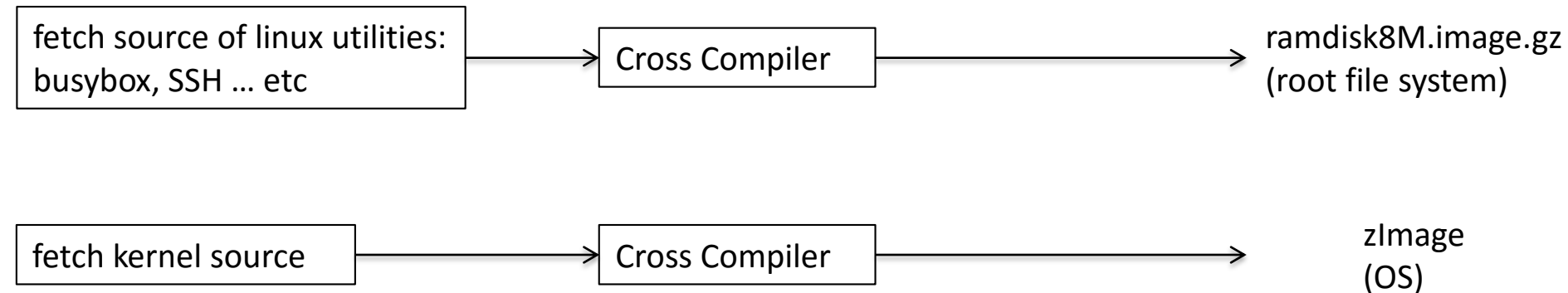
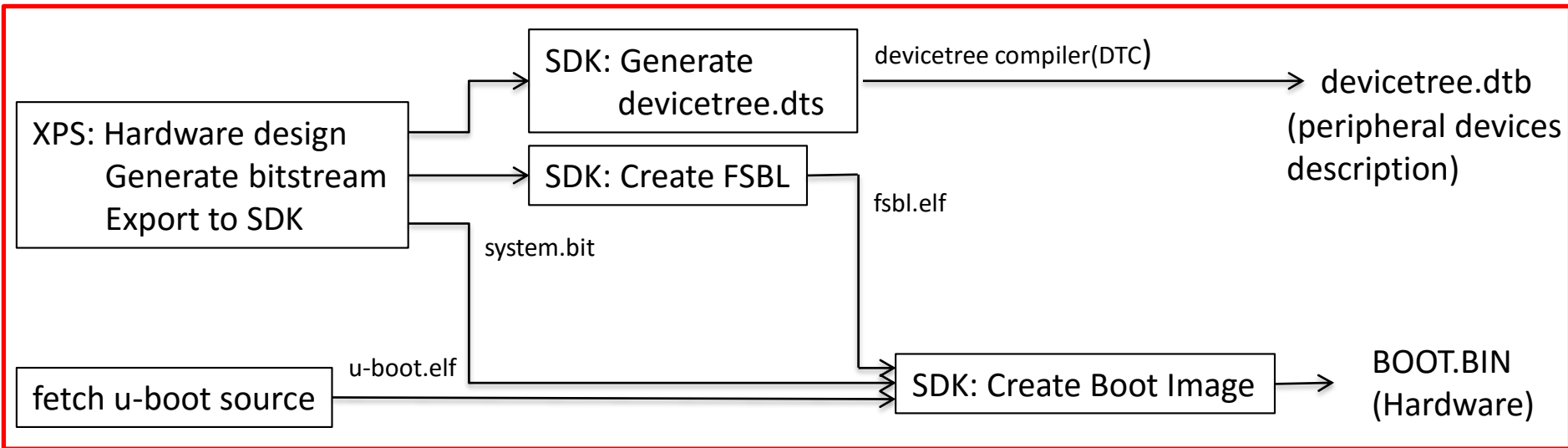


# ZedBoard Lab 4

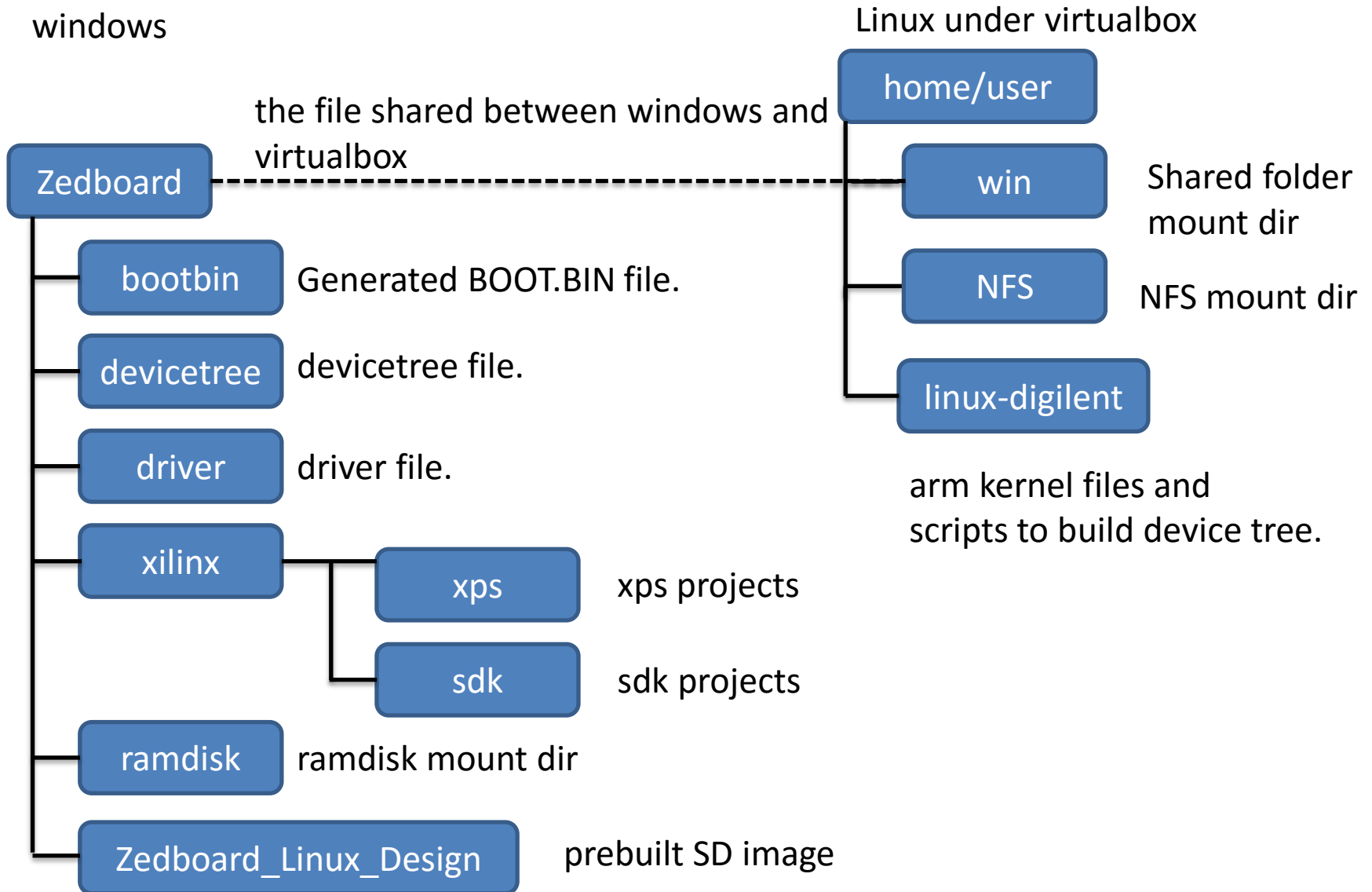
## Hardware Design

Chun-Chen Tu

[timtu@umich.edu](mailto:timtu@umich.edu)



# File placement



# Outline

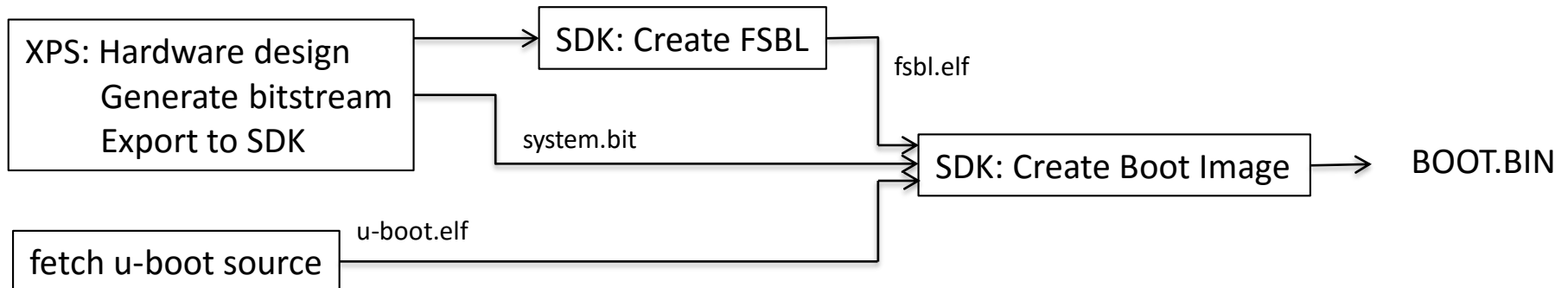
- Why don't we use files in prebuilt SD image?
  - The prebuilt SD image contain some other designs.
  - We need to build our own hardware design.
- In this slide, we will focus on building an empty hardware design.

# Requirement

- Xilinx EDK tools: 14.4 version or later  
(Note: 14.2 cannot generate devicetree files and occurs some problems when creating new project)
  - Xilinx Platform Studio (xps)
  - Xilinx Software Development Kit (sdk)
- Linux environment: Please refer to Lab 0. You'll need the followings:
  - Device tree compiler(DTC): This will appear once you've compiled kernel image.
  - Cross compiler
- My platform:
  - Win 7 with i7 cpu 32G ram.
  - Ubuntu 12.04 64-bit server version on virtualbox, single core and 8G ram

# Workflow - generate BOOT.BIN

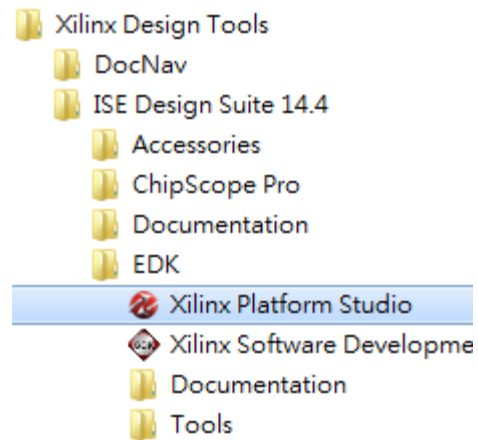
- In XPS
  - Create your design in XPS
  - Export to SDK: get **system.bit**
- In SDK
  - Generate (or recompile) FSBL project: get **fsbl.elf**
  - Create BOOT.BIN from **system.bit**, **fsbl.elf** we just generated and **u-boot.elf** (under boot\_image) from prebuilt files.



# XPS create a new project

Under Xilinx Design Tools->ISE Design Suite 14.4->EDK->Xilinx Platform Studio

Note: Another way is to open XPS through PlanAhead.



Base System Builder -- AXI flow

### Board and System Selection

Select a target development board and a System Template.

Board

☒ Create a System for the Following Development Board (Pre-selected Device Info)

Board Vendor: Avnet Board Name: ZedBoard Zynq Evaluation and Development Board Revision: C

☐ Create a System for a Custom Board

Board Configuration

Architecture: zynq Device: xc7z020 Reference Clock Frequency: 100.00 MHz

Package: clg484 Speed Grade: -1 Reset Polarity: Active High ☐ Use Stepping

Select a System

Zynq Processing System 7

System Information

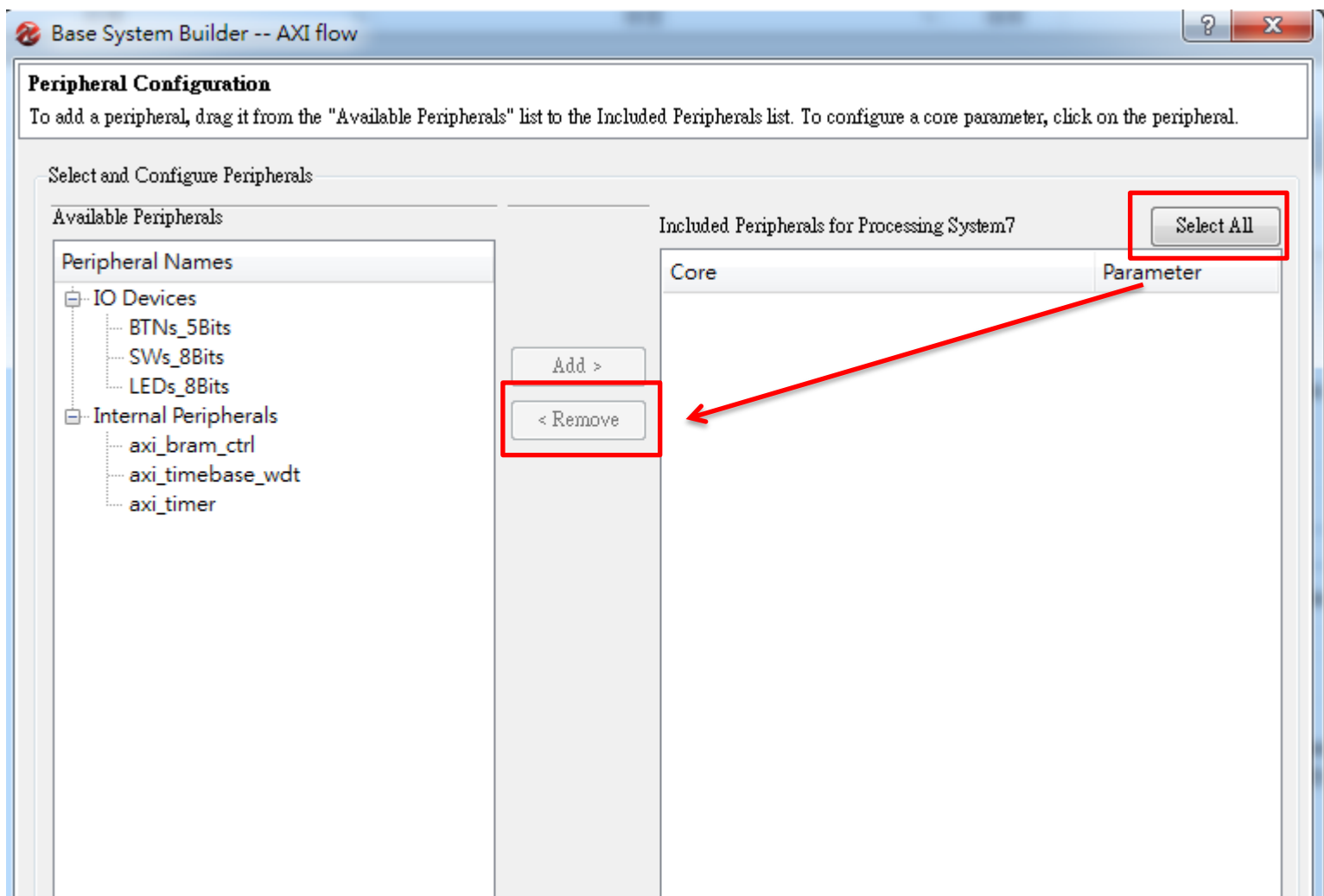
This system consists of Processing System 7 with peripheral GPIOs. Peripherals are connected on AXI interconnect. Click Next to modify the default system.

Board Vendor: Avnet

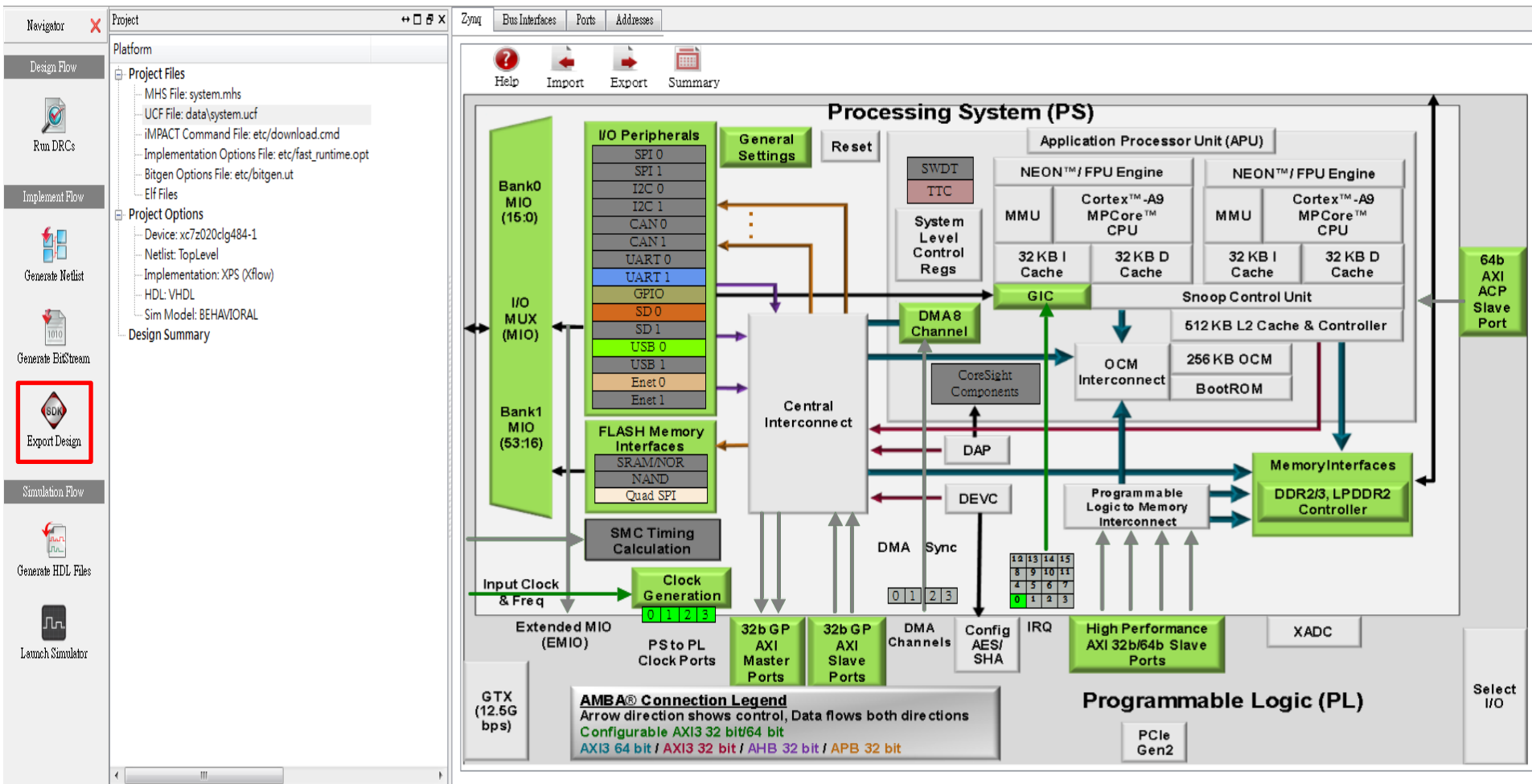
Board Name: Zedboard Zynq

Next

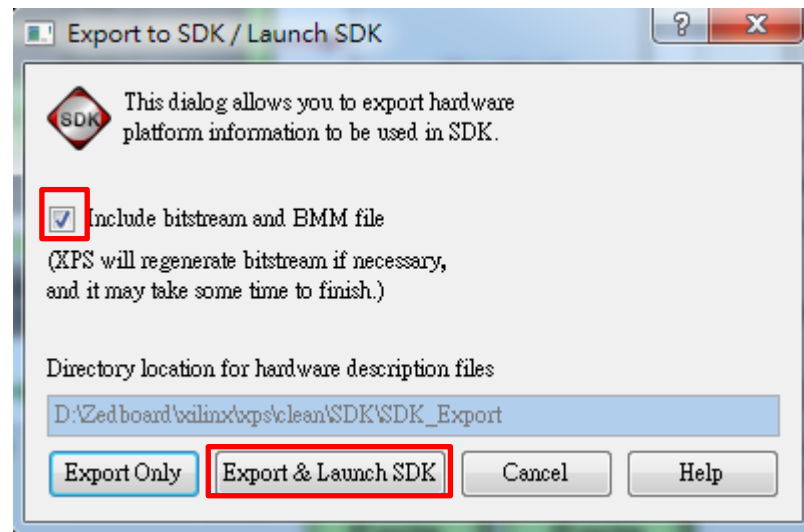




Press *Select All*  
And then *Remove*  
*Finish*



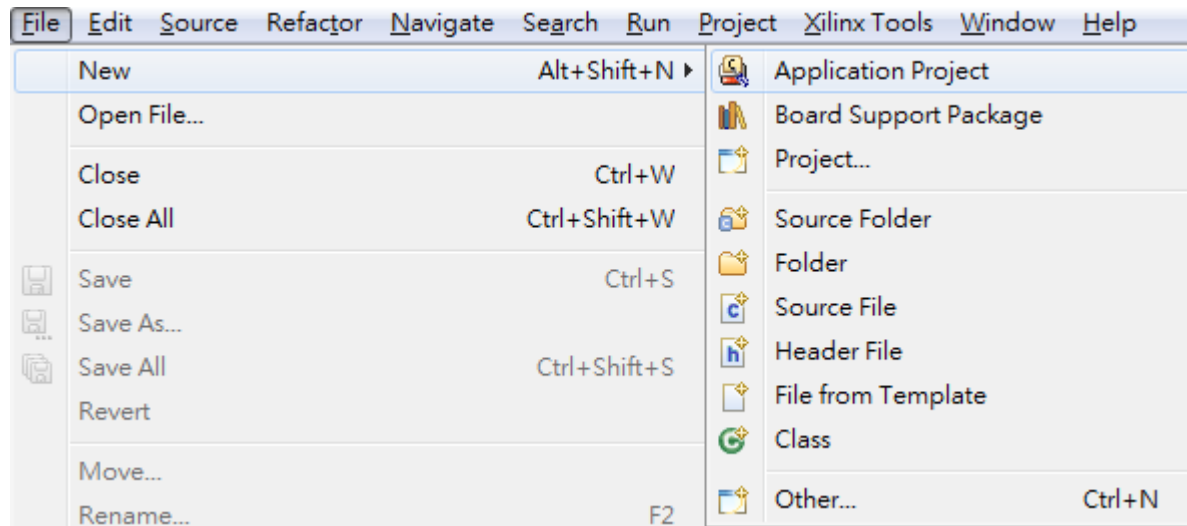
A control panel will show up. You can make some modification. But how to add something is not the topic now. Click *Export Design* to generate an empty bit stream and export it to SDK as well.



Check Include bitstream and BMM file  
and then *Export & Launch SDK for convenience*

XPS will start to generate bit stream. For an empty design, this should be several minutes. SDK will automatically activate after bit stream generate. And the hardware information will also export to SDK.

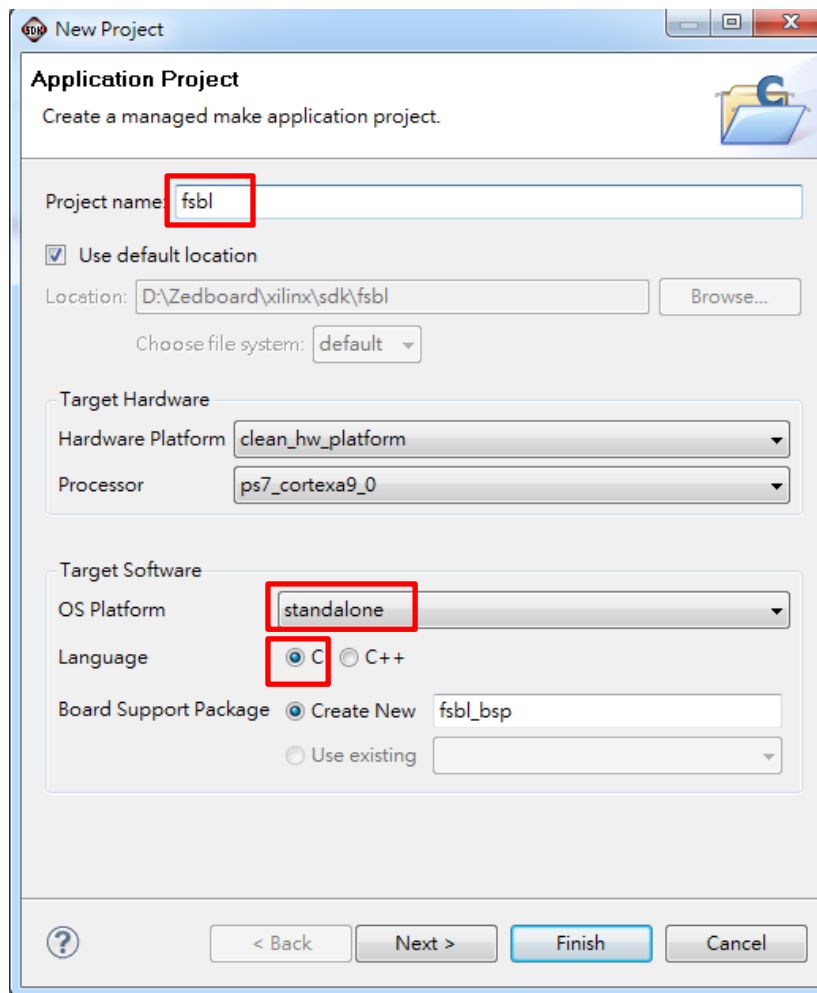
The [system.bit](#) will occur under Zedboard\xilinx\xps\clean\SDK\SDK\_Export\hw  
For the first time you launch SDK, you will need to specify workspace.  
As for me: Zedboard\xilinx\sdk



Now we want to generate **fsbl.elf**

In SDK

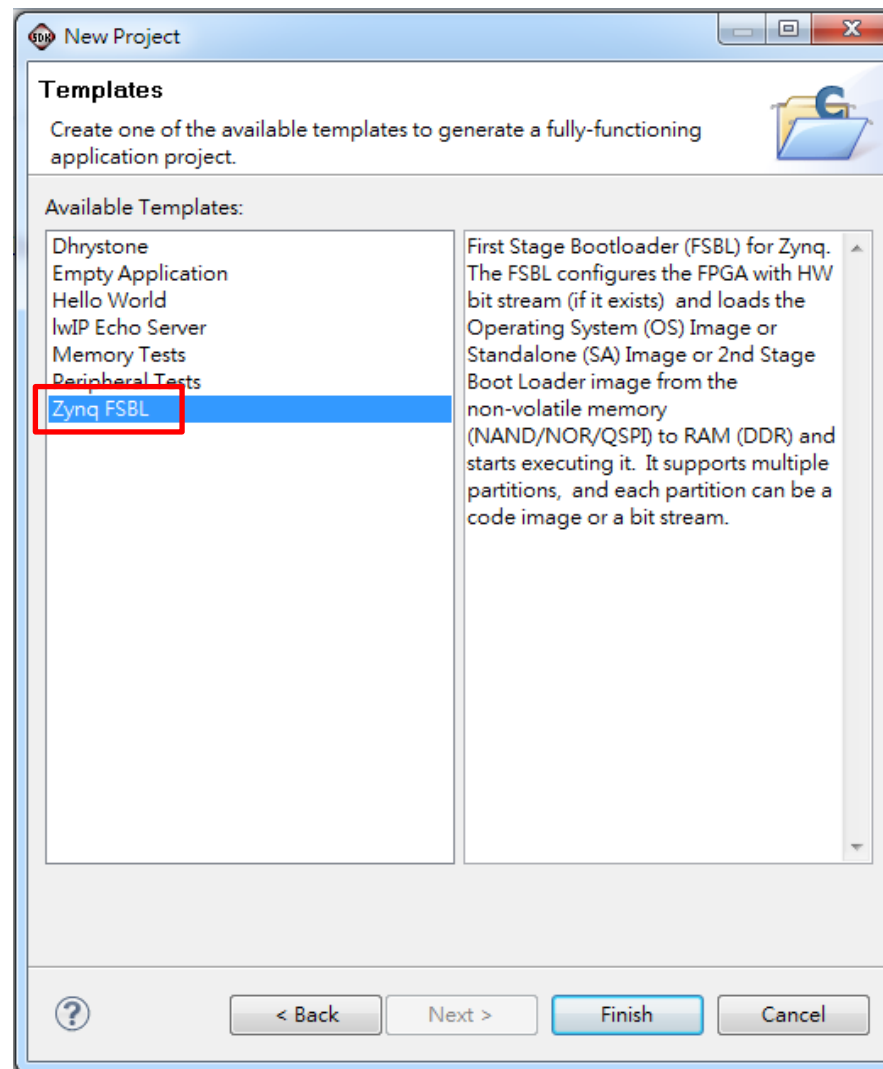
File->New->Application Project



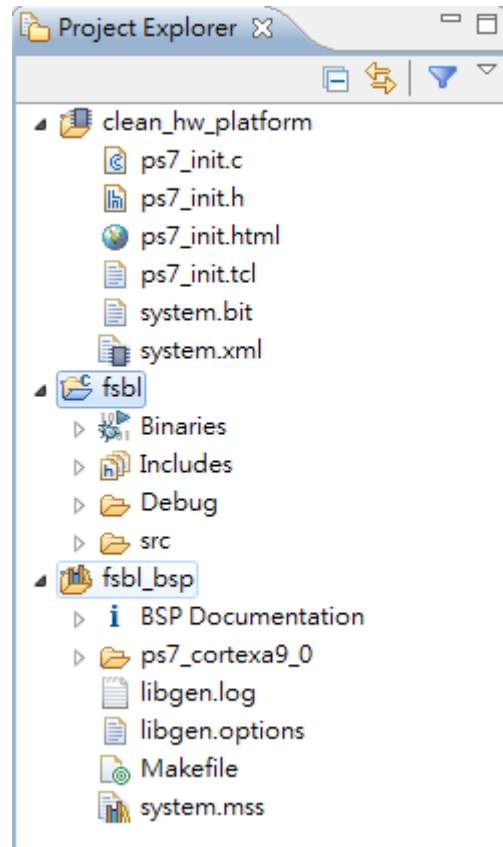
Give the new project a name, like *fsbl*

Choose *standalone* for OS Platform and *C* for Language

Remain other settings as default and then *Next*

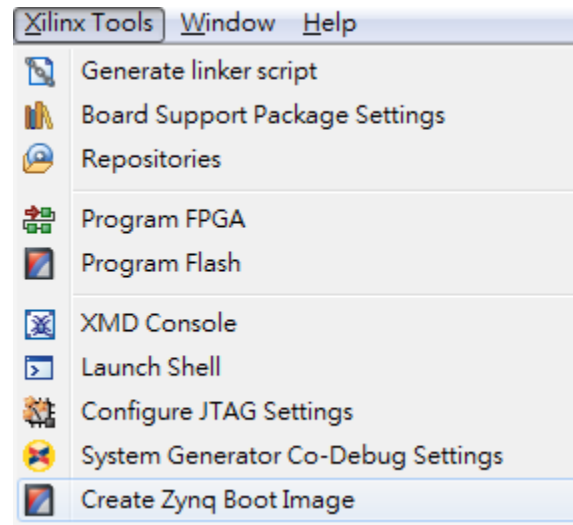


Choose *Zynq FSBL* as templates  
and then *Finish*



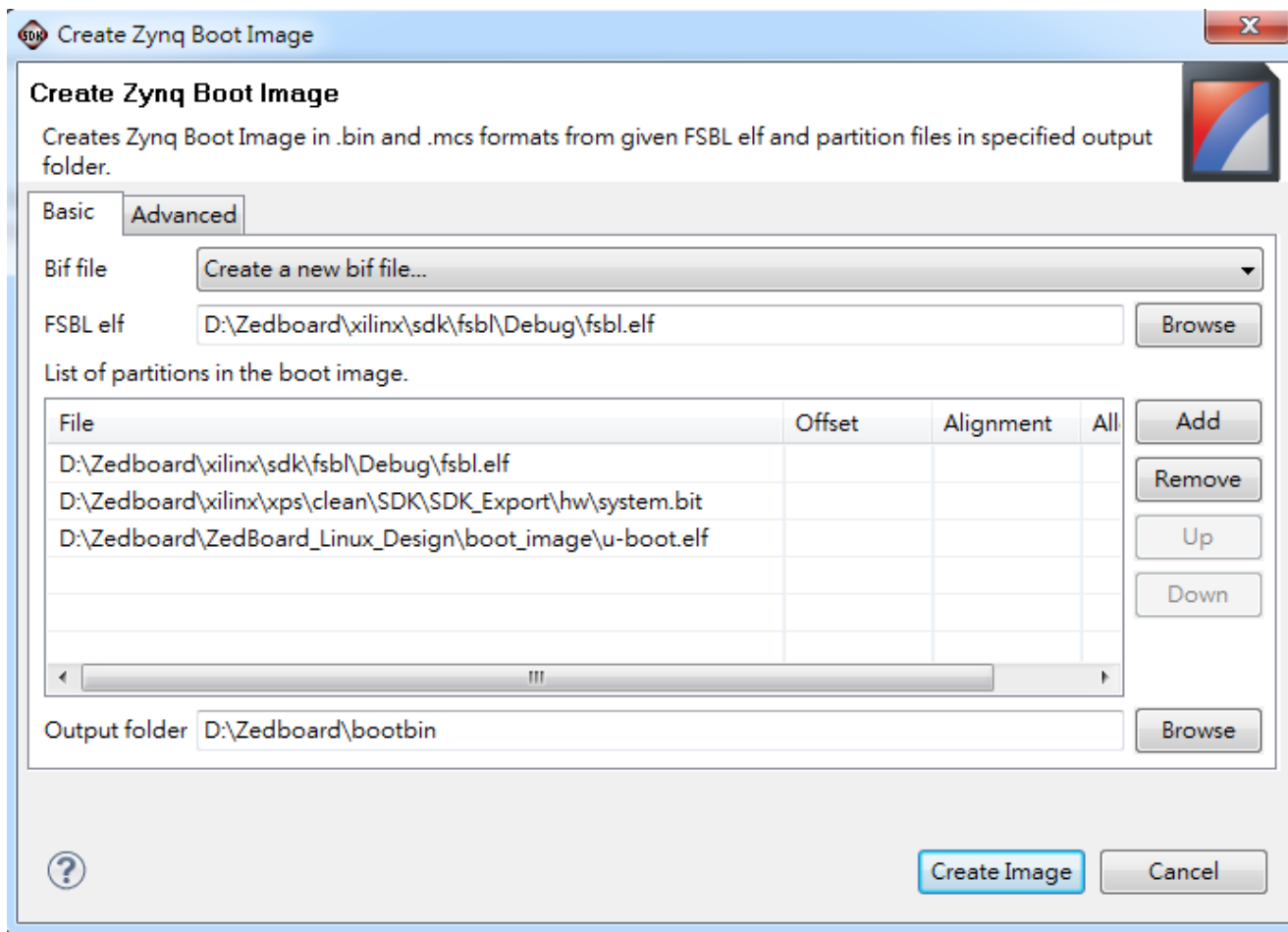
After finish, SDK will compile fsbl automatically.

The **fsbl.elf** will appear under Zedboard\xilinx\sdk\fsbl\Debug



Finally, we can create our clean BOOT.BIN  
Click Xilinx Tools -> Create Zynq Boot Image





Choose *Create a new bif file* and then select FSBL, system.bit, and u-boot.elf

FSBL: Zedboard\xilinx\sdk\fsbl\Debug\fsbl.elf

bitstream: Zedboard\xilinx\xps\clean\SDK\SDK\_Export\hw\system.bit

u-boot: Zedboard\ZedBoard\_Linux\_Design\boot\_image\u-boot.elf

Note that we use the file in prebuilt SD image.

*Create Image*

Finally, as I selected, u-boot.bin will appear under Zedboard\bootbin. Rename it as BOOT.BIN and we're done.

**Note : The order is important!! Please follow the sequence.**

# A quick test

If you just can't wait, you can try to replace the BOOT.BIN of the prebuilt image.

But you'll see this:

```
reading zImage
2458688 bytes read
reading devicetree.dtb
9648 bytes read
reading ramdisk8M.image.gz
3694324 bytes read
## Starting application at 0x00008000 ...
Uncompressing Linux... done, booting the kernel.
```

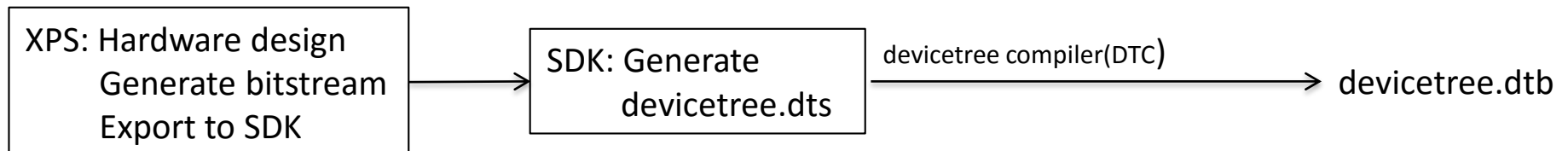
And then zedboard hangs.

This is because we're not using the right devicetree.dtb

**Note: As you modify the hardware, be sure to update your device tree blob!!**

# Workflow – generate device tree blob

- Export the hardware design to SDK:
  - In XPS, Export Design just like we have done.
- Add Device Tree Generator
  - You only have to do this as the first time you generate device tree blob.
- Create device tree project
  - SDK will then generate dts file. But it need some modification before compiling.
- Compile dts to dtb
  - As far as I know, this can only done under Linux. And your should compile kernel to get the compiler.



Linux device tree generator for the Xilinx SDK

305 commits



3 branches

10 releases

3 contributors

 branch: master ▾ device-tree / +

axivdma: Make boolean DT properties real bools ...

 Srikanth Thokala authored 3 months ago  
→ michalsimek committed 3 months agolatest commit 052cb521f6  data axivdma: Make boolean DT properties real bools

3 months ago

&lt;&gt; Code

! Issues

2


 Pull Requests

0



🔊 Pulse

 Graphs Network

HTTPS clone URL

[https://github.com](https://github.com/Xilinx/device-tree) 

You can clone with HTTPS or Subversion. ⓘ

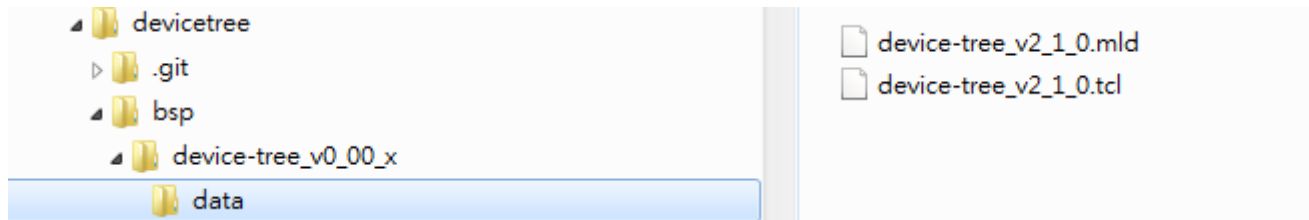
 Clone in Desktop Download ZIP

First we download the device tree repositories.

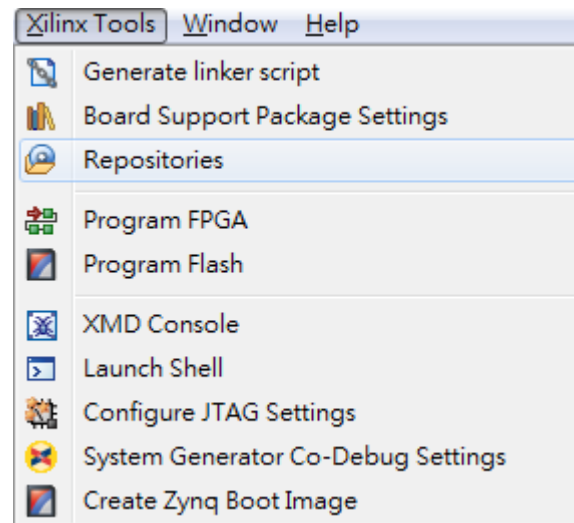
Under Windows: <https://github.com/Xilinx/device-tree>  
and you can download the Device Tree Generator

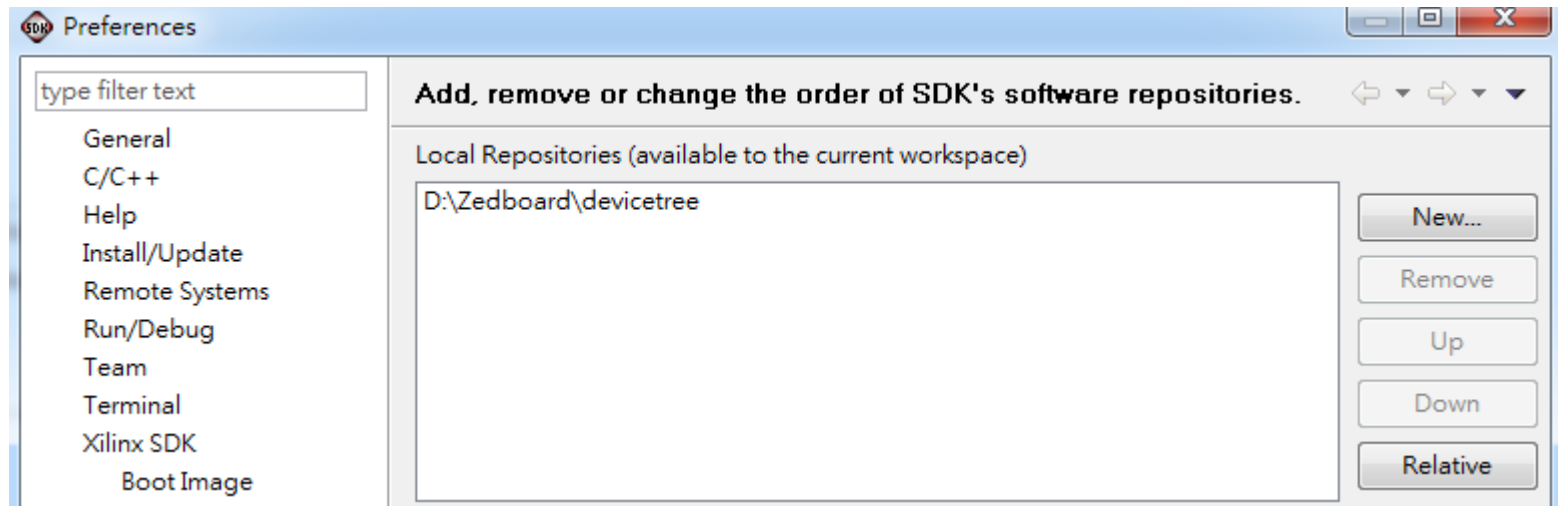
Or under Linux  
simply use the git command  
`git clone git://github.com/Xilinx/device-tree.git`

In order for SDK to correctly find the repositories, the hierarchy should look like this.

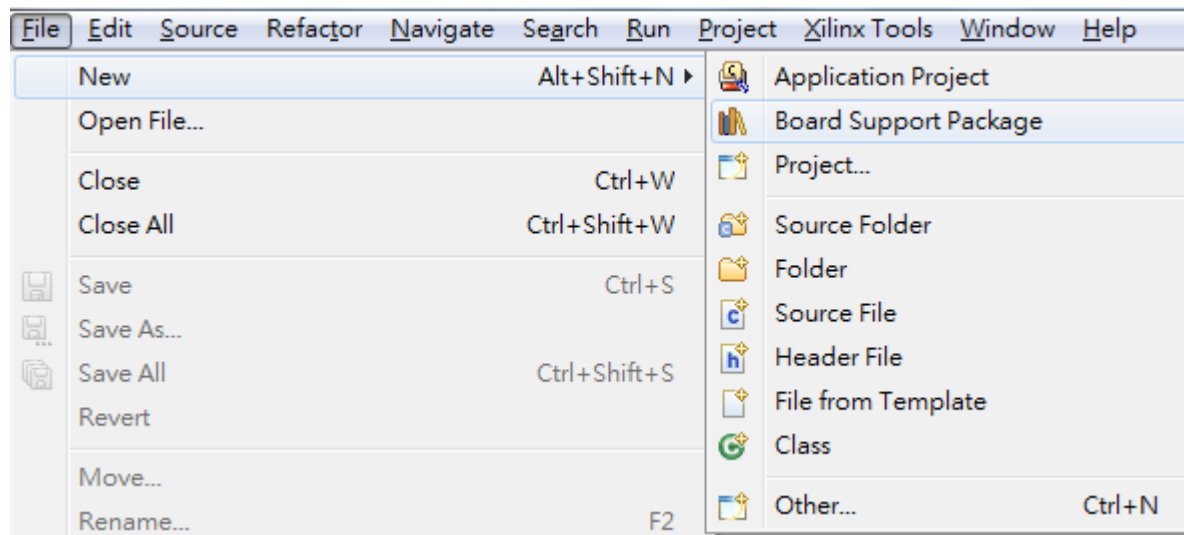


In SDK, *Xilinx Tools* -> *Repositories*

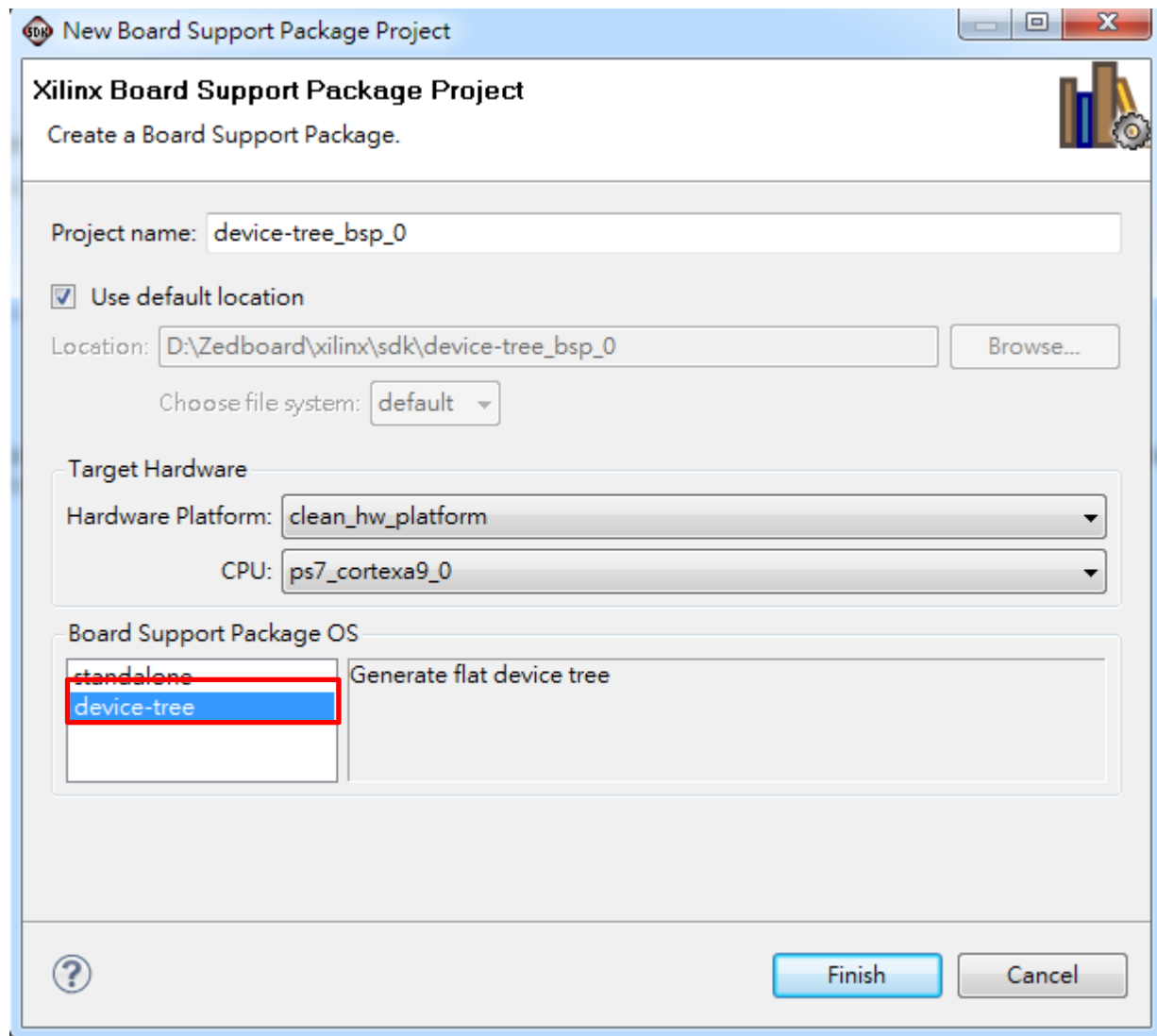




Press *New* and then add Zedboard\devicetree  
*OK*

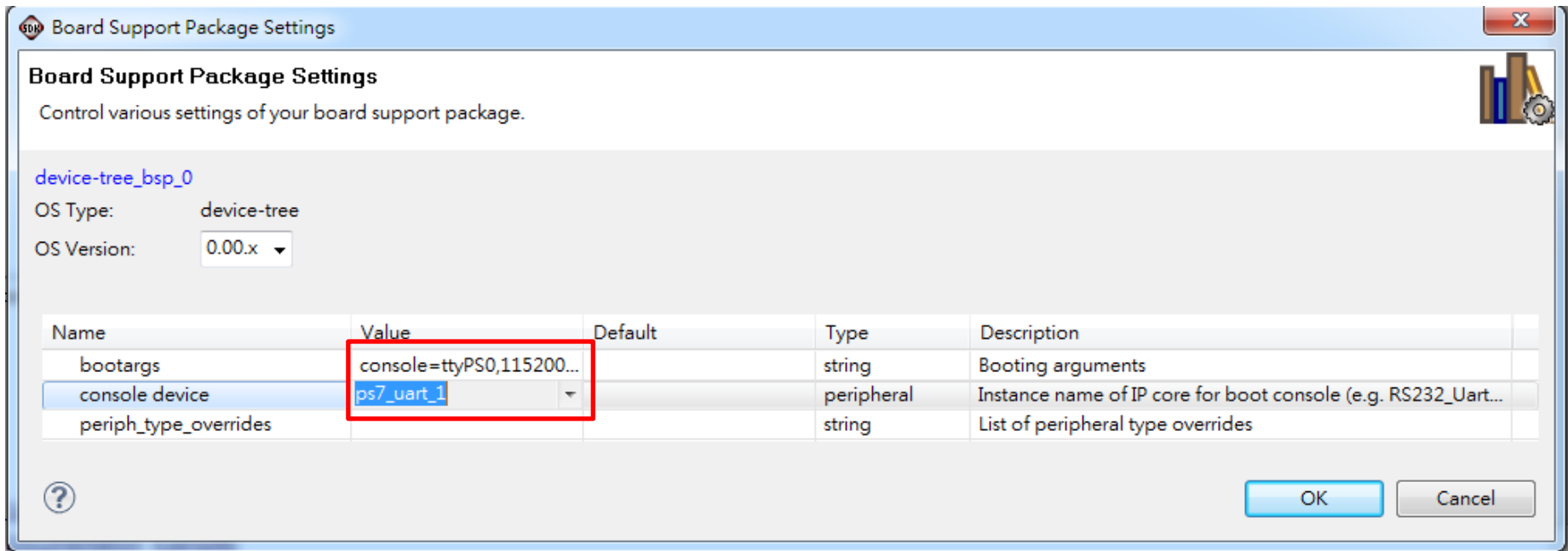


File -> New -> Board Support Package



Choose *device-tree* and then *Finish*





Set bootargs as:

```
console=ttyPS0,115200 root=/dev/ram rw initrd=0x800000,8M earlyprintk rootwait
devtmpfs.mount=1
```

And choose console device as *ps7\_uart\_1*

OK

After building, you can find *xilinx.dts* under

Zedboard\xilinx\sdk\device-tree\_bsp\_0\ps7\_cortexa9\_0\libsrc\device-tree\_v0\_00\_x

Copy *xilinx.dts* to Zedboard\devicetree and rename it as *devicetree.dts*

# Modify dts to fit zedboard

- It seems there are some inappropriate settings for the dts generating.
- Open dts files and change the followings

Line 21:  
compatible = "xlnx,zynq-7000";  
model = "Xilinx Zynq";



Line 21:  
compatible = "xlnx,zynq-**zed**";  
model = "Xilinx Zynq **ZED**";

Search for qspi

ps7\_qspi\_0: ps7-qspi@e000d000 {  
clock-names = "ref\_clk", "aper\_clk";



ps7\_qspi\_0: ps7-qspi@e000d000 {  
**bus-num=<0>;**  
clock-names = "ref\_clk", "aper\_clk";

Add a new line for bus-num setting

# Compile dts to dtb

- The compiling script is `~/win/linux-digilent/scripts/dtc/dtc`
- Compile dts to dtb by  
`cd ~/win/devicetree`  
`~/linux-digilent/scripts/dtc/dtc -I dts -O dtb -o devicetree.dtb devicetree.dts`

```
hadoop@ubuntu:~/win/devicetree$ ~/linux-digilent/scripts/dtc/dtc -I dts -O dtb -o  
devicetree.dtb devicetree.dts  
DTC: dts->dtb on file "devicetree.dts"
```

Reference: [website](#)

# Test again

Replace devicetree.dtb with the prebuilt image.  
And boot the zedboard again.

```
++ Configure static IP 192.168.1.10  
++ Starting telnet daemon  
++ Starting http daemon  
++ Starting ftp daemon  
++ Starting dropbear (ssh) daemon  
++ Starting OLED Display  
++ Exporting LEDs & SWs  
rcS Complete  
zynq>
```

Congratulation!!!!