Report on Investigation into MPSoc USB2.0 OTG functionality

Author: Greg Smart

Date : 30/04/2019

**Introduction**

The investigation was intended to reproduce results claimed by Xilinx on a ZCU102 and verify that they correlate with testingon the Mercury board. The final aim is to determine the path forward for OTG on the Mercury board.

**Investigation Procedure**

The initial process was to create a Vivado project for the ZCU102 with USB3.0 OTG to provide a baseline.

The project doesn’t contain any confidential information, and is available at <https://github.com/KutuSystems/zcu102_gpio.git>

The build is either setup to use and internal kernel or the linu-xlnx kernel at

git clone [git@github.com](mailto:git@github.com):Xilinx/linux-xlnx.git, then checkout 2018.2 branch

git checkout xlnx\_rebase\_v4.14\_2018.2

If the external kernel is used then the kernel must be manually patched with the 3 patch files.

patch -p1 < kutu\_gpio\_driver.patch

patch -p1 < msp430\_driver.patch

patch -p1 < otg\_debug.patch

The Github project is a simple PS configuration with the addition of 2 custom IP modules. The IP modules are not used in the investigation, but are in the project to provide confirmation that the patching of the Linux Kernel, Kernel configuration and device tree is being performed correctly during each build.

**1. Build Vivado Project**

Before the Petalinux system can be built, the Vivado project must be built and exported to hardware. The steps to build the FPGA design are:

1. Open Vivado 2018.2

2. cd to <github root>/zcu102\_gpio/hardware/ZCU102\_GPIO

3. type source ./create\_all\_projects.tcl

At this point Vivado will do some project generation magic, and when complete a block diagram of the system will be displayed. If modifications are to be made, they are made at this stage.

4. click on “Generate Bitstream” (bottom left of IDE)

When the bitfile generation is complete you get asked if you want to open implemented design.

5. Select “open implemented design” and click “ok”

6. when the design has been opened go to “File” menu (top left). Go down to “Export”. In the Export menu select “Export hardware ...”. A small dialog appears. Ensure “Include bitstream” is selected and click “ok”

After this stage is complete you can exit Vivado. Vivado is no longer required, all subsequent steps are done using Petalinux

**1. Build Petalinux**

After the Petalinux init script has been run (source ~/petalinux-v2018.2/settings.sh), the Petalinux project is built using the following commands.

1. petalinux-config –get-hw-description=hardware/ZCU102\_GPIO/ZCU102\_GPIO.sdk

2. petalinux-build

3. petalinux-package --force --boot --fsbl images/linux/zynqmp\_fsbl.elf --fpga images/linux/system.bit --pmufw images/linux/pmufw.elf –u-boot

when complete the files boot.bin and image.ub are in the images/linux directory. The CDC driver module is located in the directory build/tmp/work/plnx\_znqmp-xilinx-linux/linux-xlnx/4.14-xilinx-v2018.2+git999-r0/image/lib/modules/4.14.0/kernel/drivers/usb/gadget/legacy/g\_serial.ko

copy the 3 files to the SD card.

After linux is booted, load the driver using following command:

insmod /run/media/mmcblk0p1/g\_serial.ko

The console message will look like the message below

root@zcu102\_gpio:~# insmod /run/media/mmcblk0p1/g\_serial.ko

[ 35.018008] g\_serial gadget: Gadget Serial v2.4

[ 35.022473] g\_serial gadget: g\_serial ready

root@zcu102\_gpio:~#

In a working system you see the following when a PC is plugged in to the USB port.

root@zcu102\_gpio:~# [ 90.362410] dwc3 fe200000.dwc3: dwc3\_otg\_irq(): INFO: B-Device Session Valid Detect Event

[ 90.370512] dwc3 fe200000.dwc3: dwc3\_otg\_irq(): INFO: B-Device VBUS Change Event

[ 90.377899] dwc3 fe200000.dwc3: otg\_main\_thread(): INFO: Main thread entering state

[ 90.385535] dwc3 fe200000.dwc3: otg\_main\_thread(): INFO: OTG\_STATE\_B\_PERIPHERAL

[ 90.392826] dwc3 fe200000.dwc3: stop\_host(): INFO: Host already stopped

[ 90.399416] dwc3 fe200000.dwc3: start\_peripheral(): INFO: GS : Starting Peripheral

[ 90.880564] g\_serial gadget: high-speed config #2: CDC ACM config

And when it is removed:

root@zcu102\_gpio:~#

root@zcu102\_gpio:~# [ 172.584132] dwc3 fe200000.dwc3: dwc3\_otg\_irq(): INFO: B-Device VBUS Change Event

[ 172.591461] dwc3 fe200000.dwc3: stop\_peripheral(): INFO: GS : stopping peripheral

[ 172.624194] dwc3 fe200000.dwc3: otg\_main\_thread(): INFO: Main thread entering state

[ 172.631773] dwc3 fe200000.dwc3: otg\_main\_thread(): INFO: OTG\_STATE\_B\_IDLE

root@zcu102\_gpio:~#

The PC will show the ZCU102 as a CDC device

greg@kutu3:~$

greg@kutu3:~$ lsusb

Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub

Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub

Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub

Bus 001 Device 003: ID 413c:301a Dell Computer Corp.

Bus 001 Device 002: ID 413c:2113 Dell Computer Corp.

Bus 001 Device 004: ID 10c4:ea71 Cygnal Integrated Products, Inc.

Bus 001 Device 006: ID 0525:a4a7 Netchip Technology, Inc. Linux-USB Serial Gadget (CDC ACM mode)

Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub

greg@kutu3:~$

Adding a peripheral (host mode) will have similar messages

Adding and removing devices will occur in around 1 second, not more than 2-3 seconds.

**2. Kernel Modifications**

There are 3 patch files, but only one of them is relevant (usb\_otg.patch). The other patches load the other drivers. To see they have been correctly loaded, type “ls /dev” at the console. This verifies that the kernel has been built correctly

The important patch (usb\_otg.patch) contains the modification provided by Xilinx, and some debug messages have been manually enabled in order to trace the OTG state machine.

**3. Results**

The intial results were with the default Vivado configuration (USB3.0 transceiver enabled)

The results of the investigation showed some issues which demonstrated the problem. The internal linux kernel and the external kernal have the same md5 checksum, but the output files are a different size. This indicates that the internal kernel is being configured differently than the external kernel but the differences are hidden from the user.

**Test 1**

**Build with loadable modules**

In order to enable the CDC the following modules must be loaded:

* insmod configfs.ko
* insmod libcomposite.ko
* insmod u\_serial.ko
* insmod usb\_f\_serial.ko
* insmod usb\_f\_acm.ko
* insmod g\_serial.ko

When configured in this way, both kernel versions worked correctly as expected

**Test 2**

**Compile in modules, expect CDC to enumerate without any modules**

When using the external kernel, the system recognized the device but the CDC device was not enumerated. The peripheral could be plugged in and out with no problems, but the CDC didn’t enumerate. Host mode behaved normally

When using the internal kernel, the system usually recognized the the first time the PC was plugged in, but usually not more than once. When a periperal (host mode) was plugged in then the driver would crash.

The conclusion was the driver loading process did not delay driver loading correctly causing the CDC not to enumerate. I was unable to find out the exact kernel differences.

**Test 3**

**Compile in all modules except g\_serial, expect CDC to enumerate after g\_serial is manually loaded**

insmod g\_serial.ko

When using either kernel, the system operated correctly as in test 1.

The remainder of tests are using this configuration

**Test 4**

**Use Xilinx provided build (test for host only)**

This build is supposed to have the USB3.0 transceiver disabled

System unresponsive to any USB interaction (either host or periperal)

**Test 5**

**Disable USB3.0 transceiver with internal kernel**

This build is unresponsive to any USB interaction (either host or periperal)

loading g\_serial.ko provides the following result

root@zcu102\_gpio:~# insmod /run/media/mmcblk0p1/g\_serial.ko

[ 126.806819] g\_serial gadget: Gadget Serial v2.4

[ 126.811293] g\_serial gadget: g\_serial ready

[ 126.816164] dwc3 fe200000.dwc3: failed to enable ep0out

root@zcu102\_gpio:~#

**Test 6**

**Disable USB3.0 transceiver with external kernel**

The result of this build is identical to test 5

**Test 7**

**Set USB3.0 transceiver to use 100MHz clock from PCIe with PCIe transceiver disabled**

This build will not get past FSBL because PLL doesn’t lock

**Test 8**

**Set USB3.0 transceiver to use 100MHz clock from PCIe with PCIe transceiver enabled**

This build is unresponsive to any USB interaction (either host or periperal)

loading g\_serial.ko provides the fails as in previous tests

**Test 9**

**Set USB3.0 to use transceiver 1 with 26MHz clock from channel 2 (USB clock)**

This build will not get past FSBL because PLL doesn’t lock. It is hard to tell if the failure is due to the hardware being not capable of this mode (unlikely), or the clock itself not enabling.

**Test 10**

**Set USB3.0 to use transceiver 2 with 26MHz clock from channel 2 (USB clock) and disable all other transceivers**

insmod g\_serial.ko

When using either kernel, the system operated correctly as in test 1.

The remainder of tests are using this configuration

**Overall Observations**

In the kernel modifications, extra messages are enables in otg.c to monitor the OTG interrupt status and state machine behaviour. When the system is operating correctly, that status messages clearly indicate the operation of the mode switching. When the USB is disconnected, the ID pin is pulled up, and the USB regulator is disabled. When a peripheral is added, the ID pin is pulled low which causes and ID change event. When a host is added, the VBUS line (power) is driven high which causes a VBUS change event. These events are detected by the USB3320 PHY which is operating correctly. In the tests that failed, these events do not occur, so the state machine never runs. The patches provided by Xilinx only modify otg.c. If the interrupts are never received then changes to otg.c are not relevant, so you wouldn’t expect any improvement.

In the tests that try to use different transceivers, it is possible that the device tree is incorrectly configured, but it appears most likely that lower level drivers do not enable the USB2.0 functionality. The clocking of the transceiver appears to affect the USB core, but this shouldn’t affect USB2.0 controller portion of the hardware.

**Conclusion**

The conclusion is that the Xilinx kernel modifications do not address the issue, and that currently the only reliable workaround is to provide a 26MHz clock on transceiver 2 and configure the USB port a USB3.0.