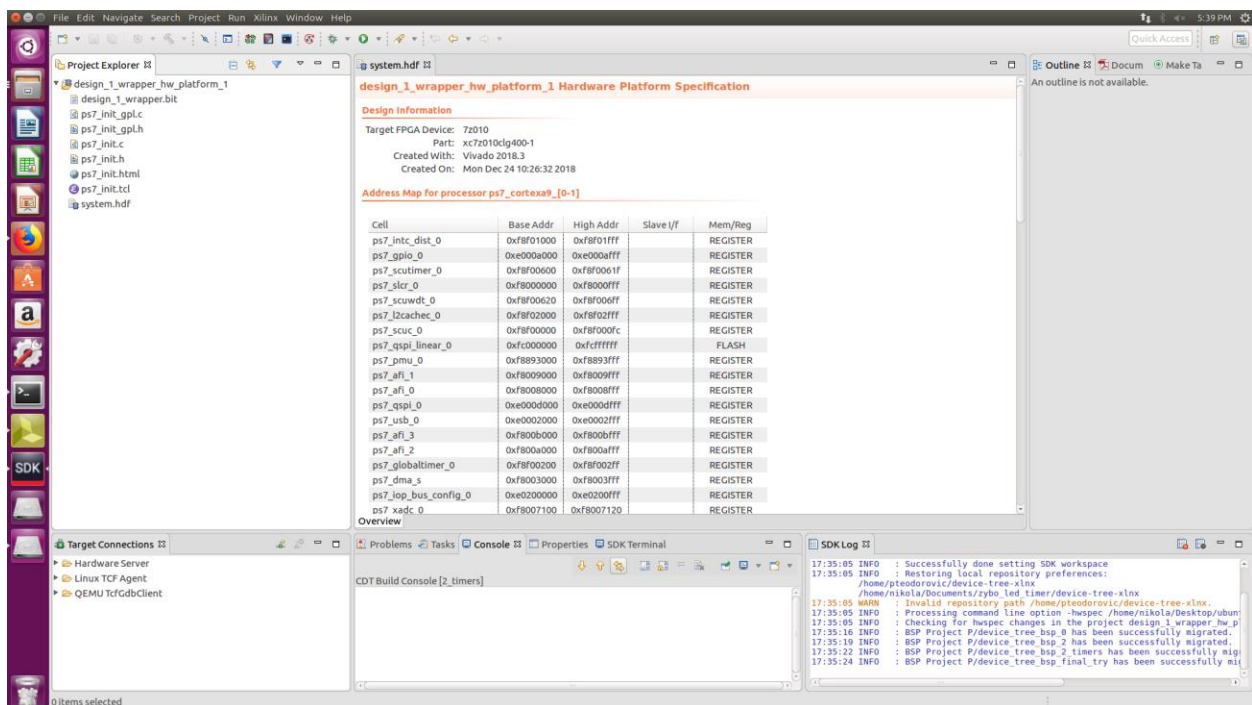
### NAPOMENA:

*Da bi tcl skripta mogla da se pokrene ispratiti sledeće upustvo za ubacivanje zybo board fajlova ako oni već nisu ubačeni.*

<https://reference.digilentinc.com/reference/software/vivado/board-files>

Nakon toga potrebno je izabrati opciju File->Export->Export Hardware(kliknuti kvadratić include bitstream), posle čega se može uključiti Vivado SDK tako što se pritisne na File->Launch SDK .

Kada se uključi SDK dobije se sledeći prozor:



Na njemu pritisnuti na *file->new->application\_project*. U prozoru koji iskoči ukucati <ime\_aplikacije>\_fsbl i pritisnuti **NEXT**, posle toga izabrati opciju Zynq FSBL i kliknuti **finish**. Tada će se sa leve strane pojaviti *example\_fsbl* folder, pritisnuti desni klik iznad njega i izabrati opciju Create Boot Image. Nakon toga iskače sledeći prozor:

Create Boot Image

Creates Zynq Boot Image in .bin format from given FSBL elf and partition files in specified output folder.

Architecture:

Zynq

☒ Create new BIF file

☐ Import from existing BIF file

Basic

Security

Output BIF file path:

/home/nikola/Desktop/ubuntu\_zybo/ubuntu\_zybo.sdk/example\_fsbl/bootimage/example

Browse...

UDF data:

Browse...

☐ Split

Output format:

BIN

Output path:

/home/nikola/Desktop/ubuntu\_zybo/ubuntu\_zybo.sdk/example\_fsbl/bootimage/BOOT.b

Browse...

Boot image partitions

File path	Encrypted	Authenticated
(bootloader) /home/nikola/Desktop/ubuntu_zybo	none	none
/home/nikola/Desktop/ubuntu_zybo/ubuntu_zybo	none	none

?

Preview BIF Changes

Cancel

Create Image

Add

Delete

Edit

Up

Down

Kliknuti na Add i doći do direktorijuma u kome se nalazi boot.elf fajl. Kada se to uradi, potrebno je kliknuti Create image, i BOOT fajl će biti napravljen na putanji specificiranoj u polju *Outpath path*.

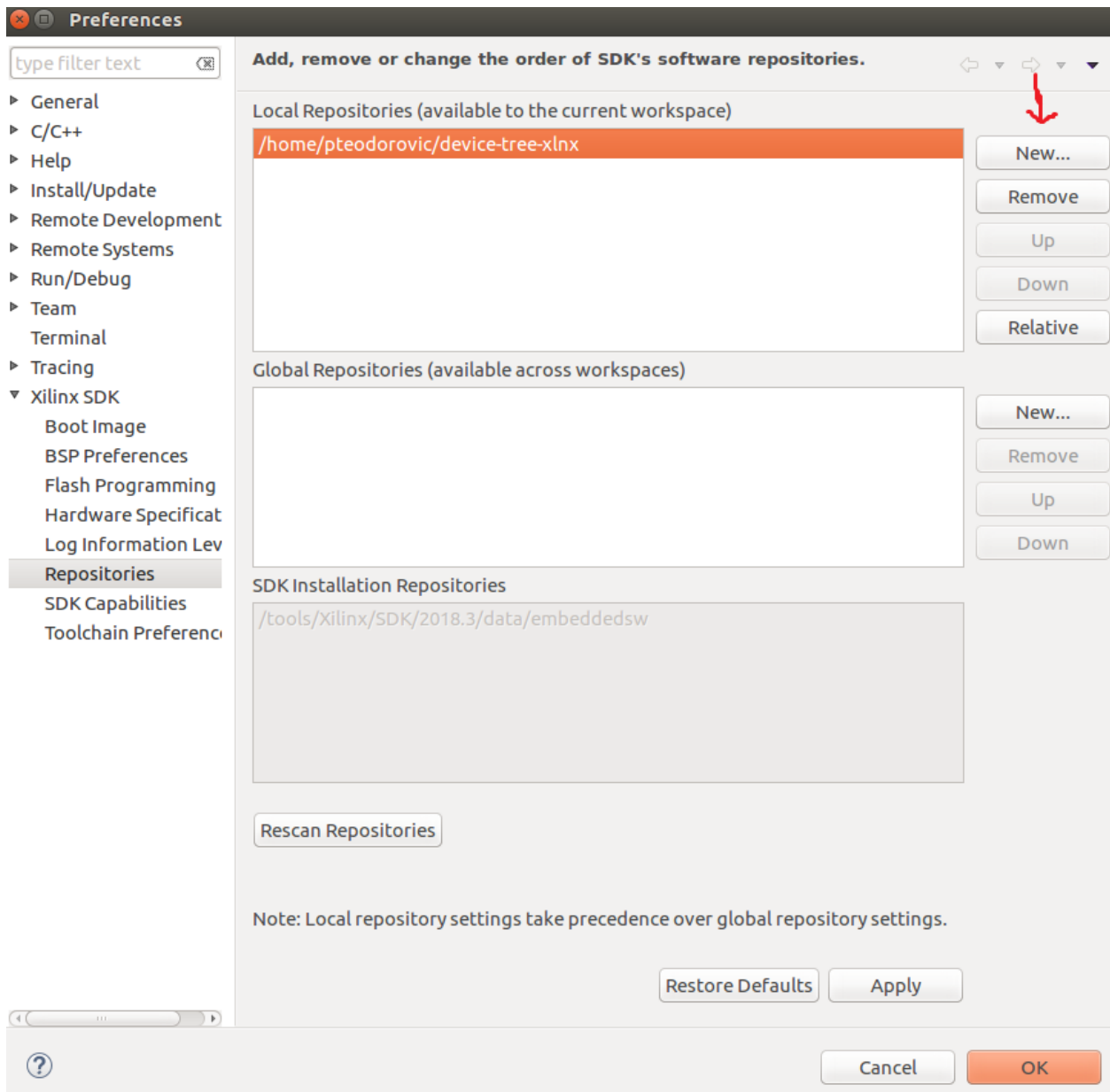


## 8) Kreiranje Device Tree Blob fajla (DTB fajl)

Prvo je potrebno skinuti devicetree.dts fajl sa sledećeg linka:

[https://github.com/SDU-Embedded/linux\\_zyng/blob/master/linux\\_zybo/devicetree.dts](https://github.com/SDU-Embedded/linux_zyng/blob/master/linux_zybo/devicetree.dts)

Nakon toga potrebno je u SDK-u pritisnuti na opciju Xilinx->Repositories. Prozor koji se tada otvori je:



Tu je potrebno pritisnuti opciju new označenu na slici, i doći do direktorijuma *device-tree-xlnx* (to je jedan od 3 direktorijuma koji je skinut u koraku 2), nakon toga pritisnuti ok.

Sada je potrebno pritisnuti File->New->Board Support Package. Prozor koji se otvara izgleda kao na sledećoj slici:

**New Board Support Package Project**

**Xilinx Board Support Package Project**  
Create a Board Support Package.

Project name:

☒ Use default location

Location:

Choose file system:

**Target Hardware**

Hardware Platform:

CPU:

Compiler:

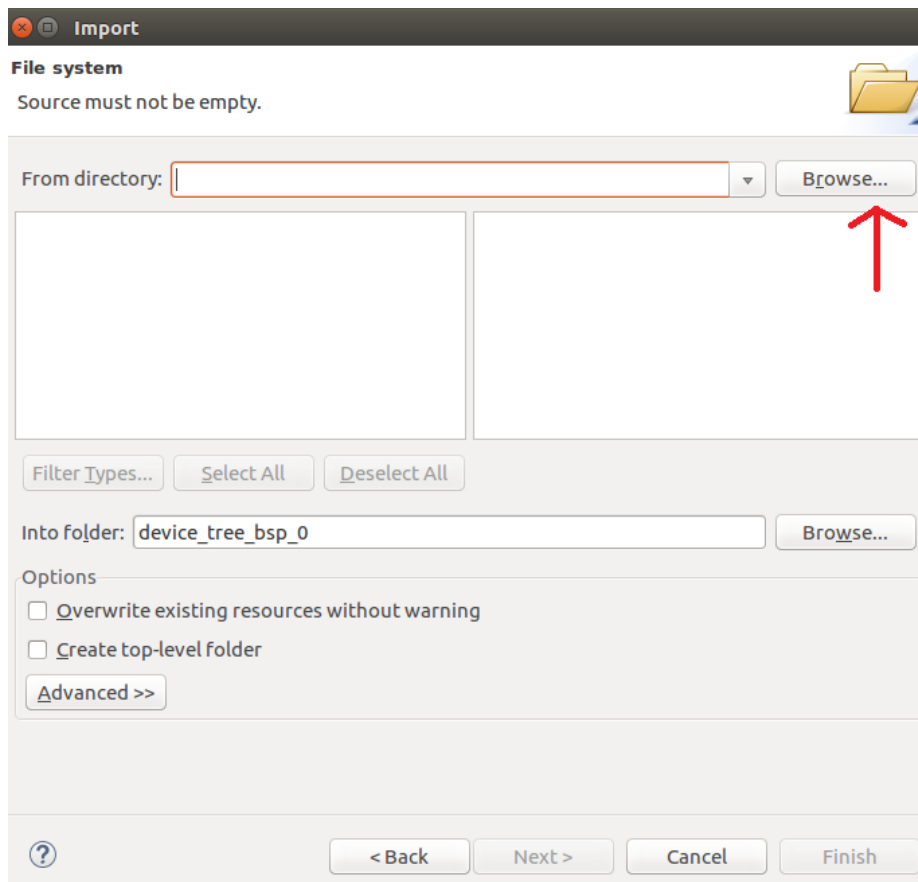
**Board Support Package OS**

☒ device\_tree ☐ flat device tree

☐ freertos10\_xilinx

☐ standalone

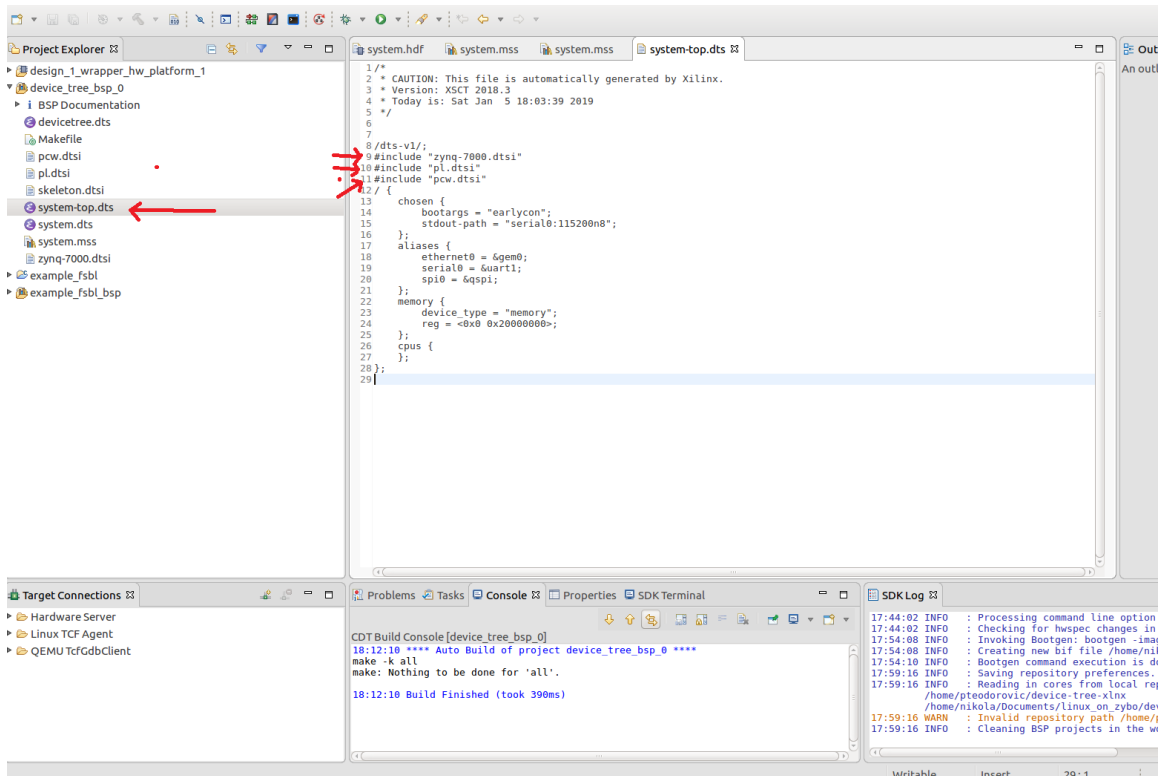
Tu izabrati opciju *device\_tree* kao što je označeno na slici, i pritisnuti *finish*. Kada se pritisne finish otvorice se prozor *board support package settings*, u njemu samo pritisnuti ok. Nakon toga pritisnuti desni klik iznad direktorijuma *device\_tree\_bsp\_0* i kliknuti import. Nakon toga otvara se prozor u kome je potrebno dvoklikom otvoriti *file\_system* direktorijum pri čemu se otvara sledeći prozor:



U njemu je pritiskom na browse potrebno navesti putanju direktorijuma u kome se nalazi devicetree.dts fajl koji skinut na početku ovog koraka. Kada se navede ta putanja iskočiće devicetree.dts fajl pored strelice sa prethodne slike, potrebno je otkāčiti taj fajl i pritisnuti finish.

## NAPOMENA:

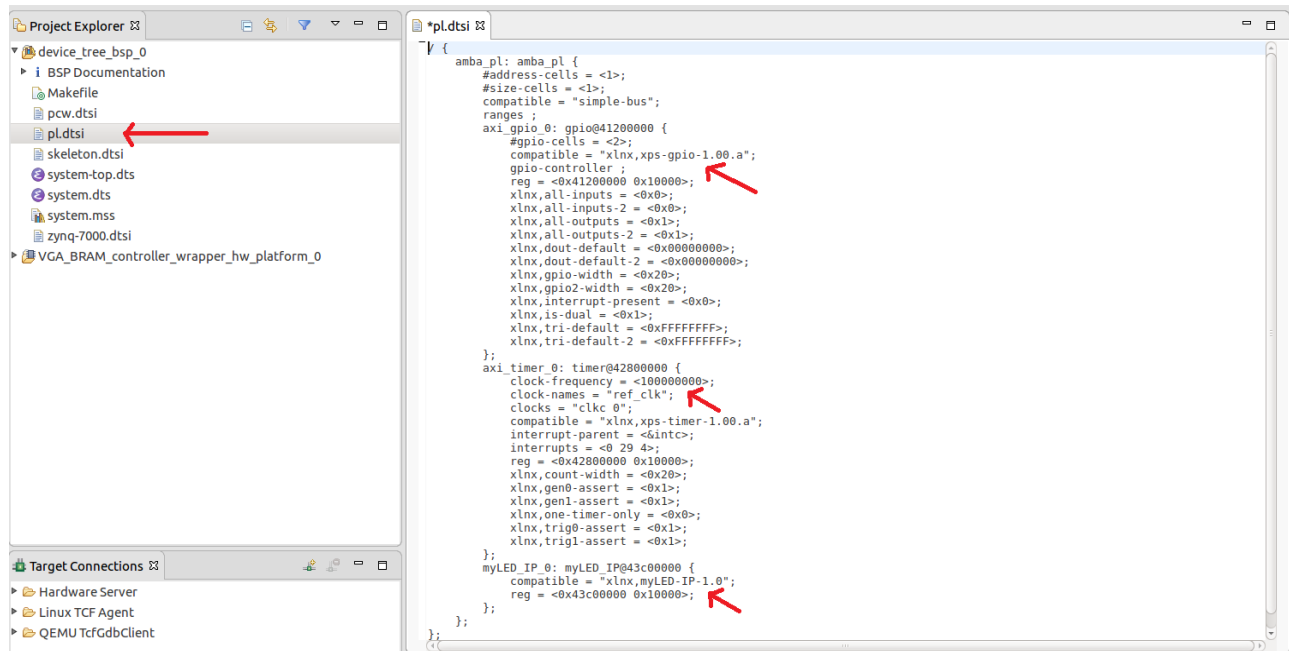
Vivado SDK 2018.3 unutar `device_tree_bsp_0` foldera generiše fajl sa sintaksnom greškom. Da bi se to prepravilo potrebno je ući u fajl sa nazivom „`system-top.dts`“ i na svim mestima umesto `#include` napisati `/include/`.



Takođe SDK može da napravi grešku u `pl.dtsi` fajlu unutar `device_tree_bsp_0` direktorijuma i da za vrednost frekvencije napiše `1e+08` što je sintaksna greška, to se treba prepraviti u `100000000`. Pored toga u ovom koraku moguće je promeniti imena `compatible` polja, jer su to polja preko kojih će karnel da poveže drajver sa modulom.

U ovom primeru ta imena treba da budu promenjena na sledeći način:

`"Xlnx,xps-gpio-1.00.a"` -> `"xlnx,gpio_bram_control"`



Sada je potrebno napraviti *Device tree Blob* i za to se koristi `dtc` kompajler. Prvo je potrebno iz linux terminala doći do `device_tree_bsp_0` fajla, koji se nalazi u direktorijumu u kome je napravljen naš projekat prilikom njegovog prvog pokretanja u Vivado alatu. U ovom slučaju ta putanja je:

```
nikola@ubuntu:~$ cd ~/Downloads/VGA/Top/result/VGA_BRAM_controller.sdk/device_tree_bsp_0/
```

Kada se dođe u taj direktorijum potrebno je ukucati sledeće:

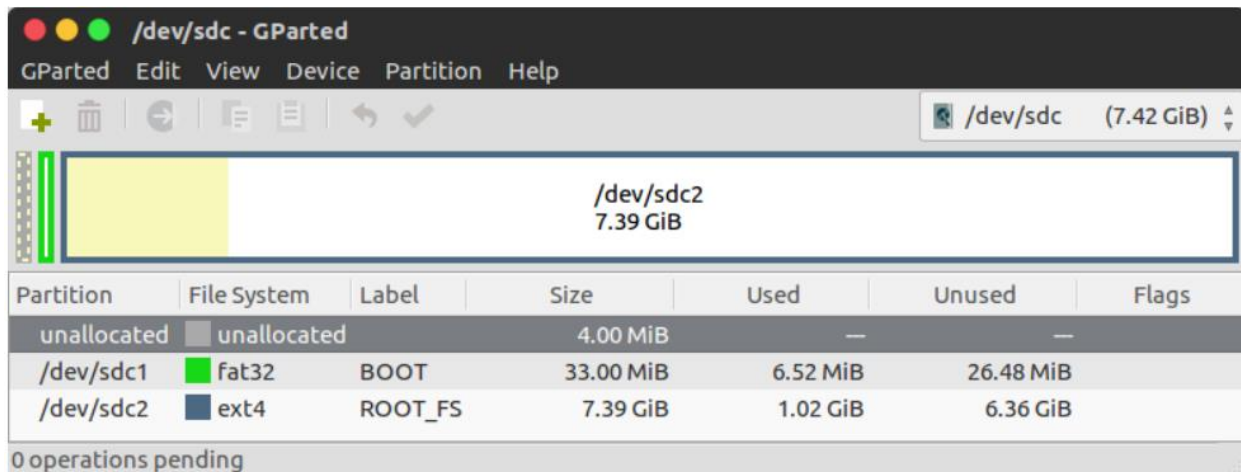
```
dtc -I dts -O dtb -o devicetree.dtb devicetree.dts
```

Nakon čega se kreira `devicetree.dtb` fajl.

## 9) Pravljenje boot-abilne SD kartice

Sada su generisani svi fajlovi potrebni za podizanje linuxa na zybo razvojnoj ploči, neophodno je još samo formatirati microSD karticu, i to na sledeći način:

- Prva 4 MB nealocirana
- 32 MB prostora tipa FAT16 ili FAT32 sa labelom BOOT
- Najmanje 1GB ext4 prostora sa labelom ROOT\_FS



Preporuka je da se koristi Gparted aplikacija za formatiranje microSD kartice. Da bi se aplikacija skinula, neophodno je u linux terminalu ukucati sledeće:

```
apt-get install gparted
```

## 10) Kopiranje generisanih fajlova na sd karticu

Na deo microSD kartice sa labelom BOOT potrebno je kopirati sledeće:

- BOOT.bin (mora da se napravi svaki put kada se menja programabilna logika)
- devicetree.dtb (mora da se napravi svaki put kada se menja programabilna logika)
- uImage (samo prvi put kada se instalira linux)

Devicetree.dtb fajl se nalazi u direktorijumu u kome se trenutno nalazimo, odnosno u direktorijumu čija je putanja u ovom slučaju:

```
nikola@ubuntu:~$ cd ~/Downloads/VGA/Top/result/VGA_BRAM_controller.sdk/device_tree_bsp_0/
```

Sa te putanje potrebno je kopirati *devicetree.dtb* fajl na microSD, u particiju sa oznakom BOOT. Kada se to kopira, potrebno je otići u direktorijum koji se nalazi u direktorijumu korak nazad u odnosu na trenutni, odnosno na sledećoj putanji:

```
nikola@ubuntu:~$ cd ~/Downloads/VGA/Top/result/VGA_BRAM_controller.sdk/example_fsbl/
```

I u tom direktorijumu je potrebno ući u *bootimage* direktorijum, u kome se nalazi *BOOT.bin* fajl, i taj fajl treba kopirati u BOOT particiju na microSD kartici.

Poslednje što je potrebno kopirati u BOOT particiju na microSD kartici je *ulmage* fajl koji se nalazi u direktorijumu koji smo skinuli u koraku 2, odnosno nalazi se u direktorijumu sa nazivom *linux-xlnx-zynmp-dt-fixes-for-4.10*. Ući u taj direktorijum, i u njemu ući u direktorijum na putanji *arch/arm/boot*. U tom direktorijumu se nalazi *ulmage* fajl koji je potrebno kopirati na microSD karticu na particiju BOOT.

Poslednje što je potrebno uraditi jeste da se u *ROOT\_fs* particiju na microSD kartici raspakuje linaro korisnički prostor koji se može naći na sledećem linku:

<http://releases.linaro.org/debian/images/developer-armhf/17.02/>

skinuti *linaro-jessie-developer-20161117-32.tar.gz* fajl, i raspakovati ga na microSD kartici korišćenjem sledeće komande:

```
sudo tar xf linaro-jessie-developer-20161117-32.tar.gz --strip-components=1 -C <path>/ROOT_FS/
```

prethodna komanda raspakovaće linaro korisnički prostor U *ROOT\_FS* particiji na microSD kartici (napomena, neophodno je uneti putanju do *ROOT\_FS* !!!).

Nakon što je raspakovan korisnički prostor potrebno je ući u root folder (ukucati *sudo su* ukoliko operativni sistem ne da da se uđe u root, pa onda ući u njega), i u njega kopirati ceo *linux-xlnx-zynmp-dt-fixes-for-4.10* direktorijum koji se može naći na ovom linku:

<https://drive.google.com/drive/folders/11X2BPOAfApdeKQJ6JTd7BffsXt6rf5a4?usp=sharing>

Kada je ovo urađeno jako je bitno ući u *ROOT\_fs* particiju na microSD kartici i ukucati *sync*, i nakon toga ući u BOOT particiju, i u njoj takođe ukucati *sync*.

```
nikola@ubuntu:/media/nikola/ROOT_FS$ sync
```

Kada je sve to urađeno potrebno je još samo unmount-ovati BOOT I ROOT\_FS particije, I kartica je spremna da se ubaci u zbybo-a (**NIPOŠTO NE IZVLAČITI KARTICU PRE NEGO ŠTO SE UNMOUNT-UJU PARTICIJE!**). To se radi sledećom komandom:

```
umount <path>/ROOT_FS
```

```
umount <path>/BOOT
```

```
nikola@ubuntu:~$ umount /media/nikola/ROOT_FS
```