

- Subject : Zybo Linux Porting 14 Steps
- Created date : 2017. 05. 27
- Recently modified date : -
- Creator : Choi Jun-Ho
- Contact : peluza6332@gmail.com
- Worked dates to make working of subject : 2017. 05. 27
- Documentation status : **finished**
- Summary :  
Zybo 개발 보드에 linux 4.4.32 포팅하는 방법
- Contents
  - 12 steps to linux porting on zybo

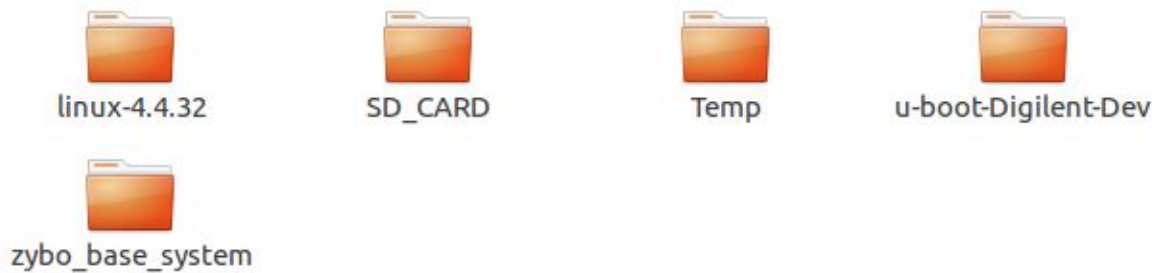


# ● 12 steps to linux porting on zybo

## Step 0 : Required Tools

Git, Vivado(+ SDK)

## Step 1: Download Sources for Porting



**linux-4.4.32** : `sudo wget https://www.kernel.org/pub/linux/kernel/v4.x/linux-4.4.32.tar.xz`

**u-boot-Digilent-Dev** : `git clone -b master-next`

`https://github.com/DigilentInc/u-boot-Digilent-Dev.git`

**Linux-Digilent-Dev** : `git clone -b master-next`

`https://github.com/DigilentInc/Linux-Digilent-Dev.git`

**zybo\_base\_system** : `sudo wget`

`https://reference.digilentinc.com/_media/reference/programmable-logic/zybo/zybo_base_system.zip`

**SD\_CARD** : 생성한 포팅에 필요한 파일들 놓아두는 임시 디렉토리

**Temp** : rsync로 파일 싱크할 때 mount할 임시 디렉토리

## Step 2: setting Environment

```
peluza@peluza-B85H3-M7:~$ subl ./bashrc
```

```
# About Zybo Linux Porting
export PATH=$PATH:/usr/local/bin
export ARCH=arm
export CROSS_COMPILE=arm-xilinx-linux-gnueabi-
export PATH=/home/peluza/Zybo_Porting_4.4.32/u-boot-Digilent-Dev/tools:$PATH
```

OR

```
peluza@peluza-B85H3-M7:~$ export PATH=$PATH:/usr/local/bin
peluza@peluza-B85H3-M7:~$ export ARCH=arm
peluza@peluza-B85H3-M7:~$ export CROSS_COMPILE=arm-xilinx-linux-gnueabi-
peluza@peluza-B85H3-M7:~$ export PATH=/home/peluza/Zybo_Porting_4.4.32/u-boot-Digilent-Dev/tools:$PATH
```

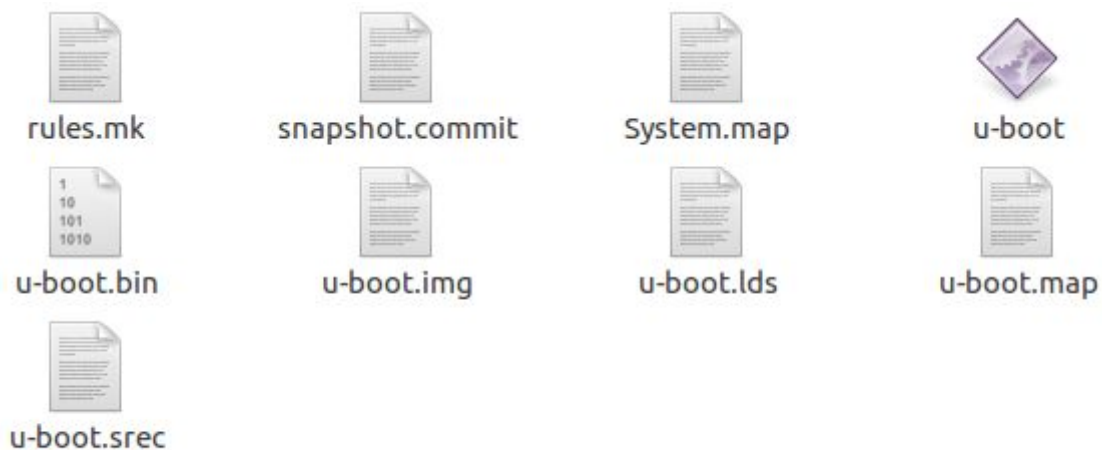


## Step 3: Compile U-Boot

우리는 zybo를 쓰므로 zynq\_zybo.h를 통한 컴파일을 해야 한다.  
make zynq\_zybo\_config를 설정한다.  
make를 통해 u-boot를 컴파일한다.

```
peluza@peluza-B85H3-M7: ~/Zybo_Porting_4.4.32/u-boot-Digilent-Dev
peluza@peluza-B85H3-M7:~$ cd Zybo_Porting_4.4.32/u-boot-Digilent-Dev/
peluza@peluza-B85H3-M7:~/Zybo_Porting_4.4.32/u-boot-Digilent-Dev$ make zynq_zybo_config
Configuring for zynq_zybo board...
peluza@peluza-B85H3-M7:~/Zybo_Porting_4.4.32/u-boot-Digilent-Dev$ make
Generating include/autoconf.mk
Generating include/autoconf.mk.dep
Generating include/spl-autoconf.mk
Generating include/tpl-autoconf.mk
arm-xilinx-linux-gnueabi-gcc -DDO_DEPS_ONLY \
-o -Os -ffunction-sections -fdata-sections -fno-common -ffixed-r9 -msoft-float -fno-sta
```

u-boot directory에



와 같이 u-boot가 컴파일 됐다.

```
peluza@peluza-B85H3-M7:~/Zybo_Porting_4.4.32/u-boot-Digilent-Dev$ cp ./u-boot ./SD_CARD/u-boot.elf
```

SD CARD 임시 directory에 u-boot.elf라는 이름으로 넣는다.



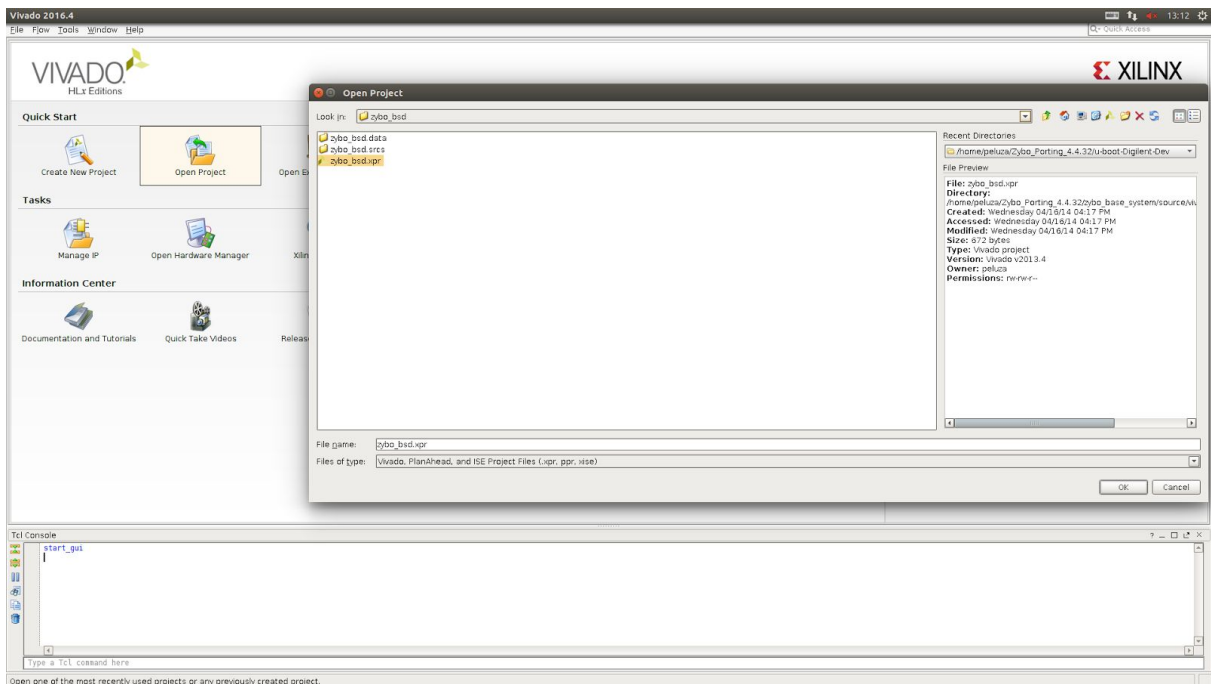
## Step 4: Open the Base System Design

vivado로 BSP의 .xpr을 연다.

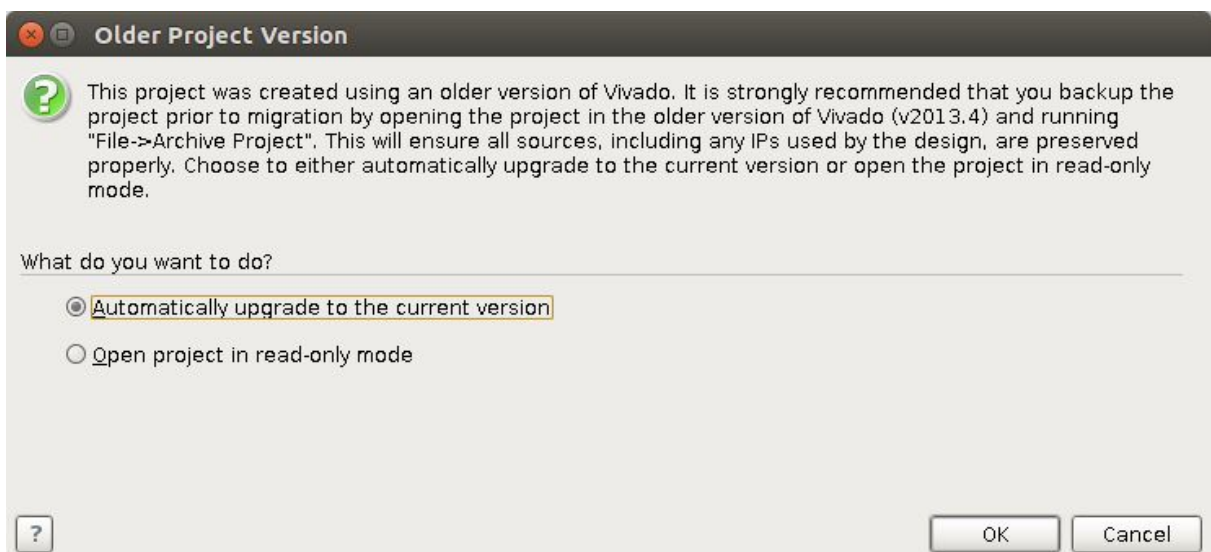
```
peluza@peluza-B85H3-M7:~/Zybo_Porting_4.4.32/u-boot-Digilent-Dev$ vivado
awk: symbol lookup error: awk: undefined symbol: mpfr_z_sub
```

```
***** Vivado v2016.4 (64-bit)
**** SW Build 1756540 on Mon Jan 23 19:11:19 MST 2017
**** IP Build 1755317 on Mon Jan 23 20:30:07 MST 2017
** Copyright 1986-2016 Xilinx, Inc. All Rights Reserved.
```

```
start_gui
```



열면 다음과 같은 메시지가 뜬다.

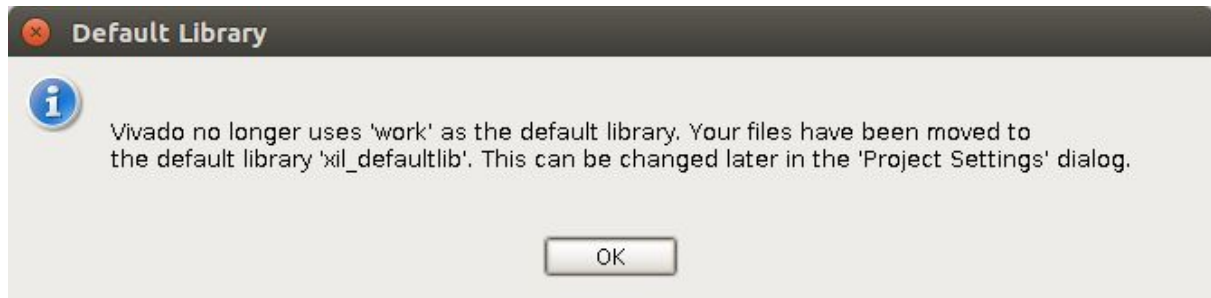


자동으로 업데이트 한다.





다른 창 하나가 뜬다.

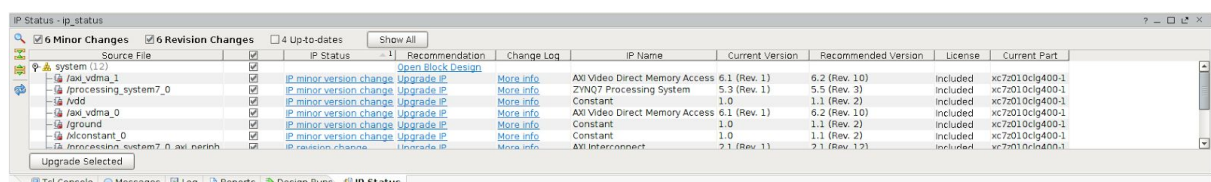


확인을 누른다.



역시나 IP 상태를 리포트 한다.

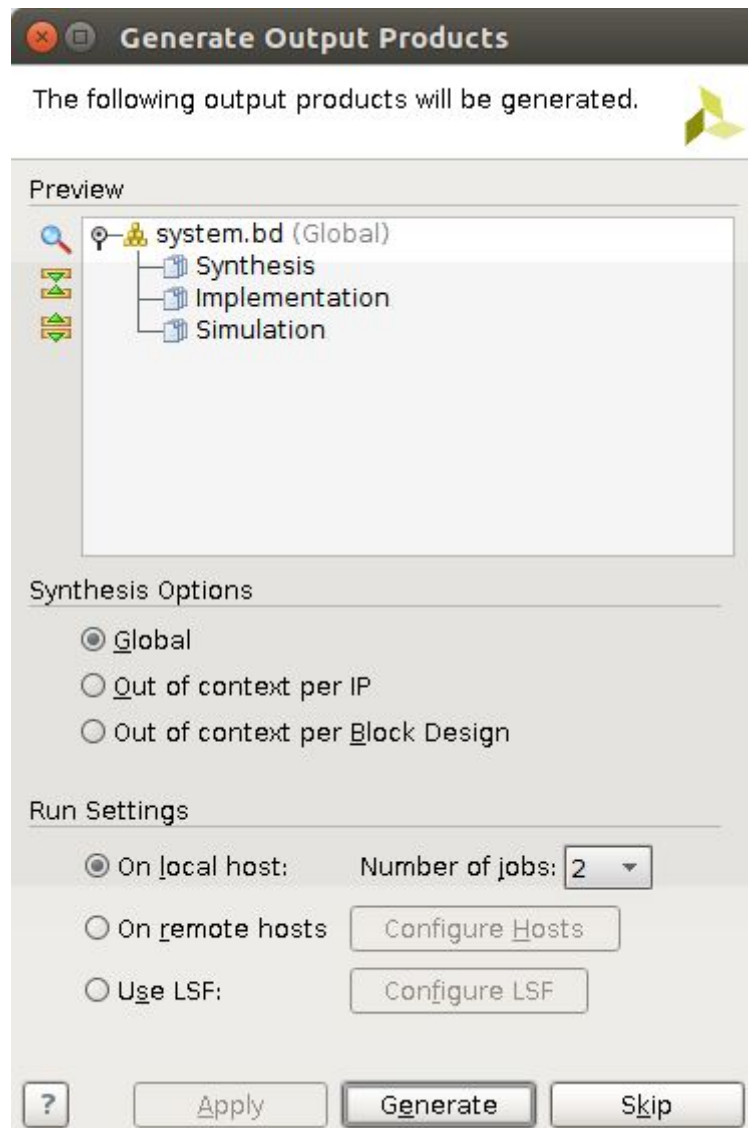
하단의 IP 상태 탭에서 IP 상태들이 나온다.



Upgrade Selected를 눌러서 IP들을 업그레이드 한다.

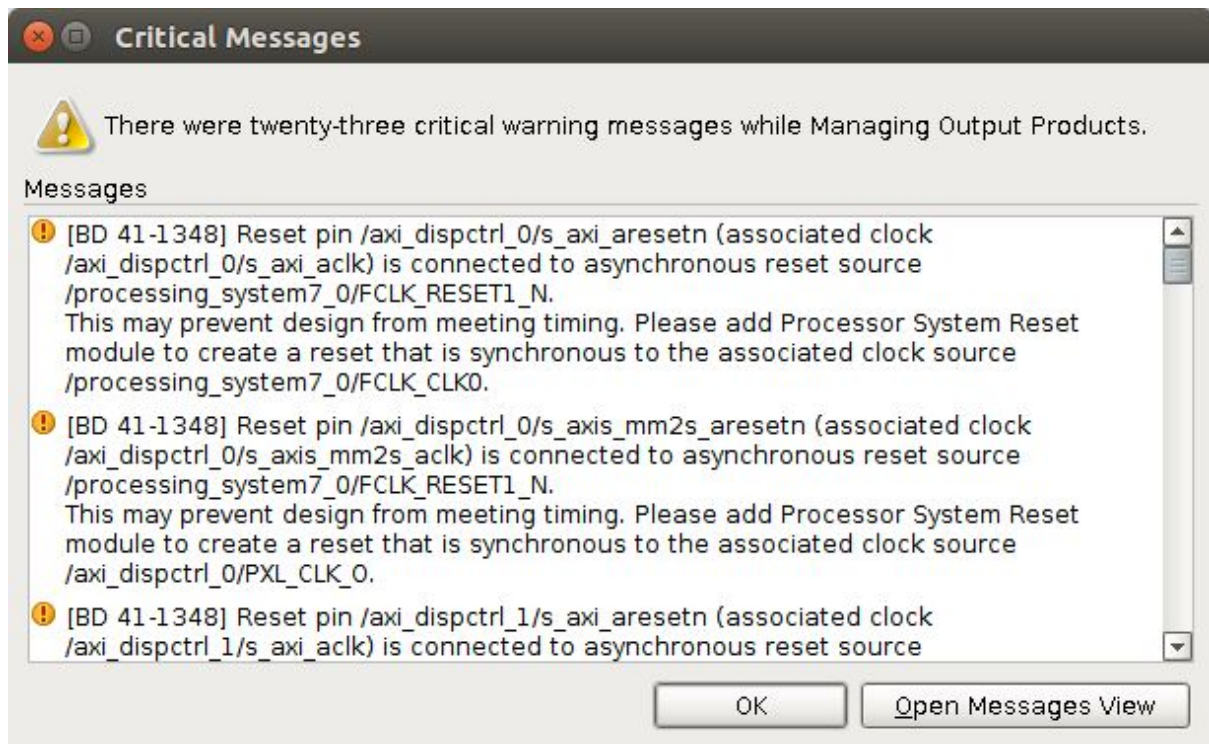


업그레이드 하고 나면 다음과 같이 Out put Products를 생성할 수 있는데

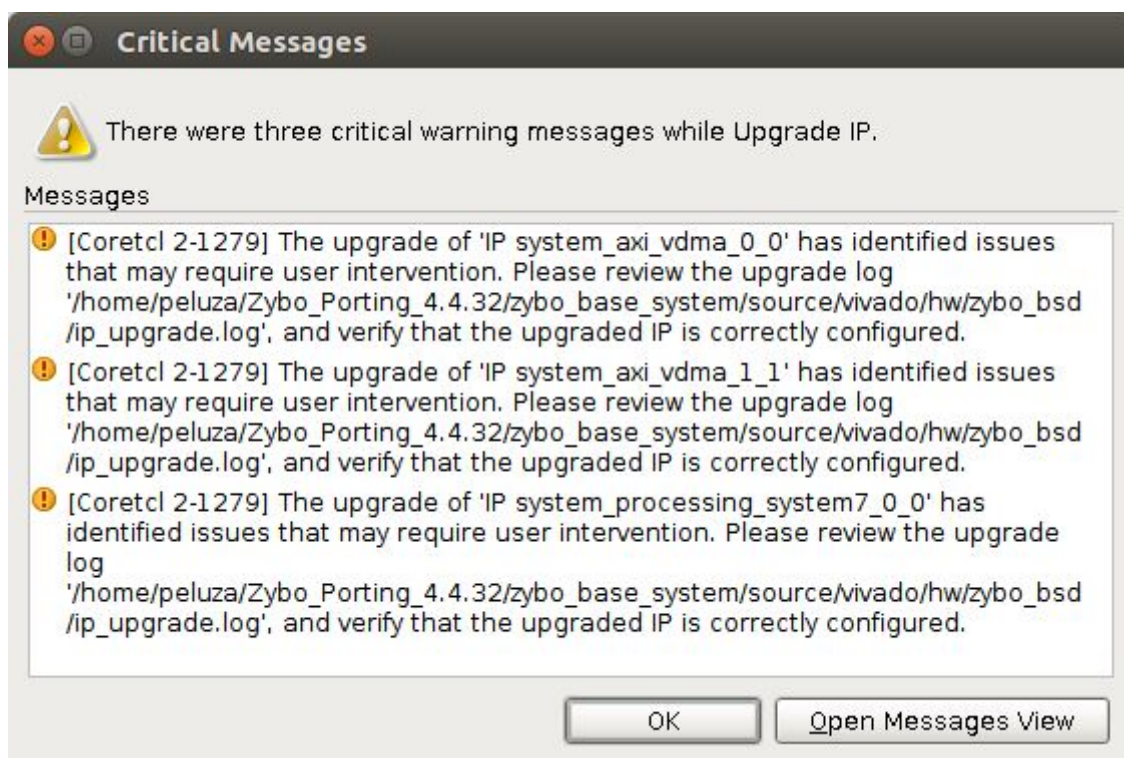


Global에 맞추고 생성한다.





무시해도 된다. OK 누른다.

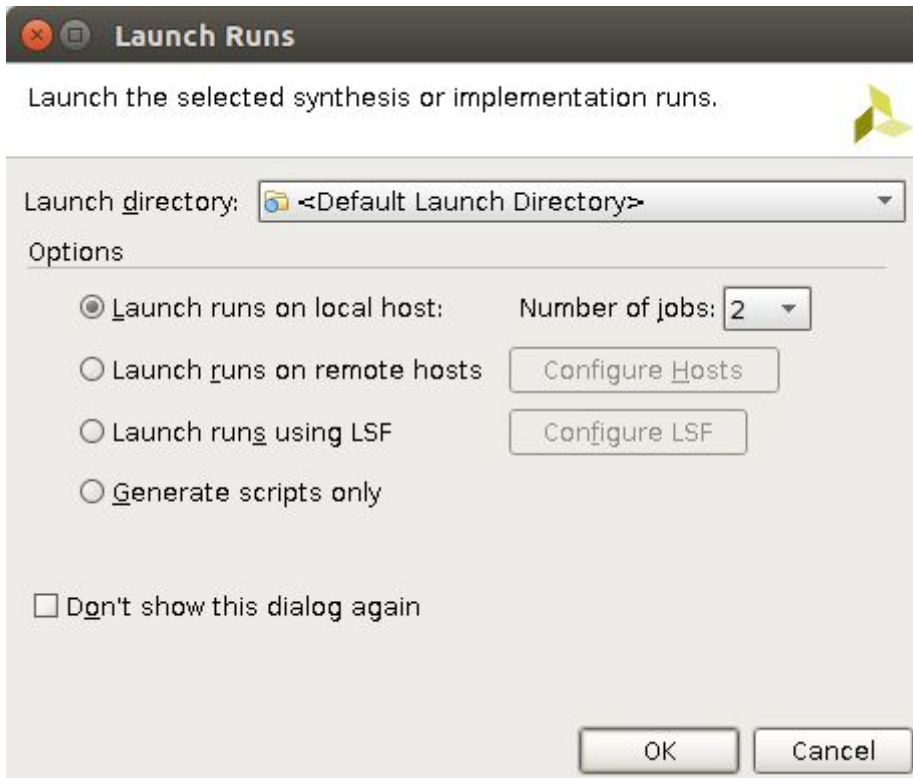


마찬가지 OK 누른다.





자 이제 Bitstream을 만든다. Generate Bitstream을 누른다.



OK 누른 후

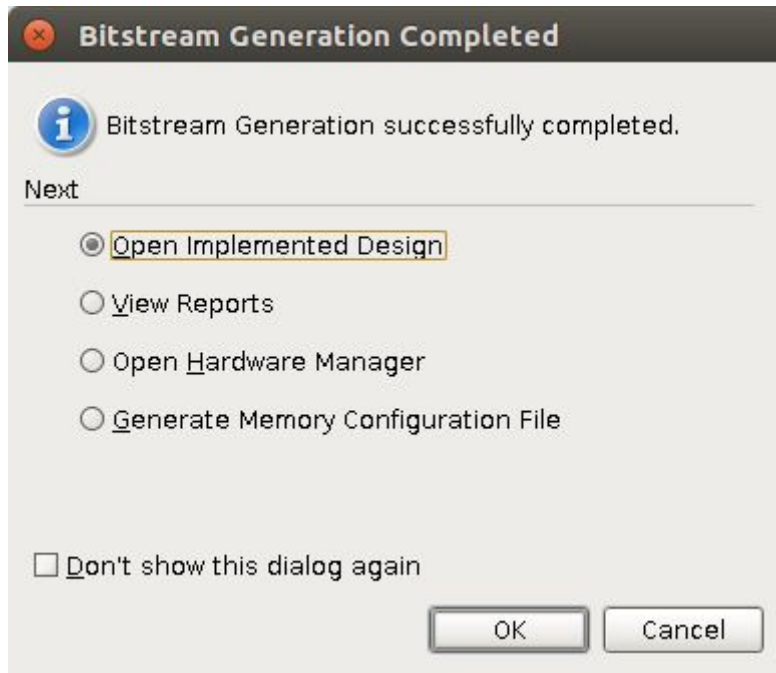
아직 Synthesis와 Implementation이 되어 있지 않아 먼저 진행한다고 하면 OK 한다.



환경에 따라 시간이 오래 걸릴 수 있다. 기다린다.





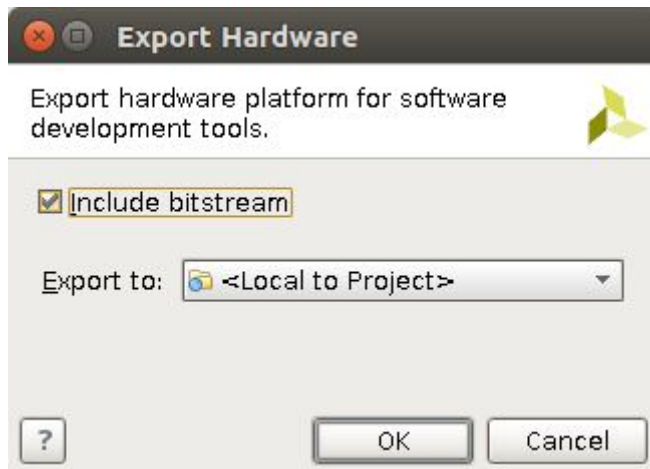


구현된 디자인을 열 필요가 없으므로 Cancel한다.

## Step 5: Export Hardware for SDK

hardware 정보와 bitstream을 SDK로 Export 한다.

File -> Export -> Export Hardware를 클릭 후



에서 bitstream을 include한 상태로 확인한다.

콘솔 창에서 zybo\_bsd.sdk 디렉토리가 잘 생성됐고

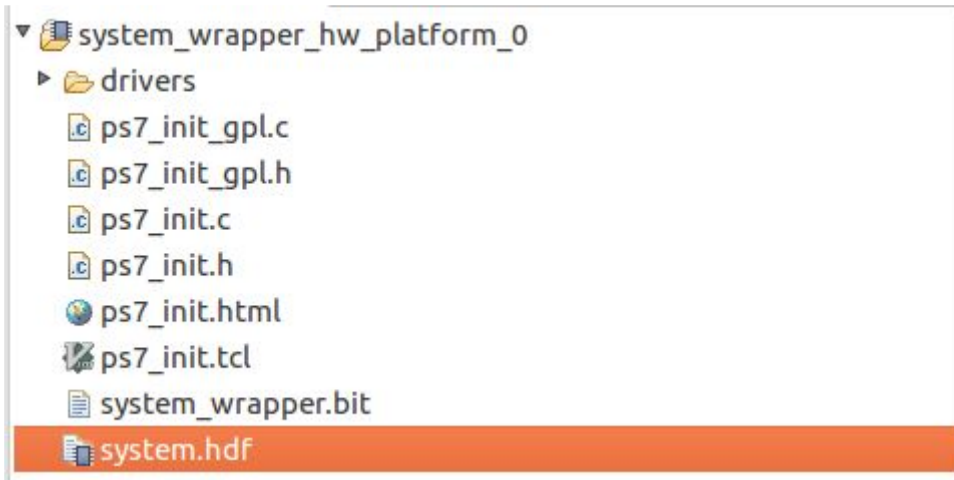
system\_wrapper.sysdef 파일 및 system\_wrapper.hdf 파일이 잘 복사됐는지 확인한다.

(hardware definition file 파일이 중요하다.)



## Step 6: Create FSBL Project

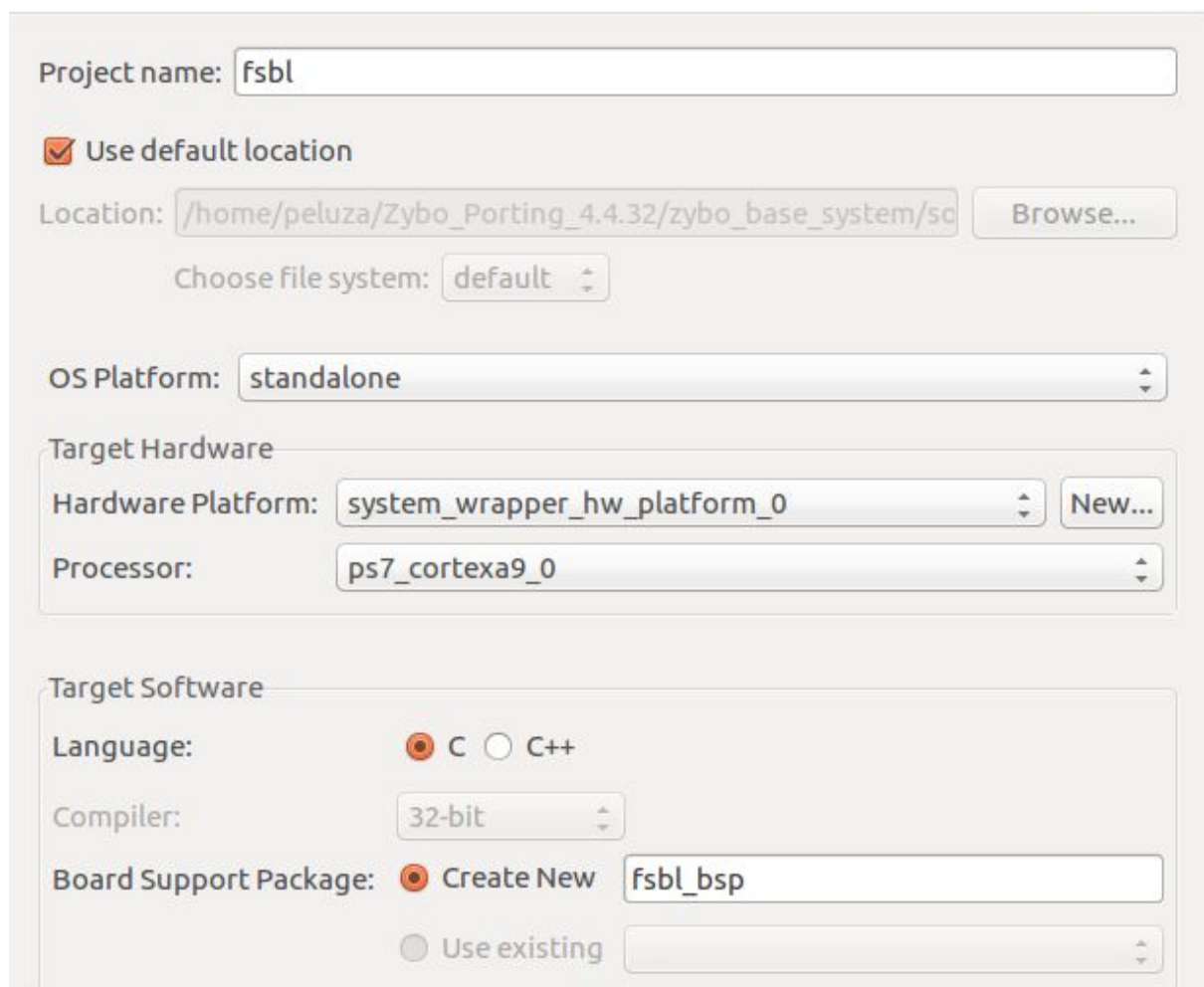
File -> Launch SDK를 눌러 SDK를 켜다.



다음과 같이 SDK 안에서 플랫폼이 잘 잡히는지 확인한다.

(그렇지 않다면, Vivado에서 Export Hardware 하는 과정에서 문제가 있거나 SDK에서 플랫폼을 인식하는데 문제가 있는 것일 수 있다.)

새 프로젝트를 누르고 이름을 fsbl로 한다.



Project name:

☒ Use default location

Location:

Choose file system:

OS Platform:

Target Hardware

Hardware Platform:

Processor:

Target Software

Language: ☒ C ☐ C++

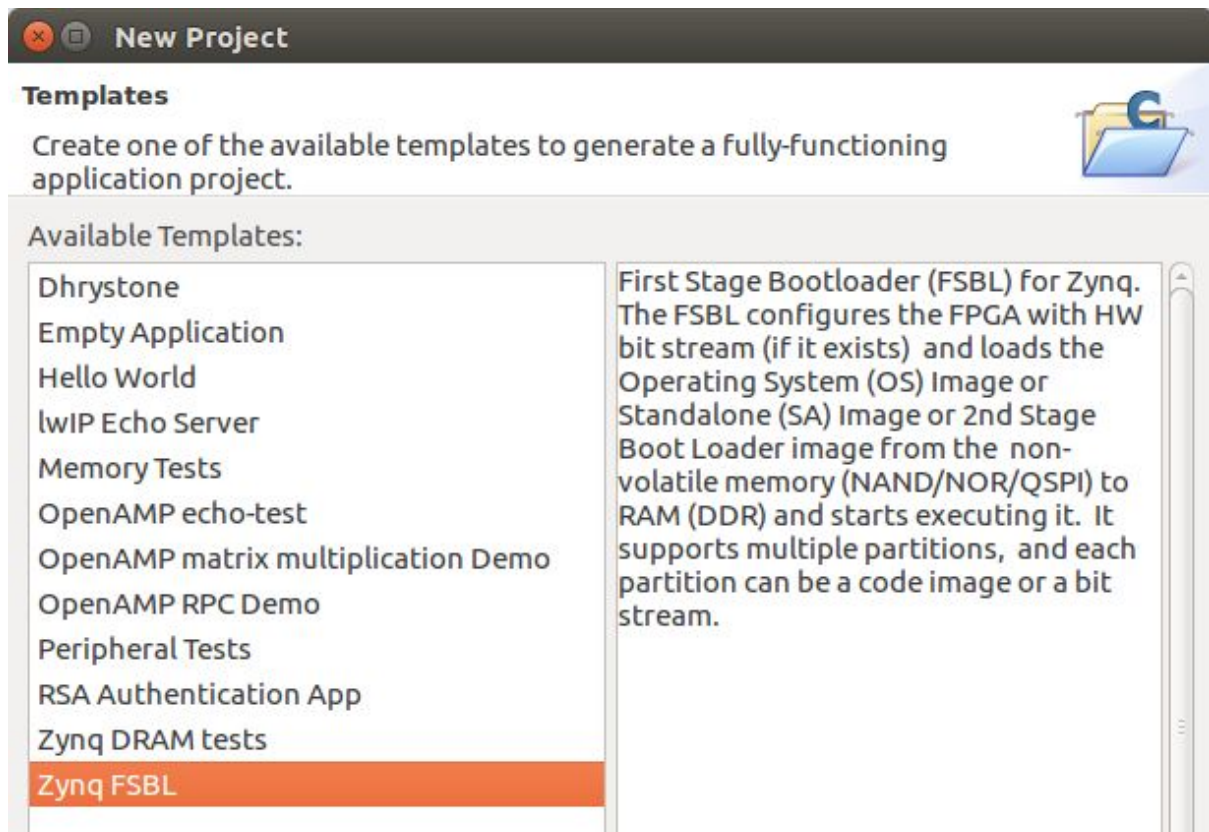
Compiler:

Board Support Package: ☒ Create New

☐ Use existing



Next를 누르고 Zynq FSBL Template을 선택하고 마친다.



잘 만들어 졌는지 확인한다.

## Step 7: FSBL Hook

```
peluza@peluza-B85H3-M7:~/Zybo_Porting_4.4.32/u-boot-Digilent-Dev$ cd ../zybo_base_system/source/vivado/SDK/fsbl/
peluza@peluza-B85H3-M7:~/Zybo_Porting_4.4.32/zybo_base_system/source/vivado/SDK/fsbl$ ls
fsbl_hooks.c
```

다음 경로에서 fsbl\_hook.c를 확인한다.

이것을 fsbl 프로젝트 source 디렉토리의 동명의 c파일에 덮어쓴다.

덮어쓴 후 Project -> Clean 및 Build All 해서 프로젝트를 빌드한다.



## Step 8: Create Boot Image

fsbl.elf, system\_wrapper.bit, u-boot.elf를 이용하여 BOOT.bin 생성

fsbl.elf은 SDK fsbl 프로젝트 빌드하면 생성된다.

system\_wrapper.bit은 vivado에서 bitstream을 인클루트하면서 generate 해서 생성한다.

u-boot.elf는 u-boot를 컴파일해서 얻는다.

**Add partition**

Add new boot image partition

Add new boot image partition

File path: /home/peluza/Zybo\_Porting\_4.4.32/SD\_CARD/u-boot.elf Browse...

Partition type: datafile

Authentication: none Encryption: none

Checksum: none

Presign: Browse...

Other

Alignment:  Offset:

Reserve:  Load:

Startup:

? Cancel OK

u-boot.elf를 SD\_CARD 디렉토리에서 찾아 추가해주고





**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/peluza/Zybo\_Porting\_4.4.32/SD\_CARD/fsbl.bif Browse...

UDF data:  Browse...

☐ Split Output format: BIN

Output path: /home/peluza/Zybo\_Porting\_4.4.32/SD\_CARD/BOOT.bin Browse...

**Boot image partitions**

File path
(bootloader) /home/peluza/Zybo_Porting_4.4.32/zybo_base_system/source/vivado/hw/zybo_bsd/zybo_bsd.sdk/fs
/home/peluza/Zybo_Porting_4.4.32/zybo_base_system/source/vivado/hw/zybo_bsd/zybo_bsd.sdk/system_wrappe
/home/peluza/Zybo_Porting_4.4.32/SD_CARD/u-boot.elf

? Preview BIF Changes Cancel Create Image

다음과 같이 BOOT.bin 이미지를 만들 준비를 마친 후 Create Image를 누른다.

다음과 같은 메시지와

13:54:02 INFO : Bootgen command execution is done.

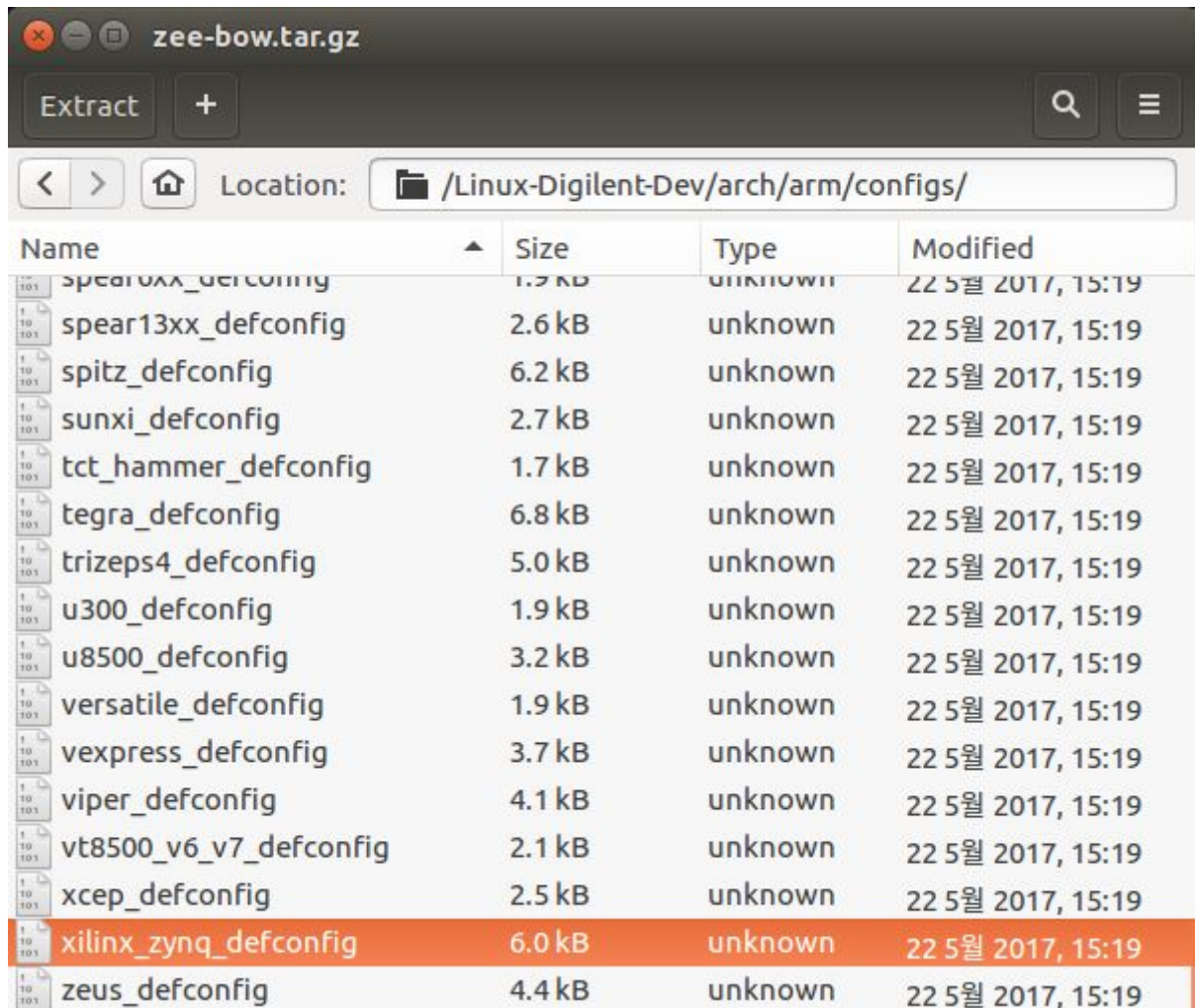


파일을 확인한다. (output.bif = fsbl.bif)



## Step 9: Configure the Kernel & Compile the Linux Kernel

4.4.32에는 xilinx\_zynq\_defconfig가 없으므로 Digilent에서 제공하는 Linux Kernel Source에서 가져온다.



4.4.32의 해당 경로에 넣어준 후

```
peluza@peluza-B85H3-M7:~/Zybo_Porting_4.4.32/linux-4.4.32$ make xilinx_zynq_defconfig
#
# configuration written to .config
#
peluza@peluza-B85H3-M7:~/Zybo_Porting_4.4.32/linux-4.4.32$ make UIIMAGE_LOADADDR=0x00008000 uImage
```

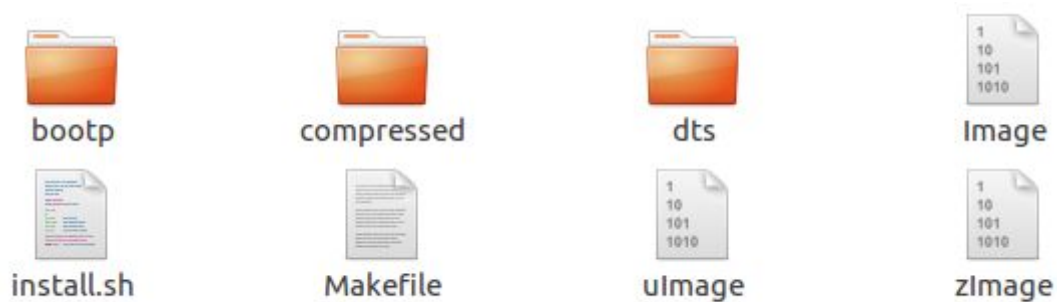
옮긴 파일로 설정한 후 ulmage로 빌드한다.  
컴파일이 될때까지 기다린다.



## Step 10: Make Uimage

```
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
Kernel: arch/arm/boot/Image is ready
Kernel: arch/arm/boot/zImage is ready
UIMAGE arch/arm/boot/uImage
Image Name:   Linux-4.4.32-xilinx
Created:      Sat May 27 14:07:25 2017
Image Type:   ARM Linux Kernel Image (uncompressed)
Data Size:    3656192 Bytes = 3570.50 kB = 3.49 MB
Load Address: 00008000
Entry Point:  00008000
Image arch/arm/boot/uImage is ready
peluza@peluza-B85H3-M7:~/Zybo_Porting_4.4.32/linux-4.4.32$
```

위 과정에서 만들어진 uImage를 확인한다. arch/arm/boot/uImage



uImage를 SD\_CARD 디렉토리로 복사한다.



## Step 11: Make Uramdisk

파일 시스템을 만든다.

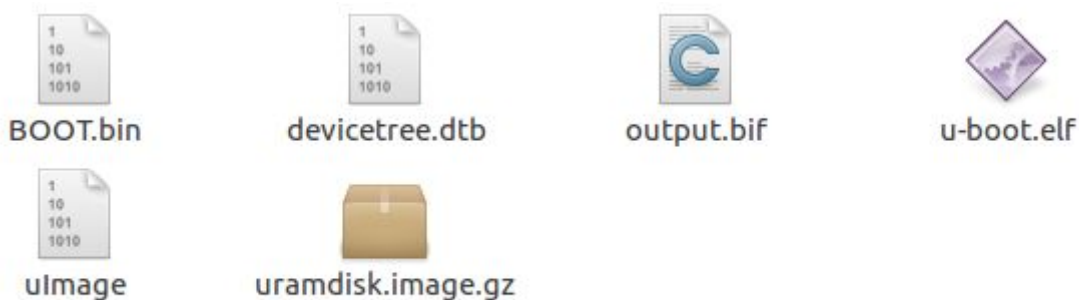
```
peluza@peluza-B85H3-M7:~/Zybo_Porting_4.4.32$ ./u-boot-Digilent-Dev/tools/mkimage -A arm -T ramdisk -C gzip -d ./arm_randisk.image.gz urandisk.image.gz
Image Name:
Created:      Sat May 27 14:19:03 2017
Image Type:   ARM Linux RAMDisk Image (gzip compressed)
Data Size:    5309954 Bytes = 5185.50 kB = 5.06 MB
Load Address: 00000000
Entry Point:  00000000
peluza@peluza-B85H3-M7:~/Zybo_Porting_4.4.32$
```

## Step 12: Generate DTB File

zynq-zybo의 device tree source를 컴파일해서 device tree blob을 만든 후 그것을 사용한다.  
(같은 경로에서 만들어진 .dtb 파일을 SD\_CARD로 복사한다.)

```
peluza@peluza-B85H3-M7:~/Embedded_Linux_Zybo/linux-4.4.32$ make zynq-zybo.dtb
DTC          arch/arm/boot/dts/zynq-zybo.dtb
```

## Step 13: Copy File Image to SD Card



BOOT.bin, ulimage, devicetree.dtb, uramdisk.image.gz를 SD Card로 복사한다.

## Step 14: Booting Linux from SD Card

```
Booting Linux on physical CPU 0x0
Linux version 4.4.32-xilinx (peluza@peluza-B85H3-M7) (gcc version 4.9.2 (Sourcery CodeBench Lite 2015.05-17) ) #1 SMP PREEMPT Sat May 27 14:07:21 KST 2017
CPU: ARMv7 Processor [413fc090] revision 0 (ARMv7), cr=18c5387d
CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
Machine model: Zynq ZYBO Development Board
cma: Reserved 16 MiB at 0x1e400000
Memory policy: Data cache writealloc
PERCPU: Embedded 12 pages/cpu @5fbcf000 s19328 r8192 d21632 u49152
Built 1 zonelists in Zone order, mobility grouping on. Total pages: 130048
Kernel command line: earlyprintk
PID hash table entries: 2048 (order: 1, 8192 bytes)
Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
Memory: 490652K/524288K available (4931K kernel code, 188K rwddata, 1740K rodata, 256K init, 223K bss, 17252K reserved, 16384K cma-reserved, 0K highmem)
```

