Maintainer's Diary: Devicetree and its stumbling blocks

Wolfram Sang

5.11.2012, ELCE 2012

About me & Devicetree

- Kernel hacker since 2008
- started working with PowerPC (which introduced devicetree to the kernel)
- now I am mainly working with ARM
- devicetree followed me :)

A simple snipplet...

```
i2c@83fc4000 { /* I2C2 on i.MX51 */
        compatible = "fsl,imx51-i2c", "fsl,imx1-i2c";
        reg = <0x83fc4000 0x4000>;
        interrupts = <63>;
};
```

...and some addition

```
i2c@83fc4000 { /* I2C2 on i.MX51 */
        compatible = "fsl,imx51-i2c", "fsl,imx1-i2c";
        reg = <0x83fc4000 0x4000>;
        interrupts = <63>;
        debug-level = <3>;
};
```

The code for querying

```
struct device_node *node = pdev->dev.of_node;
u32 dbg_level;

...

ret = of_property_read_u32(node, "debug-level",
                           &dbg_level);
```

Why the fuzz?

What is the problem?

Platform data – the old way

- is completely embedded in the kernel binary
- exchanging kernel means exchanging platform data
- internal ABI
- out of tree $\rightarrow$ bad luck

- used to describe pretty much everything

Devicetree – the new way

- a lot of devicetrees are shipped with kernel sources
- *still external ABI!*
- newer kernels must support older devicetrees

- devicetrees are OS-independent hardware descriptions (usually boards)

Major difference

- changing platform data for all users in the kernel tree $\rightarrow$ OK
- changing devicetree for all users in the kernel tree $\rightarrow$ not sufficent!

Conclusion

- think twice before adding a new binding
- think of them more like syscalls rather than platform data
- work on generic bindings if you need a new one

Some existing I2C platform data

```
/* i2c Platform Device, Driver Data */
struct mv64xxx_i2c_pdata {
    u32     freq_m;
    u32     freq_n;
    u32     timeout;    /* In milliseconds */
};
```

Proposed conversion

```
+* Marvell MV64XXX I2C controller
+
+Required properties :
+
+ - reg          : Offset and length of the register set for the device
+ - compatible   : should be "marvell,mv64xxx-i2c"
+ - interrupts   : the interrupt number
+ - frequency-m : m factor in baud rate calculation
+
+Recommended properties :
+
+ - frequency-n : n factor in baud rate calculation
+ - timeout-ms   : How long to wait for a transaction to complete
+
```

Suggestions from that

- 1:1 mappings usually don't work out
- look for existing generic solutions

Accepted conversion

```
+* Marvell MV64XXX I2C controller
+
+Required properties :
+
+ - reg              : Offset and length of the register set for the device
+ - compatible       : Should be "marvell,mv64xxx-i2c"
+ - interrupts       : The interrupt number
+ - clock-frequency : Desired I2C bus clock frequency in Hz.
```

Accepted conversion II

```
static bool __devinit
mv64xxx_find_baud_factors(const int req_freq,
                          const int tclk,
                          int *best_n,
                          int *best_m)
{
    ...
}
```

I2C timeouts - proposed binding

```
timeout-ms: How long to wait for a transaction
            to complete
```

Q: is that really a binding?

I2C timeouts - current bindings

```
fsl-i2c.txt:23:  - fsl,timeout : I2C bus timeout
                                 in microseconds.
gpio-i2c.txt:13: - i2c-gpio,timeout-ms: timeout
                                        to get data
```

Conclusion

- custom bindings are usually bad
- devicetree enforces generalization
- good, but who will do the work?

DMA - proposed binding

```
@@ -6,6 +6,7 @@ Required properties:
 - interrupts: Should contain ERROR and DMA interrupts
 - clock-frequency: Desired I2C bus clock frequency in Hz.
                    Only 100000Hz and 400000Hz modes are supported.
+- fsl,i2c-dma-channel: APBX DMA channel for the I2C

 Examples:

@@ -16,4 +17,5 @@ i2c0: i2c@80058000 {
     reg = <0x80058000 2000>;
     interrupts = <111 68>;
     clock-frequency = <100000>;
+    fsl,i2c-dma-channel = <6>;
 };
```

DMA - generic solution

still not in linux-next :(

- RFC: Februrary 2012
- V1: March 2012
- V2: March 2012
- V3: April 2012
- V4: September 2012
- V5: September 2012
- V6: September 2012

So?

- Not enough manpower?
- Not enough priority?
- Proper solution needs time?
- Developers want a solution now

- What should I do as a maintainer?

USB phy - proposed binding

```
@@ -8,6 +8,9 @@ Required properties:
+- require-transceiver: enable the flag in the driver
+- pullup-on-vbus: enable the flag in the driver
+- disable-streaming: enable the flag in the driver

@@ -58,18 +58,10 @@ static int ci13xxx_imx_vbus(struct ci13xxx *ci, int enable)
-static struct ci13xxx_platform_data ci13xxx_imx_platdata __devinitdata  = {
-    .name            = "ci13xxx_imx",
-    .flags            = CI13XXX_REQUIRE_TRANSCEIVER |
-                CI13XXX_PULLUP_ON_VBUS |
-                CI13XXX_DISABLE_STREAMING,
```

Suggestions

- Don't do 1:1 mapping
- Use existing bindings
- Don't convert what you don't need
- Some information is implicit using "compatible" binding

Configuration?

```
@@ -5,6 +5,10 @@ Required properties:
 - reg: Should contain registers location and length
 - interrupts: Should contain ERROR and DMA interrupts
 - clock-frequency: desired I2C bus clock frequency in Hz.
+
+Optional properties:
+- fsl,use-pio: Use PIO transfers instead of DMA
```

The End

Thank you for your attention!

Questions? Comments?

- right now
- anytime at this conference
- wsa@pengutronix.de