Freescale Semiconductor
Application Note

Document Number: AN4140
Rev. 0, 06/2010

i.MX51 WinCE™ Clock Setting

by Multimedia Applications Division
Freescale Semiconductor, Inc.
Austin, TX

Contents

1 Introduction

This application note gives an introduction about the i.MX51 processor clock settings under the WinCE 6.0 Board Support Package (BSP). The application note also discusses the Digital Phase Locked Loop (DPLL), Digital Phase Locked Loop-Intellectual Property (DPLL-IP), and the Clock Control Module (CCM) settings under the i.MX51 WinCE 6.0 BSP. The user can get a better understanding about the i.MX51 clock settings and configuring the i.MX51 clock tree using software.

NOTE

The software code mentioned in this application note are based on the Freescale WinCE 6.0 BSP release, WCE600-MX51TO2-FC-ER 10.

2 Clock Setting Overview

The i.MX51 processor has three internal DPLLs for providing clocks to the CCM. These DPLLs are controlled by the DPLL-IP interface, and the clock-source for the each DPLL is selected by using DPLL-IP.

The reference clock options for the DPLLs are as follows:

- Output of OSC (typical functional frequency of 24 MHz)
- Output of FPM (typical functional frequency of 32.768 MHz)

Selection of the DPLL reference clock is controlled by the CLKSS signal. If CLKSS is at logic 0, output of OSC is used as the reference clock for all the three DPLLs. If CLKSS is at logic 1, output of FPM is used as the reference clock for all the DPLLs.

The CCM controls the clocks for the i.MX51 modules. CCM takes the available clocks from the DPLLs to generate root clocks for different i.MX51 modules and the ARM core. CCM also provides clock-gating functions that helps in the optimum utilization of power. In the absence of any module, the CCM disables the clock to the module and reduces power consumption.

Figure 1 shows the i.MX51 clock framework that consists of DPLL-IPs, DPLLs, and CCM.

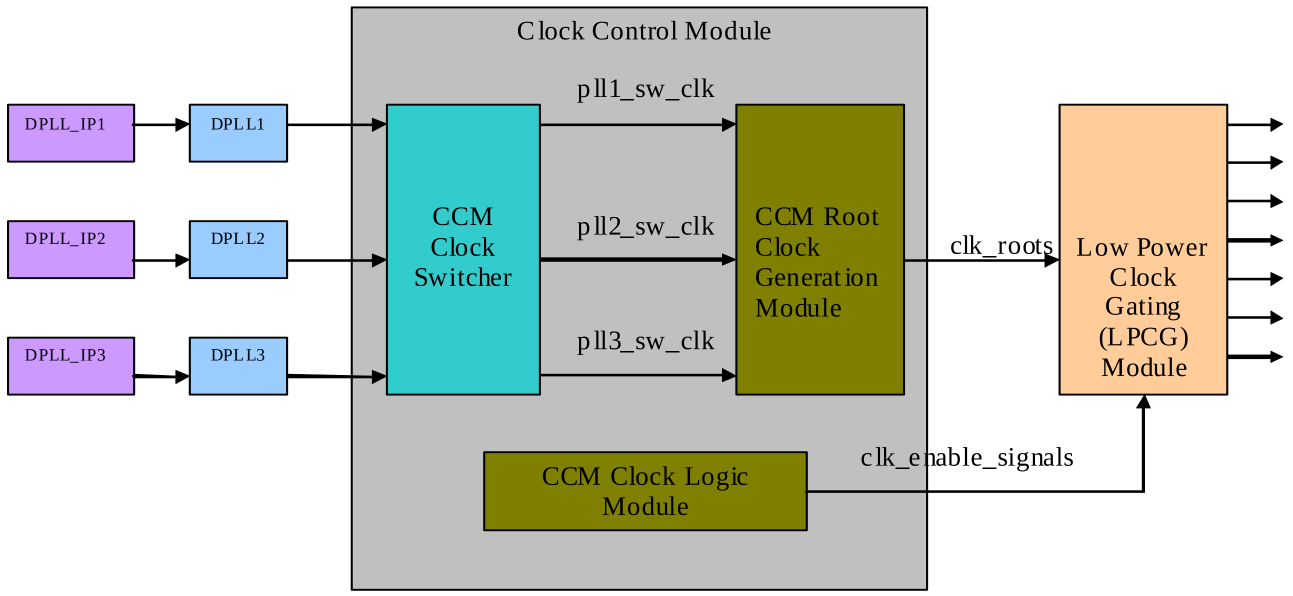

Figure 1. i.MX51 Clock Framework

The modules in Figure 1 are described as follows:

- DPLL_IP and DPLL—DPLL_IP serves as the interface for the DPLL module. DPLL_IP contains control/operation registers to control the DPLL operations. Software uses these registers to set the output clock frequency of the three DPLLs.
- CCM clock switcher—CCM clock switcher receives clock outputs from the three DPLLs or from other bypass clock-source. The CCM clock switcher has three clock outputs: pll1_sw_clk, pll2_sw_clk, and pll3_sw_clk.

- CCM root clock generation module—this module receives the three main clocks (pll1_sw_clk, pll2_sw_clk, and pll3_sw_clk) from the CCM clock switcher and generates output root clocks for the different modules in the i.MX51 processor.
- CCM clock logic module—this module generates the clk_enable_signals, which are used by the Low Power Clock Gating (LPCG) module to switch ON/OFF the distributed clocks.

3 WinCE BSP Clock Settings

This section describes the i.MX51 BSP clock settings in the WinCE platform.

3.1 DPLL/DPLL_IP Settings

In the i.MX51 Product Development Kit (PDK) design, the CLKSS signal is set to logic 0. Therefore, the 24 MHz OSC output clock is set as the reference clock for the three DPLLs. The DPLL_IP module contains registers to configure the output of each DPLL.

The output clock frequency of DPLL1, DPLL2, and DPLL3 is calculated by Equation 1 and Equation 2:

$$\textit{DPLL1/DPLL2 output clock frequency} = 4 \times \textit{DPLL reference clock} \times (MFI + MFN \div (MFD + 1)) \div (PDF + 1) \quad \textbf{Eqn. 1}$$

$$\textit{DPLL3 output clock frequency} = 2 \times \textit{DPLL reference clock} \times (MFI + MFN \div (MFD + 1)) \div (PDF + 1) \quad \textbf{Eqn. 2}$$

The terms in Equation 1 and Equation 2 are described as follows:

- DPLL reference clock—24 MHz OSC output clock is set as the DPLL reference clock for the i.MX51 PDK board design.
- MFI—integer part of a Multiplication Factor (MF)
- MFN and MFD—numerator and denominator of the MF fraction
- PDF—Predivision Factor

For the i.MX51 Windows CE 6.0 ER10 BSP, the output clocks of the three DPLLs are set in the `WINCE600\PLATFORM\iMX51-3DS\SRC\OAL\OALLIB\startup.s` file. The i.MX51 tape out version, TO2.0 setting is taken as the example for calculating the output clock frequencies based on the Equation 1 and Equation 2:

- DPLL1:
  Output clock frequency = 800 MHz
  MFI = 8, MFN = 1, MFD = 2, and PDF = 0
  DPLL1 output clock frequency = 4 × 24 × (8 + 1 ÷ (2 + 1)) ÷ (0 + 1) = 800 MHz
- DPLL2:
  Output clock frequency = 665 MHz
  MFI = 6, MFN = 89, MFD = 95, and PDF = 0
  DPLL2 output clock frequency = 4 × 24 × (6 + 89 ÷ (95 + 1)) ÷ (0 + 1) = 665 MHz
- DPLL3:
  Output clock frequency = 216 MHz
  MFI = 9, MFN = 0, MFD = 0, and PDF = 1
  DPLL3 output clock frequency = 2 × 24 × (9 + 0 ÷ (0 + 1)) ÷ (1 + 1) = 216 MHz

3.2 CCM Clock Switcher Setting

The CCM clock switcher module receives the output clocks or bypass clocks of the three DPLLs and generates three main switcher clocks for the CCM root clock generation module.

The three main switcher clocks are as follows:

- pll1_sw_clk—has a typical functional frequency of 800 MHz and is used to supply clocks in the ARM platform
- pll2_sw_clk—has a typical functional frequency of 600 MHz and is used to supply clocks for Advanced eXtensible Interface (AXI)/Advanced High-performance Bus (AHB)/Intellectual Property (IP) buses
- pll3_sw_clk—has a typical functional frequency of 216 MHz and is used to supply serial clocks for Universal Serial Bus (USB), Synchronous Serial Interface (SSI), and so on

Figure 2 shows the CCM clock switcher module architecture.

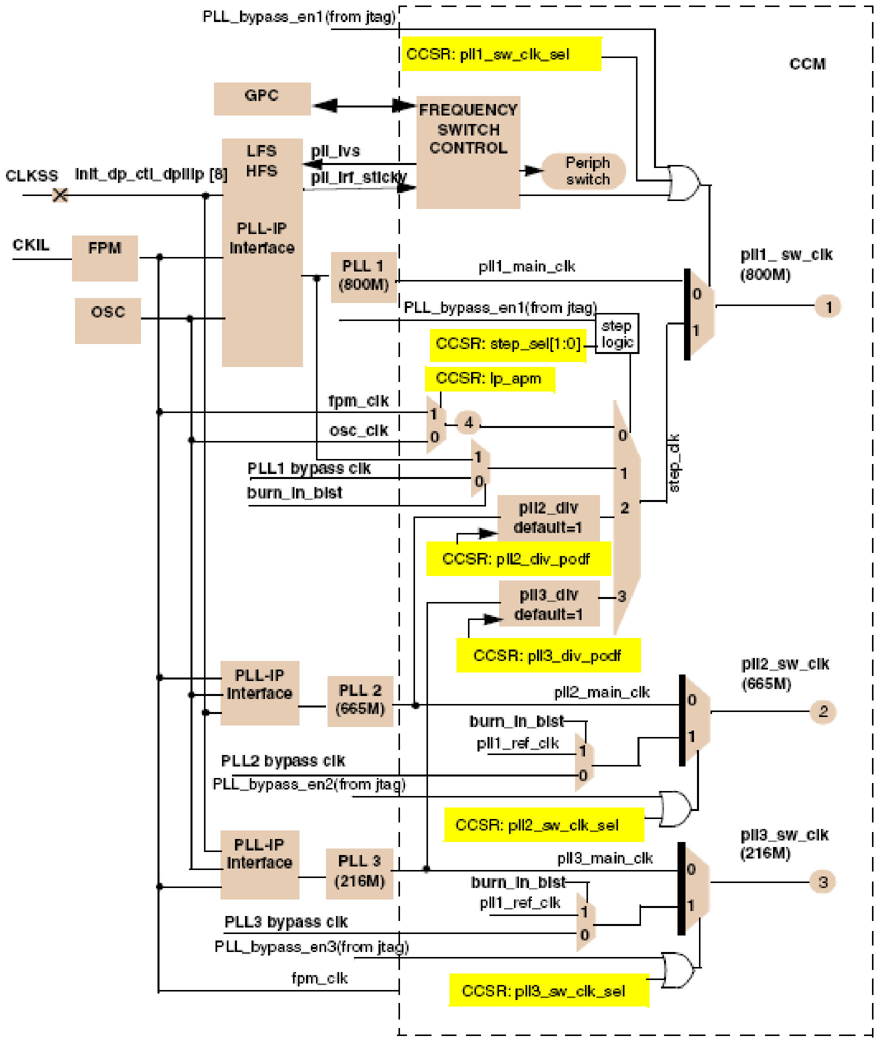

Figure 2. CCM Clock Switcher Module Architecture

As seen in Figure 2, the clock-source for each switcher clock (pll1_sw_clk, pll2_sw_clk, and pll3_sw_clk) is described as follows:

- Clock-source for pll1_sw_clk:
  - pll1_main_clk—pll1_main_clk is the clock output of DPLL1 (shown as PLL1 in Figure 2).
  - step_clk—step_clk is derived from any one of the following four clocks, based on the step_sel[1:0] field setting of the CCM Clock Switcher Register (CCSR). The four clocks are as follows:
    - lp_apm clock—lp_apm clock is derived from fpm_clk or osc_clk, depending on the lp_apm field setting of the CCSR register.
    - DPLL1 bypass clock or DPLL1 reference clock
    - DPLL2 clock with divider—the divider is determined by the pll2_div_podf[1:0] field setting of the CCSR register.
    - DPLL3 clock with divider—the divider is determined by the pll3_div_podf[1:0] field setting of the CCSR register.

  Selection between pll1_main_clk and step_clk is determined by the pll1_sw_clk_sel field of the CCSR register.
- Clock-source for pll2_sw_clk:
  - pll2_main_clk—clock output of DPLL2
  - DPLL2 bypass clock or DPLL1 reference clock

  Selection between pll2_main_clk and DPLL2 bypass clock (or DPLL1 reference clock) is determined by the pll2_sw_clk_sel field of the CCSR register.
- Clock-source for pll3_sw_clk:
  - pll3_main_clk—clock output of DPLL3
  - DPLL3 bypass clock or DPLL1 reference clock

  Selection between pll3_main_clk and DPLL3 bypass clock (or DPLL1 reference clock) is determined by the pll3_sw_clk_sel field of the CCSR register.

From the description about the clock-source for the switcher clocks, it is evident that the CCM clock switcher is configured by the CCSR register. In the i.MX51 Windows CE 6.0 ER10 BSP, the CCM clock switcher is configured in the `WINCE600\PLATFORM\iMX51-3DS\SRC\OAL\OALLIB\startup.s` file. Refer to the lines from 652 to 714 of the CCSR register setting. From this segment of code, it is evident that the three switcher clocks (pll1_sw_clk, pll2_sw_clk, and pll3_sw_clk) are derived from their corresponding DPLLs.

Therefore, the clock frequency for the each switcher clock is as follows:

- pll1_sw_clk—is the clock output of the CCM clock switcher, corresponding to DPLL1 that has a frequency of 800 MHz
- pll2_sw_clk—is the clock output of the CCM clock switcher, corresponding to DPLL2 that has a frequency of 665 MHz

- pll3_sw_clk—is the clock output of the CCM clock switcher, corresponding to DPLL3 that has a frequency of 216 MHz

NOTE

The pll1_sw_clk should be changed to step_clk (lp_apm) clock before resetting. The reason is that when DPLL1 is reset, the DPLL1 output clock is not stable. To maintain system modules running, the pll1_sw_clk should be switched to step_clk temporarily. Once the DPLL1 output clock is locked to 800 MHz, the pll1_sw_clk is switched back to 800 MHz.

3.3 Bus Clock Setting and Related Module Clocks

The CCM root clock generation module generates root clocks for the i.MX51 buses and peripherals. The clock-sources for the different peripherals are called bus clocks. This section discusses the bus clock settings in the i.MX51 processor.

The bus clocks for the i.MX51 processor are as follows:

- axi_a clock
- axi_b clock
- External Memory Interface (EMI) slow clock root
- Advanced High-performance Bus (AHB) clock root
- PERCLK root

Figure 3 shows the architecture for the bus clock tree.

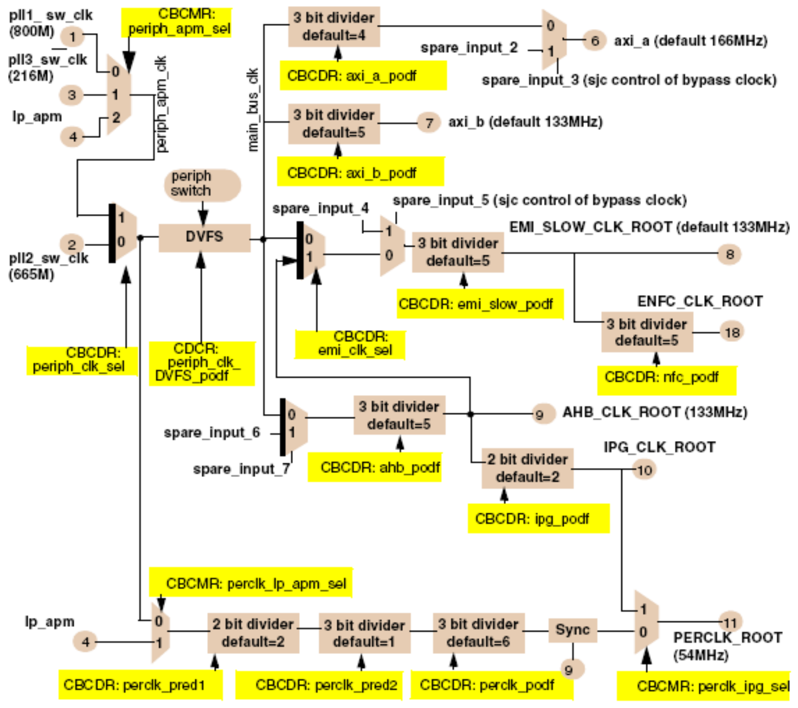

Figure 3. Bus Clock Tree Architecture

Some of the basic clocks, which are used as source-clocks for the bus clocks, are as follows:

- periph_apm_clk—periph_apm_clk is one of the source-clocks to generate main_bus_clk and is derived from any one of the three clocks: pll1_sw_clk, pll3_sw_clk, and lp_apm clock. Selection among the three clocks is determined by the periph_apm_sel field setting of the CCM Bus Clock Multiplexer Register (CBCMR). For some typical cases, pll2_sw_clk is chosen as the source-clock for the main_bus_clk. In situations when DPLL2 is reset and pll2_sw_clk is not stable, periph_apm_clk can be used as the source-clock for the main_bus_clk.
- main_bus_clk—main_bus_clk is the direct clock-source for different bus clocks and is derived from either of the two clocks: periph_apm_clk and pll2_sw_clk. Selection between the two clocks is determined by the periph_clk_sel field setting of the CBCDR register.

The bus clocks, which are shown in Figure 3, are described as follows:

- axi_a clock—axi_a clock is derived from the main_bus_clk with 3-bit divider. The 3-bit divider is configured by the axi_a_podf field of the CBCDR register.
- axi_b clock—axi_b clock is derived from the main_bus_clk with 3-bit divider. The 3-bit divider is configured by the axi_b_podf field of the CBCDR register.
- AHB_CLK_ROOT clock—AHB_CLK_ROOT is derived from the main_bus_clk with 3-bit divider. The 3-bit divider is configured by the ahb_podf field of the CBCDR register.
- EMI_SLOW_CLK_ROOT clock—EMI_SLOW_CLK_ROOT clock is derived from the main_bus_clk or AHB_CLK_ROOT. The selection between main_bus_clk and AHB_CLK_ROOT is determined by the emi_clk_sel field of the CBCDR register. The reference clock of EMI_SLOW_CLK_ROOT is divided by 3-bit divider. The 3-bit divider is configured by the emi_slow_podf field of the CBCDR register.
- IPG_CLK_ROOT clock—IPG_CLK_ROOT clock is derived from the AHB_CLK_ROOT clock with 2-bit divider. The 2-bit divider is configured by the ipg_podf field of the CBCDR register.
- PERCLK_ROOT clock—PERCLK_ROOT clock is derived from the main_bus_clk or lp_apm clock. The selection between main_bus_clk and lp_apm clock is determined by the perclk_lp_apm_sel field of the CBCDR register. The reference clock of PERCLK_ROOT is divided by three dividers: 3-bit divider configured by the perclk_podf field of the CBCDR register, 2-bit divider configured by the perclk_pred1 field of the CBCDR register, and 3-bit divider configured by the perclk_pred2 field of the CBCDR register.

Some module clocks use the bus clocks as the reference clocks. These module clocks are as follows:

- GPU2D_CLK_ROOT
  Figure 4 shows the GPU2D_CLK_ROOT clock multiplexer.

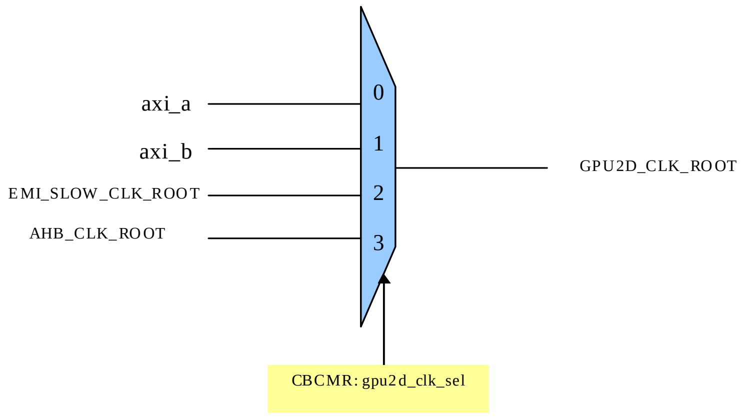

Figure 4. GPU2D_CLK_ROOT Clock Multiplexer

  From Figure 4, it is evident that GPU2D_CLK_ROOT is derived from any one of the four clocks: axi_a, axi_b, EMI_SLOW_CLK_ROOT, and AHB_CLK_ROOT. Selection among the four clocks is determined by the gpu2d_clk_sel field setting of the CBCMR register.
- ARM_AXI_CLK_ROOT
  Figure 5 shows the GPU2D_CLK_ROOT clock multiplexer.

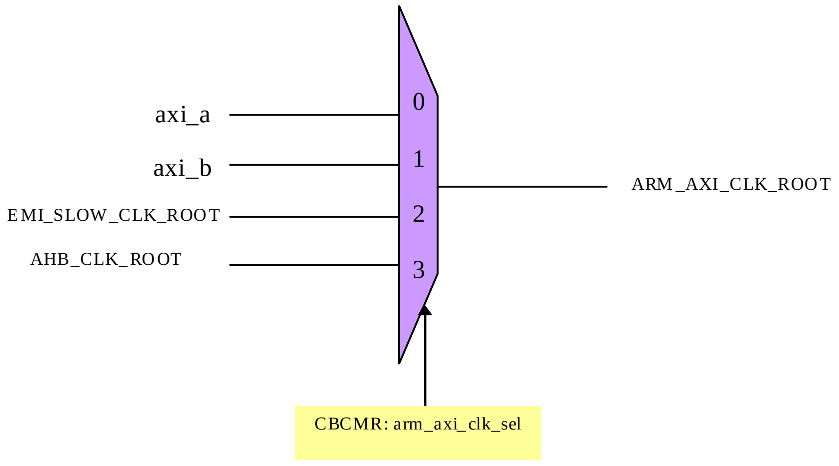

Figure 5. ARM_AXI_CLK_ROOT Clock Multiplexer

  From Figure 5, it is evident that ARM_AXI_CLK_ROOT is derived from any one of the four clocks: axi_a, axi_b, EMI_SLOW_CLK_ROOT, and AHB_CLK_ROOT. Selection among the four clocks is determined by the arm_axi_clk_sel field setting of the CBCMR register.

- IPU_HSP_CLK_ROOT—IPU_HSP_CLK_ROOT is taken as the reference clock for the Image Processing Unit (IPU) module. The IPU module takes this clock to generate the pixel clock by DI sub-module of the IPU.

  Figure 6 shows the IPU_HSP_CLK_ROOT clock multiplexer.

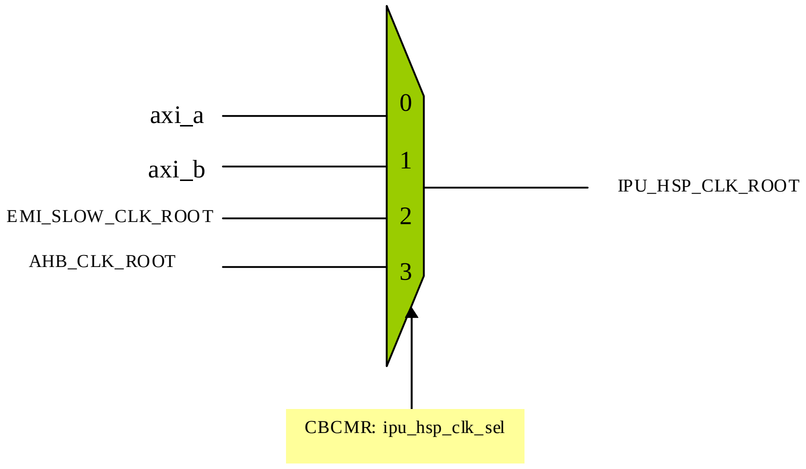

Figure 6. IPU_HSP_CLK_ROOT Clock Multiplexer

From Figure 6, it is evident that IPU_HSP_CLK_ROOT is derived from any one of the four clocks: axi_a, axi_b, EMI_SLOW_CLK_ROOT, and AHB_CLK_ROOT. Selection among the four clocks is determined by the ipu_hsp_clk_sel field setting of the CBCMR register.

- GPU_CLK_ROOT

  Figure 7 shows the GPU_CLK_ROOT clock multiplexer.

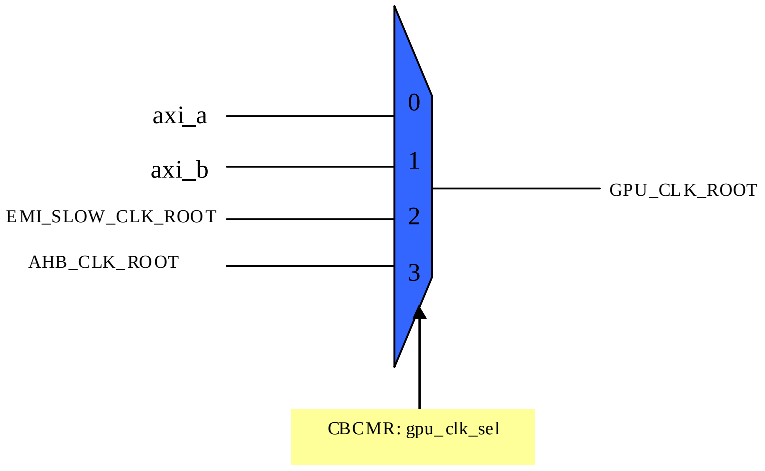

Figure 7. GPU_CLK_ROOT Clock Multiplexer

From Figure 7, it is evident that GPU_CLK_ROOT is derived from any one of the four clocks: axi_a, axi_b, EMI_SLOW_CLK_ROOT, and AHB_CLK_ROOT. Selection among the four clocks is determined by the gpu_clk_sel field setting of the CBCMR register.

- VPU_AXI_CLK_ROOT
  Figure 8 shows the VPU_AXI_CLK_ROOT clock multiplexer.

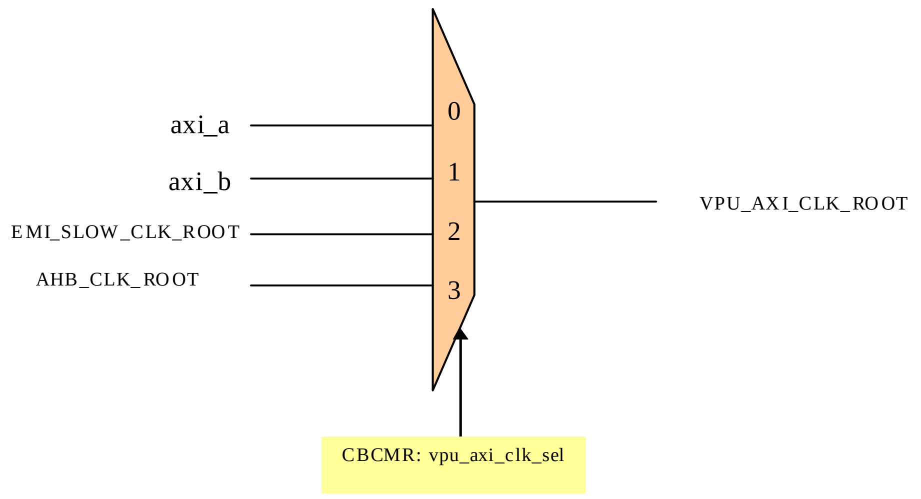

Figure 8. VPU_AXI_CLK_ROOT Clock Multiplexer

From Figure 8, it is evident that VPU_AXI_CLK_ROOT is derived from any one of the four clocks: axi_a, axi_b, EMI_SLOW_CLK_ROOT, and AHB_CLK_ROOT. Selection among the four clocks is determined by the vpu_axi_clk_sel field setting of the CBCMR register.

- DEBUG_APB_CLK_ROOT
  Figure 9 shows the DEBUG_APB_CLK_ROOT clock multiplexer.

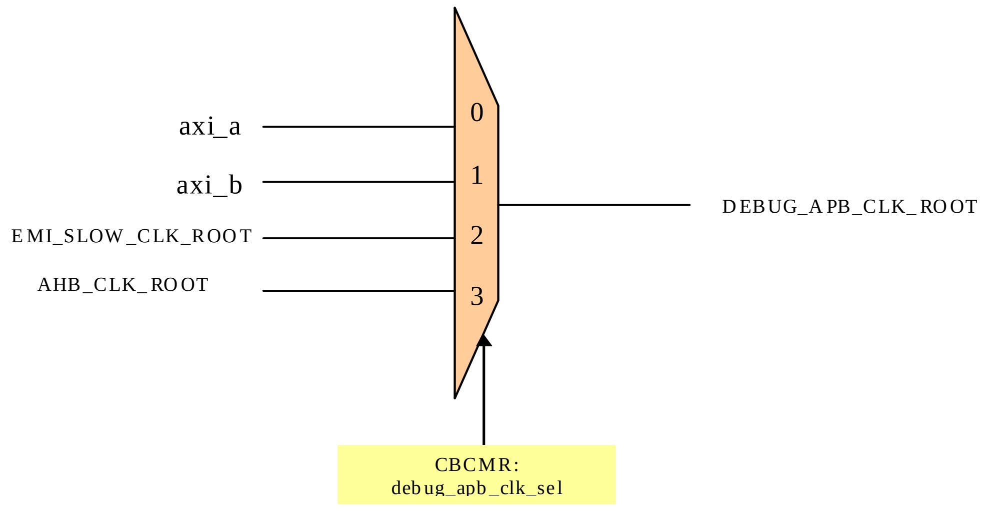

Figure 9. DEBUG_APB_CLK_ROOT Clock Multiplexer

From Figure 9, it is evident that DEBUG_APB_CLK_ROOT is derived from any one of the four clocks: axi_a, axi_b, EMI_SLOW_CLK_ROOT, and AHB_CLK_ROOT. Selection among the four clocks is determined by the debug_apb_clk_sel field setting of the CBCMR register.

- DDR_CLK_ROOT
  Figure 10 shows the DDR_CLK_ROOT clock multiplexer.

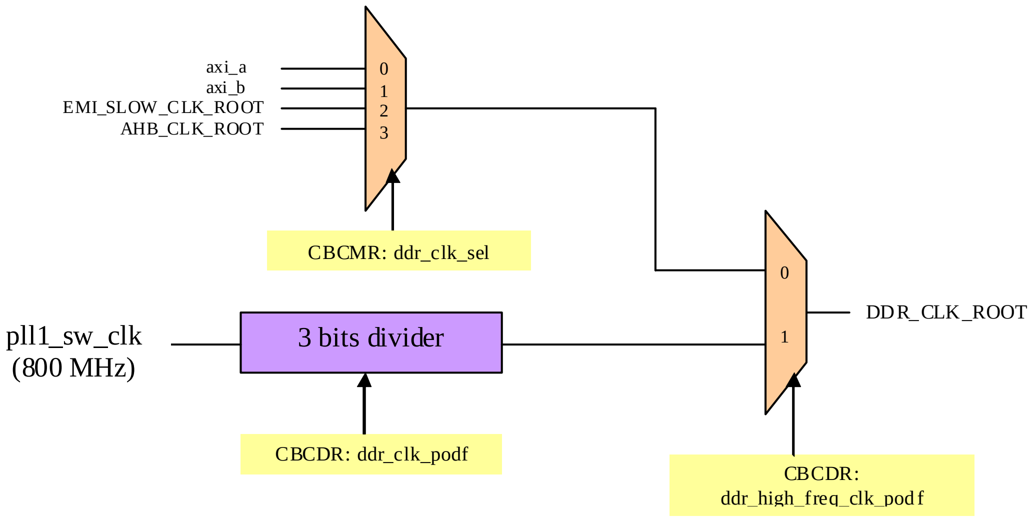

Figure 10. DDR_CLK_ROOT Clock Multiplexer

From Figure 10, it is evident that DDR_CLK_ROOT is derived from any one of the five clocks: pll1_sw_clk, axi_a, axi_b, EMI_SLOW_CLK_ROOT, and AHB_CLK_ROOT. If ddr_high_freq_clk_podf field of the CBCDR register is set to logic 1, pll1_sw_clk is taken as the reference clock for the DDR_CLK_ROOT. If ddr_high_freq_clk_podf field of the CBCDR register is set to logic 0, reference clock for the DDR_CLK_ROOT is determined by the ddr_clk_sel field of the CBCMR register. If pll1_sw_clk is taken as the reference clock for the DDR_CLK_ROOT, then the pll1_sw_clk encounters a 3-bit divider. The 3-bit divider is configured by the ddr_clk_podf field of the CBCDR register.

Therefore, from the description in this section, it is evident that the CCM bus clocks and bus clock related module clocks are configured by the CBCMR and CBCDR registers. For the i.MX51 Windows CE 6.0 ER10 BSP, the CCM bus clocks and bus clock related module clocks are configured in the `WINCE600\PLATFORM\iMX51-3DS\SRC\OAL\OALLIB\startup.s` file.

Refer to the following segment of codes of the `startup.s` file for the bus clocks setting:

- Segment 1—lines 725–734 of the `startup.s` file

```
725    ; Configure PERIPH_APM_CLK to be sourced from PLL3 (mux input 1)
726    ldr      r0, [r4, #CCM_CBCMR_OFFSET]
727    orr      r0, r0, #CCM_CBCMR_PERIPH_APM_SEL0
728    bic      r0, r0, #CCM_CBCMR_PERIPH_APM_SEL1
729    str      r0, [r4, #CCM_CBCMR_OFFSET]
730
731    ; Configure main_bus_clk to be sourced from PERIPH_APM_CLK (mux input 1)
732    ldr      r0, [r4, #CCM_CBCDR_OFFSET]
733    orr      r0, r0, #CCM_CBCDR_PERIPH_CLK_SEL
734    str      r0, [r4, #CCM_CBCDR_OFFSET]
```

In the lines 725–729 of the `startup.s` file, the periph_apm_sel field of the CBCMR register is set to logic 1. This means that the pll3_sw_clk is taken as the reference clock for the periph_apm_clk.

In the lines 731–734 of the `startup.s` file, the periph_clk_sel field of the CBCDR register is set to logic 1. This means that the periph_apm_clk is taken as the reference clock for the main_bus_clk. Therefore, from the lines 725–734, it is evident that the main_bus_clk is derived from the pll3_sw_clk.

NOTE

This is a temporary status for the main_bus_clk. After hardware reset, the default output clock frequency for the DPLL2 becomes 262.144 MHz. For WinCE eBoot, the output clock frequency of the DPLL2 is changed to 665 MHz and is restarted. Before DPLL2 is restarted, DPLL3 output (pll3_sw_clk) is temporarily used as the reference clock for the main_bus_clk. After the DPLL2 output clock is restarted and locked to the frequency of 665 MHz, the pll2_sw_clk is used as the reference clock for the main_bus_clk.

- Segment 2—lines 736–753 of the `startup.s` file
  In this segment of code, the DPLL2 is configured to restart.

NOTE

Here, DPLL2 output clock is not locked and cannot be used as the reference clock for other clocks, such as bus clocks.

- Segment 3—lines 778–793 (take the tape out version, TO2 as the example for analysis) and 815–835 of the `startup.s` file (take the tape out version, TO2 as the example for analysis)

```
778    ; CCM Bus Clock Multiplexer Register (CBCMR)
779    ;     perclk_ipg_sel = PERCLK_LP_APM (0 << 0)      = 0x00000000
780    ;     perclk_lp_apm_sel = LP_APM = (1 << 1)        = 0x00000002
781    ;     debug_apb_clk_sel = AXI A = (0 << 2)         = 0x00000000
782    ;     gpu_clk_sel = AXI A = (0 << 4)               = 0x00000000
783    ;     ipu_hsp_clk_sel = AXI_B = (1 << 6)           = 0x00000040
784    ;     arm_axi_clk_sel = AXI A = (0 << 8)           = 0x00000000
785    ;     ddr_clk_sel = AXI A = (0 << 10)              = 0x00000000
786    ;     periph_apm_sel = lp_apm_clock = (2 << 12)    = 0x00002000
787    ;     vpu_axi_clk_sel = AXI A = (0 << 14)          = 0x00000000
788    ;     gpu2d_clk_sel = AXI A = (0 << 16)            = 0x00000000
789    ;                                                  ------------
790    ;                                                    0x00002042
791    ldrlt   r0, =0x0000E3F2    ; this line is for i.MX51 TO1.x
792    ldrge   r0, =0x00002042    ; this line is for i.MX51 TO2.0
793    str     r0, [r4, #CCM_CBCMR_OFFSET]

815    ; CCM Bus Clock Divider Register (CBCDR):
816    ;
817    ;     PERCLK_PODF = /1 = (0 << 0)                  = 0x00000000
818    ;     PERCLK_PRED2 = /1 = (0 << 3)                 = 0x00000000
819    ;     PERCLK_PRED1 = /3 = (2 << 6)                 = 0x00000080
820    ;     IPG_PODF = /2 = (1 << 8)                     = 0x00000100
821    ;     AHB_PODF = /5 = (4 << 10)                    = 0x00001000
822    ;     NFC_PODF = /4 = (3 << 13)                    = 0x00006000
823    ;     AXI_A_PODF = /4 = (3 << 16)                  = 0x00030000
824    ;     AXI_B_PODF = /5 = (4 << 19)                  = 0x00200000
825    ;     EMI_SLOW_PODF = /1 = (0 << 22)               = 0x00000000
```

```
826    ;     PERIPH_CLK_SEL = PERIPH_APM_CLK = (1 << 25)  = 0x02000000
827    ;     EMI_CLK_SEL = AHB_CLK_ROOT = (1 << 26)       = 0x04000000
828    ;     DDR_CLK_PODF = /4 = (3 << 27)                = 0x18000000
829    ;     DDR_HIGH_FREQ_CLK_SEL = PLL1 = (1 << 30)     = 0x40000000
830    ;                                                  ------------
831    ;                                                    0x5E237180
832    ;
833    ldr     r0, =0x5E237180
835    str     r0, [r4, #CCM_CBCDR_OFFSET]
836
837    ; Wait for PLL2 to lock
838 PLL2_LOCK
839    ldr     r0, [r2, #DPLL_DP_CTL]
840    ands    r0, r0, #DPLL_DP_CTL_LRF
841    beq     PLL2_LOCK
842
843   ; Configure main_bus_clk to be sourced from PLL2_SW_CLK
(mux input 0)
844    ldr     r0, [r4, #CCM_CBCDR_OFFSET]
845    bic     r0, r0, #CCM_CBCDR_PERIPH_CLK_SEL
846    str     r0, [r4, #CCM_CBCDR_OFFSET]
```

Therefore, it is understood from the code segments that when DPLL2 is locked and stable, the main_bus_clk is sourced from pll2_sw_clk that has a frequency of 665 MHz (lines from 837 to 846 of the `startup.s` file).

Table 1 shows the source-clock, clock divider, related register settings for each bus clock, and bus clock related module clock after the DPLL2 output clock becomes stable.

Table 1. Bus Clocks and Bus Clock Related Module Clocks

Clock Name	Source-Clock Name	Source-Clock Frequency (MHz)	Register Setting for Source-Clock Selection	Clock Divider	Clock Frequency (MHz)
periph_apm_clk	lp_apm	24	CBCMR: periph_apm_sel = 2 (line 786 of startup.s)	NA	24
main_bus_clk	pll2_sw_clk	665	CBCDR: periph_clk_sel = 0 (lines 844–846 of startup.s)	NA	665
axi_a	main_bus_clk	665	NA	CBCDR: axi_a_podf = 3 (line 823 of startup.s) divider = 3 + 1 = 4	166 (665 ÷ 4 = 166)
axi_b	main_bus_clk	665	NA	CBCDR: axi_b_podf = 4 (line 824 of startup.s) divider = 4 + 1 = 5	133 (665 ÷ 5 = 133)
AHB_CLK_ROOT	main_bus_clk	665	NA	CBCDR: ahb_podf = 4 (line 821 of startup.s) divider = 4 + 1 = 5	133 (665 ÷ 5 = 133)

Table 1. Bus Clocks and Bus Clock Related Module Clocks (continued)

Clock Name	Source-Clock Name	Source-Clock Frequency (MHz)	Register Setting for Source-Clock Selection	Clock Divider	Clock Frequency (MHz)
EMI_SLOW_CLK_ROOT	AHB_CLK_ROOT	133	CBCDR: emi_clk_sel = 1 (line 827 of startup.s)	CBCDR: emi_slow_podf = 0 (line 825 of startup.s) divider = 0 + 1 = 1	133 (133 ÷ 1 = 133)
IPG_CLK_ROOT	AHB_CLK_ROOT	133	NA	CBCDR: ipg_podf = 1 (line 820 of startup.s) divider = 1 + 1 = 2	66.5 (133 ÷ 2 = 66.5)
PERCLK_ROOT	lp_apm	24	CBCMR: perclk_lp_apm_sel = 1 (line 780 of startup.s)	CBCDR: perclk_pred1 = 2 (line 819 of startup.s) divider1 = 2 + 1 = 3 CBCDR: perclk_pred2 = 0 (line 818 of startup.s) divider2 = 0 + 1 = 1 CBCDR: perclk_podf = 2 (line 817 of startup.s) divider3 = 2 + 1 = 3	8 (24 ÷ (3 ×1 ×1) = 8)
ENFC_CLK_ROOT	EMI_SLOW_CLK_ROOT	133	NA	CBCDR: nfc_podf = 3 (line 822 of startup.s) divider = 3 + 1 = 4	33.25 (133 ÷ 4 = 33.25)
GPU2D_CLK_ROOT	axi_a	166	CBCMR: gpu2d_clk_sel = 0 (line 788 of startup.s)	NA	166
ARM_AXI_CLK_ROOT	axi_a	166	CBCMR: arm_axi_clk_sel = 0 (line 784 of startup.s)	NA	166
IPU_HSP_CLK_ROOT	axi_b	133	CBCMR: ipu_hsp_clk_sel = 1 (line 783 of startup.s)	NA	133
GPU_CLK_ROOT	axi_a	166	CBCMR: gpu_clk_sel = 0 (line 782 of startup.s)	NA	166
VPU_AXI_CLK_ROOT	axi_a	166	CBCMR: vpu_axi_clk_sel = 0 (line 787 of startup.s)	NA	166

Table 1. Bus Clocks and Bus Clock Related Module Clocks (continued)

Clock Name	Source-Clock Name	Source-Clock Frequency (MHz)	Register Setting for Source-Clock Selection	Clock Divider	Clock Frequency (MHz)
DEBUG_APB_CLK_ROOT	axi_a	166	CBCMR: debug_apb_clk_sel = 0 (line 781 of startup.s)	NA	166
DDR_CLK_ROOT	pll1_sw_clk	800	CBCDR: ddr_high_freq_clk_sel = 1 (line 829 of startup.s)	CBCDR: ddr_clk_podf = 3 (line 828 of startup.s) divider = 3 + 1 = 4	200 (800 ÷ 4 = 200)

4 Revision History

Table 2 provides a revision history for this application note.

Table 2. Document Revision History

Rev. Number	Date	Substantive Change(s)
0	06/2010	Initial release.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or
+1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064
Japan
0120 191014 or
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
1-800 441-2447 or
+1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: AN4140
Rev. 0
06/2010