Samsung USB 2.0 PHY adaptation layer

1. Description

The architecture of the USB 2.0 PHY module in Samsung SoCs is similar among many SoCs. In spite of the similarities it proved difficult to create a one driver that would fit all these PHY controllers. Often the differences were minor and were found in particular bits of the registers of the PHY. In some rare cases the order of register writes or the PHY powering up process had to be altered. This adaptation layer is a compromise between having separate drivers and having a single driver with added support for many special cases.

2. Files description

- phy-samsung-usb2.c

  This is the main file of the adaptation layer. This file contains the probe function and provides two callbacks to the Generic PHY Framework. This two callbacks are used to power on and power off the phy. They carry out the common work that has to be done on all version of the PHY module. Depending on which SoC was chosen they execute SoC specific callbacks. The specific SoC version is selected by choosing the appropriate compatible string. In addition, this file contains struct of_device_id definitions for particular SoCs.
- phy-samsung-usb2.h

  This is the include file. It declares the structures used by this driver. In addition it should contain extern declarations for structures that describe particular SoCs.

3. Supporting SoCs

To support a new SoC a new file should be added to the drivers/phy directory. Each SoC's configuration is stored in an instance of the struct samsung_usb2_phy_config:

```
struct samsung_usb2_phy_config {
	const struct samsung_usb2_common_phy *phys;
	int (*rate_to_clk)(unsigned long, u32 *);
	unsigned int num_phys;
	bool has_mode_switch;
};
```

The num_phys is the number of phys handled by the driver. **phys* is an array that contains the configuration for each phy. The has_mode_switch property is a boolean flag that determines whether the SoC has USB host and device on a single pair of pins. If so, a special register has to be modified to change the internal routing of these pins between a USB device or host module.

For example the configuration for Exynos 4210 is following:

```
const struct samsung_usb2_phy_config exynos4210_usb2_phy_config = {
	.has_mode_switch        = 0,
	.num_phys               = EXYNOS4210_NUM_PHYS,
	.phys                   = exynos4210_phys,
	.rate_to_clk            = exynos4210_rate_to_clk,
}
```

- *int (*rate_to_clk)(unsigned long, u32 *)*

  The rate_to_clk callback is to convert the rate of the clock used as the reference clock for the PHY module to the value that should be written in the hardware register.

The exynos4210_phys configuration array is as follows:

```
static const struct samsung_usb2_common_phy exynos4210_phys[] = {
	{
		.label          = "device",
		.id             = EXYNOS4210_DEVICE,
		.power_on       = exynos4210_power_on,
		.power_off      = exynos4210_power_off,
	},
	{
		.label          = "host",
		.id             = EXYNOS4210_HOST,
		.power_on       = exynos4210_power_on,
		.power_off      = exynos4210_power_off,
	},
	{
		.label          = "hsic0",
		.id             = EXYNOS4210_HSIC0,
		.power_on       = exynos4210_power_on,
		.power_off	= exynos4210_power_off,
	},
	{
		.label		= "hsic1",
		.id		= EXYNOS4210_HSIC1,
		.power_on	= exynos4210_power_on,
		.power_off	= exynos4210_power_off,
	},
	{},
};
```

- *int (*power_on)(struct samsung_usb2_phy_instance *); int (*power_off)(struct samsung_usb2_phy_instance *);*

  These two callbacks are used to power on and power off the phy by modifying appropriate registers.

Final change to the driver is adding appropriate compatible value to the phy-samsung-usb2.c file. In case of Exynos 4210 the following lines were added to the struct of_device_id samsung_usb2_phy_of_match[] array:

```
#ifdef CONFIG_PHY_EXYNOS4210_USB2
	{
		.compatible = "samsung,exynos4210-usb2-phy",
		.data = &exynos4210_usb2_phy_config,
	},
#endif
```

To add further flexibility to the driver the Kconfig file enables to include support for selected SoCs in the compiled driver. The Kconfig entry for Exynos 4210 is following:

```
config PHY_EXYNOS4210_USB2
	bool "Support for Exynos 4210"
	depends on PHY_SAMSUNG_USB2
	depends on CPU_EXYNOS4210
	help
	  Enable USB PHY support for Exynos 4210. This option requires that
	  Samsung USB 2.0 PHY driver is enabled and means that support for this
	  particular SoC is compiled in the driver. In case of Exynos 4210 four
	  phys are available - device, host, HSCI0 and HSCI1.
```

The newly created file that supports the new SoC has to be also added to the Makefile. In case of Exynos 4210 the added line is following:

```
obj-$(CONFIG_PHY_EXYNOS4210_USB2)	+= phy-exynos4210-usb2.o
```

After completing these steps the support for the new SoC should be ready.