SOC Center of Excellence

Device Trees for Intel SoC FPGAs

Dan Negvesky

November 2016

Agenda

- Background
- Device Tree Overview
  - Hierarchy
  - Loading a device tree
  - Drivers and bindings
  - Examples
- FPGA Regions & Device Tree Overlays
- Creating a Device Tree
  - Examples
- Appendix – fully annotate device tree example

Background

- Before device trees…
  - Non-discoverable hardware (e.g. I2C, SPI, UART, Ethernet controllers, etc.) as part of an embedded system was described using C code compiled directly in the kernel (uImage or zImage).
  - Bootloader passed additional information to the kernel and a machine type integer identified the board.
- Use of device trees with ARM began in 2011 (predating the release of Altera SoC).
- With device trees…
  - Hardware description is located in a separate binary called the device tree blob or device tree binary (.dtb)
  - Bootloader loads the kernel (uImage or zImage) and the device tree binary (.dtb)

What is a Device Tree?

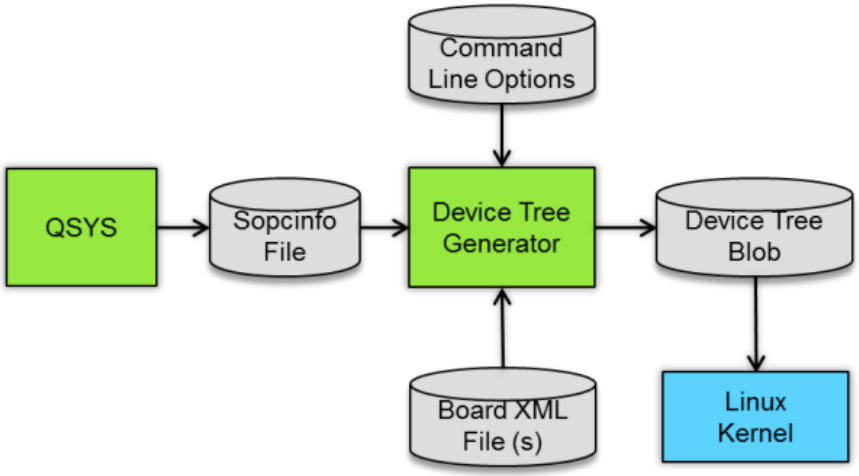
- Device trees use a hardware description language to describe the hardware available to an embedded system (e.g. Altera SoCs)
- This typically consists of non-discoverable hardware, non-discoverable buses (e.g. I2C, SPI), and external hardware attached to them (e.g. LCD display).
- They consist of a tree data structure of **nodes**, with each node having **properties** (address, interrupt, driver info) that describe the device.
- Device tree source (.dts) files are compiled into a device tree blob (.dtb), either during kernel build or separately, using a device tree compiler (DTC).
- Can be used by the bootloader (Arria 10) or the kernel.
- Can be flat or hierarchical.
- Device trees are kernel version dependent (e.g. your 3.10-ltsi kernel .dtb will not work with the 4.1.22-ltsi kernel).

Device Tree Usage

- Static
  - Base (board level) device tree defines all external hardware, HPS and FPGA peripherals used by the specific embedded system
  - The boot loader is responsible for loading the kernel image (zImage used for socfpga) and device tree blob (.dtb) into memory.
- Dynamic
  - Base (board level) device tree defines external hardware, HPS, and “FPGA Regions”
  - Device tree overlay defines FPGA peripherals to be added, or overlayed, to FPGA Region in live device tree
  - Device tree overlay file (.dtso) gets compiled and added to the root filesystem
  - Boot loader loads board level tree, then after FPGA configuration (optional), overlay gets appended to main tree by an init script
  - Applying the overlay can also (re)configure the FPGA if specified in the overlay

Creating a device tree (1/2)

- Old method
  - Kernel 3.10-ltsi examples on rocketboards.org
    - Used the sopc2dts tool (originally developed for Nios) to read .sopcinfo board and system .xml files and automatically output .dts/.dtb files
    - Creates a flattened device tree
    - Still requires the user to manually edit/create/customize a board level .xml file
    - See Device Tree Generator User Guide (v15.1)



SOC COE

Creating a device tree (2/2)

- Newer method
  - sopc2dts tool does not generate functional device trees for the newer kernels, specifically the latest ltsi kernel (currently 4.1.22)
  - sopc2dts still partially usable to extract the FPGA peripherals
  - sopc2dts may be updated in the future to produce usable device trees again - overlay
  - For kernel 4.1.22-ltsi
    - Use example hierarchical device trees that are part of the kernel source located in /arch/arm/boot/dts.
    - See full example in Appendix.
    - From Device Tree Generator User Guide :

Warning: The Linux Device Tree Generator is tested with and supports only the Linux kernel version targeted by the associated GSRD. For the 15.1 GSRD release this 3.1-ltsi. It is not recommended to use the Linux Device Tree Generator if your design targets a different Linux kernel version. It is instead recommended to manage the Device Tree manually in such a case, using the Device Tree files provided by the kernel as a baseline, and adding the FPGA IP and board information manually.

Device tree hierarchy (1/2)

- Device trees created by sopc2dts are flattened device trees (i.e. most examples on rocketboards for 3.10-ltsi kernel).
- Newer kernels use hierarchical device tree structure.
- Each board should have its own device tree.
- Different boards that use the same processor (e.g. Intel/Altera SoC devices that use the same hard processor subsystem, or HPS), can include other device trees that define common features.
  - Located in kernel tree arch/arm/boot/dts
    - socfpga.dtsi
    - socfpga_cyclone5.dtsi
    - socfpga_arria5.dtsi
    - socfpga_arria10.dtsi
- Use **#include** in board level device tree to “layer” multiple trees.

Device tree hierarchy (2/2)

Board device tree
socfpga_cyclone5_sockit.dts

```
.
.
.
.
.
#include "socfpga_cyclone5.dtsi"

/ {
     model = "Terasic SoCkit";
     compatible = "altr,socfpga-cyclone5", "altr,socfpga";
.
.
.
```

Cyclone V common device tree
socfpga_cyclone5.dtsi

```
.
.
.
.
/dts-v1/;
/* First 4KB has trampoline code for secondary cores. */
/memreserve/ 0x00000000 0x0001000;
#include "socfpga.dtsi"

/ {
     soc {
          clkmgr@ffd04000 {
               clocks {
.
.
.
```

socfpga common device tree
socfpga.dtsi

```
.
.
.
.
#include "skeleton.dtsi"
#include <dt-bindings/reset/altr,rst-mgr.h>

/ {
     #address-cells = <1>;
     #size-cells = <1>;

     aliases {
               ethernet0 = &gmac0;
               ethernet1 = &gmac1;
               serial0 = &uart0;
.
.
.
```

Bare minimum skeleton.dtsi

```
/*
 * Skeleton device tree; the bare minimum needed to boot; just
 * include and add a compatible value.  The bootloader will
 * typically populate the memory node.
 */

/ {
     #address-cells = <1>;
     #size-cells = <1>;
     chosen { };
     aliases { };
     memory { device_type = "memory"; reg = <0 0>; };
};
```

Compiled
socfpga_cyclone5_sockit.dtb
(binary device tree blob)

SOC COE

Device trees, drivers, and bindings (1/3)

- In a device tree source node entry, the **compatible** string and various properties are used to bind a device to its corresponding driver.
- In the device driver source, the **of_match_table** field of the **struct** device_driver lists the compatible strings supported by the driver.
- In the device tree binding, the required and optional properties and values of the device tree entry are defined to provide device attributes needed for the driver.
  - Device Tree bindings are documented in kernel source tree in Documents/devicetree/bindings
  - New bindings reviewed by Device Tree maintainers by sending to devicetree@vger.kernel.org.

Device trees, drivers, and bindings (2/3)

Newhaven LCD driver example

```
* TTY on a Newhaven NHD-0216K3Z-NSW-BBW LCD connected to I2C

Required properties:
- compatible: Should be "newhaven,nhd-0216k3z-nsw-bbw";
- reg: i2c address
- height: should be 2 lines
- width: should be 16 characters
- brightness: backlight brightness. Range is 1 to 8, where
              1=OFF and 8=maximum brightness.

Example:

&i2c0 {
        lcd: lcd@28 {
                compatible = "newhaven,nhd-0216k3z-nsw-bbw";
                reg = <0x28>;
                height = <2>;
                width = <16>;
                brightness = <8>;
        };
```

2.

device tree binding - newhaven_lcd.txt

phandle

```
&i2c0 {
        status = "okay";
        speed-mode = <0>;

        /*
         * adjust the falling times to decrease the i2c frequency to 50Khz
         * because the LCD module does not work at the standard 100Khz
         */
        i2c-sda-falling-time-ns = <5000>;
        i2c-scl-falling-time-ns = <5000>;

        lcd: lcd@28 {
                compatible = "newhaven,nhd-0216k3z-nsw-bbw";
                reg = <0x28>;
                eight = <2>;
                width = <16>;
                brightness = <8>;
        };
```

1.

3.

board device tree - socfpga_cyclone5_socdk.dts

```
static const struct of_device_id lcd_of_match[] = {
        { .compatible = "newhaven,nhd-0216k3z-nsw-bbw", },
        {},
};

static const struct i2c_device_id lcd_id[] = {
        { DRV_NAME, 0 },
        { }
};
MODULE_DEVICE_TABLE(i2c, lcd_id);

static struct i2c_driver lcd_i2c_driver = {
        .driver = {
                .name = DRV_NAME,
                .owner = THIS_MODULE,
                .of_match_table = lcd_of_match,
        },
        .probe = lcd_probe,
        .remove = lcd_remove,
        .id_table = lcd_id,
};
```

4.

newhaven_lcd.c driver

1. The i2c0 controller is defined in socfpga.dtsi and is referred to by &i2c0 in board level .dts file. It is enabled with **status = “okay”**
2. The device tree binding defines the required and optional properties and values that should be declared in the device tree. The lcd is defined as a child node of the parent i2c0 node.
3. A node in a device tree is bound to a driver by using the **compatible** string.
4. The of_match_table field of **struct i2c_driver** lists the compatible strings supported by the driver.

SOC COE

Device trees, drivers, and bindings (3/3)

- For the kernel to successfully load a driver (using a device tree)
  - Device must be declared in the device tree
  - Device tree properties defined correctly
    - “compatible” string that identifies the driver
    - Physical address allocated to the device
    - Interrupts (if any) used by the device
- Kernel scans “compatible” entries in device tree and matches with list of known drivers (compiled in kernel or loadable modules)

Device Tree Entry – Example 1

i2c0 entry from socfpga.dtsi

label – use phandle **&i2c0** to refer to this node in other sections of the device tree or in board level device tree

node-name@unit-address: unit-address should match first address in reg property.

Defines # of 32 bit address cells and # of 32 bit length cells used in a child node's **reg** property.

Property names and values

Device driver binding

This device is clocked by l4_sp_clk

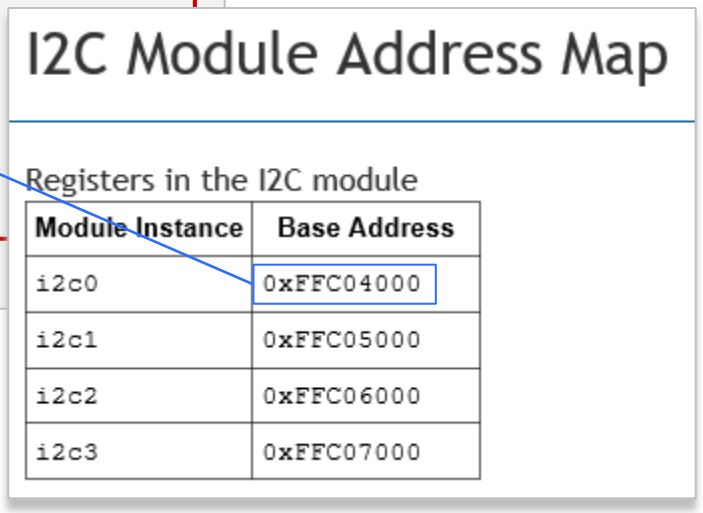
```
i2c0: i2c@ffc04000 {
        #address-cells = <1>;
        #size-cells = <0>;
        compatible = "snps,designware-i2c";
        reg = <0xffc04000 0x1000>;
        clocks = <&l4_sp_clk>;
        interrupts = <0 158 0x4>;
        status = "disabled";
};
```

This device is disabled at this level.

Interrupt type

Interrupt number

Level sense encoding



I2C Module Address Map

Registers in the I2C module

Module Instance	Base Address
i2c0	0xFFC04000
i2c1	0xFFC05000
i2c2	0xFFC06000
i2c3	0xFFC07000

From HPS Memory Map

SOC COE

Example 1 (cont'd) – interrupts (1/2)

```
i2c0: i2c@ffc04000 {
        #address-cells = <1>;
        #size-cells = <0>;
        compatible = "snps,designware-i2c";
        reg = <0xffc04000 0x1000>;
        clocks = <&l4_sp_clk>;
        interrupts = <0 158 0x4>;
        status = "disabled";
};
```

Interrupt number:
Refer to the Cyclone V SoC TRM, Generic Interrupt Controller, "GIC Interrupt Map" table.

GIC Interrupt Number	Source Block	Interrupt Name	Combined Interrupts	Triggering
190	I2C0	i2c0_IRQ	This interrupt combines: ic_rx_under_intr, ic_rx_full_intr, ic_tx_over_intr, ic_tx_empty_intr, ic_rd_req_intr, ic_tx_abrt_intr, ic_rx_done_intr, ic_activity_intr, ic_stop_det_intr, ic_start_det_intr, and ic_gen_call_intr.	Level

Interrupt type:
0 = non-SPI
Non-zero = SPI

The ARM Cortex A9 TRM defines 3 interrupt types:
Software Generated Interrupt (SGI)
Private Peripheral Interrupt (PPI)
Shared Peripheral Interrupt (SPI)

Most will be non-SPI type.

For non-SPI type interrupts, take the GIC interrupt number and subtract 32:
190 - 32 = 158

For SPI interrupts, take interrupt number and subtract 16.

SOC COE

Example 1 (con'd) – interrupts (2/2)

```
i2c0: i2c@ffc04000 {
        #address-cells = <1>;
        #size-cells = <0>;
        compatible = "snps,designware-i2c";
        reg = <0xffc04000 0x1000>;
        clocks = <&l4_sp_clk>;
        interrupts = <0 158 0x4>;
        status = "disabled";
};
```

Specifies drivers/i2c/busses/i2c-designware-platdrv.c driver, which #includes linux/interrupts.h

```
.
.
.
/*
 * These correspond to the IORESOURCE_IRQ_* defines in
 * linux/ioport.h to select the interrupt line behaviour.  When
 * requesting an interrupt without specifying a IRQF_TRIGGER, the
 * setting should be assumed to be "as already configured", which
 * may be as per machine or firmware initialisation.
 */
#define IRQF_TRIGGER_NONE           0x00000000
#define IRQF_TRIGGER_RISING         0x00000001
#define IRQF_TRIGGER_FALLING        0x00000002
#define IRQF_TRIGGER_HIGH           0x00000004
#define IRQF_TRIGGER_LOW            0x00000008
.
.
.
```

Level sense encoding:

0x0 = "as already configured" (typically from boot loader)
0x1 = rising edge
0x2 = falling edge
0x4 = high
0x8 = low

Example 2 – FPGA peripheral w/ interrupt

```
jtag_uart: serial@0x100020000 {
        compatible = "altr,juart-15.1", "altr,juart-1.0";
        reg = <0x00020000 0x00000008>;
        interrupt-parent = <&intc>;
        interrupts = <0 42 4>;
        clocks = <&clk_0>;
};
```

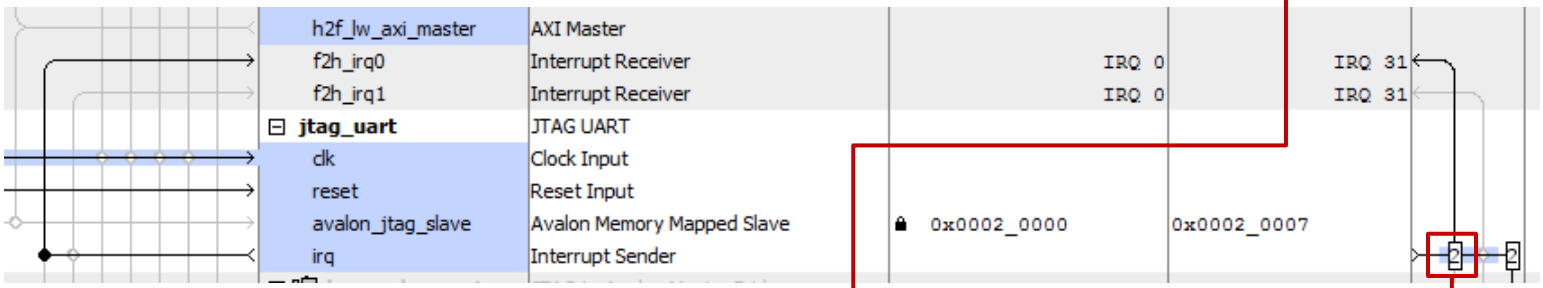
phandle to interrupt-parent, defined in socfpga.dtsi, which uses Cortex A9 GIC

For non-SPI type interrupts, take the GIC interrupt number and subtract 32:

42 = 74 - 32

There are 64 general purpose FPGA-to-HPS interrupts that allow soft IP in the FPGA to trigger interrupts in the MPU's GIC.

- f2h_irq0 - FPGA-to-HPS interrupts 0 - 31
- f2h_irq1 - FPGA-to-HPS interrupts 32 - 63



IRQ 2

Referring again to the Cyclone V SoC TRM, Generic Interrupt Controller, "GIC Interrupt Map" table.

GIC Interrupt Number	Source Block	Interrupt Name	Combined Interrupts	Triggering
72	FPGA	FPGA_IRQ0	—	Level or Edge
73	FPGA	FPGA_IRQ1	—	Level or Edge
74	FPGA	FPGA_IRQ2	—	Level or Edge

SOC COE

FPGA Regions Framework and Device Tree Overlays

FPGA Regions Framework

- FPGA Regions framework was introduced as a way to (re)configure an FPGA through an OS and allow any new hardware to be visible in the device tree (via overlay).
- FPGA Region in device tree is the parent node to custom FPGA hardware modules added to Qsys system (i.e. hardware not part of the HPS)
  - Can be defined in base (live) tree
  - Can be added to by overlay
- FPGA Region must include properties that add entries needed for configuration of the FPGA (FPGA Manager, FPGA Bridges)

FPGA Region Device Tree Properties

- Required properties
  - compatible = "fpga-region";
  - fpga-mgr : contains a phandle to an FPGA Manager (e.g. fpga-mgr = <&fpga_mgr>;)
  - fpga-bridges : contains a list of phandles to FPGA bridges (e.g. fpga-bridges = <&fpga_bridge0>, <&fpga_bridge1>, <&fpga_bridge2>, <&fpga_bridge3>;)
  - #address-cells, #size-cells : defines # of cells used to encode addresss and size cells of child node's **reg** property
  - ranges : defines child-bus-address, parent-bus-address, length
- Properties added in overlay
  - firmware-name : name of an FPGA configuration file in the firmware search path (described in firmware class documentation)
  - partial-fpga-config : boolean, used with FPGA partial reconfiguration, used with firmware-name to specify configuration file
  - external-fpga-config : boolean, used if FPGA is already configured

SOC COE

Example device tree FPGA Region entry

```
base_fpga_region: base-fpga-region {
        compatible = "fpga-region";
        fpga-mgr = <&fpga_mgr>;
        fpga-bridges = <&fpga_bridge0>, <&fpga_bridge1>,
                       <&fpga_bridge2>, <&fpga_bridge3>;

        #address-cells = <0x2>;
        #size-cells = <0x1>;
        ranges = <0x00000000 0x00000000 0xc0000000 0x20000000>,
                 <0x00000001 0x00000000 0xff200000 0x00200000>;
};
```

Handles address space mapping for child nodes

Required properties for FPGA Region

address span of LWHPS2FPGA bridge (1 cell)

Physical base address of FPGA slaves accessed via LWHPS2FPGA bridge

Virtual address mapping for device tree (2 cells)

```
fpga_bridge0: fpga_bridge@ff400000 {
        compatible = "altr,socfpga-lwhps2fpga-bridge";
        reg = <0xff400000 0x100000>;
        resets = <&rst LWHPS2FPGA_RESET>;
        reset-names = "lwhps2fpga";
        clocks = <&l4_main_clk>;
};
```

```
fpga_mgr: fpgamgr@ff706000 {
        compatible = "altr,socfpga-fpga-mgr";
        reg = <0xff706000 0x1000
               0xffb90000 0x20>;
        interrupts = <0 175 4>;
};
```

These entries are found under the **soc** node in socfpga.dtsi

SOC COE

Device Tree Overlay Overview

- Device tree overlays are used to add additional device nodes to a “live” tree currently loaded by the kernel.
  - The live device tree could be a typical board level tree (using hierarchy) with minimal device entries.
  - Live tree must contain FPGA Region, FPGA Manager, and FPGA Bridges (if they exist)
- The target of the device tree overlay is the FPGA Region; all new nodes are inserted as child nodes here (typically /soc/base_fpga_region)
- Overlay is applied in Linux by adding overlay .dtb to configfs (example to follow)
- Overlays can allow the OS to (re)configure the FPGA
  - FPGA configuration files (.rbf) added to /lib/firmware

Why use Dynamic Device Tree Overlay?

- FPGA hardware design changes
  - May be easier to only have to update an overlay file
- Can be used to (re)configure the FPGA while the OS is running (useful during development)
  - Additional hardware added to system via daughter cards or mezzanine cards plugged into base board
  - New FPGA configuration file is loaded which causes the FPGA hardware peripherals to change
  - Partial reconfiguration changes the FPGA hardware peripherals
- Overlays are the preferred methodology moving forward

Device Tree Overlay Properties

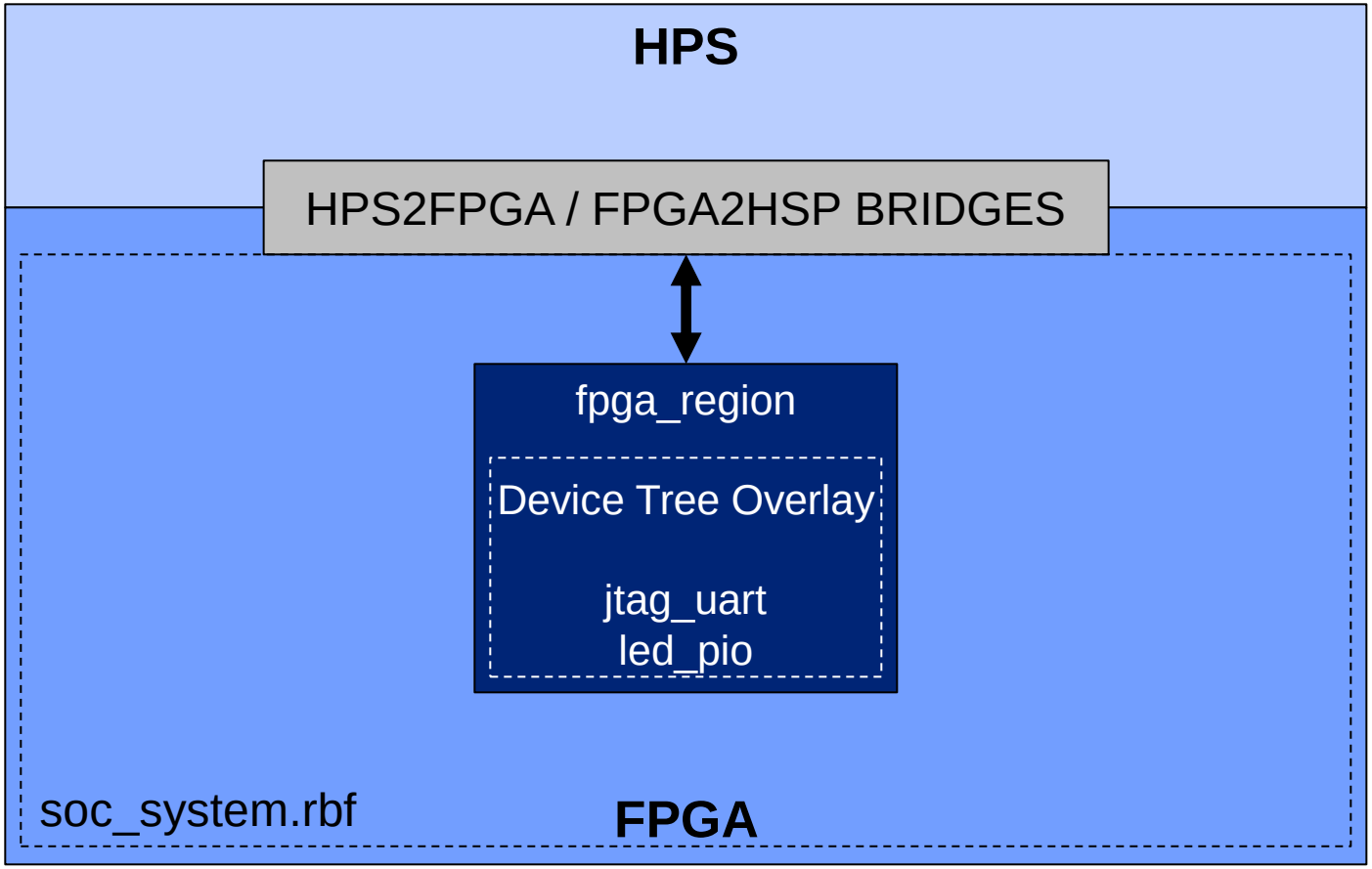
- Required properties
  - target-path = full path where overlay will be inserted in live tree **or** target = phandle of insertion point
  - address-cells and address-size = inherited from properties of parent node of insertion point
  - firmware-name : name of an FPGA configuration file in the firmware search path (described in firmware class documentation)
  - partial-fpga-config : boolean, used with FPGA partial reconfiguration, used with firmware-name to specify configuration file
  - external-fpga-config : boolean, used if FPGA is already configured (or will be configured externally)
- Optional properties
  - region-unfreeze-timeout-us = max time (in microseconds) to wait for bridges to be enabled after the region has been programmed
  - region-freeze-timeout-us = maximum time (in microseconds) to wait for bridges to be disabled before the region has been programmed
  - config-complete-timeout-us = maximum time (in microseconds) to wait for the FPGA to go to user mode after the region has been programmed

SOC COE

Device Tree Overlay Use Cases

- Full FPGA configuration (Example 1)
  - Adds 2 FPGA peripherals
  - Bridges controlled by FPGA Manager that configures FPGA
  - FPGA Bridge devices not required for full reconfiguration
- Full FPGA configuration to setup Partial Reconfiguration Regions, with hardware bridges (Example 2)
- Partial reconfiguration with soft FPGA bridges (Example 3)

Device tree overlay – Example 1 – Full FPGA Configuration and Append to Base Tree



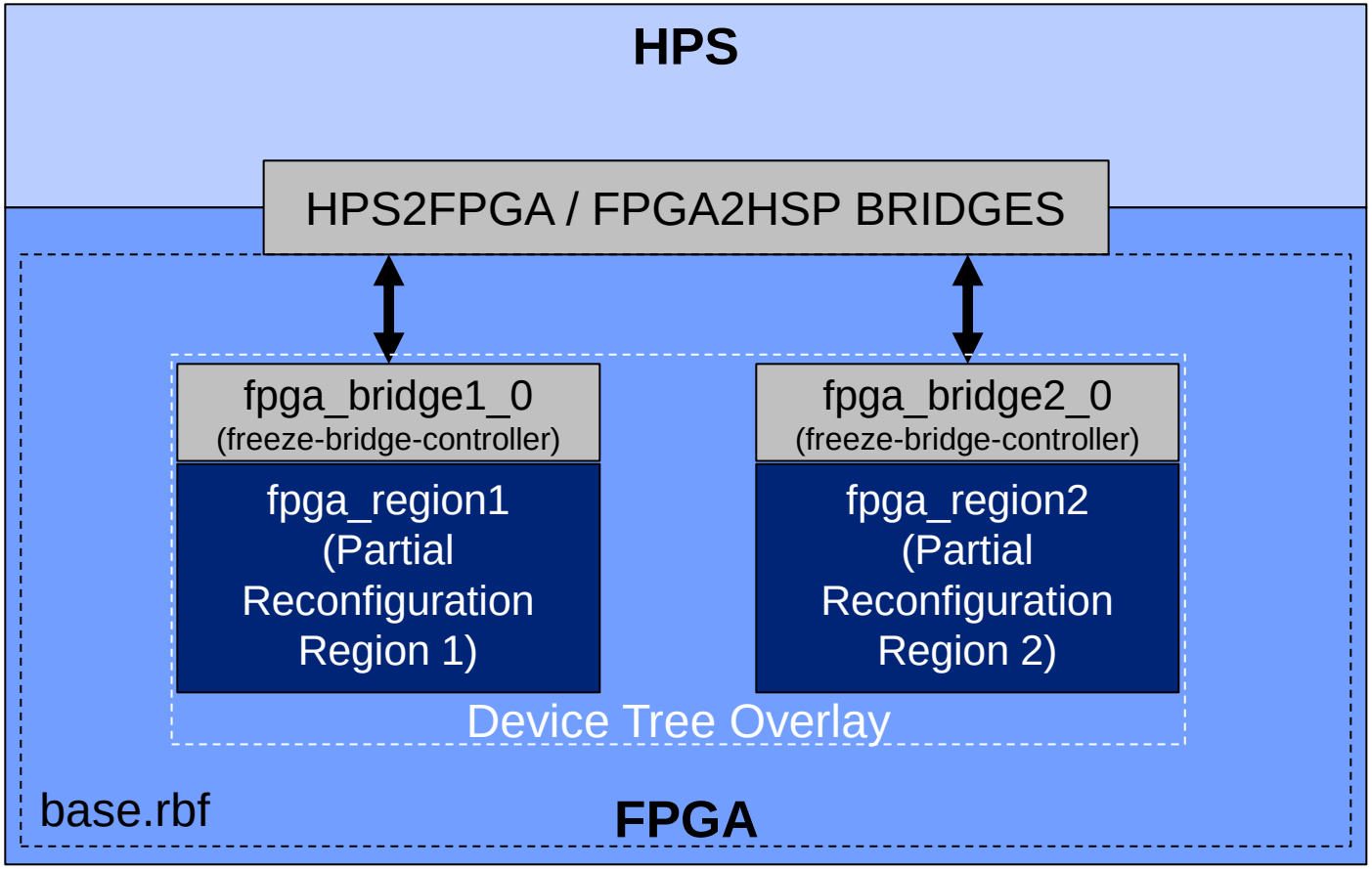
SOC COE

Device tree overlay – Example 1 – Full FPGA Configuration and Append to Base Tree

```
/dts-v1/ /plugin/;
/ {
     fragment@0 {
1.       target-path = "/soc/base_fpga_region";
          #address-cells = <1>;
          #size-cells = <1>;
          __overlay__ {
               #address-cells = <1>;
               #size-cells = <1>;

2.             firmware-name = "soc_system.rbf";

3.             jtag_uart: serial@20000 {
                    compatible = "altr,juart-1.0";
                    reg = <0x20000 0x8>;
                    interrupt-parent = <&intc>;
                    interrupts = <0 42 4>;
               };

               led_pio: gpio@10040 {
                    compatible = "altr,pio-1.0";
                    reg = <0x10040 0x20>;
                    altr,gpio-bank-width = <4>;
                    #gpio-cells = <2>;
                    gpio-controller;
               };
          };
     };
     fragment@1 {  // second child node
          ...
     };
};
```

1. Contents of the overlay will be applied to target **base_fpga_region** under **soc** node in the live tree.
2. Use of the **firmware-name** property indicates that the FPGA should be (re)configured with the given .rbf file. Any bridges listed in base_fpga_region will first be disabled, then the FPGA is configured using the FPGA manager core.
3. If FPGA configuration is successful, the bridges are re-enabled and these child devices are populated to the live tree. If configuration fails, the bridges remain disabled and the overlay is rejected.
4. **of_platform_populate** is called and device drivers are probed.

Device tree overlay – Example 2 – adding Partial Reconfiguration Regions



SOC COE

Device tree overlay – Example 2 – adding Partial Reconfiguration Regions

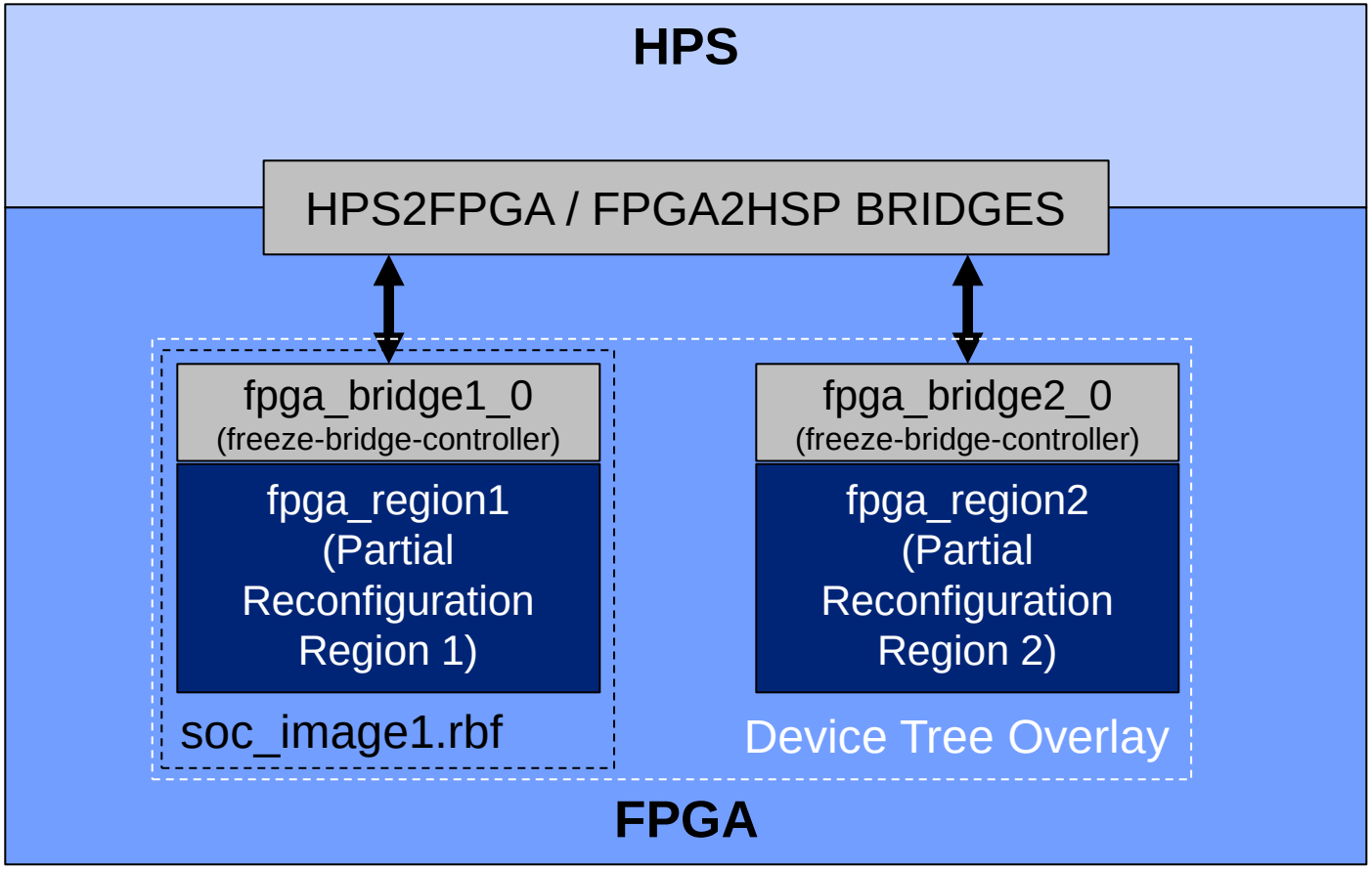
```
/dts-v1/ /plugin/;
/ {
     fragment@0 {
1.        target-path = "/soc/base_fpga_region";
          #address-cells = <1>;
          #size-cells = <1>;
          __overlay__ {
               #address-cells = <1>;
                  #size-cells = <1>;

2.        firmware-name = "base.rbf";

3.        fpga_bridge1_0: fpga_bridge@4400 {
               compatible = "altr,freeze-bridge-controller";
               reg = <0x4400 0x10>;
          };

          fpga_bridge2_0: fpga_bridge@4420 {
               compatible = "altr,freeze-bridge-controller";
               reg = <0x4420 0x10>;
          };

4.        fpga_region1 {
               compatible = "fpga-region";
               fpga-bridges = <&fpga_bridge1_0>;
               #address-cells = <0x1>;
               #size-cells = <0x1>;
               ranges;
          };

          fpga_region2 {
               compatible = "fpga-region";
               fpga-bridges = <&fpga_bridge2_0>;
               #address-cells = <0x1>;
               #size-cells = <0x1>;
               ranges;
          };
```

1. Contents of the overlay will be applied to target **base_fpga_region** under **soc** node in the live tree.
2. base.rbf configures the FPGA and sets up 2 Partial Reconfig Regions.
3. Each Region contains a soft FPGA bridge controlled by the freeze-bridge-controller driver.
4. Partial Reconfiguration Regions defined similar to **base_fpga_region**, using soft FPGA bridges.

Empty **ranges** property specifies that the parent and child address space is identical, and no address translation is required.

SOC COE

Device tree overlay – Example 3 – Reprogram a Partial Reconfiguration Region



SOC COE

Device tree overlay – Example 3 – Reprogram a Partial Reconfiguration Region

```
/dts-v1/ /plugin/;
/ {
     fragment@0 {
1.        target-path = "/soc/base_fpga_region/fpga_region1";
          #address-cells = <1>;
          #size-cells = <1>;
          __overlay__ {
               #address-cells = <1>;
                  #size-cells = <1>;

2.             firmware-name = "soc_image1.rbf";
               partial-fpga-config;
               region-unfreeze-timeout-us = <4>;
               region-freeze-timeout-us = <4>;

3.             gpio@10040 {
                    compatible = "altr,pio-1.0";
                    reg = <0x10040 0x20>;
                    clocks = <0x2>;
                    altr,gpio-bank-width = <0x4>;
                    resetvalue = <0x0>;
                    #gpio-cells = <0x2>;
                    gpio-controller;
               };
          };
     };
};
```

1. Contents of the overlay will be applied to target **fpga_region1** under **soc/base_fpga_region** node in the live tree.
2. Use of the **firmware-name** property and boolean **partial-fpga-config** indicates that the FPGA will be partially reconfigured with the given .rbf file, targeting **fpga_region1**.
3. If FPGA configuration is successful, the bridges are re-enabled and the child devices are populated to the live tree. If configuration fails, the bridges remain disabled and the overlay is rejected.

Loading the Overlay in Linux

- Makes use of Linux **configfs** to allow a userspace event to create a kernel object
  - A configfs config_item is created with **mkdir** and destroyed with **rmdir**.
  - See kernel source Documentation/filesystems/configfs/configfs.txt for complete details
- This can be accomplished with an init script added to root filesystem (examples next 2 slides) or by manually executing the commands in the script
  - SysVinit - /etc/init.d
  - systemd - /lib/systemd/system
- Copy .rbf and .dtb to /lib/firmware first
- Can be used to (re)configure the FPGA only without appending the base device tree
  - Replaces “cat” or “echo”-ing the .rbf to /dev/fpga0

Example SysVInit script

S50devicetree_overlay

```
#!/bin/sh
#

start() {
    echo "************************************************************************"
    echo "Applying overlay"
    echo "************************************************************************"
1.  mkdir /config
    mount -t configfs configfs /config
    mkdir /config/device-tree/overlays/my-board
    echo my-overlay.dtb > /config/device-tree/overlays/my-board/path
}
stop() {
    echo "************************************************************************"
    echo "Removing overlay"
    echo "************************************************************************"
2.  rmdir /config/device-tree/overlays/my-board
}
restart() {
        stop
        start
}

case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
  restart|reload)
        restart
        ;;
  *)
        echo "Usage: $0 {start|stop|restart}"
        exit 1
esac

exit $?
```

On a SysVInit system, this script is saved in the /etc/init.d folder.

1. At system start up, /config directory is created and mounted to configfs; directory is created for overlay file and file name is appended to **path** file
2. At system shutdown, overlay directory is removed

SOC COE

Example systemd script

/lib/systemd/system/devicetree-overlay.service

```
[Unit]
Description=Devicetree Overlay Service

[Service]
ExecStart=/usr/bin/devicetree-overlay.sh

[Install]
WantedBy=multi-user.target
```

Enable the service with
systemctl enable devicetree-overlay.service
to create a symlink to /etc/systemd/system

/usr/bin/devicetree-overlay.sh

```
#!/bin/sh

mkdir /config
mount -t configfs configfs /config
mkdir /config/device-tree/overlays/my-board
echo my-overlay.dtb > /config/device-tree/overlays/my-board/path
```

1.

1. At system start up, /config directory is created and mounted to configfs; directory is created for overlay file and file name is appended to **path** file. At system shutdown, overlay directory is removed when unmounting /config

SOC COE

Applying Overlay Summary

- Create overlay .dts file (use example as starting point). Base .dts file must use FPGA Regions framework.
- Compile the overlay .dts to create the overlay .dtb
- Copy .rbf (if used) and .dtb to /lib/firmware on target root filesystem
- Create init script and copy to target root filesystem OR manually execute the commands in the script
- Verify that the overlay was applied
  - cat /config/device-tree/overlays/my-board/status

Viewing Device Tree Info in Linux

- Buses, devices, device classes, drivers are internal to the kernel but are exported to user space through the **sysfs** virtual filesystem.
- Can be found in /sys (sysfs mounted in /sys)
  - /sys/bus – lists buses
  - /sys/devices – lists devices
  - /sys/class – enumerates devices by class
- View current system memory maps and all devices defined in device tree
  - cat /proc/iomem
- Additional information available in /proc/device-tree
  - ls /proc/device-tree

Creating a Device Tree for a Custom Board

Example device tree flow

- Clone the Altera kernel repository
- Create your board level .dts file (start with example from /arch/arm/boot/dts), and #include appropriate dtsi files in the kernel tree for your device (A10, C5, A5)
- Copy to /arch/arm/boot/dts
  - Typical naming convention is <soc-name>-<board-name>.dts
  - Altera uses socfpga_<device-family>_<board-name>.dts
    - socfpga_cyclone5_sockit.dts
    - socfpga_arria10_socdk_nand.dts
- Modify the kernel makefile to build your .dtb during kernel build
  - arch/arm/boot/dts/Makefile
  - add dtb-$(CONFIG_ARCH_SOCFPGA) += socfpga_cyclone5_myboard.dtb
- OR, build separately after kernel build
  - export ARCH and CROSS_COMPILE variables first or as part of make if not already done
  - make ARCH=arm CONFIG_DTB_SOURCE=arch/arm/boot/dts/my-board.dts my-board.dtb

Common Device Tree Problems

- This is by no means an exhaustive list
- Kernel hangs
  - often (but not always) indicates a problem with the device tree
- Driver fails to load
  - Check for device tree entry with compatible string for driver
  - Check that required properties are fully listed in the device tree as defined in the Device Tree binding .txt file in kernel source directory (Documentation/devicetree/bindings)
- Device driver loads but device not visible in Linux
  - Is the device enabled in the board level device tree? Many devices are disabled by default in the common .dtsi files.

References

- www.free-electrons.com
  - Linux kernel and driver development course
  - Device Tree for Dummies
- Linux kernel source Documentation folder
- Cyclone V Handbook
- http://elinux.org/Device_Tree_Usage
- ePAPR v1.1

Appendix – Fully Annotated Example Device Tree

Annotated Device Tree Example

- To create a complete device tree from scratch, one would need to know about all of the hardware available in the system
- Fortunately, the hierarchical device trees provided in the kernel source tree have already defined this for us for the Altera SoC FPGAs
- The following example uses the Arrow SoCKit device tree example provided in the kernel source tree at arch/arm/boot/dts/socfpga_cyclone5_sockit.dts
- The following slides will attempt to clearly define all the components of this device tree and describe what is required vs. optional

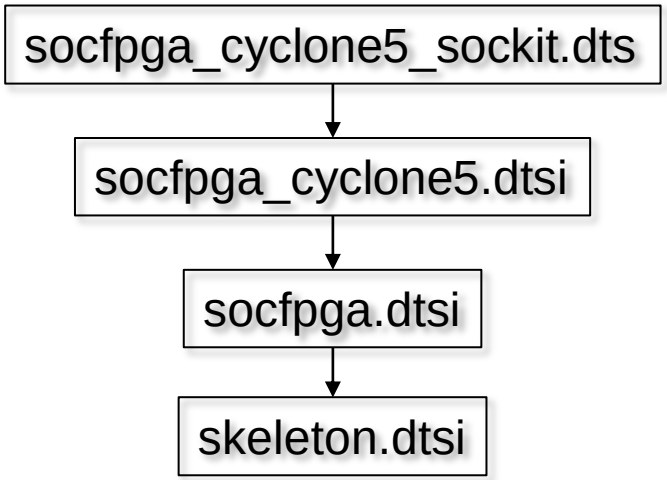
Consider changing to Chameleon96 board example since the SoCKit device tree example does not use FPGA Regions.

Annotated Device Tree Example

- The Cyclone V Hard Processor System Technical Reference Manual lists the HPS peripherals:
  - MPU subsystem featuring a single- or dual-core ARM Cortex-A9 MPCore processor
  - General-purpose direct memory access (DMA) controller
  - Two ethernet media access controllers (EMACs)
  - Two USB 2.0 on-the-go (OTG) controllers
  - NAND flash controller
  - Quad SPI flash controller
  - Secure digital/multimedia card (SD/MMC) controller
  - Two serial peripheral interface (SPI) master controllers
  - Two SPI slave controllers
  - Four inter-integrated circuit (I2C) controllers
  - 64 KB on-chip RAM
  - 64 KB on-chip boot ROM
  - Two UARTs
  - Four timers
  - Two watchdog timers
  - Three general-purpose I/O (GPIO) interfaces
  - Two controller area network (CAN) controllers
  - ARM Coresight™ debug components:
  - Debug access port (DAP)
  - Trace port interface unit (TPIU)
  - System trace macrocell (STM)
  - Program trace macrocell (PTM)
  - Embedded trace router (ETR)
  - Embedded cross trigger (ECT)
  - System manager
  - Clock manager
  - Reset manager
  - Scan manager
  - FPGA manager
  - SDRAM controller subsystem

Annotated Device Tree Example

- The base addresses for these peripherals are further defined in the Cyclone V SoC HPS Address Map and Register Definitions
- All of these peripherals should be defined in the device tree. In this example they are defined in socfpga.dtsi.
- Recall from the previous slide example how the hierarchy works using the kernel source tree dtsi files:



SOC COE

socfpga_cyclone5_sockit.dts (1/3)

```
/*
 *  Copyright (C) 2013 Steffen Trumtrar <s.trumtrar@pengutronix.de>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 */

#include "socfpga_cyclone5.dtsi"

/ {
            model = "Terasic SoCkit";
            compatible = "altr,socfpga-cyclone5", "altr,socfpga";

            chosen {
                        bootargs = "earlyprintk";
                        stdout-path = "serial0:115200n8";
            };

            memory {
                        name = "memory";
                        device_type = "memory";
                        reg = <0x0 0x40000000>; /* 1GB */
            };
```

Include this Cyclone V device tree when compiling

model: string to specify mfr. model number of device

Name of the system; can be used by OS to determine how to run on this machine or perform various checks

chosen: Default kernel command line arguments

memory: required for all device trees, specifies size and location of RAM

CONTINUED...

socfpga_cyclone5_sockit.dts (2/3)

```
            aliases {
                        /* this allow the ethaddr uboot environmnet variable contents
                         * to be added to the gmac1 device tree blob.
                         */
                        ethernet0 = &gmac1;
            };

            regulator_3_3v: vcc3p3-regulator {
                        compatible = "regulator-fixed";
                        regulator-name = "VCC3P3";
                        regulator-min-microvolt = <3300000>;
                        regulator-max-microvolt = <3300000>;
            };
};

&gmac1 {
            status = "okay";
            phy-mode = "rgmii";

            rxd0-skew-ps = <0>;
            rxd1-skew-ps = <0>;
            rxd2-skew-ps = <0>;
            rxd3-skew-ps = <0>;
            txen-skew-ps = <0>;
            txc-skew-ps = <2600>;
            rxdv-skew-ps = <0>;
            rxc-skew-ps = <2000>;
};

&mmc0 {
            vmmc-supply = <&regulator_3_3v>;
            vqmmc-supply = <&regulator_3_3v>;
};
```

aliases node must be at the root of the device tree

aliases: allows a node to be referred to by the alias name.
Note that there is also an **aliases** node in socfpga.dtsi that sets ethernet1 = &gmac1, but this setting at the highest level in the hierarchy overrides the setting in the lower level. Only gmac1 is used on the SoCKit board.

regulator entry allows for control of voltage regulators. See Documentation/devicetree/bindings/regulator/regulator.txt for additional options not used here.

gmac1 is enabled with status = “okay”. Use of phandle &gmac1 allows this to be referenced from socfpga.dtsi.

phandle to the regulator to use for vmmc for the mmc controller. If specified, probe is deferred until this regulator is found.

CONTINUED...

socfpga_cyclone5_sockit.dts (3/3)

```
&usb1 {
            status = "okay";
};

&qspi {
            flash0: n25q00@0 {
                        #address-cells = <1>;
                        #size-cells = <1>;
                        compatible = "n25q00";
                        reg = <0>;      /* chip select */
                        spi-max-frequency = <100000000>;
                        m25p,fast-read;
                        page-size = <256>;
                        block-size = <16>; /* 2^16, 64KB */
                        read-delay = <4>;  /* delay value in read data capture register */
                        tshsl-ns = <50>;
                        tsd2d-ns = <50>;
                        tchsh-ns = <4>;
                        tslch-ns = <4>;

                        partition@qspi-boot {
                                    /* 8MB for raw data. */
                                    label = "Flash 0 Raw Data";
                                    reg = <0x0 0x800000>;
                        };

                        partition@qspi-rootfs {
                                    /* 120MB for jffs2 data. */
                                    label = "Flash 0 jffs2 Filesystem";
                                    reg = <0x800000 0x7800000>;
                        };
            };
};
```

usb1 is enabled with status = “okay

SOC COE

socfpga_cyclone5.dtsi (1/2)

```
/*
 * Copyright Altera Corporation (C) 2012,2014. All rights reserved.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms and conditions of the GNU General Public License,
 * version 2, as published by the Free Software Foundation.
 *
 * This program is distributed in the hope it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License for
 * more details.
 *
 * You should have received a copy of the GNU General Public License along with
 * this program.  If not, see <http://www.gnu.org/licenses/>.
 */

/dts-v1/;
/* First 4KB has trampoline code for secondary cores. */
/memreserve/ 0x00000000 0x0001000;
#include "socfpga.dtsi"

/ {
            soc {
                        clkmgr@ffd04000 {
                                    clocks {
                                                osc1 {
                                                            clock-frequency = <25000000>;
                                                };
                                    };
                        };
                                                                                CONTINUED...
```

memreserve marks this memory space as reserved from the kernel

Include this common socfpga device tree when compiling

socfpga_cyclone5.dtsi (2/2)

```
                        mmc0: dwmmc0@ff704000 {
                                    num-slots = <1>;
                                    broken-cd;
                                    bus-width = <4>;
                                    cap-mmc-highspeed;
                                    cap-sd-highspeed;
                        };

                        ethernet@ff702000 {
                                    phy-mode = "rgmii";
                                    phy-addr = <0xffffffff>; /* probe for phy addr */
                                    status = "okay";
                        };

                        sysmgr@ffd08000 {
                                    cpu1-start-addr = <0xffd080c4>;
                        };
            };
};

&watchdog0 {
            status = "okay";
};
```

watchdog0 is enabled with status = “okay

SOC COE

socfpga.dtsi (1/28)

```
/*
 * Copyright Altera Corporation (C) 2012-2014. All rights reserved.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms and conditions of the GNU General Public License,
 * version 2, as published by the Free Software Foundation.
 *
 * This program is distributed in the hope it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License for
 * more details.
 *
 * You should have received a copy of the GNU General Public License along with
 * this program.  If not, see <http://www.gnu.org/licenses/>.
 */

#include "skeleton.dtsi"
#include <dt-bindings/reset/altr,rst-mgr.h>

/ {
     #address-cells = <1>;
     #size-cells = <1>;

     aliases {
          ethernet0 = &gmac0;
          ethernet1 = &gmac1;
          serial0 = &uart0;
          serial1 = &uart1;
          timer0 = &timer0;
          timer1 = &timer1;
          timer2 = &timer2;
          timer3 = &timer3;
     };
```

CONTINUED...

socfpga.dtsi (2/28)

```
cpus {
     #address-cells = <1>;
     #size-cells = <0>;
     enable-method = "altr,socfpga-smp";

     cpu@0 {
          compatible = "arm,cortex-a9";
          device_type = "cpu";
          reg = <0>;
          next-level-cache = <&L2>;
     };
     cpu@1 {
          compatible = "arm,cortex-a9";
          device_type = "cpu";
          reg = <1>;
          next-level-cache = <&L2>;
     };
};

intc: intc@fffed000 {
     compatible = "arm,cortex-a9-gic";
     #interrupt-cells = <3>;
     interrupt-controller;
     reg = <0xfffed000 0x1000>,
           <0xfffec100 0x100>;
};
```

CONTINUED...

socfpga.dtsi (3/28)

```
soc {
     #address-cells = <1>;
     #size-cells = <1>;
     compatible = "simple-bus";
     device_type = "soc";
     interrupt-parent = <&intc>;
     ranges;

     amba {
          compatible = "arm,amba-bus";
          #address-cells = <1>;
          #size-cells = <1>;
          ranges;

          pdma: pdma@ffe01000 {
               compatible = "arm,pl330", "arm,primecell";
               reg = <0xffe01000 0x1000>;
               interrupts = <0 104 4>,
                            <0 105 4>,
                            <0 106 4>,
                            <0 107 4>,
                            <0 108 4>,
                            <0 109 4>,
                            <0 110 4>,
                            <0 111 4>;
               #dma-cells = <1>;
               #dma-channels = <8>;
               #dma-requests = <32>;
               clocks = <&l4_main_clk>;
               clock-names = "apb_pclk";
          };
     };
```

CONTINUED...

socfpga.dtsi (4/28)

```
base_fpga_region: base-fpga-region {
     compatible = "fpga-region";
     fpga-mgr = <&fpga_mgr>;

     #address-cells = <0x1>;
     #size-cells = <0x1>;
};

can0: can@ffc00000 {
     compatible = "bosch,d_can";
     reg = <0xffc00000 0x1000>;
     interrupts = <0 131 4>, <0 132 4>, <0 133 4>, <0 134 4>;
     clocks = <&can0_clk>;
     status = "disabled";
};

can1: can@ffc01000 {
     compatible = "bosch,d_can";
     reg = <0xffc01000 0x1000>;
     interrupts = <0 135 4>, <0 136 4>, <0 137 4>, <0 138 4>;
     clocks = <&can1_clk>;
     status = "disabled";
};

clkmgr@ffd04000 {
          compatible = "altr,clk-mgr";
          reg = <0xffd04000 0x1000>;

          clocks {
               #address-cells = <1>;
               #size-cells = <0>;
```

CONTINUED...

socfpga.dtsi (5/28)

```
osc1: osc1 {
     #clock-cells = <0>;
     compatible = "fixed-clock";
};

osc2: osc2 {
     #clock-cells = <0>;
     compatible = "fixed-clock";
};

f2s_periph_ref_clk: f2s_periph_ref_clk {
     #clock-cells = <0>;
     compatible = "fixed-clock";
};

f2s_sdram_ref_clk: f2s_sdram_ref_clk {
     #clock-cells = <0>;
     compatible = "fixed-clock";
};

main_pll: main_pll {
     #address-cells = <1>;
     #size-cells = <0>;
     #clock-cells = <0>;
     compatible = "altr,socfpga-pll-clock";
     clocks = <&osc1>;
     reg = <0x40>;

     mpuclk: mpuclk {
          #clock-cells = <0>;
          compatible = "altr,socfpga-perip-clk";
          clocks = <&main_pll>;
          div-reg = <0xe0 0 9>;
          reg = <0x48>;
     };
```

CONTINUED...

socfpga.dtsi (6/28)

```
                mainclk: mainclk {
                        #clock-cells = <0>;
                        compatible = "altr,socfpga-perip-clk";
                        clocks = <&main_pll>;
                        div-reg = <0xe4 0 9>;
                        reg = <0x4C>;
                };

                dbg_base_clk: dbg_base_clk {
                        #clock-cells = <0>;
                        compatible = "altr,socfpga-perip-clk";
                        clocks = <&main_pll>, <&osc1>;
                        div-reg = <0xe8 0 9>;
                        reg = <0x50>;
                };

                main_qspi_clk: main_qspi_clk {
                        #clock-cells = <0>;
                        compatible = "altr,socfpga-perip-clk";
                        clocks = <&main_pll>;
                        reg = <0x54>;
                };

                main_nand_sdmmc_clk: main_nand_sdmmc_clk {
                        #clock-cells = <0>;
                        compatible = "altr,socfpga-perip-clk";
                        clocks = <&main_pll>;
                        reg = <0x58>;
                };

                cfg_h2f_usr0_clk: cfg_h2f_usr0_clk {
                        #clock-cells = <0>;
                        compatible = "altr,socfpga-perip-clk";
                        clocks = <&main_pll>;
                        reg = <0x5C>;
                };
        };
```

CONTINUED...

socfpga.dtsi (7/28)

```
periph_pll: periph_pll {
     #address-cells = <1>;
     #size-cells = <0>;
     #clock-cells = <0>;
     compatible = "altr,socfpga-pll-clock";
     clocks = <&osc1>, <&osc2>, <&f2s_periph_ref_clk>;
     reg = <0x80>;

     emac0_clk: emac0_clk {
          #clock-cells = <0>;
          compatible = "altr,socfpga-perip-clk";
          clocks = <&periph_pll>;
          reg = <0x88>;
     };

     emac1_clk: emac1_clk {
          #clock-cells = <0>;
          compatible = "altr,socfpga-perip-clk";
          clocks = <&periph_pll>;
          reg = <0x8C>;
     };

     per_qspi_clk: per_qsi_clk {
          #clock-cells = <0>;
          compatible = "altr,socfpga-perip-clk";
          clocks = <&periph_pll>;
          reg = <0x90>;
     };

     per_nand_mmc_clk: per_nand_mmc_clk {
          #clock-cells = <0>;
          compatible = "altr,socfpga-perip-clk";
          clocks = <&periph_pll>;
          reg = <0x94>;
     };
```

CONTINUED...

socfpga.dtsi (8/28)

```
                per_base_clk: per_base_clk {
                        #clock-cells = <0>;
                        compatible = "altr,socfpga-perip-clk";
                        clocks = <&periph_pll>;
                        reg = <0x98>;
                };

                h2f_usr1_clk: h2f_usr1_clk {
                        #clock-cells = <0>;
                        compatible = "altr,socfpga-perip-clk";
                        clocks = <&periph_pll>;
                        reg = <0x9C>;
                };
        };

        sdram_pll: sdram_pll {
                #address-cells = <1>;
                #size-cells = <0>;
                #clock-cells = <0>;
                compatible = "altr,socfpga-pll-clock";
                clocks = <&osc1>, <&osc2>, <&f2s_sdram_ref_clk>;
                reg = <0xC0>;

                ddr_dqs_clk: ddr_dqs_clk {
                        #clock-cells = <0>;
                        compatible = "altr,socfpga-perip-clk";
                        clocks = <&sdram_pll>;
                        reg = <0xC8>;
                };

                ddr_2x_dqs_clk: ddr_2x_dqs_clk {
                        #clock-cells = <0>;
                        compatible = "altr,socfpga-perip-clk";
                        clocks = <&sdram_pll>;
                        reg = <0xCC>;
                };
```

CONTINUED...

SOC COE

socfpga.dtsi (9/28)

```
                ddr_dq_clk: ddr_dq_clk {
                        #clock-cells = <0>;
                        compatible = "altr,socfpga-perip-clk";
                        clocks = <&sdram_pll>;
                        reg = <0xD0>;
                };

                h2f_usr2_clk: h2f_usr2_clk {
                        #clock-cells = <0>;
                        compatible = "altr,socfpga-perip-clk";
                        clocks = <&sdram_pll>;
                        reg = <0xD4>;
                };
        };

        mpu_periph_clk: mpu_periph_clk {
                #clock-cells = <0>;
                compatible = "altr,socfpga-perip-clk";
                clocks = <&mpuclk>;
                fixed-divider = <4>;
        };

        mpu_l2_ram_clk: mpu_l2_ram_clk {
                #clock-cells = <0>;
                compatible = "altr,socfpga-perip-clk";
                clocks = <&mpuclk>;
                fixed-divider = <2>;
        };

        l4_main_clk: l4_main_clk {
                #clock-cells = <0>;
                compatible = "altr,socfpga-gate-clk";
                clocks = <&mainclk>;
                clk-gate = <0x60 0>;
        };
```

CONTINUED...

socfpga.dtsi (10/28)

```
l3_main_clk: l3_main_clk {
     #clock-cells = <0>;
     compatible = "altr,socfpga-perip-clk";
     clocks = <&mainclk>;
     fixed-divider = <1>;
};

l3_mp_clk: l3_mp_clk {
     #clock-cells = <0>;
     compatible = "altr,socfpga-gate-clk";
     clocks = <&mainclk>;
     div-reg = <0x64 0 2>;
     clk-gate = <0x60 1>;
};

l3_sp_clk: l3_sp_clk {
     #clock-cells = <0>;
     compatible = "altr,socfpga-gate-clk";
     clocks = <&l3_mp_clk>;
     div-reg = <0x64 2 2>;
};

l4_mp_clk: l4_mp_clk {
     #clock-cells = <0>;
     compatible = "altr,socfpga-gate-clk";
     clocks = <&mainclk>, <&per_base_clk>;
     div-reg = <0x64 4 3>;
     clk-gate = <0x60 2>;
};

l4_sp_clk: l4_sp_clk {
     #clock-cells = <0>;
     compatible = "altr,socfpga-gate-clk";
     clocks = <&mainclk>, <&per_base_clk>;
     div-reg = <0x64 7 3>;
     clk-gate = <0x60 3>;
};
```

CONTINUED...

socfpga.dtsi (11/28)

```
dbg_at_clk: dbg_at_clk {
     #clock-cells = <0>;
     compatible = "altr,socfpga-gate-clk";
     clocks = <&dbg_base_clk>;
     div-reg = <0x68 0 2>;
     clk-gate = <0x60 4>;
};

dbg_clk: dbg_clk {
     #clock-cells = <0>;
     compatible = "altr,socfpga-gate-clk";
     clocks = <&dbg_at_clk>;
     div-reg = <0x68 2 2>;
     clk-gate = <0x60 5>;
};

dbg_trace_clk: dbg_trace_clk {
     #clock-cells = <0>;
     compatible = "altr,socfpga-gate-clk";
     clocks = <&dbg_base_clk>;
     div-reg = <0x6C 0 3>;
     clk-gate = <0x60 6>;
};

dbg_timer_clk: dbg_timer_clk {
     #clock-cells = <0>;
     compatible = "altr,socfpga-gate-clk";
     clocks = <&dbg_base_clk>;
     clk-gate = <0x60 7>;
};
```

CONTINUED...

socfpga.dtsi (12/28)

```
 cfg_clk: cfg_clk {
     #clock-cells = <0>;
     compatible = "altr,socfpga-gate-clk";
     clocks = <&cfg_h2f_usr0_clk>;
     clk-gate = <0x60 8>;
};

h2f_user0_clk: h2f_user0_clk {
     #clock-cells = <0>;
     compatible = "altr,socfpga-gate-clk";
     clocks = <&cfg_h2f_usr0_clk>;
     clk-gate = <0x60 9>;
};

emac_0_clk: emac_0_clk {
     #clock-cells = <0>;
     compatible = "altr,socfpga-gate-clk";
     clocks = <&emac0_clk>;
     clk-gate = <0xa0 0>;
};

emac_1_clk: emac_1_clk {
     #clock-cells = <0>;
     compatible = "altr,socfpga-gate-clk";
     clocks = <&emac1_clk>;
     clk-gate = <0xa0 1>;
};

usb_mp_clk: usb_mp_clk {
     #clock-cells = <0>;
     compatible = "altr,socfpga-gate-clk";
     clocks = <&per_base_clk>;
     clk-gate = <0xa0 2>;
     div-reg = <0xa4 0 3>;
};
```

CONTINUED...

socfpga.dtsi (13/28)

```
spi_m_clk: spi_m_clk {
     #clock-cells = <0>;
     compatible = "altr,socfpga-gate-clk";
     clocks = <&per_base_clk>;
     clk-gate = <0xa0 3>;
     div-reg = <0xa4 3 3>;
};

can0_clk: can0_clk {
     #clock-cells = <0>;
     compatible = "altr,socfpga-gate-clk";
     clocks = <&per_base_clk>;
     clk-gate = <0xa0 4>;
     div-reg = <0xa4 6 3>;
};

can1_clk: can1_clk {
     #clock-cells = <0>;
     compatible = "altr,socfpga-gate-clk";
     clocks = <&per_base_clk>;
     clk-gate = <0xa0 5>;
     div-reg = <0xa4 9 3>;
};

gpio_db_clk: gpio_db_clk {
     #clock-cells = <0>;
     compatible = "altr,socfpga-gate-clk";
     clocks = <&per_base_clk>;
     clk-gate = <0xa0 6>;
     div-reg = <0xa8 0 24>;
};
```

CONTINUED...

socfpga.dtsi (14/28)

```
 h2f_user1_clk: h2f_user1_clk {
      #clock-cells = <0>;
      compatible = "altr,socfpga-gate-clk";
      clocks = <&h2f_usr1_clk>;
      clk-gate = <0xa0 7>;
};

sdmmc_clk: sdmmc_clk {
      #clock-cells = <0>;
      compatible = "altr,socfpga-gate-clk";
      clocks = <&f2s_periph_ref_clk>, <&main_nand_sdmmc_clk>, <&per_nand_mmc_clk>;
      clk-gate = <0xa0 8>;
      clk-phase = <0 135>;
};

sdmmc_clk_divided: sdmmc_clk_divided {
      #clock-cells = <0>;
      compatible = "altr,socfpga-gate-clk";
      clocks = <&sdmmc_clk>;
      clk-gate = <0xa0 8>;
      fixed-divider = <4>;
};

nand_x_clk: nand_x_clk {
      #clock-cells = <0>;
      compatible = "altr,socfpga-gate-clk";
      clocks = <&f2s_periph_ref_clk>, <&main_nand_sdmmc_clk>, <&per_nand_mmc_clk>;
      clk-gate = <0xa0 9>;
};

nand_clk: nand_clk {
      #clock-cells = <0>;
      compatible = "altr,socfpga-gate-clk";
      clocks = <&f2s_periph_ref_clk>, <&main_nand_sdmmc_clk>, <&per_nand_mmc_clk>;
      clk-gate = <0xa0 10>;
      fixed-divider = <4>;
};                                                          CONTINUED...
```

socfpga.dtsi (15/28)

```
                qspi_clk: qspi_clk {
                     #clock-cells = <0>;
                     compatible = "altr,socfpga-gate-clk";
                     clocks = <&f2s_periph_ref_clk>, <&main_qspi_clk>, <&per_qspi_clk>;
                     clk-gate = <0xa0 11>;
                };
           };
     };

fpga_bridge0: fpga-bridge@ff400000 {
     compatible = "altr,socfpga-lwhps2fpga-bridge";
     reg = <0xff400000 0x100000>;
     resets = <&rst LWHPS2FPGA_RESET>;
     reset-names = "lwhps2fpga";
     clocks = <&l4_main_clk>;
};

fpga_bridge1: fpga-bridge@ff500000 {
     compatible = "altr,socfpga-hps2fpga-bridge";
     reg = <0xff500000 0x10000>;
     resets = <&rst HPS2FPGA_RESET>;
     reset-names = "hps2fpga";
     clocks = <&l4_main_clk>;
};

fpga_bridge2: fpga-bridge@ff600000 {
     compatible = "altr,socfpga-fpga2hps-bridge";
     reg = <0xff600000 0x100000>;
     resets = <&rst FPGA2HPS_RESET>;
     reset-names = "fpga2hps";
     clocks = <&l4_main_clk>;
};

fpga_bridge3: fpga2sdram-bridge {
     compatible = "altr,socfpga-fpga2sdram-bridge";
};                                                                    CONTINUED...
```

SOC COE

socfpga.dtsi (16/28)

```
 fpga_mgr: fpga-mgr@ff706000 {
     compatible = "altr,socfpga-fpga-mgr";
     reg = <0xff706000 0x1000
            0xffb90000 0x20>;
     interrupts = <0 175 4>;
};

gmac0: ethernet@ff700000 {
     compatible = "altr,socfpga-stmmac", "snps,dwmac-3.70a", "snps,dwmac";
     altr,sysmgr-syscon = <&sysmgr 0x60 0>;
     reg = <0xff700000 0x2000>;
     interrupts = <0 115 4>;
     interrupt-names = "macirq";
     mac-address = [00 00 00 00 00 00];/* Filled in by U-Boot */
     clocks = <&emac0_clk>;
     clock-names = "stmmaceth";
     resets = <&rst EMAC0_RESET>;
     reset-names = "stmmaceth";
     snps,multicast-filter-bins = <256>;
     snps,perfect-filter-entries = <128>;
     tx-fifo-depth = <4096>;
     rx-fifo-depth = <4096>;
     status = "disabled";
};
```

CONTINUED...

socfpga.dtsi (17/28)

```
 gmac1: ethernet@ff702000 {
     compatible = "altr,socfpga-stmmac", "snps,dwmac-3.70a", "snps,dwmac";
     altr,sysmgr-syscon = <&sysmgr 0x60 2>;
     reg = <0xff702000 0x2000>;
     interrupts = <0 120 4>;
     interrupt-names = "macirq";
     mac-address = [00 00 00 00 00 00];/* Filled in by U-Boot */
     clocks = <&emac1_clk>;
     clock-names = "stmmaceth";
     resets = <&rst EMAC1_RESET>;
     reset-names = "stmmaceth";
     snps,multicast-filter-bins = <256>;
     snps,perfect-filter-entries = <128>;
     tx-fifo-depth = <4096>;
     rx-fifo-depth = <4096>;
     status = "disabled";
};

i2c0: i2c@ffc04000 {
     #address-cells = <1>;
     #size-cells = <0>;
     compatible = "snps,designware-i2c";
     reg = <0xffc04000 0x1000>;
     clocks = <&l4_sp_clk>;
     interrupts = <0 158 0x4>;
     status = "disabled";
};

i2c1: i2c@ffc05000 {
     #address-cells = <1>;
     #size-cells = <0>;
     compatible = "snps,designware-i2c";
     reg = <0xffc05000 0x1000>;
     clocks = <&l4_sp_clk>;
     interrupts = <0 159 0x4>;
     status = "disabled";
};
```

CONTINUED...

socfpga.dtsi (18/28)

```
 i2c2: i2c@ffc06000 {
      #address-cells = <1>;
      #size-cells = <0>;
      compatible = "snps,designware-i2c";
      reg = <0xffc06000 0x1000>;
      clocks = <&l4_sp_clk>;
      interrupts = <0 160 0x4>;
      status = "disabled";
};

i2c3: i2c@ffc07000 {
      #address-cells = <1>;
      #size-cells = <0>;
      compatible = "snps,designware-i2c";
      reg = <0xffc07000 0x1000>;
      clocks = <&l4_sp_clk>;
      interrupts = <0 161 0x4>;
      status = "disabled";
};

gpio0: gpio@ff708000 {
      #address-cells = <1>;
      #size-cells = <0>;
      compatible = "snps,dw-apb-gpio";
      reg = <0xff708000 0x1000>;
      clocks = <&l4_mp_clk>;
      status = "disabled";
```

CONTINUED...

socfpga.dtsi (19/28)

```
                porta: gpio-controller@0 {
                        compatible = "snps,dw-apb-gpio-port";
                        gpio-controller;
                        #gpio-cells = <2>;
                        snps,nr-gpios = <29>;
                        reg = <0>;
                        interrupt-controller;
                        #interrupt-controller = <2>;
                        interrupts = <0 164 4>;
                };
        };

        gpio1: gpio@ff709000 {
                #address-cells = <1>;
                #size-cells = <0>;
                compatible = "snps,dw-apb-gpio";
                reg = <0xff709000 0x1000>;
                clocks = <&l4_mp_clk>;
                status = "disabled";

                portb: gpio-controller@0 {
                        compatible = "snps,dw-apb-gpio-port";
                        gpio-controller;
                        #gpio-cells = <2>;
                        snps,nr-gpios = <29>;
                        reg = <0>;
                        interrupt-controller;
                        #interrupt-cells = <2>;
                        interrupts = <0 165 4>;
                };
        };
```

CONTINUED...

socfpga.dtsi (20/28)

```
gpio2: gpio@ff70a000 {
              #address-cells = <1>;
              #size-cells = <0>;
              compatible = "snps,dw-apb-gpio";
              reg = <0xff70a000 0x1000>;
              clocks = <&l4_mp_clk>;
              status = "disabled";

              portc: gpio-controller@0 {
                   compatible = "snps,dw-apb-gpio-port";
                   gpio-controller;
                   #gpio-cells = <2>;
                   snps,nr-gpios = <27>;
                   reg = <0>;
                   interrupt-controller;
                   #interrupt-cells = <2>;
                   interrupts = <0 166 4>;
              };
         };

         sdr: sdr@ffc25000 {
              compatible = "altr,sdr-ctl", "syscon";
              reg = <0xffc25000 0x1000>;
         };

         sdramedac {
              compatible = "altr,sdram-edac";
              altr,sdr-syscon = <&sdr>;
              interrupts = <0 39 4>;
         };
```

CONTINUED...

socfpga.dtsi (21/28)

```
L2: l2-cache@fffef000 {
     compatible = "arm,pl310-cache", "syscon";
     reg = <0xfffef000 0x1000>;
     interrupts = <0 38 0x04>;
     cache-unified;
     cache-level = <2>;
     arm,tag-latency = <1 1 1>;
     arm,data-latency = <2 1 1>;
     prefetch-data = <1>;
     prefetch-instr = <1>;
};

l3regs@0xff800000 {
     compatible = "altr,l3regs", "syscon";
     reg = <0xff800000 0x1000>;
};

mmc: dwmmc0@ff704000 {
     compatible = "altr,socfpga-dw-mshc";
     reg = <0xff704000 0x1000>;
     interrupts = <0 139 4>;
     fifo-depth = <0x400>;
     #address-cells = <1>;
     #size-cells = <0>;
     clocks = <&l4_mp_clk>, <&sdmmc_clk_divided>;
     clock-names = "biu", "ciu";
};

ocram: sram@ffff0000 {
     compatible = "mmio-sram";
     reg = <0xffff0000 0x10000>;
};
```

CONTINUED...

socfpga.dtsi (22/28)

```
nand: nand@ff900000 {
     #address-cells = <1>;
     #size-cells = <1>;
     compatible = "denali,denali-nand-dt";
     reg = <0xff900000 0x100000>, <0xffb80000 0x10000>;
     reg-names = "nand_data", "denali_reg";
     interrupts = <0 144 4>;
     dma-mask = <0xffffffff>;
     clocks = <&nand_clk>;
     have-hw-ecc-fixup;
     status = "disabled";

     partition@nand-boot {
          /* 8MB for raw data. */
          label = "NAND Flash Boot Area 8MB";
          reg = <0x0 0x800000>;
     };
     partition@nand-rootfs {
          /* 128MB jffs2 root filesystem. */
          label = "NAND Flash jffs2 Root Filesystem 128MB";
          reg = <0x800000 0x8000000>;
     };
     partition@nand-128 {
          label = "NAND Flash 128 MB";
          reg = <0x8800000 0x8000000>;
     };
     partition@nand-64 {
          label = "NAND Flash 64 MB";
          reg = <0x10800000 0x4000000>;
     };
     partition@nand-32 {
          label = "NAND Flash 32 MB";
          reg = <0x14800000 0x2000000>;
     };
```

CONTINUED...

socfpga.dtsi (23/28)

```
                partition@nand-16 {
                     label = "NAND Flash 16 MB";
                     reg = <0x16800000 0x1000000>;
                };
        };

        l2edac@xffd08140 {
                compatible = "altr,l2-edac";
                reg = <0xffd08140 0x4>;
                interrupts = <0 36 1>, <0 37 1>;
                 };

        ocramedac@ffd08144 {
                compatible = "altr,ocram-edac";
                reg = <0xffd08144 0x4>;
                iram = <&ocram>;
                interrupts = <0 178 1>, <0 179 1>;
        };

        qspi: spi@ff705000 {
                compatible = "cdns,qspi-nor";
                #address-cells = <1>;
                #size-cells = <0>;
                reg = <0xff705000 0x1000>,
                     <0xffa00000 0x1000>;
                interrupts = <0 151 4>;
                clocks = <&qspi_clk>;
                is-decoded-cs = <1>;
                fifo-depth = <128>;
                status = "disabled";
                m25p,fast-read;
        };
```

CONTINUED...

socfpga.dtsi (24/28)

```
spi0: spi@fff00000 {
     compatible = "snps,dw-apb-ssi";
     #address-cells = <1>;
     #size-cells = <0>;
     reg = <0xfff00000 0x1000>;
     interrupts = <0 154 4>;
     num-cs = <4>;
     tx-dma-channel = <&pdma 16>;
     rx-dma-channel = <&pdma 17>;
     clocks = <&per_base_clk>;
     status = "disabled";
};

scu: snoop-control-unit@fffec000 {
     compatible = "arm,cortex-a9-scu";
     reg = <0xfffec000 0x100>;
};

spi1: spi@fff01000 {
     compatible = "snps,dw-apb-ssi";
     #address-cells = <1>;
     #size-cells = <0>;
     reg = <0xfff01000 0x1000>;
     interrupts = <0 155 4>;
     num-cs = <4>;
     tx-dma-channel = <&pdma 20>;
     rx-dma-channel = <&pdma 21>;
     clocks = <&per_base_clk>;
     status = "disabled";
};
```

CONTINUED...

socfpga.dtsi (25/28)

```
/* Local timer */
timer@fffec600 {
     compatible = "arm,cortex-a9-twd-timer";
     reg = <0xfffec600 0x100>;
     interrupts = <1 13 0xf04>;
     clocks = <&mpu_periph_clk>;
};

timer0: timer0@ffc08000 {
     compatible = "snps,dw-apb-timer";
     interrupts = <0 167 4>;
     reg = <0xffc08000 0x1000>;
     clocks = <&l4_sp_clk>;
     clock-names = "timer";
};

timer1: timer1@ffc09000 {
     compatible = "snps,dw-apb-timer";
     interrupts = <0 168 4>;
     reg = <0xffc09000 0x1000>;
     clocks = <&l4_sp_clk>;
     clock-names = "timer";
};

timer2: timer2@ffd00000 {
     compatible = "snps,dw-apb-timer";
     interrupts = <0 169 4>;
     reg = <0xffd00000 0x1000>;
     clocks = <&osc1>;
     clock-names = "timer";
};
```

CONTINUED...

socfpga.dtsi (26/28)

```
timer3: timer3@ffd01000 {
     compatible = "snps,dw-apb-timer";
     interrupts = <0 170 4>;
     reg = <0xffd01000 0x1000>;
     clocks = <&osc1>;
     clock-names = "timer";
};

uart0: serial0@ffc02000 {
     compatible = "snps,dw-apb-uart";
     reg = <0xffc02000 0x1000>;
     interrupts = <0 162 4>;
     reg-shift = <2>;
     reg-io-width = <4>;
     clocks = <&l4_sp_clk>;
     dmas = <&pdma 28>,
            <&pdma 29>;
     dma-names = "tx", "rx";
};

uart1: serial1@ffc03000 {
     compatible = "snps,dw-apb-uart";
     reg = <0xffc03000 0x1000>;
     interrupts = <0 163 4>;
     reg-shift = <2>;
     reg-io-width = <4>;
     clocks = <&l4_sp_clk>;
     dmas = <&pdma 30>,
            <&pdma 31>;
     dma-names = "tx", "rx";
};
```

CONTINUED...

socfpga.dtsi (27/28)

```
rst: rstmgr@ffd05000 {
     #reset-cells = <1>;
     compatible = "altr,rst-mgr";
     reg = <0xffd05000 0x1000>;
     altr,modrst-offset = <0x10>;
};

usbphy0: usbphy@0 {
     #phy-cells = <0>;
     compatible = "usb-nop-xceiv";
     status = "okay";
};

usb0: usb@ffb00000 {
     compatible = "snps,dwc2";
     reg = <0xffb00000 0xffff>;
     interrupts = <0 125 4>;
     clocks = <&usb_mp_clk>;
     clock-names = "otg";
     phys = <&usbphy0>;
     phy-names = "usb2-phy";
     status = "disabled";
};

usb1: usb@ffb40000 {
     compatible = "snps,dwc2";
     reg = <0xffb40000 0xffff>;
     interrupts = <0 128 4>;
     clocks = <&usb_mp_clk>;
     clock-names = "otg";
     phys = <&usbphy0>;
     phy-names = "usb2-phy";
     status = "disabled";
};
```

CONTINUED...

socfpga.dtsi (28/28)

```
            watchdog0: watchdog@ffd02000 {
                 compatible = "snps,dw-wdt";
                 reg = <0xffd02000 0x1000>;
                 interrupts = <0 171 4>;
                 clocks = <&osc1>;
                 status = "disabled";
            };

            watchdog1: watchdog@ffd03000 {
                 compatible = "snps,dw-wdt";
                 reg = <0xffd03000 0x1000>;
                 interrupts = <0 172 4>;
                 clocks = <&osc1>;
                 status = "disabled";
            };

            sysmgr: sysmgr@ffd08000 {
                 compatible = "altr,sys-mgr", "syscon";
                 reg = <0xffd08000 0x4000>;
            };
      };
};
```

skeleton.dtsi

```
/*
 * Skeleton device tree; the bare minimum needed to boot; just include and
 * add a compatible value.  The bootloader will typically populate the memory
 * node.
 */

/ {
            #address-cells = <1>;
            #size-cells = <1>;
            chosen { };
            aliases { };
            memory { device_type = "memory"; reg = <0 0>; };
};
```

SOC COE

Thank you