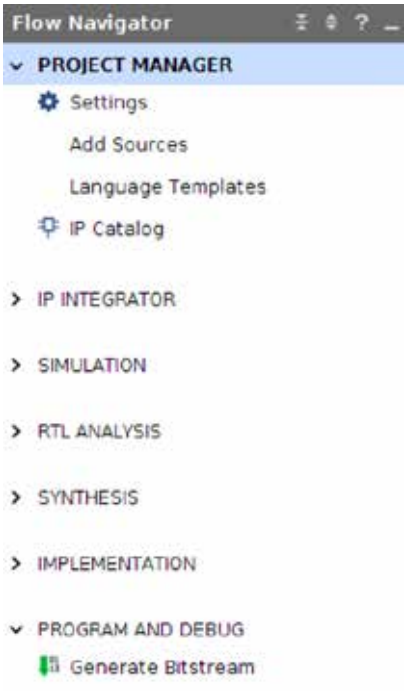
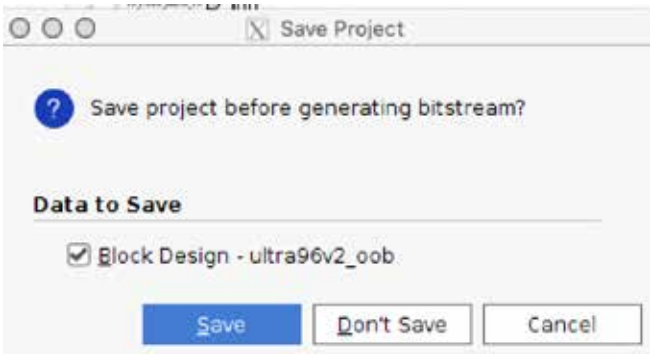
Generating an FPGA bit file for the Ultra96

This document describes how to generate a bit file for the Ultra96.



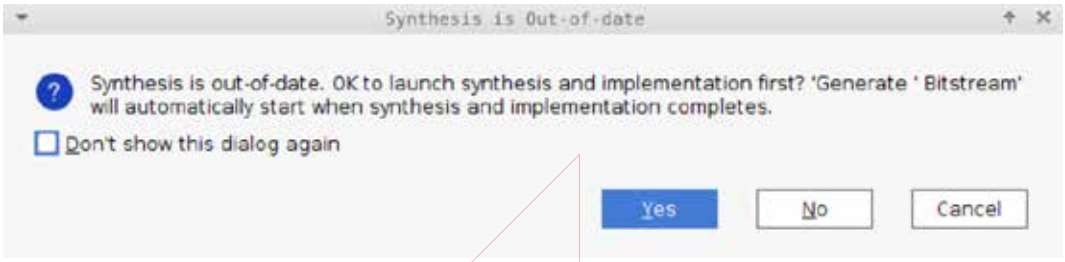
In the **Flow Navigator** click **Generate Bitstream**

You may see the following popup:



Click **Save**

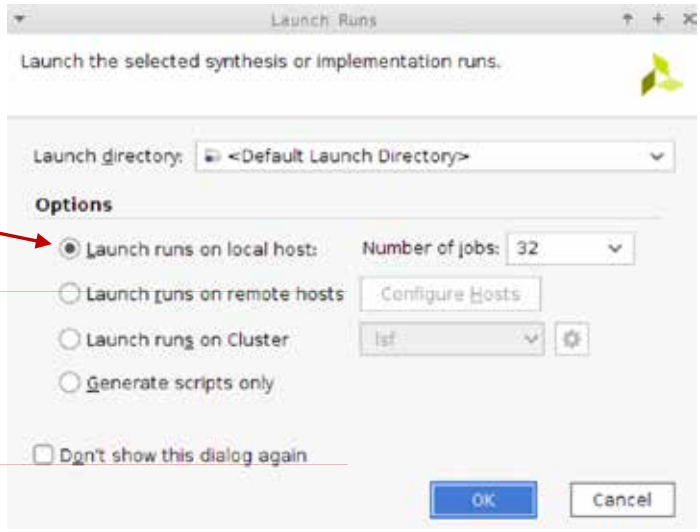
The following popup will appear:



Click **Yes**

Generating an FPGA bit file for the Ultra96

The following popup will appear:

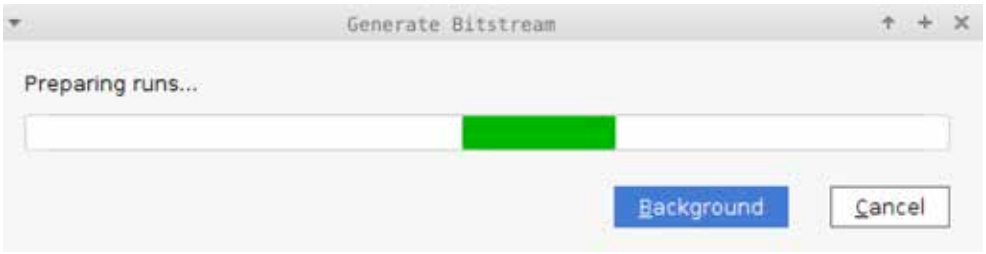


Select **Launch runs on local host**

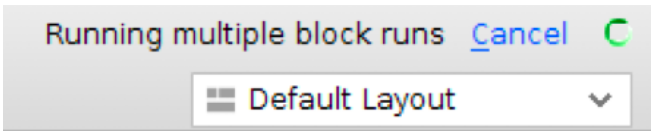
If you are running on one of the big LRC machines you can set the **Number of jobs** to 32

Click **OK**

The following popup will appear:



Let the jobs run in the foreground. Recall that the status of the runs is dynamically displayed in the upper right hand corner of the Vivado workspace:

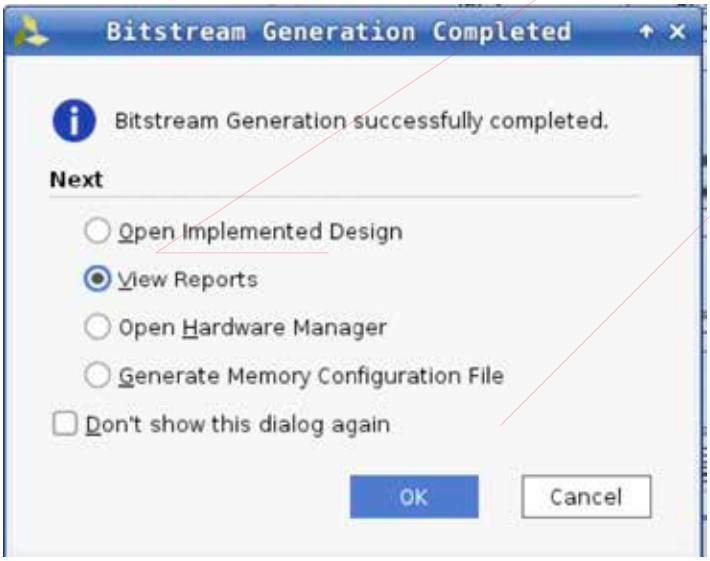


There is also a console window that will show the status of each operation:

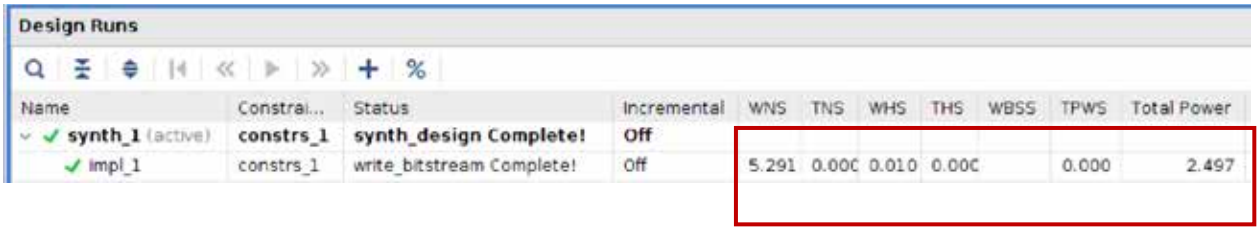
Pay special attention to errors and critical warnings.

Generating an FPGA bit file for the Ultra96

Once the bitstream is generated select the **View Reports** button and click **OK**



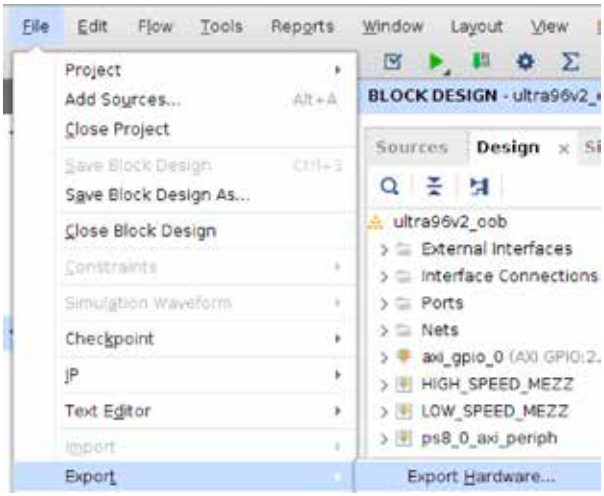
There will be a number of reports that should be looked at if the design does not synthesize, place or route correctly. The key report to look at is under **Design Runs**. It will show the final STA (static timing analysis) runs for the design. The STA section highlighted below indicates that the design meets setup and hold time constraints:



The next step is Export the hardware:

Under **File** select **Export**

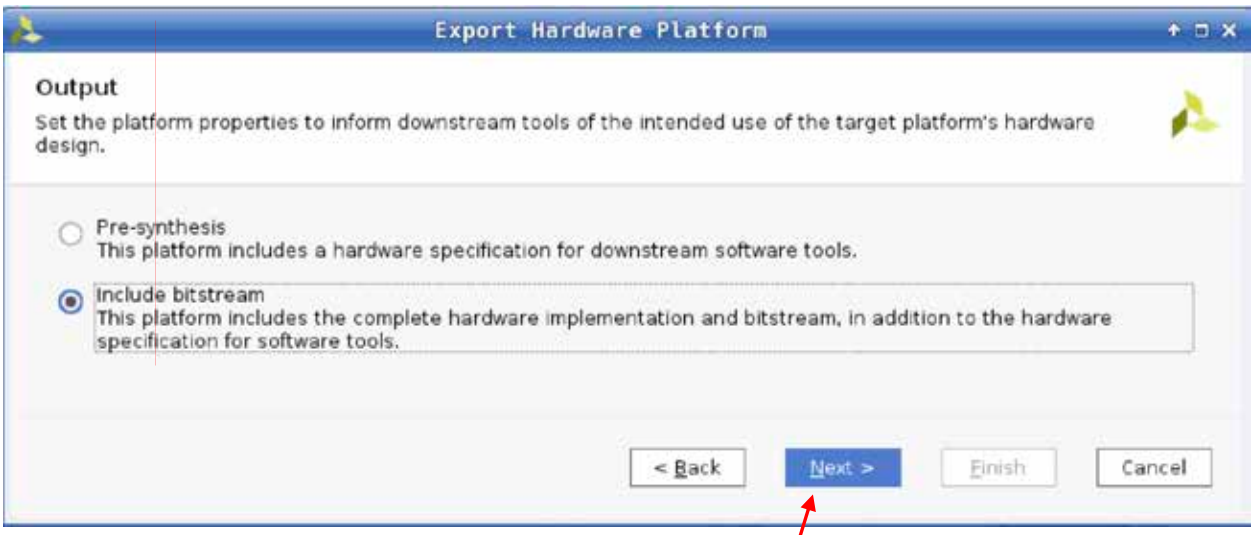
And then **Export Hardware**:



The following popup will appear:

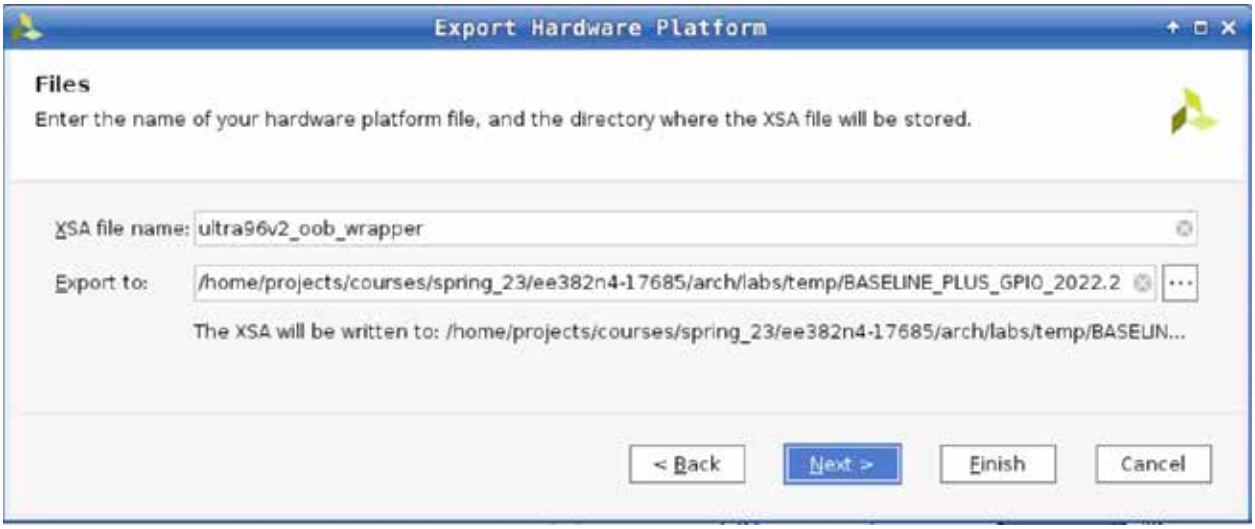
Select **Next**

Select **Include Bitstream**

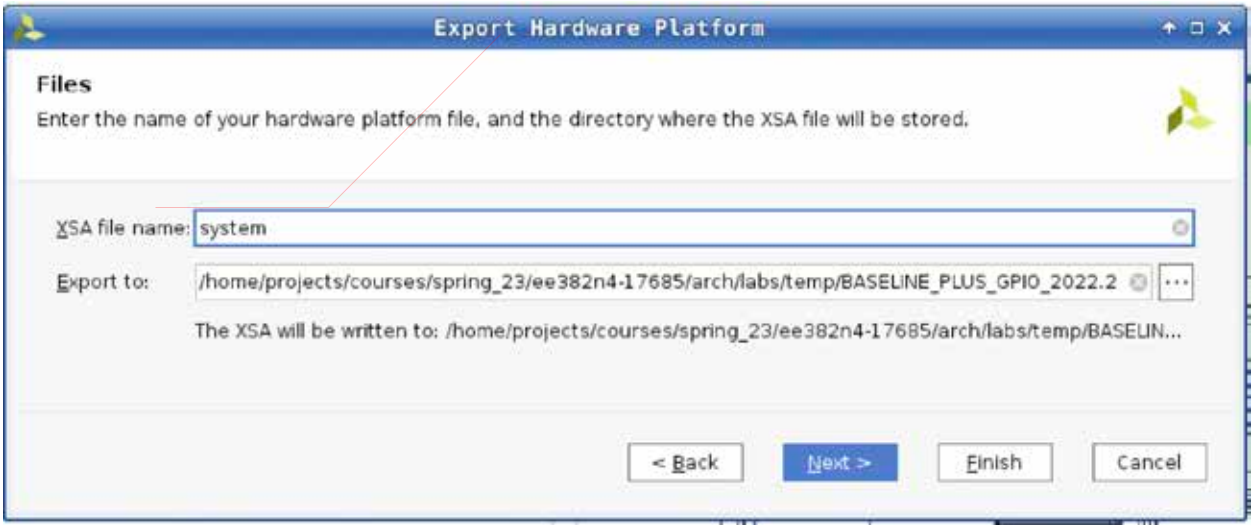


Select **Next**

Generating an FPGA bit file for the Ultra96



Change “ultra96v2_oob_wrapper” to “system”



You may see this popup:

click **Yes**

The following popup will be displayed:

click **Finish**

The system.xsa file is used by Petalinux to build a new DTB file. More about that later

The generated bit file is located in the following sub-directory:

```
../ultra96v2_oob.runs/impl_1/ultra96v2_oob_wrapper.bit
```

The bit file can now be loaded into the FPGA using the fpgautil program. The first step is to SFTP the `ultra96v2_oob_wrapper.bit` file to your Ultra96. This is a two-step process because of the UT VPN restrictions. 1) SFTP the file to your laptop and 2) SFTP from your laptop to the Ultra96. To dynamically load the file into the FPGA fabric execute:

```
fgpautil -b ultra96v2_oob_wrapper.bit
```

Note: The bit file will remain in the FPGA until the next reboot. At which time the default bit file will be reloaded during the bootup process.

Follow these procedures to make the new bit file the default file used by the first stage bootloader (FSBL) **Note: this procedure can completely brick the system if you do not follow these steps correctly.**

```
sudo bash  < ----- Enter your password
cd /media/
mkdir <your_user_name>
cd <your_user_name>
mkdir BOOT
mkdir rootfs
ls /dev/mmc*
```

You should see 3 entries: `/dev/mmcblk0` `/dev/mmcblk0p1` `/dev/mmcblk0p2` There are two partitions on the SDC.

Execute the lsblk command to see the two partitions.

```
lsblk

NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
mmcblk0     179:0    0 29.7G  0 disk
  mmcblk0p1 179:1    0    1G  0 part
  mmcblk0p2 179:2    0 28.7G  0 part
```

The next thing to do is to mount both partitions:

```
mount /dev/mmcblk0p1 BOOT/
mount /dev/mmcblk0p2 rootfs/
```

You will now see the following when you execute the `lsblk` command

```
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
mmcblk0     179:0    0 29.7G  0 disk
  mmcblk0p1 179:1    0    1G  0 part /media/<your name>/BOOT
  mmcblk0p2 179:2    0 28.7G  0 part /media/<your name>/rootfs
```

Change directory into the boot partition and list the directory contents.

```
cd BOOT
ll
```

You will see something like this:

```
-rwxr-xr-x 1 root root     51744 Dec 22 15:01 bl31.bin
-rwxr-xr-x 1 root root    154648 Dec 22 15:01 bl31.elf
-rwxr-xr-x 1 root root    829680 Dec 22 15:01 BOOT.BIN
-rwxr-xr-x 1 root root       444 Dec 22 15:01 boot.scr
-rwxr-xr-x 1 root root  15034880 Dec 22 15:01 Image
-rwxr-xr-x 1 root root   7177836 Dec 22 15:01 image.ub
-rwxr-xr-x 1 root root    141568 Dec 22 15:01 pmufw.elf
-rwxr-xr-x 1 root root   5568793 Dec 22 15:01 system.bit <-----
-rwxr-xr-x 1 root root     51293 Dec 22 15:01 system.dtb
-rwxr-xr-x 1 root root     59801 Dec 22 15:01 system.dts
-rwxr-xr-x 1 root root   3361363 Dec 22 15:01 System.map.linux
-rwxr-xr-x 1 root root    543216 Dec 22 15:01 u-boot.bin
-rwxr-xr-x 1 root root    609544 Dec 22 15:01 u-boot.elf
-rwxr-xr-x 1 root root    524288 Dec 22 15:01 uboot.env
-rwxr-xr-x 1 root root       445 Dec 22 15:01 uEnv.txt
-rwxr-xr-x 1 root root  14838336 Dec 22 15:01 uImage
-rwxr-xr-x 1 root root    112440 Dec 22 15:01 zynqmp_fsbl.elf
```

NOTE: The bit file that UBoot is looking for is: `system.bit`

Generating an FPGA bit file for the Ultra96

You will need copy the `ultra96v2_oob_wrapper.bit` to the boot partition. First though you need to save the original bit file in case you have a problem

```
cp /media/<your_user_name>/BOOT/system.bit \
        /media/<your_user_name>/BOOT/system.bit.save
cp ultra96v2_oob_wrapper.bit /media/<your_user_name>/BOOT/system.bit
```

If you brick your system, you will need to remove the SDC and repair it on a Linux workstation.