NXP Semiconductors
Engineering Bulletin

Document Number: EB821
Rev. 0, 06/2016

LDB Clock Switch Procedure & i.MX6 Asynchronous Clock Switching Guidelines

Contents

1 Overview

Incorrect switching of clock sources can cause glitches which can lock up downstream clocking logic. Due to the absence of a clock gate after a specific clock multiplexer an exact clock switch sequence is required to ensure that the downstream clocking logic is not impacted.

This document briefly describes the NXP recommended steps to switch clocks when using asynchronous multiplexers. The document also describes the special procedure required when modifying the LVDS/LDB clock source as stated in the Errata ERR009219 CCM: Asynchronous clock switching of the LDB clock source can cause unpredictable behavior.

2 Clock switching multiplexers

There are a multitude of multiplexers available throughout the clock generation logic that provide alternate clock sources for the system clocks controlled by the Clock Control Module (CCM). The CCM utilizes several synchronous glitchless clock multiplexers as well as asynchronous glitchy clock multiplexers.

Synchronous multiplexers ensure there are no glitches between the transitions of two asynchronous clocks and that there will be no pulses that are of a frequency higher than either input clock. In order for the synchronous multiplexer to work properly, both the current clock and the clock to be selected must remain active during the entire selection process.

Asynchronous multiplexers or glitchy multiplexers (MUX), allow the clock to switch immediately after the multiplexer select is changed. Since both clock sources to the multiplexer are asynchronous, switching the clocks from one source to the other can cause a glitch to be generated, regardless of the input clock source. This immediate switch of two asynchronous clock domains can cause the output clock to glitch. If the input and output clocks are not gated, this clock glitch can propagate to the logic that follows the clock multiplexer, causing the logic to behave unpredictably.

3 LDB clock switch

Certain applications require the source clock of the LVDS Display Bridge (LDB) to be modified to accommodate various display clock frequency requirements. The clock source can be modified programming an asynchronous clock multiplexer (CCM_CS2CDR[LDB_DIx_CLK_SEL]) in software.

An incorrect programming when switching clocks inside the asynchronous (glitchy) LDB clock MUX prevents the clock being output. Since clock sources to the multiplexer are asynchronous, switching the clocks from the parent source (MMDC_CH1) to the destination source (PLL5 for example) can cause a glitch to be generated, regardless of the input clock source (PFD

or PLL). If the input clocks are not gated, clock glitches may propagate to the downstream logic that follows the clock multiplexer, causing unpredictable behavior.

A clock gate has not been implemented after the asynchronous clock multiplexer for the LDB_DI0_IPU clock and LDB_DI1_IPU clocks as shown in Figure 1. Due to the absence of this clock gate on this LDB_DIx_IPU clock path, a glitch generated when the clock source is switched, can lock up the LDB clock divider causing a loss of the LDB_DIx_IPU clock under certain conditions. Analysis has shown that the LDB clock divider is more susceptible to a lock up condition when the VDD_SOC supply voltage is lowered.

The input clocks to an asynchronous multiplexer are required to be gated before switching the source clock in the CCM clock multiplexer if the output of the glitchy multiplexer cannot be gated. This applies to any enabled clock tree path in the customer application that utilizes glitchy multiplexers without clock output gating abilities. If the glitchy LDB MUX (CCM_CS2CDR[LDB_Dix_clk_sel]) is used in the clock path for a customer application then the input clocks to the multiplexer must be gated before switching clock sources.

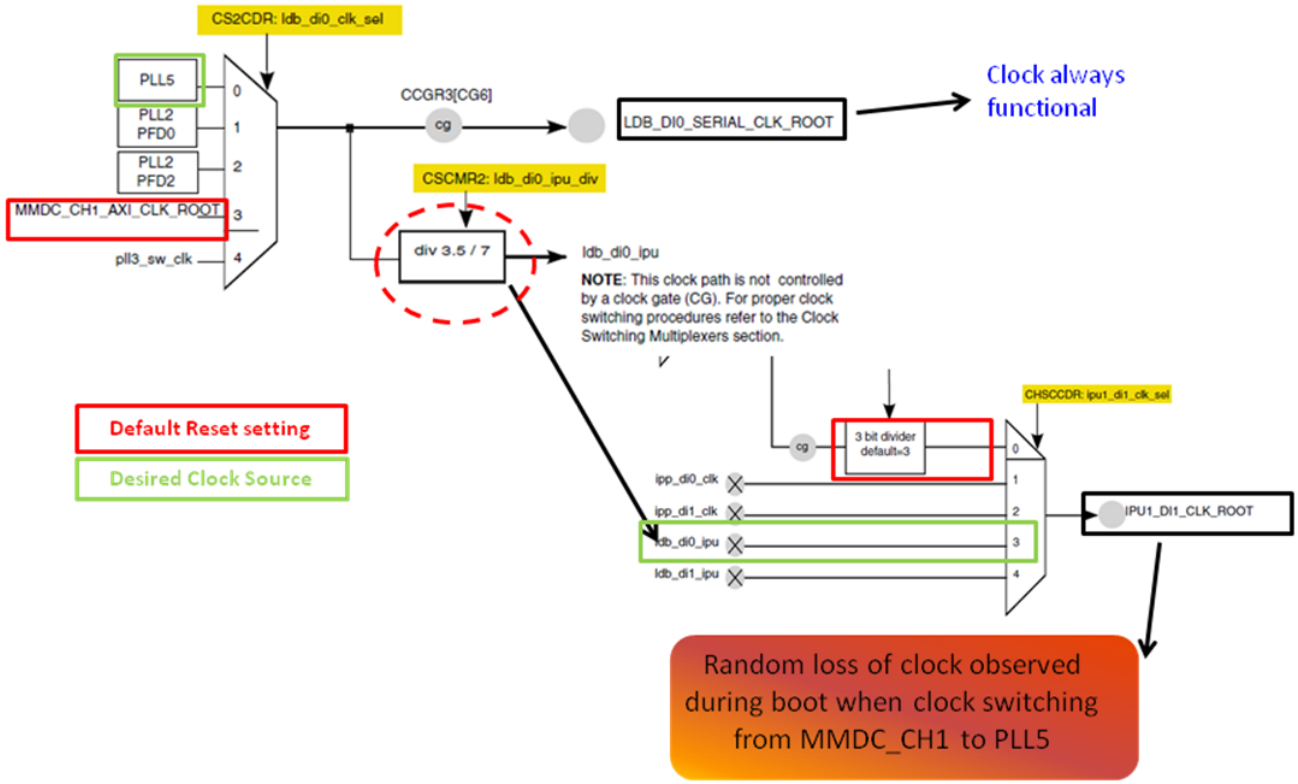

Figure 1. LDB Clocking on i.MX 6Dual/6Quad/6DualLite/6Solo

Since the clock gate CCGR3[CG6] can be used for the LDB_DI0_SERIAL_CLK_ROOT, users can disable the CCGR3[CG6] clock gate prior to switching the clock in the glitchy LDB MUX (*CS2CDR[LDB_Dix_clk_sel]*). This will ensure that no glitches propagate to downstream logic.

4 Switching procedure using a glitchy multiplexor

The recommended software procedure ensures that the input clocks to the asynchronous multiplexer are gated off before switching the source clock since the output of the glitchy multiplexer is not gated by a clock gate.

4.1 Clock switch concept

Software must follow the correct procedure to change clock sources.

- Clock switch must occur only when the MUX output is not in use.
- Software should gate (disable) the source clocks before changing the output clock of the multiplexer.
- The ideal procedure is to gate all input clocks without exception prior to switching. This is the cleanest approach since all mux inputs are disabled and there is no chance of glitch propagation downstream. However disabling all clocks is not always viable in the application, hence the minimum requirement for the LDB clock is to gate the current source input clock, the final destination clock, and pll3_sw_clk while using a special switch procedure.
- Once switched over, the source clocks can be re-enabled for a glitch free clock switch.
- It is recommended to perform the Clock switch procedure early in the initialization sequence to minimize clock tree impacts. The other clock switching should also be grouped together in the initial bootloader if possible.
- No software changes are required if the default MMDC_CH1 path is used since no clock switching is required.

Switching of glitchy multiplexers must not be performed in the DCD boot configuration since the correct procedure can not be implemented using this approach.

As long as the clock multiplexer switch occurs early on in the initialization code, the required procedure should have minimal impact since most modules requiring the PFD clock sources are not yet initialized. If it is performed later, then users must ensure that no module is using the clock that has to be gated for the clock switch to occur.

4.2 LDB clock source change detailed steps

Further details of the LDB Clock mux (CS2CDR) are shown in Figure 2 below.The following example lists the steps required to change the default LDB clock source from MMDC_CH1 to PLL5.

NOTE

Only the current source, the future source, and pll3_sw_clk are manipulated.

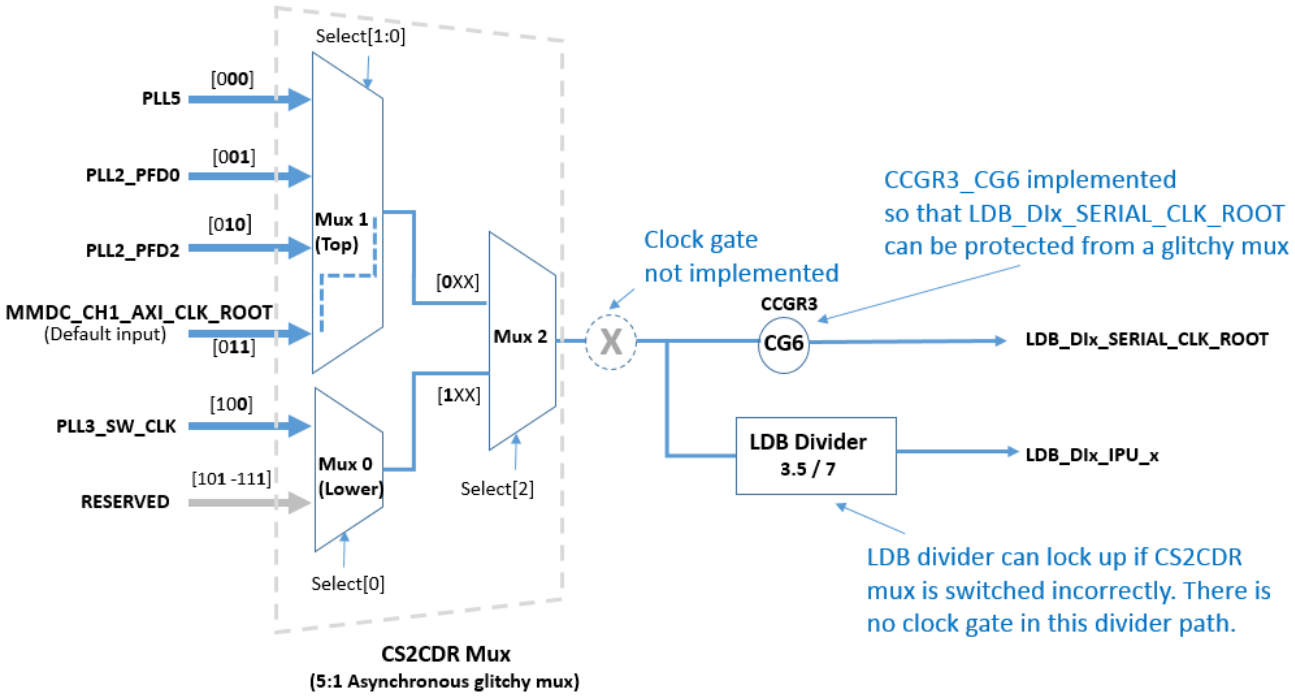

Figure 2. Missing clock gate

1. Disable the new source clock (PLL5 in this example).
2. Disable the old source clock. In this example, it is MMDC_CH1. To disable MMDC_CH1, first disable the MMDC_CH1 handshake in CCM_CCDR and then use the following steps:
   a. Set periph2_clk2_sel to be sourced from pll3_sw_clk.
   b. Switch periph2_clk_sel to periph2_clk2 clock source.
   c. Wait for the periph2_clk_sel_busy bit to clear in CCM_CDHIPR register.
3. Switch pll3_sw_clk to bypass source by writing to pll3_sw_clk_sel bit in CCM_CCSR register.

NOTE

This step is only needed if the LDB clock switch is done when the system is already up and running. It is best to disable PLL3 if you are changing the parent very early in the boot process, before any active PLL3 user. There is no need to write the CCM_CCSR bit to switch pll3_sw_clk and bypass the source, if PLL3 is not enabled.

NOTE

The pll3_sw_clk bypass source is used for NXP test purposes and hence should not be used in the customer application other than described in this procedure. It is an unconditional requirement for the safe switch procedure to have pll3_sw_clk disabled.

4. Change the CCM_CS2CDR[LDB_DIx_CLK_SEL] value to use the new source clock with the following procedure that treats the msb in a special way.
   a. The old source in this example is MMDC_CH1 (b'011) and the new source is PLL5 (b'000).
   b. Write LDB_DIx_CLK_SEL with the old source but set the msb. For example, b'011 ==> write b'111. MSB controls if the upper or lower mux is selected.
   c. Write LDB_DIx_CLK_SEL with the two least significant bits of the new source but keep the msb set. For example, b'000 ==> write b'100. The two least significant bits control the upper mux.
   d. Write LDB_DIx_CLK_SEL with the new source value that has a cleared msb now. For example, b'000 ==> write b'000.
5. Re-enable the old source clock if needed elsewhere in the system. In this example it is MMDC_CH1 which needs to be re-enabled with the following steps:
   a. Switch periph2_clk_sel to pll2_main_clk options (original parent of MMDC_CH1).
   b. Wait for the periph2_clk_sel_busy bit to clear in CCM_CDHIPR register.
6. Switch pll3_sw_clk to PLL3 by writing to pll3_sw_clk_sel bit in CCM_CCSR register.
7. Enable MMDC_CH1 handshake in CCM_CCDR.
8. Enable new source clock, PLLs, and associated PFDs as needed. Ensure PFD's are reset properly.
9. The clock switch is now complete; continue with customer application or boot process.

The Linux kernel, for example, switches to PLL5 and disables all PFD clock sources. The PFD clock sources come out of reset in an enabled state.

Since PLL2-PFD0 and PLL2-PFD2 are inputs to the mux, it would be best to disable these clocks. But if the application code does not need them as new source clock and follows the special switch procedure, there is no need to disable these PFD clock sources.

If the customer application is switching to either PLL2-PFD0 or PLL2-PFD2, then the PFDs would need to be disabled, just as PLL5 was disabled in the example above. For systems using multiple LDB clocks (LDB_DI1_IPU), the above procedure would have to be followed when switching clock sources as well.

NOTE

For more information see the example sequence at: http://git.freescale.com/git/cgit.cgi/imx/linux-2.6-imx.git/commit/?h=imx_3.10.17_1.0.1_ga&id=eecbe9a52587cf9eec30132fb9b8a6761f3a1e6d.

5 General clock switching recommendations

The input clocks to the asynchronous multiplexer are required to be gated before switching the source clock in the CCM clock multiplexer and the output should also be gated.

For serial clocks, software should first disable the module, and then gate its clock in the LPCG. Then it should move the multiplexer controlling the source of the clocks to another PLL, and reset the module and its clocks. Only then is it safe to disable the PLL. The multiplexer for the serial clocks is not glitchless.

NOTE

The Boot ROM does configure certain muxes based on the particular boot mode used by the application. Correct clock switching techniques are used by the ROM. The reset values of certain CCM registers hence may not match the documented reset values of these muxes. Any further clock switching in the application must use the recommended clock switch procedures.

Examples of other multiplexors on i.MX6DQ/DLS which do not have a gating option on the output are listed below:

- CDCCR - SPIDF Clock mux
- CSMR1 - ACLK_CLK_ROOT mux (not used in i.MX6DLS)

NOTE

If during a clock switch operation, PLL2/PLL3, which generate the PFD clocks are powered up from the power down state or made to go through a relock cycle, customer application software is required to reset the respective PFDs using the PFDx_CLKGATE bits. The phase fractional dividers (PFDs) used in PLL2 and PLL3 can go into an unknown state if they are not properly reset before being used as clock source. The PFDs can be in the clock gated state during PLL relock, but must be un-clock gated only after lock is achieved. For further details, see EB790.

Synchronous clock multiplexers ensure there are no glitches between the transitions of two asynchronous clocks and that there will be no pulses that are of a frequency higher than either input clock. In order for the synchronous multiplexer to work properly, both the current clock and the clock to be selected must remain active during the entire selection process.

5.1 Clock switching when enabling spread spectrum

The System PLL (PLL2) supports spread spectrum modulation for use in applications to minimize radiated emissions. To enable this feature requires relocking the System PLL. Since the System PLL is source of many system clocks it is recommended to enable Spread spectrum early in the boot sequence to minimize system software impact, resulting from the clock tree changes.

To relock the System PLL does require users to switch the system to an alternate clock while system PLL relocks. This clock switching procedure is similar to the procedure used for switching asynchronous clock multiplexors.

Additional handshaking procedures with the MMDC are required for other i.MX 6 series devices when the MMDC clock is being changed and sourced from another PLL, instead of the default System PLL2. The code to perform this clock switch should execute out of internal RAM - On Chip RAM(OCRAM).

There are several steps required to change clocking.

- If you are changing clock setup, you must bypass PLL first.
- PLL must be powered down in order to enable spread spectrum.

- When you re-lock PLL you must wait for the PLL to lock.

Proper procedure is to:

1. Bypass PLL.
2. Power down PLL.
3. Enable spread spectrum.
4. Re-enable PLL.
5. Wait for cycles amounting to PLL lock time(11250 cycles are required for PLL2) or have the loop continue till the LOCK bit is set. Waiting for the LOCK bit is the recommended approach.
6. Take PLL out of bypass (unbypass).

6 References

1. i.MX 6Dual/6Quad Applications Processor Reference Manual - IMX6DQRM
2. Chip Errata for the i.MX 6Dual/6Quad and i.MX 6DualPlus/6QuadPlus - IMX6DQCE
3. EB790: Configuration of Phase Fractional Dividers
4. i.MX 6Solo/6DualLite Applications Processor Reference Manual - IMX6SDLRM
5. Chip Errata for the i.MX 6Solo/6DualLite - Errata - IMX6SDLCE

7 Revision history

TThe following table summarizes the revisions to this document since the release.

Table 1. Revision history

Revision	Date	Substantive changes
0	06/2016	Initial release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

EB821
Rev. 0
06/2016