Enabling uSDHC1 of i.MX8MM Based On L5.4.24_2.1.0 BSP

This article will describe how to enable uSDHC1 & uSDHC3 for i.MX8MM+DDR4+eMMC boards.

We released 2 versions of i.MX8MM EVK board: one is i.MX8MMINILPD4-EVK, the other is i.MX8MMiniD4-EVK. The main difference between these two development boards is LPDDR4, DDR4, eMMC and NAND Flash.

--i.MX8MMINILPD4-EVK

LPDDR4

eMMC: uSDHC3 port multiplexed via NAND pads.

--i.MX8MMiniD4-EVK

DDR4

NAND Flash: NAND port muliplexed via NAND pads.

The uSDHC1 and uSDHC2 of these two EVK boards are connected to WIFI and MicrosSD card slot respectively.

In u-boot, the i.MX8MMINILPD4-EVK board initialized 2 uSDHC ports: uSDHC2 and uSDHC3, respectively for MicrosSD card and eMMC. The i.MX8MMiniD4-EVK board initialized 1 uSDHC port: uSDHC2 port for MicroSD card.

Open uboot-imx/board/freescale/imx8mm_evk/spl.c, in board_mmc_init () function, We can see these codes:

```
for (i = 0; i < CONFIG_SYS_FSL_USDHC_NUM; i++)
        switch (i) {
        case 0:
            init_clk_usdhc(1);
            usdhc_cfg[0].sdhc_clk = mxc_get_clock(MXC_ESDHC2_CLK);
            imx_iomux_v3_setup_multiple_pads(
                        usdhc2_pads, ARRAY_SIZE(usdhc2_pads));
            gpio_request(USDHC2_PWR_GPIO, "usdhc2_reset");
            gpio_direction_output(USDHC2_PWR_GPIO, 0);
            udelay(500);
            gpio_direction_output(USDHC2_PWR_GPIO, 1);
            break;
        case 1:
            init_clk_usdhc(2);
            usdhc_cfg[1].sdhc_clk = mxc_get_clock(MXC_ESDHC3_CLK);
            imx_iomux_v3_setup_multiple_pads(
                        usdhc3_pads, ARRAY_SIZE(usdhc3_pads));
            udelay(1500);
            break;
        default:
            printf("Warning: you configured more USDHC controllers"
                "(%d) than supported by the board\n", i + 1);
            return -EINVAL;
        }
```

Open ./include/configs/imx8mm_evk.h, we can find CONFIG_SYS_FSL_USDHC_NUM, which is defined like below:

```
#ifdef CONFIG_TARGET_IMX8MM_DDR4_EVK
#define CONFIG_SYS_FSL_USDHC_NUM	1
#else
#define CONFIG_SYS_FSL_USDHC_NUM	2
#endif
```

Some customers use the I.MX8MM+DDR4+EMMC solution, and when porting based on IMX8MMD4-EVK, they will encounter the problem of uSDHC3 not working, because the uSDHC3 clock is not initialized. We must let CONFIG_SYS_FSL_USDHC_NUM=2 to run the uSDHC3 clock initialization code.

The following is the reference code for enabling uSDHC1 in spl.c and u-boot. Users can try to debug the board based on these codes and start the board from uSDHC1.

- ## spl.c

-(1)For eMMC-8bit

```
static iomux_v3_cfg_t const usdhc1_pads[] = {
	IMX8MM_PAD_SD1_CLK_USDHC1_CLK | MUX_PAD_CTRL(USDHC_PAD_CTRL),
	IMX8MM_PAD_SD1_CMD_USDHC1_CMD | MUX_PAD_CTRL(USDHC_PAD_CTRL),
	IMX8MM_PAD_SD1_DATA0_USDHC1_DATA0 | MUX_PAD_CTRL(USDHC_PAD_CTRL),
	IMX8MM_PAD_SD1_DATA1_USDHC1_DATA1 | MUX_PAD_CTRL(USDHC_PAD_CTRL),
	IMX8MM_PAD_SD1_DATA2_USDHC1_DATA2 | MUX_PAD_CTRL(USDHC_PAD_CTRL),
	IMX8MM_PAD_SD1_DATA3_USDHC1_DATA3 | MUX_PAD_CTRL(USDHC_PAD_CTRL),
	IMX8MM_PAD_SD1_DATA4_USDHC1_DATA4 | MUX_PAD_CTRL(USDHC_PAD_CTRL),
	IMX8MM_PAD_SD1_DATA5_USDHC1_DATA5 | MUX_PAD_CTRL(USDHC_PAD_CTRL),
	IMX8MM_PAD_SD1_DATA6_USDHC1_DATA6 | MUX_PAD_CTRL(USDHC_PAD_CTRL),
	IMX8MM_PAD_SD1_DATA7_USDHC1_DATA7 | MUX_PAD_CTRL(USDHC_PAD_CTRL),
};
static struct fsl_esdhc_cfg usdhc_cfg[3] = {
	{USDHC1_BASE_ADDR, 0, 8},
	{USDHC2_BASE_ADDR, 0, 4},
	{USDHC3_BASE_ADDR, 0, 8},
};

int board_mmc_getcd(struct mmc *mmc)
{
	struct fsl_esdhc_cfg *cfg = (struct fsl_esdhc_cfg *)mmc->priv;
	int ret = 0;

	switch (cfg->esdhc_base) {
	case USDHC1_BASE_ADDR:
	case USDHC3_BASE_ADDR:
		ret = 1;
```

```
            break;
        case USDHC2_BASE_ADDR:
            imx_iomux_v3_setup_pad(usdhc2_cd_pad);
            gpio_request(USDHC2_CD_GPIO, "usdhc2 cd");
            gpio_direction_input(USDHC2_CD_GPIO);

            /*
             * Since it is the DAT3 pin, this pin is pulled to
             * low voltage if no card
             */
            ret = gpio_get_value(USDHC2_CD_GPIO);

            imx_iomux_v3_setup_pad(usdhc2_dat3_pad);
            return ret;
        }
        return 1;
}
```

-(2)For SD card--4bit

```
#define USDHC1_CD_GPIO     IMX_GPIO_NR(2, 5) /* Using SD1_DATA3 TO BE DETECT PIN */

static iomux_v3_cfg_t const usdhc1_pads[] = {
        IMX8MM_PAD_SD1_CLK_USDHC1_CLK | MUX_PAD_CTRL(USDHC_PAD_CTRL),
        IMX8MM_PAD_SD1_CMD_USDHC1_CMD | MUX_PAD_CTRL(USDHC_PAD_CTRL),
        IMX8MM_PAD_SD1_DATA0_USDHC1_DATA0 | MUX_PAD_CTRL(USDHC_PAD_CTRL),
        IMX8MM_PAD_SD1_DATA1_USDHC1_DATA1 | MUX_PAD_CTRL(USDHC_PAD_CTRL),
        IMX8MM_PAD_SD1_DATA2_USDHC1_DATA2 | MUX_PAD_CTRL(USDHC_PAD_CTRL),
        IMX8MM_PAD_SD1_DATA3_USDHC1_DATA3 | MUX_PAD_CTRL(USDHC_PAD_CTRL),
};

/* Multipelex SD1_DATA3 To be GPIO for Card detect */
static iomux_v3_cfg_t const usdhc1_cd_pad =
        IMX8MM_PAD_SD1_DATA3_GPIO2_IO5 | MUX_PAD_CTRL(USDHC_GPIO_PAD_CTRL);
/* After detection, SD1_DATA3 pin back to DATA3 funtion */
static iomux_v3_cfg_t const usdhc1_dat3_pad =
        IMX8MM_PAD_SD1_DATA3_USDHC1_DATA3 |
        MUX_PAD_CTRL(USDHC_PAD_CTRL);

static struct fsl_esdhc_cfg usdhc_cfg[3] = {
        {USDHC1_BASE_ADDR, 0, 4},
        {USDHC2_BASE_ADDR, 0, 4},
        {USDHC3_BASE_ADDR, 0, 8},
};
int board_mmc_getcd(struct mmc *mmc)
```

```
{
	struct fsl_esdhc_cfg *cfg = (struct fsl_esdhc_cfg *)mmc->priv;
	int ret = 0;

	switch (cfg->esdhc_base) {
	case USDHC3_BASE_ADDR:
		ret = 1;
		break;
	case USDHC2_BASE_ADDR:
		imx_iomux_v3_setup_pad(usdhc2_cd_pad);
		gpio_request(USDHC2_CD_GPIO, "usdhc2 cd");
		gpio_direction_input(USDHC2_CD_GPIO);

		/*
		 * Since it is the DAT3 pin, this pin is pulled to
		 * low voltage if no card
		 */
		ret = gpio_get_value(USDHC2_CD_GPIO);

		imx_iomux_v3_setup_pad(usdhc2_dat3_pad);
		return ret;
	case USDHC1_BASE_ADDR:
		imx_iomux_v3_setup_pad(usdhc1_cd_pad);
		gpio_request(USDHC1_CD_GPIO, "usdhc1 cd");
		gpio_direction_input(USDHC1_CD_GPIO);
		ret = gpio_get_value(USDHC1_CD_GPIO);
		imx_iomux_v3_setup_pad(usdhc1_dat3_pad);
		return ret;
	}

	return 1;
}

--(3) clock initialization
int mmc_clock_init(bd_t *bis,int nSD_num)
{
	int ret;
	switch (i) {
	case 0:
		init_clk_usdhc(0);
		usdhc_cfg[0].sdhc_clk = mxc_get_clock(MXC_ESDHC1_CLK);
		imx_iomux_v3_setup_multiple_pads(
		usdhc1_pads, ARRAY_SIZE(usdhc1_pads));
		break;
```

```
        case 1:
                init_clk_usdhc(1);
                usdhc_cfg[1].sdhc_clk = mxc_get_clock(MXC_ESDHC2_CLK);
                imx_iomux_v3_setup_multiple_pads(
                usdhc2_pads, ARRAY_SIZE(usdhc2_pads));
                gpio_request(USDHC2_PWR_GPIO, "usdhc2_reset");
                gpio_direction_output(USDHC2_PWR_GPIO, 0);
                udelay(500);
                gpio_direction_output(USDHC2_PWR_GPIO, 1);
                break;

        case 2:
                init_clk_usdhc(2);
                usdhc_cfg[2].sdhc_clk = mxc_get_clock(MXC_ESDHC3_CLK);
                imx_iomux_v3_setup_multiple_pads(
                        usdhc3_pads, ARRAY_SIZE(usdhc3_pads));
                break;

                default:
                        printf("Warning: you configured more USDHC controllers"
                                "(%d) than supported by the board\n", nSD_num);
                        return -EINVAL;
                }
        ret = fsl_esdhc_initialize(bis, &usdhc_cfg[nSD_num]);
        if (ret)
                        return ret;
        printf("eSDHC %d controllers have been initialized\n",nSD_num+1);
        return 0;
}

int board_mmc_init(bd_t *bis)
{
        mmc_clock_init(bis ,0); /* initialize eSDHC1 CLOCK */
        mmc_clock_init(bis ,1); /* initialize eSDHC2 CLOCK */
        mmc_clock_init(bis ,2); /* initialize eSDHC3 CLOCK */

        return 0;
}
```

- device tree in u-boot(imx8mm-evk.dts)

```
pinctrl_usdhc1_gpio: usdhc1grpgpio {
	fsl,pins = <
		MX8MM_IOMUXC_GPIO1_IO06_GPIO1_IO6	0x1c4	/* CD detect */
	>;
};
pinctrl_usdhc1: usdhc1grp {
	fsl,pins = <
		MX8MM_IOMUXC_SD1_CLK_USDHC1_CLK		0x190
		MX8MM_IOMUXC_SD1_CMD_USDHC1_CMD		0x1d0
		MX8MM_IOMUXC_SD1_DATA0_USDHC1_DATA0	0x1d0
		MX8MM_IOMUXC_SD1_DATA1_USDHC1_DATA1	0x1d0
		MX8MM_IOMUXC_SD1_DATA2_USDHC1_DATA2	0x1d0
		MX8MM_IOMUXC_SD1_DATA3_USDHC1_DATA3	0x1d0
		/* if using 4bit microSD card, remove 5 lines below */
		MX8MM_IOMUXC_SD1_DATA4_USDHC1_DATA4	0x1d0
		MX8MM_IOMUXC_SD1_DATA5_USDHC1_DATA5	0x1d0
		MX8MM_IOMUXC_SD1_DATA6_USDHC1_DATA6	0x1d0
		MX8MM_IOMUXC_SD1_DATA7_USDHC1_DATA7	0x1d0
		MX8MM_IOMUXC_SD1_STROBE_USDHC1_STROBE	0x190
	>;
};

pinctrl_usdhc1_100mhz: usdhc1grp100mhz {
	fsl,pins = <
		MX8MM_IOMUXC_SD1_CLK_USDHC1_CLK		0x194
		MX8MM_IOMUXC_SD1_CMD_USDHC1_CMD		0x1d4
		MX8MM_IOMUXC_SD1_DATA0_USDHC1_DATA0	0x1d4
		MX8MM_IOMUXC_SD1_DATA1_USDHC1_DATA1	0x1d4
		MX8MM_IOMUXC_SD1_DATA2_USDHC1_DATA2	0x1d4
		MX8MM_IOMUXC_SD1_DATA3_USDHC1_DATA3	0x1d4
		/* if using 4bit microSD card, remove 5 lines below */
		MX8MM_IOMUXC_SD1_DATA4_USDHC1_DATA4	0x1d4
		MX8MM_IOMUXC_SD1_DATA5_USDHC1_DATA5	0x1d4
		MX8MM_IOMUXC_SD1_DATA6_USDHC1_DATA6	0x1d4
		MX8MM_IOMUXC_SD1_DATA7_USDHC1_DATA7	0x1d4
		MX8MM_IOMUXC_SD1_STROBE_USDHC1_STROBE	0x194
	>;
};
pinctrl_usdhc3_200mhz: usdhc3grp200mhz {
	fsl,pins = <
		MX8MM_IOMUXC_SD1_CLK_USDHC1_CLK		0x196
		MX8MM_IOMUXC_SD1_CMD_USDHC1_CMD		0x1d6
```

```
                MX8MM_IOMUXC_SD1_DATA0_USDHC1_DATA0        0x1d6
                MX8MM_IOMUXC_SD1_DATA1_USDHC1_DATA1        0x1d6
                MX8MM_IOMUXC_SD1_DATA2_USDHC1_DATA2        0x1d6
                MX8MM_IOMUXC_SD1_DATA3_USDHC1_DATA3        0x1d6
                /* if using 4bit microSD card, remove 5 lines below */
                MX8MM_IOMUXC_SD1_DATA4_USDHC1_DATA4        0x1d6
                MX8MM_IOMUXC_SD1_DATA5_USDHC1_DATA5        0x1d6
                MX8MM_IOMUXC_SD1_DATA6_USDHC1_DATA6        0x1d6
                MX8MM_IOMUXC_SD1_DATA7_USDHC1_DATA7        0x1d6
                MX8MM_IOMUXC_SD1_STROBE_USDHC1_STROBE          0x196
            >;
        };
```

If eMMC is connected:

```
&usdhc1 {
        pinctrl-names = "default", "state_100mhz", "state_200mhz";
        pinctrl-0 = <&pinctrl_usdhc1>;
        pinctrl-1 = <&pinctrl_usdhc1_100mhz>;
        pinctrl-2 = <&pinctrl_usdhc1_200mhz>;
        bus-width = <8>;
        non-removable;
        status = "okay";
};
```

if microSD card is connected:

```
&usdhc1 {
        pinctrl-names = "default", "state_100mhz", "state_200mhz";
        pinctrl-0 = <&pinctrl_usdhc1>, <&pinctrl_usdhc1_gpio>;
        pinctrl-1 = <&pinctrl_usdhc1_100mhz>, <&pinctrl_usdhc1_gpio>;
        pinctrl-2 = <&pinctrl_usdhc1_200mhz>, <&pinctrl_usdhc1_gpio>;
        cd-gpios = <&gpio1 6 GPIO_ACTIVE_LOW>;    /* Assume using GPIO1_IO06 to be CD pin */
        bus-width = <4>;
        status = "okay";
};
```

● Boot Setting From uSHDC1

--MicroSD card

BOOT_CFG[15]=0

BOOT_CFG[14:12]=001

BOOT_CFG[11:10]=00

BOOT_CFG[9]=0

BOOT_CFG[8]=0

BOOT_CFG[7]=0

BOOT_CFG[6:4]=001
BOOT_CFG[3:1]=000
BOOT_CFG[0]=0

--eMMC

BOOT_CFG[15]=0
BOOT_CFG[14:12]=010
BOOT_CFG[11:10]=00
BOOT_CFG[9]=0
BOOT_CFG[8]=0
BOOT_CFG[7]=0
BOOT_CFG[6:4]=010
BOOT_CFG[3:2]=00
BOOT_CFG[1]=1(0->3.3V IO; 1->1.8V IO)
BOOT_CFG[0]=1(0->3.3V IO; 1->1.8V IO)

NXP TIC Team
Weidong Sun
11/30/2020