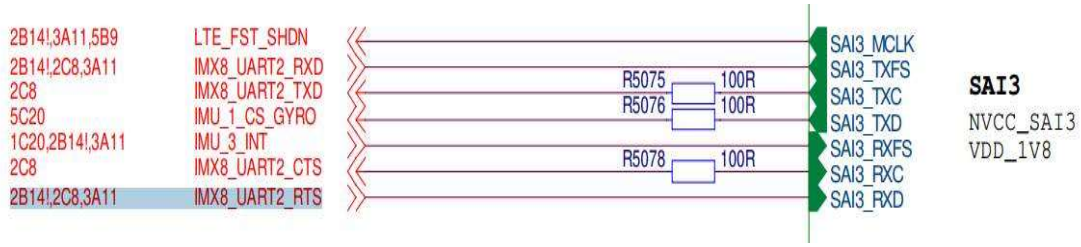
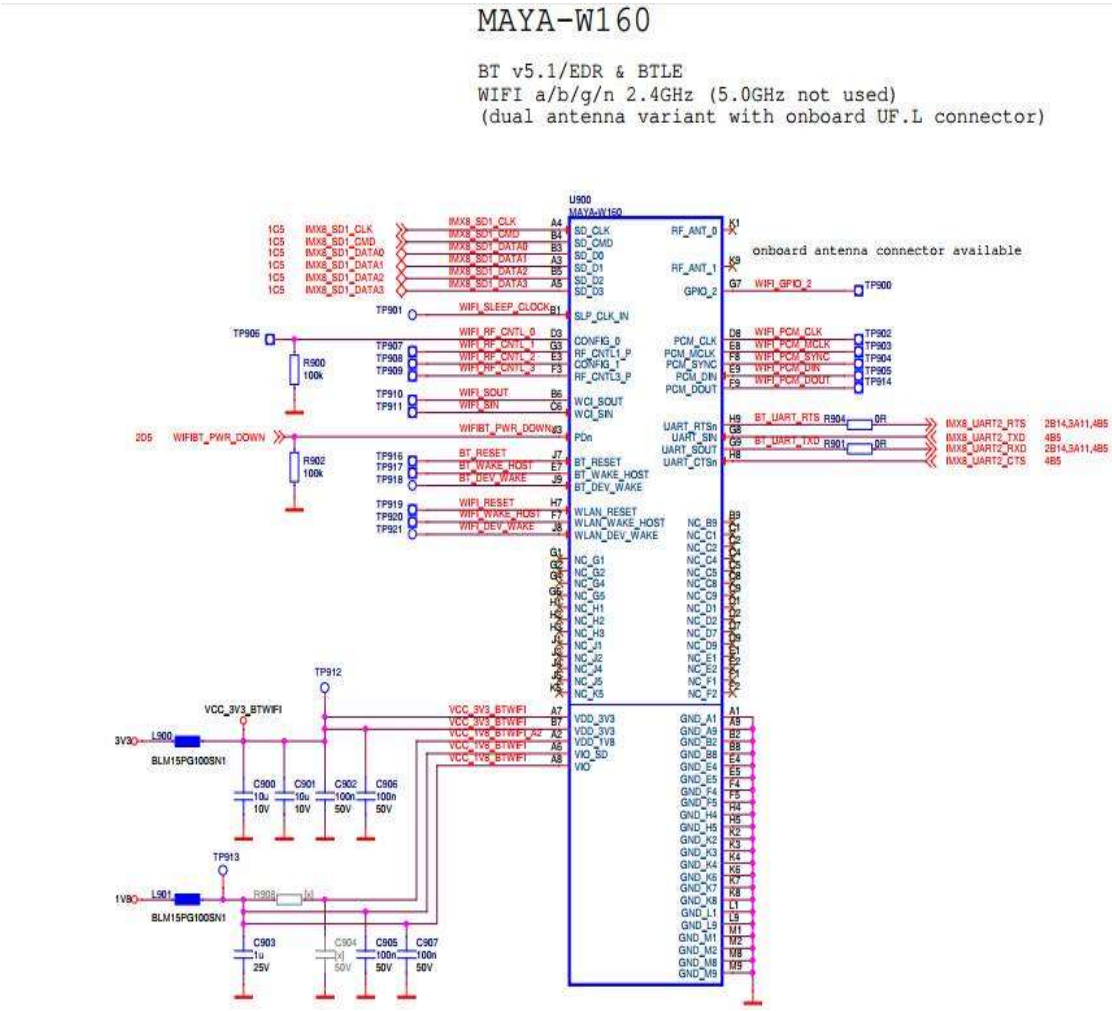
Enabling IW416 Bluetooth on iMX8MM + Linux BSP

Using UART2 of iMX8MM For Bluetooth Communication

1. Hardware Connections





[Checking & Analysis]

--UART2 IO level between host & Bluetooth

- ✓ BT side : VIO = 1.8V
- ✓ CPU side : NVCC_SAI3 = 1.8V

OK, matching level is no problem.

--PDn pin of MAYA_160

The pin should be pulled up to High before loading wifi/bt driver, which is power switch of MAYA_160, the pin can also be used to reset the module.

--IO directions between CPU UART2 & UART of MAYA_160

CPU UART2 DCE mode	CPU UART2 IOMUX PAD	MAYA_160 UART
UART2_DCE_RX (Input)	SAI3_TXFS	UART_SOUT (Output)
UART2_DCE_TX (Output)	SAI3_TXC	UART_SIN (Input)
UART2_DCE_CTS_B (Output)	SAI3_RXC	UART_CTSn (Input)
UART2_DCE_RTS_B (Input)	SAI3_RXD	UART_RTSn (output)

Comparing above schematic, UART IO connections are no problem.

OK, Hardware design is no problem.

2. Software (taking L5.10.35_2.0.0 as an example)

2.1 Comment original UART2 in device tree (imx8mm-evk.dtsi)

By default, UART2 is used for console, so original settings should be commented.

```
/*
	pinctrl_uart2: uart2grp {
		fsl,pins = <
			MX8MM_IOMUXC_UART2_RXD_UART2_DCE_RX	0x140
			MX8MM_IOMUXC_UART2_TXD_UART2_DCE_TX	0x140
		>;
	};
*/

/*
&uart2 { /* console */
	pinctrl-names = "default";
	pinctrl-0 = <&pinctrl_uart2>;
	status = "okay";
};
*/
```

2.2 adding UART2 to the device tree according hardware design

2.2.1 UART2 pins IOMUX

```
	pinctrl_uart2: uart2grp {
		fsl,pins = <
			MX8MM_IOMUXC_SAI3_TXFS_UART2_DCE_RX 0x140
			MX8MM_IOMUXC_SAI3_TXC_UART2_DCE_TX 0x140
			MX8MM_IOMUXC_SAI3_RXC_UART2_DCE_CTS_B 0x140
			MX8MM_IOMUXC_SAI3_RXD_UART2_DCE_RTS_B 0x140
		>;
	};
```

2.2.2 Adding UART2 node to the device tree

```
	&uart2 {
		pinctrl-names = "default";
		pinctrl-0 = <&pinctrl_uart2>;
```

```
        assigned-clocks = <&clk IMX8MM_CLK_UART2>;
        assigned-clock-parents = <&clk IMX8MM_SYS_PLL1_80M>;
        fsl,uart-has-rtscts;
        status = "okay";
    };
```

2.3 Handling PDn pin

Assume that above design uses **SD1_RESET_B**(R23 pin) to control PDn of MAYA_160.

Add settings below:

✓ **Using mmc power sequence driver**

```
{
……
        modem_reset: modem-reset {
            compatible = "gpio-reset";
            reset-gpios = <&gpio2 6 GPIO_ACTIVE_LOW>;
            reset-delay-us = <2000>;
            reset-post-delay-ms = <40>;
            #reset-cells = <0>;
        };
        usdhc1_pwrseq: usdhc1_pwrseq {
            compatible = "mmc-pwrseq-simple";
            pinctrl-names = "default";
            pinctrl-0 = <&pinctrl_usdhc1_gpio>;
            reset-gpios = <&gpio2 10 GPIO_ACTIVE_LOW>;
        };
……
```

✓ **Adding it to IOMUX node**

```
……
            pinctrl_usdhc1_gpio: usdhc1grpgpio {
                fsl,pins = <
                    MX8MN_IOMUXC_SD1_RESET_B_GPIO2_IO10  0x41
                >;
            };
……
```

✓ **Adding *usdhc1_pwrseq* to usdhc1 node**

```
&usdhc1 {
        ……
            mmc-pwrseq = <&usdhc1_pwrseq>;
        …
}
```

Up to now, modification in device tree is done.

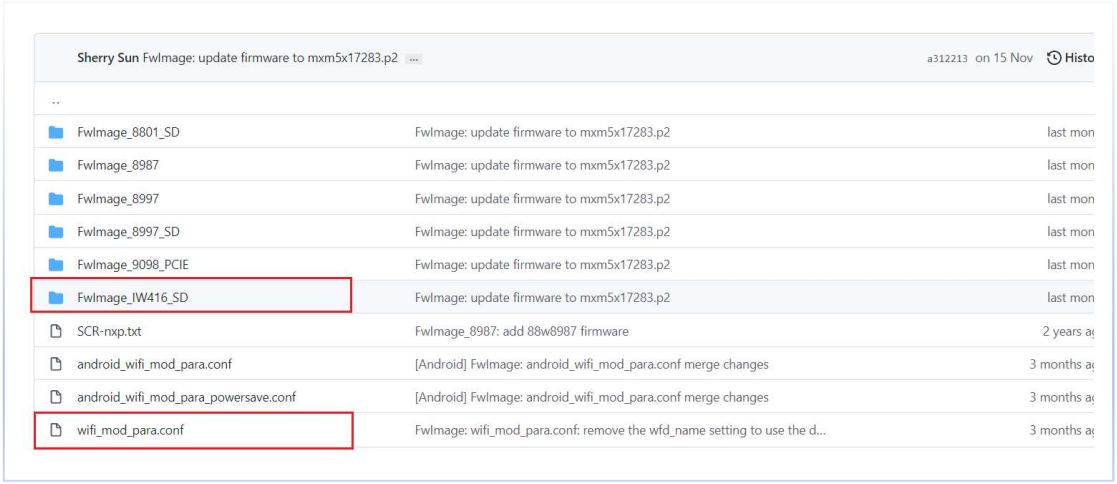
3. Getting IW416 Mass Market driver

3.1 downloading driver

```
#git clone http://source.codeaurora.org/external/imx/mwifiex.git
#cd mwifiex
#git tag
android-11.0.0_2.0.0
android-11.0.0_2.2.0
android-11.0.0_2.2.0-imx8ulp-er
android-11.0.0_2.4.0
android-11.0.0_2.6.0
automotive-11.0.0_2.1.0
automotive-11.0.0_2.3.0
lf-5.10.35-2.0.0
lf-5.10.52-2.1.0
lf-5.10.72-2.2.0
lf-5.10.y-1.0.0
rel_imx_5.4.24_2.1.0
rel_imx_5.4.47_2.2.0
rel_imx_5.4.70_2.3.0
rel_imx_5.4.70_2.3.2
#git checkout lf-5.10.35-2.0.0
```

3.2 downloading firmware

https://github.com/NXP/imx-firmware/tree/lf-5.10.72_2.2.0/nxp



[Note]

The firmware should be placed in /lib/firmware/nxp on board. The firmware is for WiFi & Bluetooth, so if users want to enable Bluetooth, she should load wifi driver firstly. Wifi_mod_para.conf is configuration file , which will be used when using modprobe to load wifi driver, it is also should be placed in /lib/firmware/nxp on board.

In linux bsp, Bluetooth uses standard hci driver in linux kernel. So after loading wifi driver & firmware with Combo, Bluetooth can also be used.

3.3 Compiling WiFi driver

Before compiling driver, uses should export cross-compile toolchain from Yocto, and install it to /opt, detailed steps are described in i.MX_Linux_Users_Guide.pdf, open the file and search 'standalone' , to find '**4.5.12 How to build U-Boot and Kernel in standalone environment**'

When installation is done, it should be like below in /opt.

```
weidong@ubuntu:~/imx-yocto-bsp/L5.10.72_2.2.0/build-wayland/tmp/work/imx8mm_lpddr4_evk-poky-linux/linux-imx/5.10.72+gitAUTOINC+a68e31b63f-r0/build$ ls /opt/fsl-imx-xwayland/5.10-hardknott/
environment-setup-cortexa53-crypto-poky-linux
site-config-cortexa53-crypto-poky-linux
sysroots
version-cortexa53-crypto-poky-linux
weidong@ubuntu:~/imx-yocto-bsp/L5.10.72_2.2.0/build-wayland/tmp/work/imx8mm_lpddr4_evk-poky-linux/linux-imx/5.10.r4_evk-poky-linux/linux-imx/5.10.72+gitAUTOINC+a68e31b63f-r0/build$
```

Now, go back to the directory of wifi driver source code 'mwifiex', and run commands like below:

source /opt/fsl-imx-xwayland/5.10-hardknott/environment-setup-cortexa53-crypto-poky-linux

export KERNELDIR=/home/weidong/imx-yocto-bsp/L5.10.72_2.2.0/build-wayland/tmp/work/imx8mm_lpddr4_evk-poky-linux/linux-imx/5.10.r4_evk-poky-linux/linux-imx/5.10.72+gitAUTOINC+a68e31b63f-r0/build

[Note]

The above KERNELDIR is mine, and taken L5.10.72_2.2.0 yocto as an example, it should be modified to yours.

Then begin to compile it.

make build

When compilation is done, mlan.ko & moal.ko will be generated in the current directory.

Uses can copy these 2 files, firmware & wifi_mod_para.conf to board, and place firmware & wifi_mod_para.conf to /lib/firmware/nxp . mlan.ko & moal.ko should be placed to /lib/modules/5.10.35-lts-5.10.y+gef3f2cfc6010/extra/ if using modprobe to load them. If using insmod them, home directory is OK.

If you can find mlan.ko & moal.ko file, it means that driver is ok in your system, don't need to compile it again, you can check if firmware exists in /lib/firmware/nxp, if no, copy firmware to the directoy, then load wifi driver with modprobe command;

modprobe moal mod_para=nxp/wifi_mod_para.conf

4. Starting Bluetooth

if UART1/UART2/UART3 are used in device tree, UART2 device is ttymxc1, so before starting bluetooth, we should check UART device:
# ls /dev/ttymxc*
the purpose is to confirm which device is UART2.
hciattach /dev/ttymxc1 any 115200 flow
hciconfig hci0 up
hciconfig
Setting UART bandrate to be 3Mbps:
hcitool -i hci0 cmd 0x3f 0x0009 0xc0 0xc6 0x2d 0x00
killall hciattach
hciattach /dev/ttymxc0 any -s 3000000 3000000 flow
hciconfig hci0 up

NXP global CAS connectivity team
Weidong Sun
12-29-2021