BIOS MCSDK 2.0

PCIE Boot Example

Applies to patch release based on 02.00.00.10
Publication Date: June 23, 2011
Version 1.1

Texas Instruments, Incorporated
20450 Century Boulevard
Germantown, MD 20874 USA

Document License

Contributors to this document

Contents

1 Overview 1
2 Revision History 1
3 References 1
4 DDR Init Boot Image 1
4.1 Procedure to build ddrinit 2
5 HelloWorld Boot Image 2
5.1 Procedure to build HelloWorld 2
6 POST Boot Image 3
6.1 Procedure to prepare POST boot image 3
7 PCIE Linux Host Loader Code 3
7.1 Procedure to build and run Linux host loader 4
7.2 How HelloWorld boot example works 4
7.3 How POST boot example works 5
8 Test Setup and Expected Results 5

PCIE Boot Example

1 Overview

The PCIE boot example is created to help customer quickly boot DSP through PCIE. The boot example includes:

- A HelloWorld boot example from all cores, which has two CCS projects to build the DDR initialization boot image and HelloWorld boot image.
- A simple POST boot example from core 0 in addition to HelloWorld boot example.
- Linux host PCIE loader code to map between PC memory and DSP memory. It loads the boot example into DSP via PCIE for boot demo purpose.

2 Revision History

Revision	Details
1.1	Add a new PCIE boot demo for “HelloWorld”
1.0	Initial Version

3 References

[1] KeyStone Architecture Peripheral Component Interconnect Express (PCIe) User Guide (Rev. A, http://www.ti.com/litv/pdf/sprugs6a)

4 DDR Init Boot Image

The DDR Init project uses the BIOS MCSDK Platform Library to initialize the DDR.

4.1 Procedure to build ddrinit

- Import the project from tools\boot_loader\examples\pcie\pcieboot_ddrinit\evmc66xxl in CCSv5
- Clean and re-build the project
- The pcieboot_ddrinit_evm66xxl.map and pcieboot_ddrinit_evm66xxl.out will be generated under tools\boot_loader\examples\pcie\pcieboot_ddrinit\evmc66xxl\bin
- Run pcieboot_ddrinit_elf2HBin.bat under tools\boot_loader\examples\pcie\pcieboot_ddrinit\evmc66xxl\bin, the batch file does the following file conversion:
    - Uses Code Gen utility hex6x.exe utility to convert the ELF format .out file to a ASCII hex format boot table file
    - Uses Bttbl2Hfile.exe to convert the boot table file to a header text file.
    - Uses hfile2array.exe to convert the header text file to a header file with array of the image data
    - Moves the converted header file to tools\boot_loader\examples\pcie\linux_host_loader folder

5 HelloWorld Boot Image

The HelloWorld project uses the BIOS MCSDK Platform Library to initialize the UART, it will print the “Hello World” and booting information for all the DSP cores to the UART once it runs.

5.1 Procedure to build HelloWorld

- Import the project from tools\boot_loader\examples\pcie\pcieboot_helloworld\evmc66xxl in CCSv5
- Clean and re-build the project
- The pcieboot_helloworld_evm66xxl.map and pcieboot_helloworld_evm66xxl.out will be generated under tools\boot_loader\examples\pcie\pcieboot_helloworld\evmc66xxl\bin
- Run helloworld_elf2HBin.bat under tools\boot_loader\examples\pcie\pcieboot_helloworld\evmc66xxl\bin, the batch file does the same file conversion as pcieboot_ddrinit_elf2HBin.bat does.

6 POST Boot Image

The existing POST is used as another PCIE boot example. The POST uses the BIOS MCSDK Platform Library to do a board test and results can be displayed via UART.

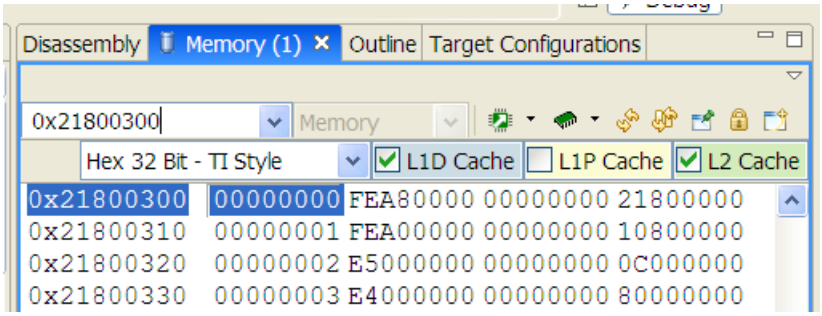
6.1 Procedure to prepare POST boot image

- Copy post_evm66xxl.out from tools\post\evmc66xxl\bin to tools\boot_loader\examples\pcie\pcieboot_post\evmc66xxl\bin folder.
- Run pcieboot_post_elf2HBin.bat under tools\boot_loader\examples\pcie\pcieboot_post\evmc66xxl\bin, the batch file does the same file conversion as pcieboot_ddrinit_elf2HBin.bat does as well.

7 PCIE Linux Host Loader Code

The PCIE Linux host loader code has several functions:

- Do a memory mapping between PC memory and DSP memory (4 blocks of memories are requested by DSP via PCIE registers BAR0, BAR1, BAR2 and BAR3 masks, with a size of 16K, 512K, 4M and 16M respectively for PCIE application registers, local L2, MSMC and DDR3. The BAR masks are configured inside PCIE initialization code when selects PCIE boot mode on EVM.
- Configure the PCIE inbound address translation through the accessing of application registers as below example for IB_BARn, IB_STARTn_LO, IB_STARTn_HI and IB_OFFSETn (n = 0, 1, 2, 3).



- Provide DSP memory read/write API:
  - Uint32 ReadDSPMemory(Uint32 coreNum, Uint32 DSPMemAddr, Uint32 *buffer, Uint32 length)
  - Uint32 WriteDSPMemory(Uint32 coreNum, Uint32 DSPMemAddr, Uint32 *buffer, Uint32 length)
- Parse the boot example header array file for boot entry address, section size and start address of sections and load the boot data into DSP memory via API.

- Write the boot entry address into the magic address on core 0 via API.

7.1 Procedure to build and run Linux host loader

- Create a folder (e.g. pcie_test) in a Linux machine. Copy pciedemo.c, Makefile, pcieDdrInit.h, pcieBootCode.h and post6678.h from tools\boot_loader\examples\pcie\linux_host_loader to the folder.
- Change directory to “pice_test”, update KDIR inside Makefile to your system (you can find it out by “uname –a” command)
- Type “make”, a pciedemo.ko file should be created
- By default, this will build the “HelloWorld” demo, which is controlled by:

  ```
  #define HELLO_WORLD_DEMO  1
  #define POST_DEMO         0
  ```

  in pciedemo.c. If one wants to build the POST demo, please change above to:

  ```
  #define HELLO_WORLD_DEMO  0
  #define POST_DEMO         1
  ```

  Then, type “make clean” and type “make” to rebuild the pciedemo.ko.
- To insert the module into kernel, type “sudo insmod pciedemo.ko”; to view the kernel message, type “dmesg”; to remove the module from kernel, type “sudo rmmod pciedemo.ko”

7.2 How HelloWorld boot example works

The Linux host first pushes the DDR init boot image data to L2 memory of core 0, then writes the boot entry address of the DDR init boot image to the magic address (0x87FFFC for TMS320C6678 device) on core 0, both via PCIE. When the EVM is in PCIE boot mode, the IBL code running on the DSP core 0 polls the entry address and jumps to that address and starts to boot (initialize the DDR). After DDR is properly initialized, the DDR init code clears the magic address and keeps on polling it.

Linux host then pushes the HelloWorld boot image data to DDR memory, then writes the boot entry address of the HelloWorld boot image to the magic address on core 0 to boot core 0. Core 0 starts to boot and print the “Hello World” booting information, and then boot all the other cores by writing the address of _c_int00 to the magic address on other cores and sending an IPC interrupt to other cores. The RBL running on other cores will jump to _c_int00 and start to boot, each core will write 0xBABEFACE to its magic address by running a function write_boot_magic_number().

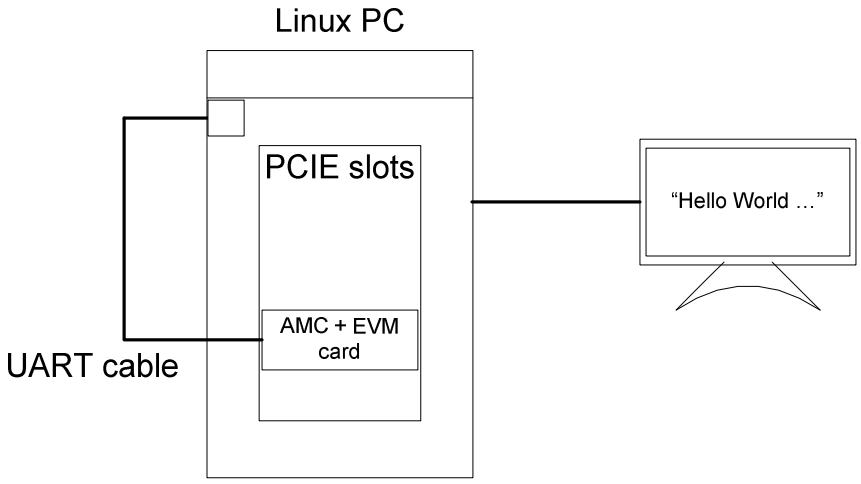
Note that Host boot application needs to wait for some time after pushing the DDR init boot image and before pushing the HelloWorld boot image to the DDR, this will ensure DDR is properly initialized.

7.3 How POST boot example works

The POST example uses L2 only. The Linux host first pushes the POST boot image data to L2 memory of core 0, then writes the boot entry address of the POST to the magic address on core 0, both via PCIE. The IBL code running on the DSP core 0 polls the entry address and jumps to that address and starts to boot.

8 Test Setup and Expected Results

An AMC to PCIE adaptor card, a TMS320C66xxL EVM card and a Linux PC are required to do the test, as the diagram shown below:



- Before connect the system, please update IBL with the latest from MCSDK 2.0 GA by following the steps in tools\boot_loader\ibl\doc\evmc66xx-instructions.txt. When perform step 1: "Programing "IBL" on the EEPROM at bus address 0x51", make sure using "swap_data = 0" in tools\writer\eeprom\evmc66xxl\bin\eepromwriter_input.txt.
- Set EVM card to PCIE boot via following switch setting:

SW3 (pin1, 2, 3, 4)	SW4 (pin1, 2, 3, 4)	SW5 (pin1, 2, 3, 4)	SW6 (pin1, 2, 3, 4)	SW9 (pin1)
(off, on, on, off)	(on, on, on, on)	(on, on, on, off)	(off, on, on, on)	(on)

- Assemble the EVM card into the adaptor card
- UART port can be accessed either through Mini-USB connector (USB1) or through 3-pin RS232 Serial port header (COM1). The selection can be made through UART route select connector COM_SEL1 as follows:

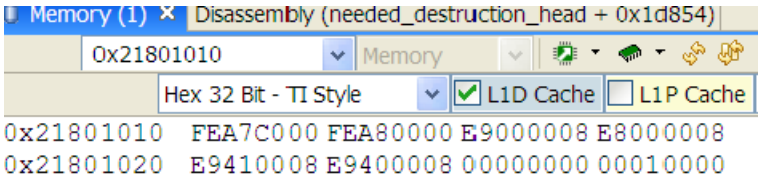
  - UART over USB Connector (Default): Shunts installed over COM_SEL1.3-COM_SEL1.1 and COM_SEL1.4-COM_SEL1.2
  - UART over 3-Pin Header LAN1: Shunts installed over COM_SEL1.3-COM_SEL1.5 and COM_SEL1.4-COM_SEL1.6
  - Connect the URAT cable from EVM card to a Linux PC's USB port or serial port based on the desired access method

- Completely shut off the PC power supply (by disconnecting the power cord), insert the AMC adaptor card (with TMS320C66xxL EVM card mounted) into an open PCIE slot in PC's motherboard
- Supply the power to PC, wait for a few seconds and power on the PC.
- Make sure the PCIE device is correctly enumerated by PC by checking below,
  - Either enter PC's BIOS setting when PC is booting up, a new PCIE device should be populated in the PCIE slot where Shannon card is inserted, shown as a "Multimedia device".
  - Or, type "lspci –n" under Linux command shell after Linux OS is loaded, a TI device (VENDOR_ID: 0x104c) should be in the list:

    local-ubuntu:~$ lspci -n
    00:00.0 0600: 8086:2774
    00:1b.0 0403: 8086:27d8 (rev 01)
    ….
    00:1f.3 0c05: 8086:27da (rev 01)
    01:00.0 0480: 104c:8888 (rev 01)
    03:00.0 0200: 14e4:1677 (rev 01)

    Similarly, one can type "lspci",
    local-ubuntu:~$ lspci
    ….
    00:1f.3 SMBus: Intel Corporation N10/ICH 7 Family SMBus Controller (rev 01)
    01:00.0 Multimedia controller: Texas Instruments Device 8888 (rev 01)
    ....

  - The PCIE BARn (n = 0, 1, 2, … , 5) registers are written by Linux PC after enumeration, they should be non-zero. Optionally, if a JTAG emulator is available, one can verify this by looking at address starting from 0x21801010 for 6 32-bit word. Below is an example.

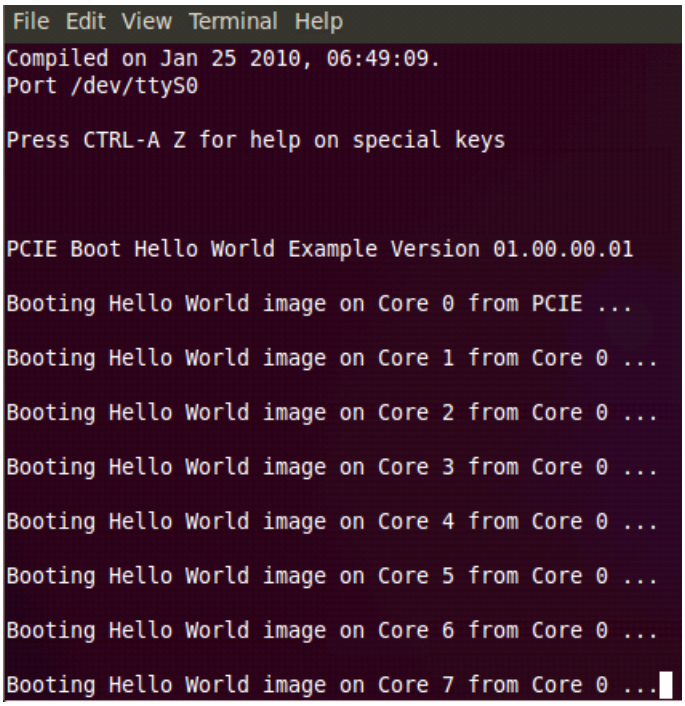


- Prepare pciedemo.ko in the Linux PC, please refer to section 7.1

- On the Linux PC open a new terminal window to run minicom. First run “sudo minicom –s” to set the correct configuration: 115200bps, 8-N-1, Hardware flow control: OFF, Software flow control: OFF, and select the correct Serial Device. Save then run “sudo minicom” to monitor the port.

- Type “sudo insmod pciedemo.ko”,

    - For the HelloWorld demo, one should see the following printed on the minicom:



      Optionally, if a JTAG emulator is available, one can verify that the PC registers for cores other than core 0 should be inside DDR; and magic address for cores other than core 0 should be written with 0xBABEFACE.

    - For the POST demo, one should see the following printed on the minicom:

- To view the kernel log, one can type “dmesg”

  [1365.860539] pciedemo: module license 'unspecified' taints kernel.
  [1365.861808] Finding the device....
  [1365.861821] Found TI device
  [1365.861825] TI device: vendor=0x0000104c, dev=0x00008888, drv=0x00000000, irq=0x00ff, bus=0xf74ca400
  ….