NXP Semiconductors
Application Note

Document Number: AN12225
Rev. 0, 07/2018

How to Reduce SoC Power when Running M4 with A53 on i.MX8M

Contents

1. Introduction

On i.MX8M EVK board, when user measures current with AMP applications, he may find that the VDD_SOC current is much higher than expected. This document discusses about the root cause of the issue and the solution.

2. Chip overview

This section provides main features of i.MX 8M chips.

2.1. i.MX8M chip overview

The i.MX 8M family is a set of NXP products focused on delivering the latest and greatest video and audio experience combining state-of-the-art media-specific features with high-performance processing while optimized for lowest power consumption. The i.MX 8M Quad Media Applications Processor is the first product of the growing i.MX 8M family targeting the consumer market.

- Built in TSMC 28HPC to achieve both high performance and low power consumption.
- Relies on a powerful fully coherent core based on a quad Cortex®-A53 cluster with graphics processing handled by the GC7000Lite GPU from Vivante supporting the latest graphic APIs.
- Advanced security modules for secure boot, cipher acceleration and DRM support.

- General purpose Cortex-M4 processor for low power processing.
- A wide range of audio interfaces including I2S, AC97, TDM and S/PDIF.
- Large set of peripherals that are commonly used in consumer/industrial markets including USB 3.0, PCIe and Ethernet.

3. High VDD_SOC current issue

Here we will compare SOC current when only kernel is loaded on A53 and when both A53 kernel and M4 image are loaded.

- SOC current when only kernel is loaded on A53

i.MX8MQ GA EVK dtb, no display, eth down, SDCard boot, idle		
	ARM	SOC
Voltage(V)	0.896	0.877
Playback Current (mA)	65	205
Power (mW)	58.24	179.785

- SOC current when both A53 kernel and M4 image are loaded.

i.MX8MQ GA EVK dtb, no display, eth down, SDCard boot, idle. Use SDK hello_world M4 image.		
	ARM	SOC
Voltage(V)	0.896	0.877
Playback Current (mA)	65	500
Power (mW)	58.24	438.5

Comparing the SOC current, we should see that **when M4 image is loaded, SOC current increases from 205 mA to 500 mA**, which is definitely abnormal.

4. Root cause of high VDD_SOC current issue

Normally, in kernel initialization, it disables all peripheral's clock gate.

But in a AMP applications environment, when kernel detects that M4 core is running, it will not disable all peripheral's clock gates, that is, all peripherals are enabled. So it will consume much more current.

In drivers/clk/imx/clk.h, function imx_clk_gate3 is used to enable/disable peripherals' clock gates.

```
static inline struct clk *imx_clk_gate3(const char *name, const char *parent,
        void __iomem *reg, u8 shift)
{
    /*
     * per design team's suggestion, clk root is NOT consuming
     * much power, and clk root enable/disable does NOT have domain
     * control, so they suggest to leave clk root always on when
     * M4 is enabled.
     */
    if (imx_src_is_m4_enabled())
        return clk_register_fixed_factor(NULL, name, parent,
                        CLK_SET_RATE_PARENT, 1, 1);
    else
        return clk_register_gate(NULL, name, parent,
            CLK_SET_RATE_PARENT | CLK_OPS_PARENT_ENABLE,
            reg, shift, 0, &imx_ccm_lock);
}
```

Macro `imx_src_is_m4_enabled()` is used to check if M4 is running.

So, when M4 is detected running, the peripheral's clock will be kept enabled.

In drivers/clk/imx/clk-imx8mq.c, function , when searching with imx_clk_gate3, you can find that a lot of peripheral's clock root gates are controlled by `imx_clk_gate3`.

```
clks[IMX8MQ_CLK_DRAM_ALT_CG] = imx_clk_gate3("dram_alt_cg", "dram_alt_src", base + 0xa000,
28);
 clks[IMX8MQ_CLK_DRAM_APB_CG] = imx_clk_gate3("dram_apb_cg", "dram_apb_src", base + 0xa080,
28);
  clks[IMX8MQ_CLK_VPU_G1_CG] = imx_clk_gate3("vpu_g1_cg", "vpu_g1_src", base + 0xa100, 28);
  …
  clks[IMX8MQ_CLK_PCIE2_PHY_CG] = imx_clk_gate3("pcie2_phy_cg", "pcie2_phy_src", base +
0xc080, 28);
  clks[IMX8MQ_CLK_PCIE2_AUX_CG] = imx_clk_gate3("pcie2_aux_cg", "pcie2_aux_src", base +
0xc100, 28);
    clks[IMX8MQ_CLK_ECSPI3_CG] = imx_clk_gate3("ecspi3_cg", "ecspi3_src", base + 0xc180, 28);
```

5. Solution

5.1. Preliminary knowledge on clock

There're some preliminary knowledges that we should point out.

- On iMX8M, we should be aware of that **almost all peripheral modules have clock gates and those gates can be separately controlled by 4 power domains**. See peripheral modules' clock gate registers below:

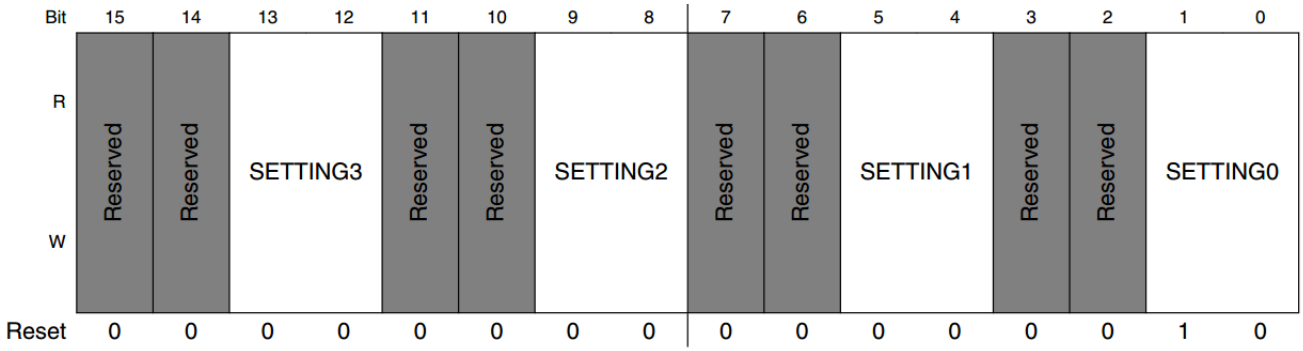

Figure 1. **Clock gate registers for peripheral modules**

SETTING0	Clock gate control setting for domain 0. This field can only be written by domain 0. 00 Domain clocks not needed 01 Domain clocks needed when in RUN 10 Domain clocks needed when in RUN and WAIT 11 Domain clocks needed all the time

Figure 2. **Clock gate control setting**

But **clock roots don't have this power domain control**. See clock roots register below:

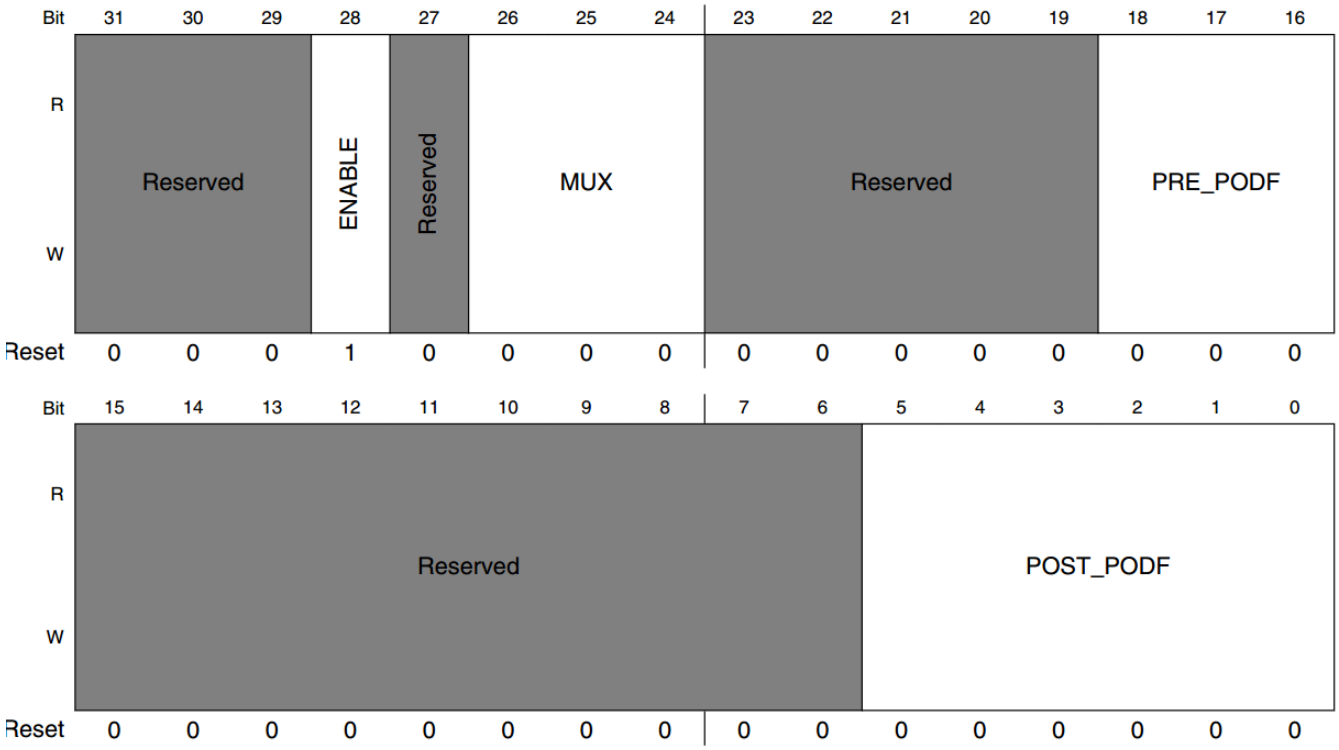

Figure 3. **Clock roots register**

So **kernel will be responsible for overall control of clock roots**.

- As kernel will control all clock roots and kernel don't know what root clocks that M4 image needed (that's why kernel enables all clock roots), we need to know what clock roots M4 image needed.

5.2. Code change

Now that we know why kernel enable all clock gates when M4 core is running and we need to know what clock roots M4 image needed, we can do the change on code now.

- Disable M4 core detection in clock gate function

  In drivers/clk/imx/clk.h, change imx_clk_gate3 as following:

```
static inline struct clk *imx_clk_gate3(const char *name, const char *parent,
                        void __iomem *reg, u8 shift)
    {
      return clk_register_gate(NULL, name, parent,
             CLK_SET_RATE_PARENT | CLK_OPS_PARENT_ENABLE,
             reg, shift, 0, &imx_ccm_lock);
 }
```

- Add M4 needed clocks

  In drivers/clk/imx/clk-imx8mq.c, add clks_init_on_m4 as following:

```
static int const clks_init_on_m4[] __initconst = {
          IMX8MQ_CLK_M4_CG,
        IMX8MQ_CLK_SAI1_CG, IMX8MQ_CLK_SAI2_CG,
        IMX8MQ_CLK_UART2_CG,
};
```

  The clks_init_on_m4 here is just an example. In real user applications, user should know what M4 root clocks are needed.

- Add clock init code in function imx8mq_clocks_init. We can add it right after initialization of A53 needed clock roots.

```
/* enable all the clocks just for bringup */
    for (i = 0; i < ARRAY_SIZE(clks_init_on);  i++)
        clk_prepare_enable(clks[clks_init_on[i]]);

    for (i = 0; i < ARRAY_SIZE(clks_init_on_m4);  i++)
    clk_prepare_enable(clks[clks_init_on_m4[i]]);
```

6. Power Consumption Test

After code change, we can now recompile kernel and boot board again. New power data is listed below:

i.MX8MQ GA EVK dtb, no display, eth down, SDCard boot, idle. Use SDK hello_world M4 image.		
	ARM	SOC
Voltage(V)	0.896	0.877
Playback Current (mA)	65	250
Power (mW)	58.24	438.5

The SOC current drops nearly 250mA.

7. Conclusion

This document mainly describes a power issue on i.MX8MQ platform and how to fix this issue. As there're no mechanism to communicate clock roots on i.MX8M, the solution should be applied according to different M4 user cases.
Users who are designing AMP applications on i.MX8M can make use of this document to optimize power on their code.

8. Revision history

Revision number	Date	Substantive changes
0	7/2018	Initial release

9. Reference

1. i.MX8MQ Reference Mannual

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: AN12225
Rev. 0
07/2018