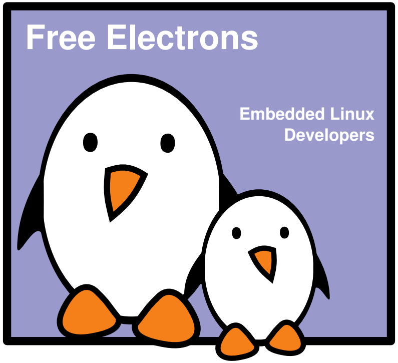
Avoiding duplication in DTs

Thomas Petazzoni
Free Electrons
thomas.petazzoni@free-electrons.com

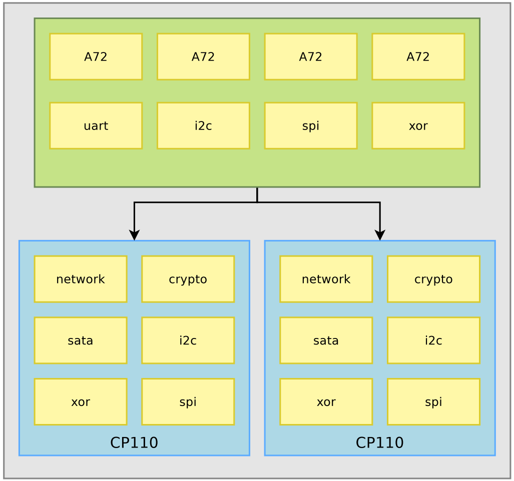


Marvell Armada 7K/8K

- “Modular” architecture
- SoC functionality split into two blocks: AP (Application) and CP (Communication Processor)
- AP: CPU cores, interrupt controllers, caches, timers, UART, XOR engines, I2C, SPI, SDHCI, GPIO
- CP: network, PCIe, RTC, GPIO, USB, SATA, XOR, SPI, I2C, NAND, crypto
- Two AP variants: AP806 dual core and AP806 quad core
- Only one CP variant currently: CP110
- Gotcha: we may have multiple instances of the same AP or the same CP in a given SoC.
- Already the case with the Marvell Armada 8020 and 8040

Marvell Armada 7K/8K family

- 7020: AP806 dual, one CP110
- 7040: AP806 quad, one CP110
- 8020: AP806 dual, two CP110
- 8040: AP806 quad, two CP110
- Two CP110: doubles the number of I/O interfaces



Armada 8040

Current approach: duplication

armada-8040.dtsi

```
#include "armada-ap806-quad.dtsi"
#include "armada-80x0.dtsi"
```

armada-80x0.dtsi

```
#include "armada-cp110-master.dtsi"
#include "armada-cp110-slave.dtsi"
```

Current approach: duplication

```
/ {
  cp110-master {
    compatible = "simple-bus";
    interrupt-parent = <&cpm_icu>;

    config-space@f2000000 {
      compatible = "simple-bus";
      ranges = <0x0 0x0 0xf2000000 0x2000000>;

      cpm_ethernet: ethernet@0 {
        clocks = <&cpm_clk 1 3>, <&cpm_clk 1 9> ...;
        marvell,system-controller = <&cpm_syscon0>;
      };

      cpm_syscon0: system-controller@440000 {
        cpm_clk: clock { };
      };

    };

    cpm_pcie0: pcie@f2600000 {
      reg = <0 0xf2600000 0 0x10000>, <0 0xf6f00000 0 0x80000>;
      ranges = <0x81000000 0 0xf9000000 0  0xf9000000 0 0x10000
        0x82000000 0 0xf6000000 0  0xf6000000 0 0xf00000>;
    };
  };
};
```

```
/ {
  cp110-slave {
    compatible = "simple-bus";
    interrupt-parent = <&cps_icu>;

    config-space@f4000000 {
      compatible = "simple-bus";
      ranges = <0x0 0x0 0xf4000000 0x2000000>;

      cps_ethernet: ethernet@0 {
        clocks = <&cps_clk 1 3>, <&cps_clk 1 9>, ...;
        marvell,system-controller = <&cps_syscon0>;
      };

      cps_syscon0: system-controller@440000 {
        cps_clk: clock { };
      };

    };

    cps_pcie0: pcie@f4600000 {
      reg = <0 0xf4600000 0 0x10000>, <0 0xfaf00000 0 0x80000>;
      ranges = <0x81000000 0 0xfd000000 0  0xfd000000 0 0x10000
        0x82000000 0 0xfa000000 0  0xfa000000 0 0xf00000>;
    };
  };
};
```

Why the duplication?

- Register base address
  - OK for most IP blocks thanks to the bus translation described by the `ranges` property.
  - Doesn't work for PCIe due to PCI config space and PCI MEM and I/O areas → **problem**
- Interrupts
  - All interrupts go into an interrupt controller called ICU inside the CP110
  - OK thanks to the top-level `interrupt-parent` property
- Clocks
  - Clock controller in each CP110
  - Clock references must therefore be *local* to the current CP110 → **problem**
- Other *phandles*
  - System controller reference → **problem**

→ This problem will get worse as we plan on having several APs, and also have 4 CPs or even more.

Possible solution: C preprocessor

- `CP110_LABEL` macro to generate the label names: `cpm_<foo>` or `cps_<foo>`
- The rest is C preprocessor sorcery

armada-common.dtsi

```
#define PASTER(x, y) x ## _ ## y
#define EVALUATOR(x, y) PASTER(x, y)
#define CP110_LABEL(name) EVALUATOR(CP110_NAME, name)
```

Possible solution: C preprocessor

armada-cp110-master.dtsi

```
#include "armada-common.dtsi"

#define CP110_NAME          cpm
#define CP110_BASE          0xf2000000
#define CP110_PCIE_IO_BASE  0xf9000000
#define CP110_PCIE_MEM_BASE 0xf6000000
#define CP110_SPI_BUS_ID(n) (0 + (n))

/ {
	cp110-master {
		#address-cells = <2>;
		#size-cells = <2>;
		compatible = "simple-bus";
		interrupt-parent = <&cpm_icu>;
		ranges;

		#include "armada-cp110.dtsi"
	};
};

#undef CP110_NAME
#undef CP110_BASE
#undef CP110_PCIE_IO_BASE
#undef CP110_PCIE_MEM_BASE
#undef CP110_SPI_BUS_ID
```

armada-cp110-slave.dtsi

```
#include "armada-common.dtsi"

#define CP110_NAME          cps
#define CP110_BASE          0xf4000000
#define CP110_PCIE_IO_BASE  0xfd000000
#define CP110_PCIE_MEM_BASE 0xfa000000
#define CP110_SPI_BUS_ID(n) (2 + (n))

/ {
	cp110-slave {
		#address-cells = <2>;
		#size-cells = <2>;
		compatible = "simple-bus";
		interrupt-parent = <&cps_icu>;
		ranges;

		#include "armada-cp110.dtsi"
	};
};

#undef CP110_NAME
#undef CP110_BASE
#undef CP110_PCIE_IO_BASE
#undef CP110_PCIE_MEM_BASE
#undef CP110_SPI_BUS_ID
```

Possible solution: C preprocessor

armada-cp110.dtsi

```
#define CP110_PCIEx_IO_BASE(iface)   (CP110_PCIE_IO_BASE + (iface *  0x10000))
#define CP110_PCIEx_MEM_BASE(iface)  (CP110_PCIE_MEM_BASE + (iface *  0x1000000))
#define CP110_PCIEx_REG0_BASE(iface) (CP110_BASE + 0x600000 + (iface) * 0x20000)
#define CP110_PCIEx_REG1_BASE(iface) (CP110_PCIEx_MEM_BASE(iface) + 0xf00000)

config-space@CP110_BASE {
        compatible = "simple-bus";
        ranges = <0x0 0x0 CP110_BASE 0x2000000>;

        CP110_LABEL(ethernet): ethernet@0 {
                reg = <0x0 0x100000>, <0x129000 0xb000>;
                clocks = <&CP110_LABEL(clk) 1 3>, <&CP110_LABEL(clk) 1 9>, <&CP110_LABEL(clk) 1 5>;
        };
};

#if CP110_BASE == 0xf2000000
CP110_LABEL(pcie0): pcie@f2600000 {
#else
CP110_LABEL(pcie0): pcie@f4600000 {
#endif
        reg = <0 CP110_PCIEx_REG0_BASE(0) 0 0x10000>, <0 CP110_PCIEx_REG1_BASE(0) 0 0x80000>;

        ranges = <0x81000000 0 CP110_PCIEx_IO_BASE(1) 0  CP110_PCIEx_IO_BASE(1) 0 0x10000
          0x82000000 0 CP110_PCIEx_MEM_BASE(1) 0  CP110_PCIEx_MEM_BASE(1) 0 0xf00000>;
};
```

Other options ?

- C preprocessor solution works
  - Not super pretty
  - But it works today, with no extra tooling/development
- Do we have other options ?
  - Scripting to generate the `.dts` file at build time ?
  - Extensions to the DT language ?
  - Something else ?