Flattened Device Trees for embedded FreeBSD

Rafał Jaworowski
raj@semihalf.com, raj@FreeBSD.org

BSDCan 2010, Ottawa

Presentation outline

- Introduction
- Integration of FDT with FreeBSD
  - Tools, environment
  - loader(8)
  - kernel
- Using the FDT
- Current state summary
  - ARM, PowerPC

Introduction

- Problem
  - Embedded systems vary greatly in the design, components interconnects and non-enumerable resources utilization
  - Simple firmware / early stage bootloaders (typically no comprehensive and uniform data about hardware configuration delivered to the OS)
  - Challenge: describe non-enumerable resources of a computer system in a portable way

Introduction cont'd

- Examples
  - Memory layout (offsets, ranges)
  - Network MAC-PHY binding
  - Interrupts hierarchy and IRQ lines routing
  - GPIO / multi-purpose pin assignment
  - I^2C slave id (address)
- Current embedded FreeBSD approaches

Why FDT?

- **Flattened Device Tree overview**
  - Established and mature, independent of platform and architecture
  - Embraced by the Power.org for the *ePAPR* specification
  - Central idea inherited from Open Firmware (IEEE 1275) *device-tree* notion
  - Does NOT require OF (or any other specific firmware)

FDT basics

- Hardware platform resources described in human readable text source format
- The source description is converted (*compiled*) into a binary object i.e. the flattened device tree
- The OS (kernel, drivers) learns about hardware resources from this blob without any a priori knowledge

Terminology, definitions

- Device Tree Source (DTS)
- Device Tree Blob (DTB)
  - Big endian
- Device Tree Compiler (DTC)
- *Bindings* definitions
  - Content conventions
  - Define meaning of allowed values and their ranges
  - Specific to a particular node type

DTS example

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

        PowerPC,8555@0 {
                device_type = "cpu";
                reg = <0x0>;
                d-cache-line-size = <32>;       // 32 bytes
                i-cache-line-size = <32>;       // 32 bytes
                d-cache-size = <0x8000>;        // L1, 32K
                i-cache-size = <0x8000>;        // L1, 32K
                timebase-frequency = <0>;
                bus-frequency = <0>;
                clock-frequency = <0>;
                next-level-cache = <&L2>;
        };
};

memory {
        device_type = "memory";
        reg = <0x0 0x8000000>;  // 128M at 0x0
};
```

DTS example cont'd

```
soc8555@e0000000 {
        #address-cells = <1>;
        #size-cells = <1>;
        device_type = "soc";
        compatible = "simple-bus";
        ranges = <0x0 0xe0000000 0x100000>;
        bus-frequency = <0>;

        ...

        i2c@3000 {
                #address-cells = <1>;
                #size-cells = <0>;
                compatible = "fsl-i2c";
                reg = <0x3000 0x100>;
                interrupts = <43 2>;
                interrupt-parent = <&mpic>;
                dfsrr;
        };

        ...

        enet0: ethernet@24000 {
                #address-cells = <1>;
                #size-cells = <1>;
                device_type = "network";
                model = "TSEC";
                compatible = "gianfar";
                reg = <0x24000 0x1000>;
                ranges = <0x0 0x24000 0x1000>;
                local-mac-address = [ 00 00 00 00 00 00 ];
                interrupts = <29 2 30 2 34 2>;
                interrupt-parent = <&mpic>;
                tbi-handle = <&tbi0>;
                phy-handle = <&phy0>;
        };
...
```

Integrating FDT with FreeBSD

- Reuse of existing tools
  - *dtc* package
  - The device tree compiler utility: *dtc*
  - Helper library: *libfdt*
- Compliancy with native FreeBSD interfaces and frameworks
- Baseline code, build environment
  - FreeBSD 9-CURRENT (Nov 2009)

Integration areas

- Build system
  - WITH_FDT knob
  - The *dtc* utility as a bootstrap tool
- loader(8)
  - For platforms with regular booting environment
  - Reusing *libfdt*
- FreeBSD kernel support
  - Reusing *libfdt*

Usage scenarios

- **Stand-alone device tree blob**
  - Full FreeBSD booting set-up, using loader(8)
  - FDT blob is stand-alone i.e. a physically separate file
  - Delivered to the kernel by loader(8)
- **Statically embedded blob**
  - Simplified booting environments, no loader(8)
  - DTB is integral part of the kernel image file
- **Both cases: the DTB is prepared beforehand**

loader(8) extensions

- Leverage existing mechanisms
  - The stand-alone DTB file treated as yet another type of a raw binary kernel module
  - Loaded and unloaded before kernel boot
- Dedicated *fdt* command
  - Inspect and manipulate the loaded blob

```
fdt cd <fdt_path>
fdt header
fdt ls [fdt_path]
fdt mknode [fdt_path/]<node_name>
fdt mkprop [node_path/]<property_name> <string | [ byte1 byte2 .. ] | <uint32_1 uint32_2 .. > >
fdt prop [node_path/[prop_name value_to_set]]
fdt pwd
fdt rm [node_path/]<node_name | property_name>
```

loader(8) examples

```
loader> load -t dtb boot/mpc8555cds.dtb

loader> lsmod
 ...
 0x162f92c: boot/mpc8555cds.dtb (dtb, 0x1eb2)
loader>
```

```
loader> fdt header

Flattened device tree header (0x162f92c):
 magic                   = 0xd00dfeed
 size                    = 7858
 off_dt_struct           = 0x00000038
 off_dt_strings          = 0x000018ac
 off_mem_rsvmap          = 0x00000028
 version                 = 17
 last compatible version = 16
 boot_cpuid              = 0
 size_dt_strings         = 518
 size_dt_struct          = 6260
loader>
```

```
loader> fdt prop /cpus/PowerPC,8572@0
device_type = "cpu"
reg = <0x00000000>
d-cache-line-size = <0x00000020>
i-cache-line-size = <0x00000020>
d-cache-size = <0x00008000>
i-cache-size = <0x00008000>
timebase-frequency = <0x00000000>
bus-frequency = <0x23c34600>
clock-frequency = <0x00000000>
next-level-cache = <0x00000001>

loader> fdt prop /cpus/PowerPC,8572@0/clock-frequency <15000000>
loader> fdt prop /cpus/PowerPC,8572@0
...
clock-frequency = <0x00e4e1c0>
...
```

loader(8) examples cont'd

```
loader> fdt ls

/aliases
/cpus
/cpus/PowerPC,8555@0
/memory
/soc8555@e0000000
/soc8555@e0000000/ecm-law@0
/soc8555@e0000000/ecm@1000
/soc8555@e0000000/memory-controller@2000
/soc8555@e0000000/l2-cache-controller@20000
/soc8555@e0000000/i2c@3000
/soc8555@e0000000/dma@21300
/soc8555@e0000000/dma@21300/dma-channel@0
/soc8555@e0000000/dma@21300/dma-channel@80
/soc8555@e0000000/dma@21300/dma-channel@100
/soc8555@e0000000/dma@21300/dma-channel@180
/soc8555@e0000000/ethernet@24000
/soc8555@e0000000/ethernet@24000/mdio@520
/soc8555@e0000000/ethernet@24000/mdio@520/ethernet-phy@0
/soc8555@e0000000/ethernet@24000/mdio@520/ethernet-phy@1
/soc8555@e0000000/ethernet@24000/mdio@520/tbi-phy@11
/soc8555@e0000000/ethernet@25000
/soc8555@e0000000/ethernet@25000/mdio@520
/soc8555@e0000000/ethernet@25000/mdio@520/tbi-phy@11
/soc8555@e0000000/serial@4500
/soc8555@e0000000/serial@4600
/soc8555@e0000000/crypto@30000
/soc8555@e0000000/pic@40000
/soc8555@e0000000/cpm@919c0
/soc8555@e0000000/cpm@919c0/muram@80000
/soc8555@e0000000/cpm@919c0/muram@80000/data@0
/soc8555@e0000000/cpm@919c0/brg@919f0
/soc8555@e0000000/cpm@919c0/pic@90c00
/pci@e0008000
/pci@e0008000/i8259@19000
/pci@e0009000
loader>
```

FreeBSD kernel and FDT

- Early system initialization
  - Adapt to FDT-based approach
- Integration with existing Open Firmware framework
- Integration with FreeBSD native *NEWBUS* device drivers scheme
- Conversion of individual drivers to the new conventions

FreeBSD Open Firmware infrastructure

- PowerPC (Apple), Sparc64
  - Genuine Open Firmware services
- Close relationship of FDT and OF device-tree
  - Limited to device tree data retrieval
- OFW kernel interfaces
  - OFW_* (low-level access to OF API calls)
  - OFW_BUS_* (standard device node *properties* access, translation to NEWBUS device kernel objects)

FDT as OFW provider

- Back-end implementation of OFW_* methods
  - Device tree data retrieved from the DTB
- Using OFW_BUS_* by higher level FDT infrastructure
  - Simplify node and properties management
- Client code sees FDT as genuine OF
  - Only to the extent of device tree retrieval
  - Other methods (device I/O, memory management etc.) not implemented and return error when called

FDT-OFW integration demo

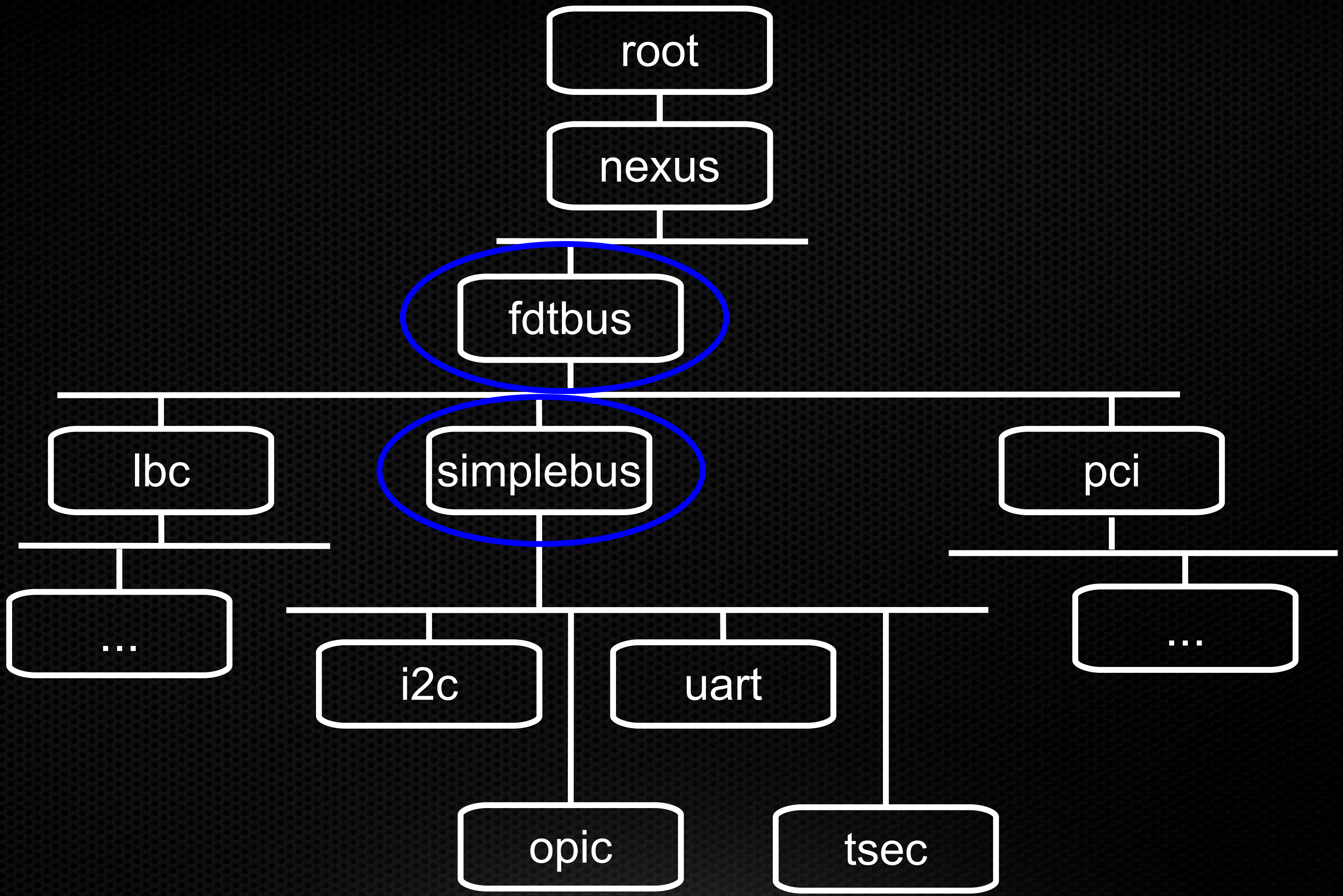
```
# ofwdump -a
Node 0xc06309a0:
  Node 0xc0630a04: aliases
  Node 0xc0630b04: cpus
    Node 0xc0630b30: PowerPC,8555@0
  Node 0xc0630bec: memory
  Node 0xc0630c24: soc8555@e0000000
    Node 0xc0630cac: ecm-law@0
    Node 0xc0630cfc: ecm@1000
    Node 0xc0630d6c: memory-controller@2000
    Node 0xc0630dec: l2-cache-controller@20000
    Node 0xc0630ea4: i2c@3000
    Node 0xc0630f40: dma@21300
      Node 0xc0630fd8: dma-channel@0
      Node 0xc0631074: dma-channel@80
      Node 0xc0631110: dma-channel@100
      Node 0xc06311ac: dma-channel@180
    Node 0xc063124c: ethernet@24000
      Node 0xc0631360: mdio@520
        Node 0xc06313c4: ethernet-phy@0
        Node 0xc063143c: ethernet-phy@1
        Node 0xc06314b4: tbi-phy@11
    Node 0xc0631504: ethernet@25000
      Node 0xc0631618: mdio@520
        Node 0xc0631678: tbi-phy@11
    Node 0xc06316c8: serial@4500
    Node 0xc063175c: serial@4600
    Node 0xc06317f0: crypto@30000
    Node 0xc0631898: pic@40000
    Node 0xc0631930: cpm@919c0
      Node 0xc06319a8: muram@80000
        Node 0xc06319f0: data@0
      Node 0xc0631a40: brg@919f0
      Node 0xc0631aa8: pic@90c00
  Node 0xc0631b58: pci@e0008000
    Node 0xc0631fa8: i8259@19000
  Node 0xc0632068: pci@e0009000
#

# ls -al /dev/openfirm
crw-------  1 root  wheel    0,  24 Jan  1 00:00 /dev/openfirm
```

Integration with *NEWBUS*

- Device drivers hierarchy, object oriented
- Legacy on-chip representation models
  - Multiple, incompatible: mbus(4), obio(4), ocpbus(4) etc.
- FDT kernel infrastructure
  - Generic, common *replacement* entities
  - fdtbus(4)
  - simplebus(4)
  - Abstract bus drivers
  - Modularity (arch/platform dependent, fixups)

Device drivers hierarchy



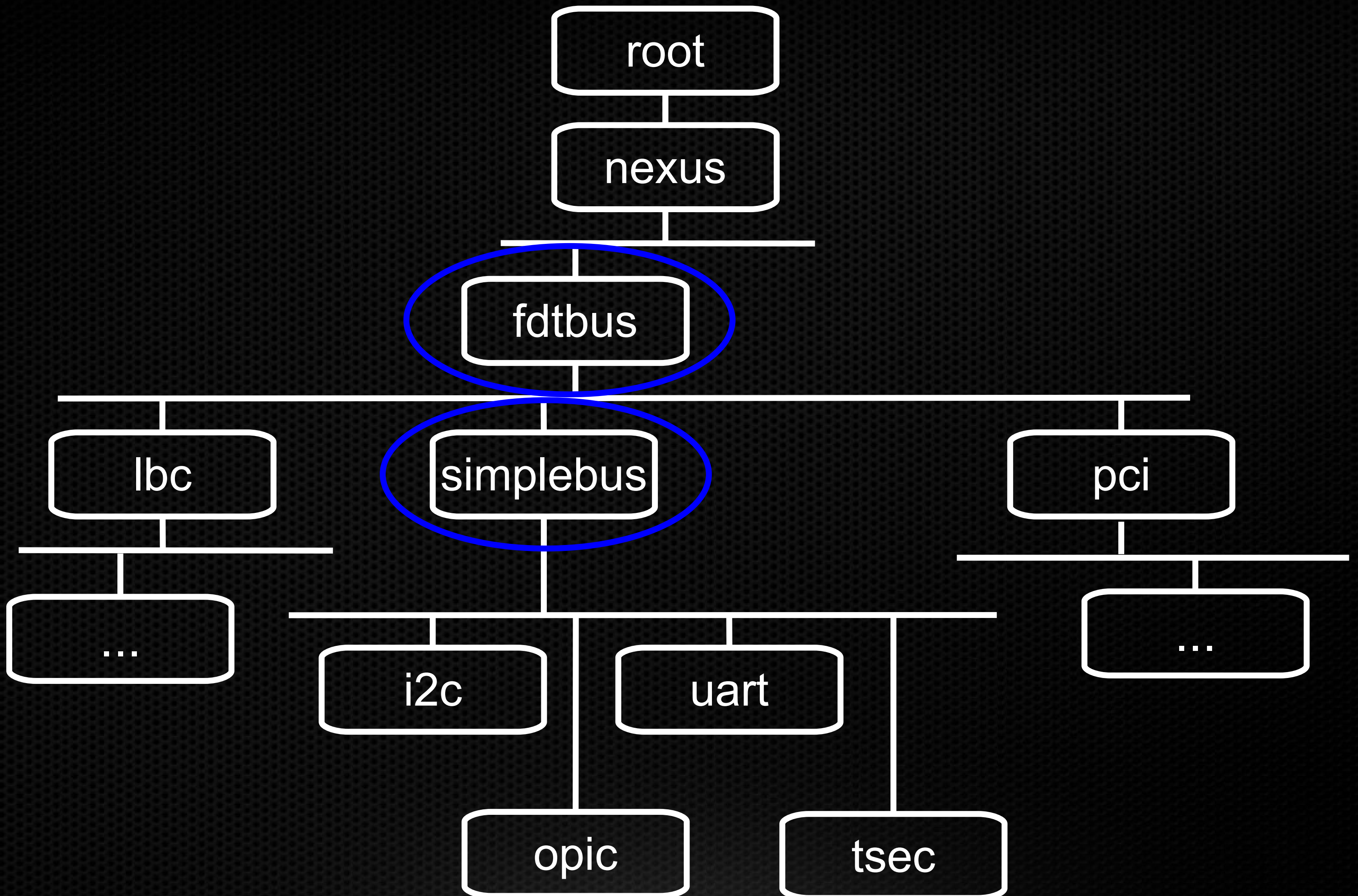
FDT kernel infrastructure

- fdtbus(4)
  - Focal point of FDT-newbus integration
  - Direct replacement of the legacy on-chip abstractions
- Main responsibilities
  - Creating newbus children reflecting FDT nodes
  - Translating resources info from FDT to FreeBSD native
  - Managing IRQ resources
  - Managing MEM, I/O resources
  - Generic, common environment for FDT-oriented drivers

FDT kernel infrastructure cont'd

- **simplebus(4)**
  - Representing ePAPR-style „simple-bus" node
  - Grouping integrated peripherals
    - Interrupt controller(s)
    - Ethernet
    - UART etc.
- **Main responsibilities**
  - Parent to all „simple-bus" subnodes
  - Passing resources requests to the fdtbus(4) layer

Device drivers hierarchy (MPC85xx)



FDT-newbus integration by example

```
soc8555@e0000000 {
        #address-cells = <1>;
        #size-cells = <1>;
        device_type = "soc";
        compatible = "simple-bus";
        ranges = <0x0 0xe0000000 0x100000>;
        bus-frequency = <0>;

        ...

        i2c@3000 {
                #address-cells = <1>;
                #size-cells = <0>;
                compatible = "fsl-i2c";
                reg = <0x3000 0x100>;
                interrupts = <43 2>;
                interrupt-parent = <&mpic>;
                dfsrr;
        };

        ...

        enet0: ethernet@24000 {
                #address-cells = <1>;
                #size-cells = <1>;
                device_type = "network";
                model = "TSEC";
                compatible = "gianfar";
                reg = <0x24000 0x1000>;
                ranges = <0x0 0x24000 0x1000>;
                local-mac-address = [ 00 00 00 00 00 00 ];
                interrupts = <29 2 30 2 34 2>;
                interrupt-parent = <&mpic>;
                tbi-handle = <&tbi0>;
                phy-handle = <&phy0>;
        };
...
```

```
# devinfo
nexus0
  fdtbus0
    lbc0
      cfi0
        cfid0
      cfi1
        cfid1
      rtc0
    simplebus0
      i2c0
        iicbus0
          iic0
      tsec0
        miibus0
          ciphy0
      tsec1
        miibus1
          ciphy1
      uart0
      uart1
      sec0
      openpic0
#
```

Conversion of (simple) device drivers

- Prerequisites
  - Device tree description of a system (blob)
  - fdtbus(4), simplebus(4) infrastructure
- Basic steps
  - Declare the driver in hierarchy

```
DRIVER_MODULE(openpic, simplebus, openpic_fdt_driver, openpic_devclass, 0, 0);
```

  - Have the probe routine check for node compatibility

```
if (!ofw_bus_is_compatible(dev, "chrp,open-pic"))
    return (ENXIO);
```

  - Further adjust

Using FDT

- Two modes of operation
  - Stand-alone DTB file
  - Statically embedded as part of kernel image
  - /sys/boot/fdt/dts
- Kernel options

```
# Enable FDT support.
options         FDT

# Provide a preferred (default) device tree source (DTS) file for the kernel.
# The indicated DTS file will be converted (compiled) into a binary form
# during kernel build stage.
makeoptions     FDT_DTS_FILE=sheevaplug.dts

# Statically embed device tree blob (DTB) into a kernel image. This option
# allows using device tree on platforms which do not (cannot) run loader(8);
# in these cases we need to embed the DTB as part of kernel data. This option
# requires a DTS file to be specified with FDT_DTS_FILE makeoption.
options         FDT_DTB_STATIC
```

Current support state

- Freescale MPC85xx family
  - DTS files provided by the silicon vendor
  - All existing peripheral drivers converted to FDT (PIC, UART, Ethernet, crypto, PCI / PCI-Express etc.)
  - Some minor optimizations still required
- Marvell Orion, Kirkwood, Discovery families
  - Hey, we're pioneering FDT deployment on ARM!
  - DTS files developed as part of this project (for 6 boards), including some *bindings* definition
  - All existing peripheral drivers converted to FDT

Summary

- Benefits
  - Uniform and extensible way of representing hardware devices, compliant with industry standards (ePAPR, Open Firmware)
  - Independent of architecture and platform (portable across ARM, MIPS, PowerPC etc.)
  - Encourages code sharing and reduction (e.g. *uart(4)* attachment)
  - Multi-platform kernels (great for cheap testing: a single kernel image can be tested against many configurations)
- Cost
  - Vendor *dtc* / *libfdt* dependency
  - Maintenance of device tree sources and bindings

Future considerations

- Adoption on more embedded platforms
  - MIPS
  - Other ARM, more PowerPC
- Remaining infrastructure elements
  - Optional *device.hints(5)*
  - RMAN resource representation shortage (u_long rm_start)
- Long term
  - Maintenance of DTS, FreeBSD-specific bindings definitions

Acknowledgments

- The FreeBSD Foundation
- M. Warner Losh (The FreeBSD Project)
- Nathan Whitehorn (The FreeBSD Project)
- Phil Brownfield (Freescale)
- Łukasz Wójcik, Michał Hajduk (both Semihalf)

Questions, please?

Flattened Device Trees for embedded FreeBSD

Rafał Jaworowski
raj@semihalf.com, raj@FreeBSD.org

BSDCan 2010, Ottawa