ESWIN

EIC7700X SOC Manual

Public Release / Rev 1.1

2024/07/05

Notice

INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND DOES NOT REPRESENT A COMMITMENT ON THE PART OF ESWIN. UNLESS OTHERWISE SPECIFIED IN THE CONTRACT, ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS DOCUMENT ARE PROVIDED “AS IS” WITHOUTH WARRANTIES, GUARANTEES, OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED.

Change History

Version No	Date	Descriptions
V8.0	2024-1-31	Engineering Draft.
V1.0	2024-7-5	Engineering Final, Public Release

Content

Content of Figures

Content of Tables

1 Introduction

1.1 Overview

EIC7700X is a low power and high performance SoC chip for edge AI application base on TSMC 12nm FFC, embedded CPU which support RISC-V instruction set. EIC7700X Integrates independent NPU with computing power up to 19.95TOPS INT8, and provides a complete set of peripheral interface include audio and video.

1.2 Application

1.2.1 AI Box

The data delay and network bandwidth pressure handled by the traditional cloud infrastructure can no longer meet all applications and scenarios. Edge computing can provide solutions for this part of the market, and the front and rear devices can work together to achieve smooth interconnection. Through edge AI Box, not only can many intelligent products efficiently collaborate, but it can also connect with enterprise business and seamlessly integrate with multiple industry businesses.

1.2.2 Industrial Vision

The characteristic of industrial machine vision systems is to provide a degree of production automation, mainly in dangerous working environments that are not suitable for manual work or situations where artificial vision cannot meet the requirements. Machine vision is commonly used to replace artificial vision, and machine vision detection can greatly improve production efficiency and automation, which is the foundation for achieving intelligent manufacturing.

1.2.3 Educational Scenarios

Education Smart Station is an edge intelligent computing device for educational scenarios, with super strong audio and video decoding processing capabilities, high computing power, fast deployment, and easy management. The product is widely used in teaching interactive live streaming, online teaching and research, remote course supervision, intelligent learning analysis and other scenarios, providing efficient and high-quality intelligent processing and analysis capabilities for various types of audio and video data.

1.3 Architecture

1.3.1 Block diagram

EIC7700X SoC chip block diagram as bellow.

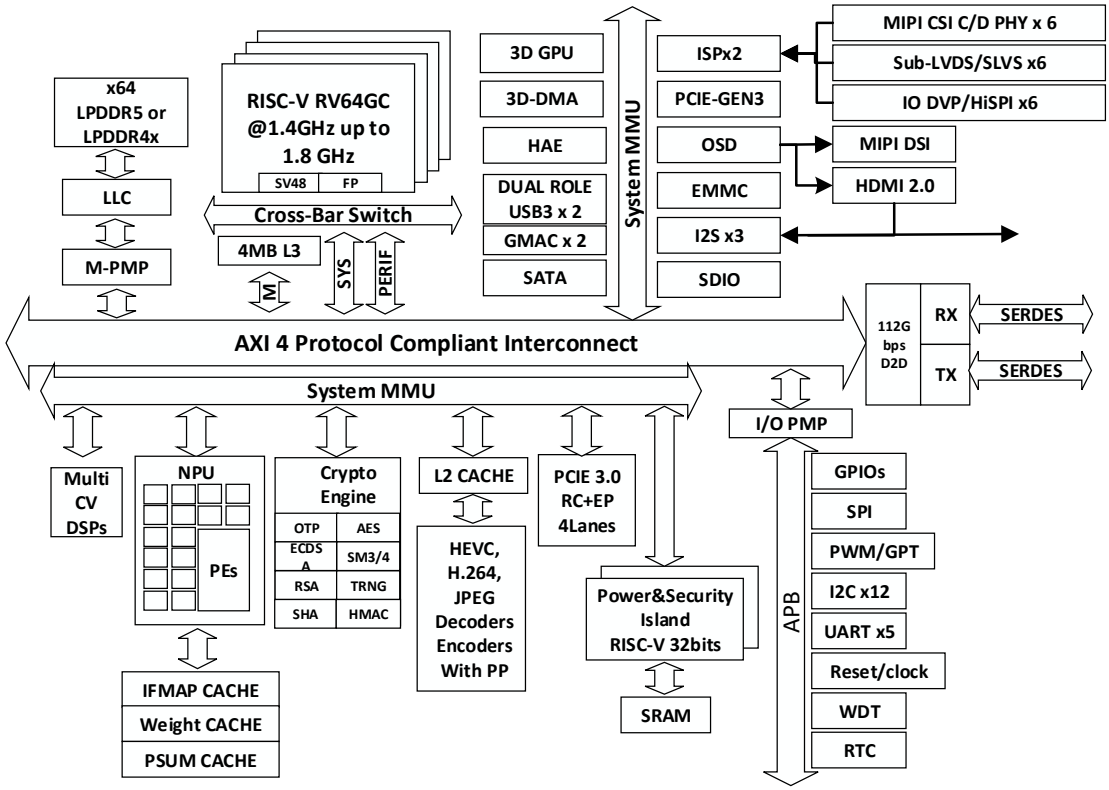


Figure 1-1 EIC700X block diagram

1.3.2 Processor

- RISC-V RV64GC 4core@1.4GHz, up to 1.8GHz
- Private L1 Cache 32K(I)+32KB(D)，private L2Cache 256KB，shared L3 Cache 4MB

1.3.3 Co-processor

- RV321 single core L1Cahe 64KB(I) + 64KB(D)

1.3.4 Intelligent accelerator

- NPU accelerator，up to 19.95 TOPS(INT8)，9.975TOPS(FP16 or INT16)
- Multiple DSPs @1040MHz for computer vision, support 512bit SIMD

1.3.5 Memory interface

- 8GB/16GB 64 bits LPDDR4x @4266MHz
- 8GB/16GB 64 bits LPDDR5 @6400MHz
- eMMC 5.1
- 2x SDIO
- SATA3@6GBps

1.3.6 Video encoder

- H.265(HEVC) 4K@60fps，13x1080@30fps
- H.264(AVC) 4K@60fps，10x1080p@30fps

1.3.7 JPEG encoder

- JPEG ISO/IEC 10918-1，ITU-T T.81
- Baseline process (support Huffman coding Interleaved YUV420, YUV422，Monochrome)
- Lossless process (support 8-bit with Huffman coding Interleaved YUV420，Monochrome)
- MJPEG format (T.81 Annex H) in AVI container

1.3.8 Video decoder

- H.265(H.264） 8K@50fps，4x4K@30fps，32x1080p@30fps
- H.264(AVC) 8K@30fps，4x4K@30fps，16x1080p@30fps

1.3.9 Video image proccess

- HAE(2D GPU)
- 3D GPU (support OpenGL-ES 3.2、EGL 1.4、OpenCL 1.2/2.1 EP2、Vulkan 1.2、Android NN HAL)
- ISP
- Distortion correction

1.3.10 Video input

- Support MIPI DPHY v2.1, CPHY v1.2 and Sub LVDS/SLVS/HiSPi interface
- Support up to 6 channels camera input, with 2lane(trio) per channel
- Support maximum 3.5Gbps/lane MIPI-DPHY interface
- Support maximum 3.5Gbps/trio MIPI-CPHY interface
- Single maximum 1.0Gbps/lane LVDS/Sub-LVDS/HiSPi interface
- Channel connection mode settings:
  - Mode0: Reserved
  - Mode1: Reserved
  - Mode2: Channel 0,1 for 4lanes/3trios sensor
    Channel 2,3 for 4lanes/3trios sensor
    Channel 4,5 for 4lanes/3trios sensor
  - Mode3: Channel 0,1 for 4lanes/3trios senor
    Channel 2,3 for 4lanes/3trios sensor
    Channel 4 for 2lanes/2trios sensor
    Channel 5 for 2lanes/2trios sensor
  - Mode4: Channel 0,1 for 4lanes/3trios sensor
    Channel 2 for 2lanes/2trios sensor
    Channel 3 for 2lanes/2trios sensor
    Channel 4 for 2lanes/2trios sensor
    Channel 5 for 2lanes/2trios sensor
  - Mode5: Channel 0 for 2lanes/2trios sensor
    Channel 1 for 2lanes/2trios sensor
    Channel 2 for 2lanes/2trios sensor
    Channel 3 for 2lanes/2trios sensor
    Channel 4 for 2lanes/2trios sensor
    Channel 5 for 2lanes/2trios sensor

1.3.11 Video output

- OSD(three layer which are video ,mouse,background)
- HDMI2.0 support HDCP1.4/2.2
- MIPI-DSI TX 2.5GHz 4lanes

1.3.12 High-Speed interface

- Dual mode (RC + EP) PCI-Express Gen3.0 4xLanes
- 2 x 10/100/1000 BASE-T GMAC
- DRD (Host + Device) 2 x USB 3.0

1.3.13 Security

- TEE (Trusted Execution Environment)，TRNG， RSA4096，AES，SM4，DES，HMAC(MD5,SHA-1,SHA-224.SHA256,SM-3,SHA512/224,SHA-512/256,SHA-384),CRC32,dual core hardware accelerate
- 16KB OTP

1.3.14 Other module

- 12 x I2C @ 1Mbps
- 5 x UARTs
- 2 x SPI@50MHz
- PWM
- 16xMaibox
- 112xGPIOs
- 4 x WDT
- RTC
- 3xJTAG@20MHz
- 3 x I2S (slave + master)@192kbps
- FAN_CTRL
- PVT

1.4 System Boot

1.4.1 Boot Mode

Two modes of startup are supported:

- Secure boot mode
  Booting from internal ROM is only allowed and verification signatures for bootrom and uboot are required.
- Non-secure mode
  According to boot_sel system can be booted from external storage, it can be configured whether the bootrom and uboot need to verify signatures.

The following boot medium are supported:

- UART
- NOR Flash
- eMMC
- USB

1.4.2 Boot Medium

The selection of boot mode and medium is decided by the security control bit and boot_sel, whose combination is shown in Table 1-1.

Table 1-1 Boot mode and medium selection table

security mode	boot_sel[3]	boot_sel[2]	boot_sel[1:0]	bootrom src	boot CPU	uboot src
1'b1	X	X	2'b00	ROM	SCPU	UART

security mode	boot_sel[3]	boot_sel[2]	boot_sel[1:0]	bootrom src	boot CPU	uboot src
			2'b01			eMMC
			2'b10			Boot-SPI
			2'b11			USB
1'b0	1'b0	1'b0	2'b00	ROM	SCPU	UART
			2'b10			Boot-SPI
			2'b11			USB
		1'b1	2'b00	Boot-SPI		UART
			2'b01			eMMC
			2'b10			Boot-SPI
			2'b11			USB
	1'b1	X	2'b00	Boot-SPI	MCPU	UART
			2'b01			eMMC
			2'b10			Boot-SPI
			2'b11			USB

The security bit is automatically read from the OTP after the chip is powered on, and boot_sel is controlled by the external pins (BOOT_SEL0, BOOT_SEL1, BOOT_SEL2, BOOT_SEL3) of the chip.

In secure mode, bootrom is only allowed to be loaded from internal ROM, and bootrom code can only be executed by SCPU, opensbi and uboot code can be loaded from the corresponding source according to boot_sel[1:0] selection.

In non-secure mode, bootrom can be loaded from internal ROM or external NOR Flash according to boot_sel[2]. boot_sel[3] selects booting from SCPU or MCPU. The code of opensbi and uboot can be loaded from the corresponding source according to the boot_sel[1:0] selection. In non-secure mode, bootrom can only be loaded from external NOR Flash when booting from MCPU (boot_sel[3]= 1'b1).

1.4.3 Boot Flow

The system boot procedure flow is shown in Figure 1-2.

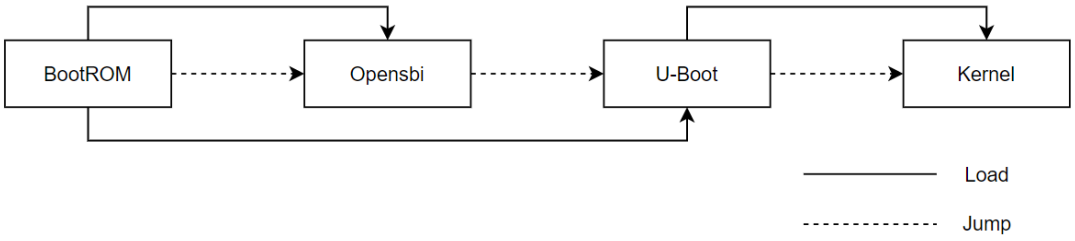


Figure 1-2 System boot procedure flow

BootROM runs on SCPU/MCPU (non-secure mode only), opensbi, uboot and kernel run on MCPU. The workflow of bootrom is mainly divided into three stages: system initialization, image loading and execution,

and security services, as shown in Figure 1-3.

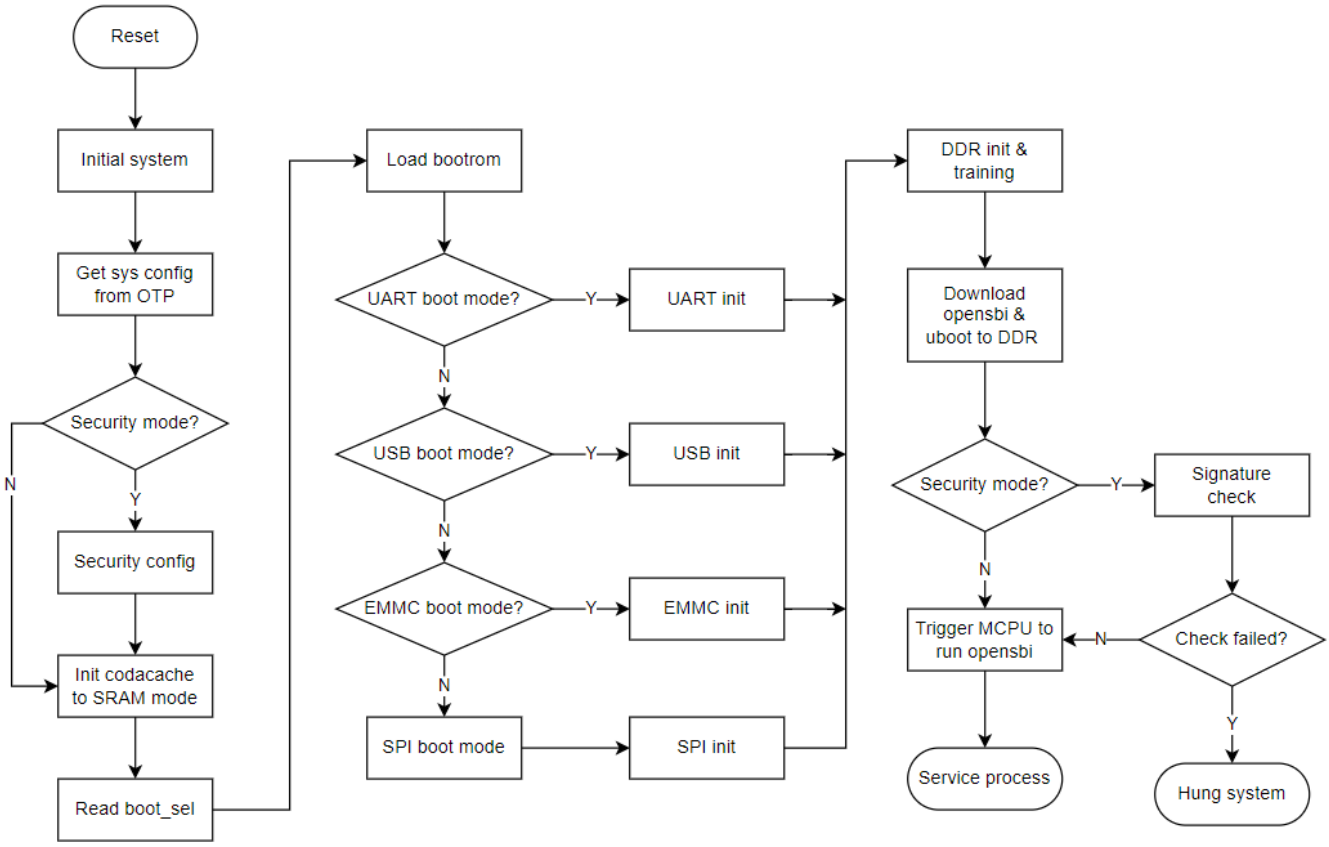


Figure 1-3 Bootrom workflow

- System initialization
  In the system initialization stage, the basic configuration of each module is completed, including:
  - Switch the main frequency of SCPU;
  - Configure the scu and set the relevant registers in sec_csr;
  - Initialize or configure uart, crypto engine, interrupt, malloc in sequence;
  - Read security mode from OTP control register;
  - In security mode, PMP initialization, security configuration and other preparations are needed;
  - In non-security mode, signature verification is not required when loading images, and MEM-PMP protection is not required for each system resource.
  - Initialize codacache to SRAM mode, codacache is 4Mbyte,which can temporarily store the code required for DDR phy initialization.
- Image loading and execution
  During booting, multiple images can be loaded consecutively, and the specific number can be obtained from the bootchain. The system chooses to boot from different media according to the boot_sel, which is divided into the following four situations:
  - UART

    When booting from UART, the image format accepted by the bootrom must be ihex, and the load address of the image must be configured in the boot file.
  - USB

When booting from USB, the device need to be emulated as a disk.

- eMMC

  When booting from eMMC, the eMMC driver needs to be initialized.

- NOR Flash

  When booting from NOR Flash, SPI-Flash needs to be initialized.

First, it receives the DDR init code, then receives the bootloader image package, loads it into the DDR, and finally triggers the MCPU to execute the bootloader. After receiving the complete image, if it is in security mode, bootrom needs to verify the signature of the image, and only after the signature verification is passed, the image can be processed and executed.

- Security service

  After all images are loaded and processed, the bootrom enters the security service processing state.

2 Hardware

2.1 Package and Pinout

2.1.1 Package

EIC7700X uses the flip-chip ball grid array (FCBGA) package. It has 1067 pins, its body size is 23 mm x 23 mm (0.91 in. x 0.91 in.), and its ball pitch is 0.65 mm (0.03 in.).

Figure 2-1 to Figure 2-4 show the package of EIC7700. Figure 2-5 shows the package dimensions.

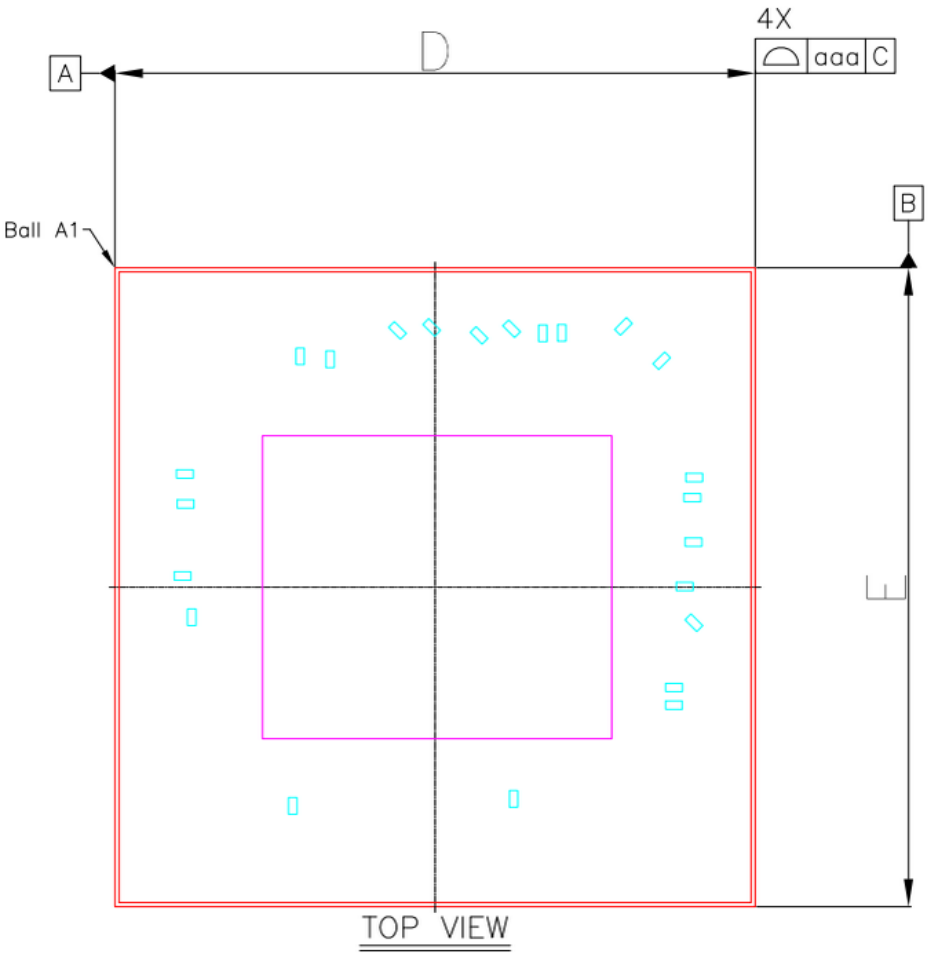


Figure 2-1 Top view

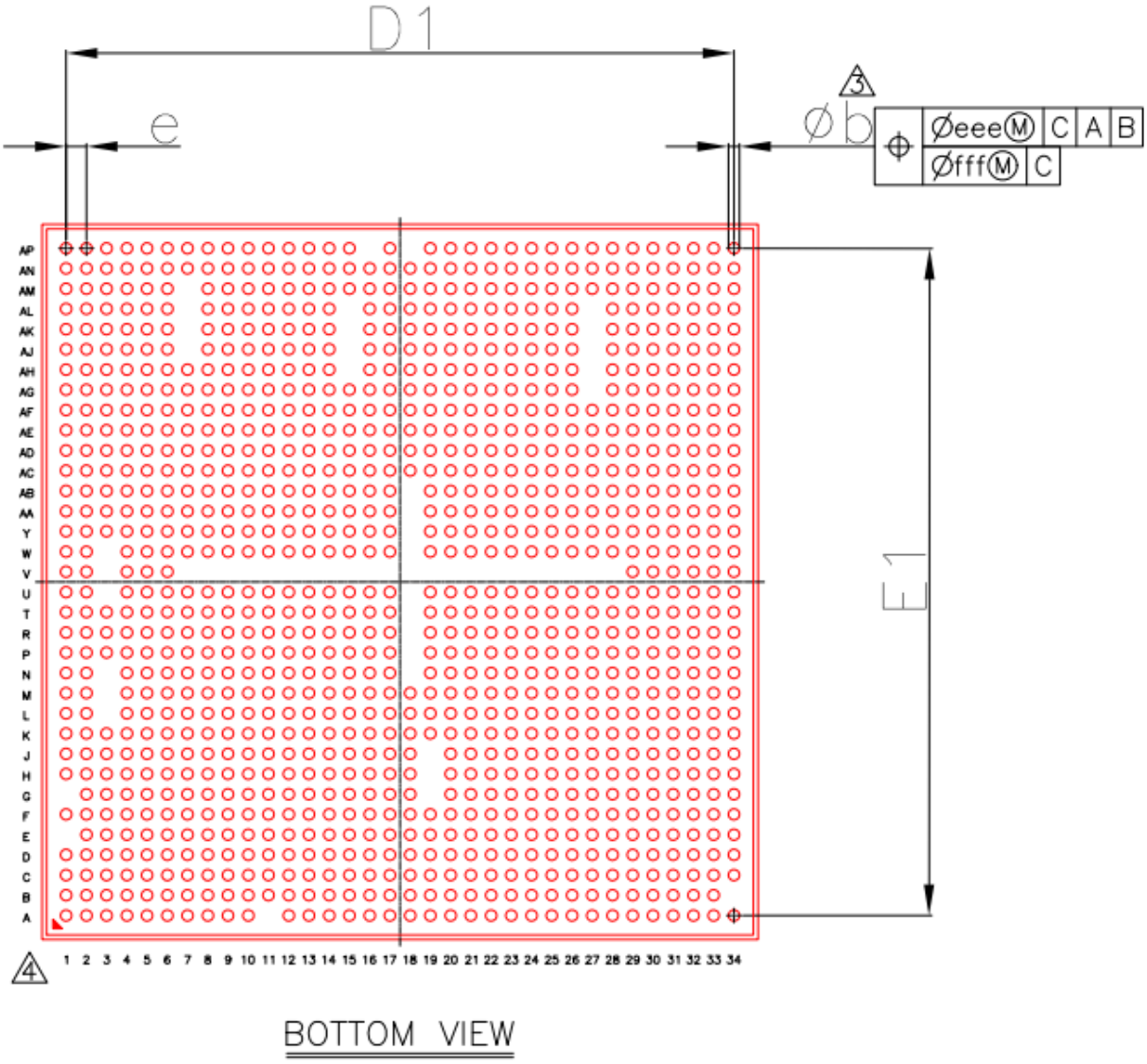


Figure 2-2 Bottom view

Figure 2-3 Side view

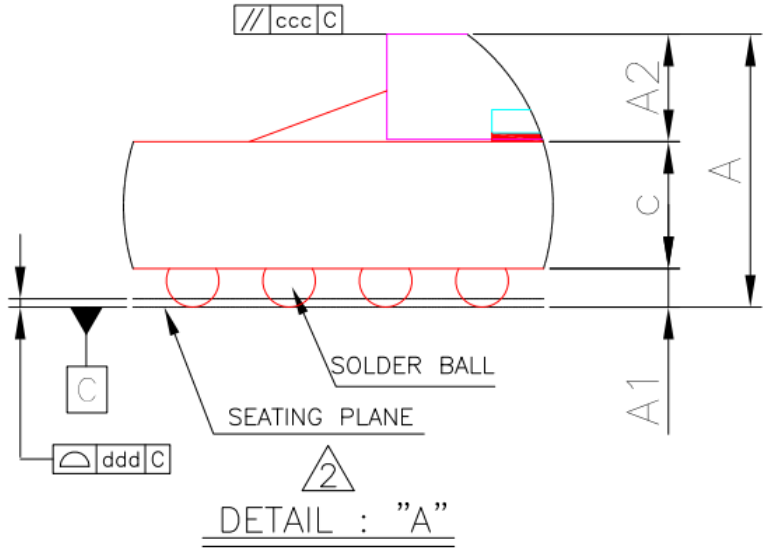


Figure 2-4 Enlarged view of detail "A"

Symbol	Dimension in mm			Dimension in inch		
	MIN	NOM	MAX	MIN	NOM	MAX
A	1.690	1.778	1.866	0.067	0.070	0.073
A1	0.200	0.250	0.300	0.008	0.010	0.012
A2	0.840	0.860	0.880	0.033	0.034	0.035
c	0.598	0.668	0.738	0.024	0.026	0.029
D	22.900	23.000	23.100	0.902	0.906	0.909
E	22.900	23.000	23.100	0.902	0.906	0.909
D1	---	21.450	---	---	0.844	---
E1	---	21.450	---	---	0.844	---
e	---	0.650	---	---	0.026	---
b	0.310	0.360	0.410	0.012	0.014	0.016
aaa	0.200			0.008		
ccc	0.200			0.008		
ddd	0.150			0.006		
eee	0.150			0.006		
fff	0.080			0.003		
MD/ME	34/34					

NOTE :
1. CONTROLLING DIMENSION : MILLIMETER.
2. PRIMARY DATUM C AND SEATING PLANE ARE DEFINED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.
3. DIMENSION b IS MEASURED AT THE MAXIMUM SOLDER BALL DIAMETER, PARALLEL TO PRIMARY DATUM C.
4. THE PATTERN OF PIN 1 FIDUCIAL IS FOR REFERENCE ONLY.
5. BALL PLACEMENT USE 0.35 mm SOLDER BALL.
BGA PAD SOLDER MASK OPENING = 0.30 mm.

Figure 2-5 Package dimensions

2.1.2 Pinout

Table 2-1 lists the pin quantity of the EIC7700X by type.

Table 2-1 Pin quantity

Pin Type	Quantity
I/O	112
Digital power	177
Digital GND	443
Total	1097

2.2 Pin Number List

Table 2-2 Pin Number Order Information

Signal Name	Ball Number	Signal Name	Ball Number
C0_LPDDR_ATO	G22	C1_LPDDR_ATO	L29
C0_LPDDR_DTO	G23	C1_LPDDR_DTO	R29
C0_LPDDR_RESET_N	F24	C1_LPDDR_RESET_N	R28
C0_LPDDR_ZN	F22	C1_LPDDR_ZN	N28
C0_LPDDRA_CA0	A17	C1_LPDDRA_CA0	J34
C0_LPDDRA_CA1	B17	C1_LPDDRA_CA1	J33
C0_LPDDRA_CA2	A15	C1_LPDDRA_CA2	G34
C0_LPDDRA_CA3	F16	C1_LPDDRA_CA3	G29
C0_LPDDRA_CA4	B15	C1_LPDDRA_CA4	G33
C0_LPDDRA_CA5	D16	C1_LPDDRA_CA5	G31
C0_LPDDRA_CLK_C	E17	C1_LPDDRA_CLK_C	H30
C0_LPDDRA_CLK_T	D17	C1_LPDDRA_CLK_T	H31
C0_LPDDRA_DMI0	D20	C1_LPDDRA_DMI0	L31
C0_LPDDRA_DMI1	E14	C1_LPDDRA_DMI1	E32
C0_LPDDRA_DQ0	A18	C1_LPDDRA_DQ0	K34
C0_LPDDRA_DQ1	B18	C1_LPDDRA_DQ1	K33
C0_LPDDRA_DQ2	A19	C1_LPDDRA_DQ2	L34
C0_LPDDRA_DQ3	B19	C1_LPDDRA_DQ3	L33
C0_LPDDRA_DQ4	A20	C1_LPDDRA_DQ4	M34
C0_LPDDRA_DQ5	B20	C1_LPDDRA_DQ5	M33
C0_LPDDRA_DQ6	A21	C1_LPDDRA_DQ6	N34
C0_LPDDRA_DQ7	B21	C1_LPDDRA_DQ7	N33
C0_LPDDRA_DQ8	C11	C1_LPDDRA_DQ8	B32
C0_LPDDRA_DQ9	B12	C1_LPDDRA_DQ9	B33
C0_LPDDRA_DQ10	B11	C1_LPDDRA_DQ10	C34
C0_LPDDRA_DQ11	B13	C1_LPDDRA_DQ11	C33
C0_LPDDRA_DQ12	A13	C1_LPDDRA_DQ12	D34
C0_LPDDRA_DQ13	A12	C1_LPDDRA_DQ13	E34
C0_LPDDRA_DQ14	B14	C1_LPDDRA_DQ14	F34
C0_LPDDRA_DQ15	A14	C1_LPDDRA_DQ15	F33
C0_LPDDRA_DQS0_C	E21	C1_LPDDRA_DQS0_C	M30
C0_LPDDRA_DQS0_T	D21	C1_LPDDRA_DQS0_T	M31

Signal Name	Ball Number	Signal Name	Ball Number
C0_LPDDRA_DQS1_C	E13	C1_LPDDRA_DQS1_C	D32
C0_LPDDRA_DQS1_T	D13	C1_LPDDRA_DQS1_T	D33
C0_LPDDRA_LP4CKE0_LP5CS0	A16	C1_LPDDRA_LP4CKE0_LP5CS0	H34
C0_LPDDRA_LP4CKE1_LP5CS1	B16	C1_LPDDRA_LP4CKE1_LP5CS1	H33
C0_LPDDRA_LP4CS0_LP5CA6	D18	C1_LPDDRA_LP4CS0_LP5CA6	J31
C0_LPDDRA_LP4CS1	F18	C1_LPDDRA_LP4CS1	J29
C0_LPDDRA_WCK0_C	E19	C1_LPDDRA_WCK0_C	K30
C0_LPDDRA_WCK0_T	D19	C1_LPDDRA_WCK0_T	K31
C0_LPDDRA_WCK1_C	E15	C1_LPDDRA_WCK1_C	F31
C0_LPDDRA_WCK1_T	D15	C1_LPDDRA_WCK1_T	F32
C0_LPDDRB_CA0	A26	C1_LPDDRB_CA0	V34
C0_LPDDRB_CA1	B26	C1_LPDDRB_CA1	V33
C0_LPDDRB_CA2	A28	C1_LPDDRB_CA2	Y34
C0_LPDDRB_CA3	F28	C1_LPDDRB_CA3	W29
C0_LPDDRB_CA4	B28	C1_LPDDRB_CA4	Y33
C0_LPDDRB_CA5	D28	C1_LPDDRB_CA5	W31
C0_LPDDRB_CLK_C	E27	C1_LPDDRB_CLK_C	V30
C0_LPDDRB_CLK_T	D27	C1_LPDDRB_CLK_T	V31
C0_LPDDRB_DMI0	D30	C1_LPDDRB_DMI0	AA31
C0_LPDDRB_DMI1	D24	C1_LPDDRB_DMI1	R31
C0_LPDDRB_DQ0	A29	C1_LPDDRB_DQ0	AA34
C0_LPDDRB_DQ1	B29	C1_LPDDRB_DQ1	AA33
C0_LPDDRB_DQ2	A30	C1_LPDDRB_DQ2	AB34
C0_LPDDRB_DQ3	B30	C1_LPDDRB_DQ3	AB33
C0_LPDDRB_DQ4	A31	C1_LPDDRB_DQ4	AC34
C0_LPDDRB_DQ5	A32	C1_LPDDRB_DQ5	AC33
C0_LPDDRB_DQ6	B31	C1_LPDDRB_DQ6	AD34
C0_LPDDRB_DQ7	A33	C1_LPDDRB_DQ7	AD33
C0_LPDDRB_DQ8	A22	C1_LPDDRB_DQ8	P34
C0_LPDDRB_DQ9	B22	C1_LPDDRB_DQ9	P33
C0_LPDDRB_DQ10	A23	C1_LPDDRB_DQ10	R34
C0_LPDDRB_DQ11	B23	C1_LPDDRB_DQ11	R33
C0_LPDDRB_DQ12	A24	C1_LPDDRB_DQ12	T34
C0_LPDDRB_DQ13	B24	C1_LPDDRB_DQ13	T33
C0_LPDDRB_DQ14	A25	C1_LPDDRB_DQ14	U34
C0_LPDDRB_DQ15	B25	C1_LPDDRB_DQ15	U33
C0_LPDDRB_DQS0_C	E29	C1_LPDDRB_DQS0_C	AB30
C0_LPDDRB_DQS0_T	D29	C1_LPDDRB_DQS0_T	AB31
C0_LPDDRB_DQS1_C	E23	C1_LPDDRB_DQS1_C	P30
C0_LPDDRB_DQS1_T	D23	C1_LPDDRB_DQS1_T	P31
C0_LPDDRB_LP4CKE0_LP5CS0	B27	C1_LPDDRB_LP4CKE0_LP5CS0	W33
C0_LPDDRB_LP4CKE1_LP5CS1	A27	C1_LPDDRB_LP4CKE1_LP5CS1	W34
C0_LPDDRB_LP4CS0_LP5CA6	D26	C1_LPDDRB_LP4CS0_LP5CA6	U31
C0_LPDDRB_LP4CS1	F26	C1_LPDDRB_LP4CS1	U29
C0_LPDDRB_WCK0_C	D31	C1_LPDDRB_WCK0_C	Y30
C0_LPDDRB_WCK0_T	C31	C1_LPDDRB_WCK0_T	Y31
C0_LPDDRB_WCK1_C	E25	C1_LPDDRB_WCK1_C	T30
C0_LPDDRB_WCK1_T	D25	C1_LPDDRB_WCK1_T	T31

Signal Name	Ball Number	Signal Name	Ball Number
EMMC_CLK	A8	SDIO0_CLK	B3
EMMC_CMD	B9	SDIO0_CMD	A3
EMMC_D0	A9	SDIO0_D0	A4
EMMC_D1	B10	SDIO0_D1	B4
EMMC_D2	A10	SDIO0_D2	A2
EMMC_D3	B8	SDIO0_D3	B2
EMMC_D4	B7	SDIO1_CLK	E8
EMMC_D5	A7	SDIO1_CMD	D8
EMMC_D6	A6	SDIO1_D0	D9
EMMC_D7	A5	SDIO1_D1	E9
EMMC_DS	B6	SDIO1_D2	D7
EMMC_RSTN_OUT	B5	SDIO1_D3	E7
MIPI_CSI0_ATB	AK28	MIPI_CSI3_ATB	AG28
MIPI_CSI0_CKN	AN27	MIPI_CSI3_CKN	AK33
MIPI_CSI0_CKP	AP27	MIPI_CSI3_CKP	AK34
MIPI_CSI0_D0N	AN26	MIPI_CSI3_D0N	AJ33
MIPI_CSI0_D0P	AP26	MIPI_CSI3_D0P	AJ34
MIPI_CSI0_D1N	AN28	MIPI_CSI3_D1N	AL33
MIPI_CSI0_D1P	AP28	MIPI_CSI3_D1P	AL34
MIPI_CSI0_MCLK	AJ12	MIPI_CSI3_MCLK	AM13
MIPI_CSI0_REXT	AL30	MIPI_CSI3_REXT	AK31
MIPI_CSI0_XHS	AJ11	MIPI_CSI3_XHS	AL13
MIPI_CSI0_XVS	AK10	MIPI_CSI3_XVS	AL12
MIPI_CSI1_ATB	AJ28	MIPI_CSI4_ATB	AH30
MIPI_CSI1_CKN	AP30	MIPI_CSI4_CKN	AG33
MIPI_CSI1_CKP	AN30	MIPI_CSI4_CKP	AG34
MIPI_CSI1_D0N	AN31	MIPI_CSI4_D0N	AH33
MIPI_CSI1_D0P	AP31	MIPI_CSI4_D0P	AH34
MIPI_CSI1_D1N	AN29	MIPI_CSI4_D1N	AF33
MIPI_CSI1_D1P	AP29	MIPI_CSI4_D1P	AF34
MIPI_CSI1_MCLK	AM11	MIPI_CSI4_MCLK	AM14
MIPI_CSI1_REXT	AL31	MIPI_CSI4_REXT	AJ30
MIPI_CSI1_XHS	AL11	MIPI_CSI4_XHS	AN13
MIPI_CSI1_XVS	AK11	MIPI_CSI4_XVS	AP13
MIPI_CSI2_ATB	AH28	MIPI_CSI5_ATB	AH31
MIPI_CSI2_CKN	AN33	MIPI_CSI5_CKN	AF30
MIPI_CSI2_CKP	AP33	MIPI_CSI5_CKP	AF31
MIPI_CSI2_D0N	AN32	MIPI_CSI5_D0N	AE30
MIPI_CSI2_D0P	AP32	MIPI_CSI5_D0P	AE31
MIPI_CSI2_D1N	AM34	MIPI_CSI5_D1N	AG30
MIPI_CSI2_D1P	AM33	MIPI_CSI5_D1P	AG31
MIPI_CSI2_MCLK	AM12	MIPI_CSI5_MCLK	AL14
MIPI_CSI2_REXT	AK30	MIPI_CSI5_REXT	AJ31
MIPI_CSI2_XHS	AP12	MIPI_CSI5_XHS	AK13
MIPI_CSI2_XVS	AN12	MIPI_CSI5_XVS	AK14
USB0_DM	F1	USB1_DM	C1
USB0_DP	F2	USB1_DP	C2
USB0_ID	E5	USB1_ID	C4

Signal Name	Ball Number	Signal Name	Ball Number
USB0_PWREN	H6	USB1_PWREN	H5
USB0_REXT	D6	USB1_REXT	D5
USB0_SSRX_M	G5	USB1_SSRX_M	E2
USB0_SSRX_P	G6	USB1_SSRX_P	E3
USB0_SSTX_M	F4	USB1_SSTX_M	D1
USB0_SSTX_P	F5	USB1_SSTX_P	D2
USB0_VBUS	E6	USB1_VBUS	D4
DVDD_USB0	K10	DVDD_USB1	K11
RGMII0_CLK_125	H4	RGMII1_CLK_125	P1
RGMII0_INTB	J2	RGMII1_INTB	P2
RGMII0_MDC	N2	RGMII1_MDC	T4
RGMII0_MDIO	N1	RGMII1_MDIO	T5
RGMII0_RXCLK	J5	RGMII1_RXCLK	P3
RGMII0_RXD0	J3	RGMII1_RXD0	P5
RGMII0_RXD1	K3	RGMII1_RXD1	R5
RGMII0_RXD2	K4	RGMII1_RXD2	R4
RGMII0_RXD3	K5	RGMII1_RXD3	R3
RGMII0_RXDV	J4	RGMII1_RXDV	P4
RGMII0_TXCLK	K1	RGMII1_TXCLK	R1
RGMII0_TXD0	L1	RGMII1_TXD0	T1
RGMII0_TXD1	L2	RGMII1_TXD1	T2
RGMII0_TXD2	M1	RGMII1_TXD2	U1
RGMII0_TXD3	M2	RGMII1_TXD3	U2
RGMII0_TXEN	K2	RGMII1_TXEN	R2
HDMI_CEC	AJ21	PCIE_CLKN	AN15
HDMI_HPD	AN25	PCIE_CLKP	AP15
HDMI_REXT	AP25	PCIE_CLKREQ_N	AH14
HDMI_SCL	AL21	PCIE_PERST_N	AJ14
HDMI_SDA	AK21	PCIE_REXT	AH19
HDMI_TX0N	AP22	PCIE_RX0N	AK16
HDMI_TX0P	AN22	PCIE_RX0P	AJ16
HDMI_TX1N	AP23	PCIE_RX1N	AL17
HDMI_TX1P	AN23	PCIE_RX1P	AK17
HDMI_TX2N	AP24	PCIE_RX2N	AK18
HDMI_TX2P	AN24	PCIE_RX2P	AJ18
HDMI_TXCKN	AP21	PCIE_RX3N	AL19
HDMI_TXCKP	AN21	PCIE_RX3P	AK19
MIPI_DSI_ATB	AL28	PCIE_TX0N	AN16
MIPI_DSI_CKN	AK24	PCIE_TX0P	AM16
MIPI_DSI_CKP	AL24	PCIE_TX1N	AP17
MIPI_DSI_D0N	AL22	PCIE_TX1P	AN17
MIPI_DSI_D0P	AK22	PCIE_TX2N	AN18
MIPI_DSI_D1N	AK23	PCIE_TX2P	AM18
MIPI_DSI_D1P	AL23	PCIE_TX3N	AP19
MIPI_DSI_D2N	AL25	PCIE_TX3P	AN19
MIPI_DSI_D2P	AK25	PCIE_WAKE_N	AH17
MIPI_DSI_D3N	AK26	SATA_REXT	F7
MIPI_DSI_D3P	AL26	SATA_RXM	H1

Signal Name	Ball Number	Signal Name	Ball Number
MIPI_DSI_REXT	AL29	SATA_RXP	H2
KEY_RESET_N	AP3	SATA_TXM	G3
CHIP_MODE	G11	SATA_TXP	G2
MODE_SET0	F14	BOOT_SEL0	D11
MODE_SET1	G12	BOOT_SEL1	E11
MODE_SET2	G13	BOOT_SEL2	F11
MODE_SET3	G14	BOOT_SEL3	F12
S_MODE	G15	LPDDR_REF_CLK	H27
POR_SEL	AN3	RST_OUT_N	AN2
I2S_MCLK	AL5	JTAG0_TCK	AF1
I2S0_BCLK	AN6	JTAG0_TDI	AF2
I2S0_SDI	AP7	JTAG0_TDO	AG2
I2S0_SDO	AP6	JTAG0_TMS	AH1
I2S0_WCLK	AN7	JTAG1_TCK	AG4
I2S1_BCLK	AP4	JTAG1_TDI	AG5
I2S1_SDI	AP5	JTAG1_TDO	AH5
I2S1_SDO	AN5	JTAG1_TMS	AH4
I2S1_WCLK	AN4	JTAG2_TCK	AJ2
I2S2_BCLK	AK5	JTAG2_TDI	AJ3
I2S2_SDI	AM5	JTAG2_TDO	AJ1
I2S2_SDO	AM6	JTAG2_TMS	AJ4
I2S2_WCLK	AL6	JTAG2_TRST	AH2
SPI0_CLK	V1	I2C0_SCL	AE5
SPI0_CS_N	W1	I2C0_SDA	AF5
SPI0_D0	V2	I2C1_SCL	AE3
SPI0_D1	Y1	I2C1_SDA	AE4
SPI0_D2	W2	I2C10_SCL	AE1
SPI0_D3	Y2	I2C10_SDA	AE2
SPI1_CLK	AA1	I2C11_SCL	AF4
SPI1_CS0_N	AC1	I2C11_SDA	AF3
SPI1_CS1_N	AB2	I2C2_SCL	AK8
SPI1_D0	AA2	I2C2_SDA	AL8
SPI1_D1	AC3	I2C3_SCL	AM8
SPI1_D2	AC2	I2C3_SDA	AL9
SPI1_D3	AB1	I2C4_SCL	AN8
SPI2_CS0_N	AJ5	I2C4_SDA	AP8
SPI2_CS1_N	AJ6	I2C5_SCL	AM9
SPI3_CLK	AC4	I2C5_SDA	AL10
SPI3_CS_N	AC5	GPIO0	AM4
SPI3_DI	AD4	GPIO106	AL2
SPI3_DO	AD5	GPIO11	AH3
UART0_RX	Y4	GPIO111	AL4
UART0_TX	Y3	GPIO27	AL1
UART1_CTS	AA4	GPIO28	AM1
UART1_RTS	AB5	GPIO29	AM2
UART1_RX	Y5	GPIO34	AL3
UART1_TX	AA5	GPIO5	AG1
UART2_RX	AB4	GPIO92	AK4

Signal Name	Ball Number	Signal Name	Ball Number
UART2_TX	AB3	GPIO93	AK1
AVDDHV_PLL	L28	GPIO95	AK2
AVDDHV_PLL	AB17	FAN_PWM	AD2
AVDDHV_PLL	AC17	FAN_TACH	AD1
VDD_LPDDR	L20	VDDQ_LPDDR	J21
VDD_LPDDR	L21	VDDQ_LPDDR	J22
VDD_LPDDR	L22	VDDQ_LPDDR	J24
VDD_LPDDR	L23	VDDQ_LPDDR	K22
VDD_LPDDR	L24	VDDQ_LPDDR	K24
VDD_LPDDR	L25	VDDQ_LPDDR	K25
VDD_LPDDR	M25	VDDQ_LPDDR	K26
VDD_LPDDR	N25	VDDQ_LPDDR	M26
VDD_LPDDR	P25	VDDQ_LPDDR	M27
VDD_LPDDR	R25	VDDQ_LPDDR	P26
VDD_LPDDR	T25	VDDQ_LPDDR	P27
VDD_LPDDR	U25	VDDQ_LPDDR	T26
VDD2H_LPDDR	H22	VDDQ_LPDDR	T27
VDD2H_LPDDR	H23	VDDQ_LPDDR	U26
VDD2H_LPDDR	H25	VDD_NPU	L4
VDD2H_LPDDR	M28	VDD_NPU	L5
VDD2H_LPDDR	P28	VDD_NPU	L7
VDD2H_LPDDR	T28	VDD_NPU	L8
VAA_VDD2H_LPDDR0	G25	VDD_NPU	L11
VAA_VDD2H_LPDDR1	N29	VDD_NPU	L12
VDD_SOC	L18	VDD_NPU	L15
VDD_SOC	M18	VDD_NPU	L16
VDD_SOC	N17	VDD_NPU	M7
VDD_SOC	P17	VDD_NPU	M8
VDD_SOC	P19	VDD_NPU	M11
VDD_SOC	P20	VDD_NPU	M12
VDD_SOC	P22	VDD_NPU	M15
VDD_SOC	P23	VDD_NPU	M16
VDD_SOC	R19	VDD_NPU	N4
VDD_SOC	R20	VDD_NPU	N5
VDD_SOC	R22	VDD_NPU	N9
VDD_SOC	R23	VDD_NPU	N10
VDD_SOC	U17	VDD_NPU	N13
VDD_SOC	U20	VDD_NPU	N14
VDD_SOC	U21	VDD_NPU	P10
VDD_SOC	U22	VDD_NPU	P13
VDD_SOC	W20	VDD_NPU	P14
VDD_SOC	W23	VDD_NPU	R11
VDD_SOC	W24	VDD_NPU	R12
VDD_SOC	W27	VDD_NPU	R15
VDD_SOC	Y17	VDD_NPU	R16
VDD_SOC	Y20	VDD_NPU	T11
VDD_SOC	Y23	VDD_NPU	T12
VDD_SOC	Y24	VDD_NPU	T15

Signal Name	Ball Number	Signal Name	Ball Number
VDD_SOC	Y27	VDD_NPU	T16
VDD_SOC	AA21	VDD_NPU	U4
VDD_SOC	AA22	VDD_NPU	U5
VDD_SOC	AA25	VDD_NPU	U9
VDD_SOC	AA26	VDD_NPU	U10
VDD_SOC	AB19	VDD_NPU	U13
VDD_SOC	AB21	VDD_NPU	U14
VDD_SOC	AB22	VDD_NPU	W4
VDD_SOC	AB25	VDD_NPU	W5
VDD_SOC	AB26	VDD_NPU	W7
VDD_SOC	AC19	VDD_NPU	W9
VDD_SOC	AC21	VDD_NPU	W10
VDD_SOC	AC22	VDD_NPU	W13
VDD18_POR	AB16	VDD_NPU	W14
VDD18_PVT_DDR	J28	VDD_NPU	Y9
VDD18_PVT_SOC	W16	VDD_NPU	Y10
VDDIO_EMMCSD	J13	VDD_NPU	Y13
VDDIO_EMMCSD	K13	VDD_NPU	Y14
VDDIO33_EMMCSD	J15	VDD_CPU	AA11
VDDIO33_EMMCSD	K15	VDD_CPU	AA12
VDDIO_OTP	AF15	VDD_CPU	AB11
VDDIO18	J17	VDD_CPU	AB12
VDDIO18	K17	VDD_CPU	AB14
VDDIO18	AA7	VDD_CPU	AC13
VDDIO18	AA8	VDD_CPU	AC14
VDDIO18	AB7	VDD_CPU	AD13
VDDIO18	AB8	VDD_CPU	AD14
VDDIO18	AC9	VDD_CPU	AE11
VDDIO18	AC10	VDD_CPU	AE12
VDDIO18	AD9	VDD_CPU	AE14
VDDIO18	AD10	VDD_CPU	AF11
VDDIO18	AE7	VDD_CPU	AF12
VDDIO18	AE8	VDD_CPU	AG11
VDDIO18	AF7	VDD_CPU	AG12
VDDIO18	AF8	VDD_CPU	AH11
VDDIO18	AG7	VDD_CPU	AH12
VDDIO18	AG8	VP_CSI01	AE23
VDDIO18	AH7	VP_CSI01	AF23
VDDIO18	AH8	VP_CSI23	AE25
VDDIO18_RGMII	P8	VP_CSI23	AF25
VDDIO18_RGMII	R8	VP_CSI45	AE27
VDDIO18_RGMII	T8	VP_CSI45	AF27
VDDIO33_RGMII0	P6	VPH_CSI01	AD24
VDDIO33_RGMII0	P7	VPH_CSI23	AD25
VDDIO33_RGMII1	T6	VPH_CSI45	AD26
VDDIO33_RGMII1	T7	VP_DSI	AF21
VDDIO33_USB	H10	VPH_DSI	AE21
VDDIO33_USB	H11	VP_HDMI	AF19

Signal Name	Ball Number	Signal Name	Ball Number
VP_USB0	J9	VPH_HDMI	AE19
VP_USB1	J11	VP_SATA	J7
VPTX_USB0	J10	VPH_SATA	G8
VPTX_USB1	J12	VPTX_SATA	J8
VP_PCIE	AE16	VPH_PCIE	AD16
VP_PCIE	AE17	VPH_PCIE	AD17
XIN_24M	AP10	VSSA_PVT_DDR	H28
XOUT_24M	AN10	VSSA_PVT_SOC	W15
VSS	A1	VSS	V6
VSS	A34	VSS	V29
VSS	B1	VSS	V32
VSS	C3	VSS	W6
VSS	C5	VSS	W8
VSS	C6	VSS	W11
VSS	C7	VSS	W12
VSS	C8	VSS	W17
VSS	C9	VSS	W19
VSS	C10	VSS	W21
VSS	C12	VSS	W22
VSS	C13	VSS	W25
VSS	C14	VSS	W26
VSS	C15	VSS	W28
VSS	C16	VSS	W30
VSS	C17	VSS	W32
VSS	C18	VSS	Y6
VSS	C19	VSS	Y7
VSS	C20	VSS	Y8
VSS	C21	VSS	Y11
VSS	C22	VSS	Y12
VSS	C23	VSS	Y15
VSS	C24	VSS	Y16
VSS	C25	VSS	Y19
VSS	C26	VSS	Y21
VSS	C27	VSS	Y22
VSS	C28	VSS	Y25
VSS	C29	VSS	Y26
VSS	C30	VSS	Y28
VSS	C32	VSS	Y29
VSS	D3	VSS	Y32
VSS	D10	VSS	AA3
VSS	D12	VSS	AA6
VSS	D14	VSS	AA9
VSS	D22	VSS	AA10
VSS	E4	VSS	AA13
VSS	E10	VSS	AA14
VSS	E12	VSS	AA15
VSS	E16	VSS	AA16
VSS	E18	VSS	AA17

Signal Name	Ball Number	Signal Name	Ball Number
VSS	E20	VSS	AA19
VSS	E22	VSS	AA20
VSS	E24	VSS	AA23
VSS	E26	VSS	AA24
VSS	E28	VSS	AA27
VSS	E30	VSS	AA28
VSS	E31	VSS	AA29
VSS	E33	VSS	AA30
VSS	F3	VSS	AA32
VSS	F6	VSS	AB6
VSS	F8	VSS	AB9
VSS	F9	VSS	AB10
VSS	F10	VSS	AB13
VSS	F13	VSS	AB15
VSS	F15	VSS	AB20
VSS	F17	VSS	AB23
VSS	F19	VSS	AB24
VSS	F20	VSS	AB27
VSS	F21	VSS	AB28
VSS	F23	VSS	AB29
VSS	F25	VSS	AB32
VSS	F27	VSS	AC6
VSS	F29	VSS	AC7
VSS	F30	VSS	AC8
VSS	G4	VSS	AC11
VSS	G7	VSS	AC12
VSS	G9	VSS	AC15
VSS	G10	VSS	AC16
VSS	G16	VSS	AC18
VSS	G17	VSS	AC20
VSS	G18	VSS	AC23
VSS	G20	VSS	AC24
VSS	G21	VSS	AC25
VSS	G24	VSS	AC26
VSS	G26	VSS	AC27
VSS	G27	VSS	AC28
VSS	G28	VSS	AC29
VSS	G30	VSS	AC30
VSS	G32	VSS	AC31
VSS	H3	VSS	AC32
VSS	H7	VSS	AD3
VSS	H8	VSS	AD6
VSS	H9	VSS	AD7
VSS	H12	VSS	AD8
VSS	H13	VSS	AD11
VSS	H14	VSS	AD12
VSS	H15	VSS	AD15
VSS	H16	VSS	AD18

Signal Name	Ball Number	Signal Name	Ball Number
VSS	H17	VSS	AD19
VSS	H18	VSS	AD20
VSS	H20	VSS	AD21
VSS	H21	VSS	AD22
VSS	H24	VSS	AD23
VSS	H26	VSS	AD27
VSS	H29	VSS	AD28
VSS	H32	VSS	AD29
VSS	J1	VSS	AD30
VSS	J6	VSS	AD31
VSS	J14	VSS	AD32
VSS	J16	VSS	AE6
VSS	J18	VSS	AE9
VSS	J20	VSS	AE10
VSS	J23	VSS	AE13
VSS	J25	VSS	AE15
VSS	J26	VSS	AE18
VSS	J27	VSS	AE20
VSS	J30	VSS	AE22
VSS	J32	VSS	AE24
VSS	K6	VSS	AE26
VSS	K7	VSS	AE28
VSS	K8	VSS	AE29
VSS	K9	VSS	AE32
VSS	K12	VSS	AE33
VSS	K14	VSS	AE34
VSS	K16	VSS	AF6
VSS	K18	VSS	AF9
VSS	K19	VSS	AF10
VSS	K20	VSS	AF13
VSS	K21	VSS	AF14
VSS	K23	VSS	AF16
VSS	K27	VSS	AF17
VSS	K28	VSS	AF18
VSS	K29	VSS	AF20
VSS	K32	VSS	AF22
VSS	L6	VSS	AF24
VSS	L9	VSS	AF26
VSS	L10	VSS	AF28
VSS	L13	VSS	AF29
VSS	L14	VSS	AF32
VSS	L17	VSS	AG3
VSS	L19	VSS	AG6
VSS	L26	VSS	AG9
VSS	L27	VSS	AG10
VSS	L30	VSS	AG13
VSS	L32	VSS	AG14
VSS	M4	VSS	AG15

Signal Name	Ball Number	Signal Name	Ball Number
VSS	M5	VSS	AG16
VSS	M6	VSS	AG17
VSS	M9	VSS	AG18
VSS	M10	VSS	AG19
VSS	M13	VSS	AG20
VSS	M14	VSS	AG21
VSS	M17	VSS	AG22
VSS	M19	VSS	AG23
VSS	M20	VSS	AG24
VSS	M21	VSS	AG25
VSS	M22	VSS	AG26
VSS	M23	VSS	AG29
VSS	M24	VSS	AG32
VSS	M29	VSS	AH6
VSS	M32	VSS	AH9
VSS	N6	VSS	AH10
VSS	N7	VSS	AH13
VSS	N8	VSS	AH16
VSS	N11	VSS	AH18
VSS	N12	VSS	AH20
VSS	N15	VSS	AH21
VSS	N16	VSS	AH22
VSS	N19	VSS	AH23
VSS	N20	VSS	AH24
VSS	N21	VSS	AH25
VSS	N22	VSS	AH26
VSS	N23	VSS	AH29
VSS	N24	VSS	AH32
VSS	N26	VSS	AJ8
VSS	N27	VSS	AJ9
VSS	N30	VSS	AJ10
VSS	N31	VSS	AJ13
VSS	N32	VSS	AJ17
VSS	P9	VSS	AJ19
VSS	P11	VSS	AJ20
VSS	P12	VSS	AJ22
VSS	P15	VSS	AJ23
VSS	P16	VSS	AJ24
VSS	P21	VSS	AJ25
VSS	P24	VSS	AJ26
VSS	P29	VSS	AJ29
VSS	P32	VSS	AJ32
VSS	R6	VSS	AK3
VSS	R7	VSS	AK6
VSS	R9	VSS	AK9
VSS	R10	VSS	AK12
VSS	R13	VSS	AK20
VSS	R14	VSS	AK29

Signal Name	Ball Number	Signal Name	Ball Number
VSS	R17	VSS	AK32
VSS	R21	VSS	AL16
VSS	R24	VSS	AL18
VSS	R26	VSS	AL20
VSS	R27	VSS	AL32
VSS	R30	VSS	AM3
VSS	R32	VSS	AM10
VSS	T3	VSS	AM15
VSS	T9	VSS	AM17
VSS	T10	VSS	AM19
VSS	T13	VSS	AM20
VSS	T14	VSS	AM21
VSS	T17	VSS	AM22
VSS	T19	VSS	AM23
VSS	T20	VSS	AM24
VSS	T21	VSS	AM25
VSS	T22	VSS	AM26
VSS	T23	VSS	AM27
VSS	T24	VSS	AM28
VSS	T29	VSS	AM29
VSS	T32	VSS	AM30
VSS	U6	VSS	AM31
VSS	U7	VSS	AM32
VSS	U8	VSS	AN1
VSS	U11	VSS	AN9
VSS	U12	VSS	AN11
VSS	U15	VSS	AN14
VSS	U16	VSS	AN20
VSS	U19	VSS	AN34
VSS	U23	VSS	AP1
VSS	U24	VSS	AP2
VSS	U27	VSS	AP9
VSS	U28	VSS	AP11
VSS	U30	VSS	AP14
VSS	U32	VSS	AP20
VSS	V4	VSS	AP34
VSS	V5		

2.3 Signal Description

The EIC7700X application processor pin assignment is described in the following table

Table 2-3 Signal Description

Signal Name	Description	Type	Power Supply
LPDDR Chanel0			
C0_LPDDR_ATO	C0 Analog test output	O	VDDQ_LPDDR
C0_LPDDR_DTO	C0 Digital test output	I/O	VDD2H_LPDDR

Signal Name	Description	Type	Power Supply
C0_LPDDR_RESET_N	C0 DRAM reset signal output, active low.	O	VDD2H_LPDDR
C0_LPDDR_ZN	C0 Calibration reference, connected to VDDQ through a 240Ω ±1% resistor.	I/O	VDDQ_LPDDR
C0_LPDDRA_CA0	C0 Rank A CA0	O	VDDQ_LPDDR
C0_LPDDRA_CA1	C0 Rank A CA1	O	VDDQ_LPDDR
C0_LPDDRA_CA2	C0 Rank A CA2	O	VDDQ_LPDDR
C0_LPDDRA_CA3	C0 Rank A CA3	O	VDDQ_LPDDR
C0_LPDDRA_CA4	C0 Rank A CA4	O	VDDQ_LPDDR
C0_LPDDRA_CA5	C0 Rank A CA5	O	VDDQ_LPDDR
C0_LPDDRA_CLK_C	C0 Rank A differential clock output negative.	O	VDDQ_LPDDR
C0_LPDDRA_CLK_T	C0 Rank A differential clock output positive.	O	VDDQ_LPDDR
C0_LPDDRA_DMI0	C0 Rank A DM0 and DBI0	O	VDDQ_LPDDR
C0_LPDDRA_DMI1	C0 Rank A DM1 and DBI1	O	VDDQ_LPDDR
C0_LPDDRA_DQ0	C0 Rank A Data0	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ1	C0 Rank A Data1	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ2	C0 Rank A Data2	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ3	C0 Rank A Data3	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ4	C0 Rank A Data4	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ5	C0 Rank A Data5	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ6	C0 Rank A Data6	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ7	C0 Rank A Data7	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ8	C0 Rank A Data8	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ9	C0 Rank A Data9	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ10	C0 Rank A Data10	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ11	C0 Rank A Data11	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ12	C0 Rank A Data12	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ13	C0 Rank A Data13	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ14	C0 Rank A Data14	I/O	VDDQ_LPDDR
C0_LPDDRA_DQ15	C0 Rank A Data15	I/O	VDDQ_LPDDR
C0_LPDDRA_DQS0_C	C0 Rank A Byte0 Data strobe negative	I/O	VDDQ_LPDDR
C0_LPDDRA_DQS0_T	C0 Rank A Byte0 Data strobe positive	I/O	VDDQ_LPDDR
C0_LPDDRA_DQS1_C	C0 Rank A Byte1 Data strobe negative	I/O	VDDQ_LPDDR
C0_LPDDRA_DQS1_T	C0 Rank A Byte1 Data strobe positive	I/O	VDDQ_LPDDR
C0_LPDDRA_LP4CKE0_LP5CS0	LPDDR5: C0 Rank A CS0 LPDDR4/x:C0 Rank A CKE0	O	VDD2H_LPDDR
C0_LPDDRA_LP4CKE1_LP5CS1	LPDDR5: C0 Rank A CS1 LPDDR4/x:C0 Rank A CKE1	O	VDD2H_LPDDR
C0_LPDDRA_LP4CS0_LP5CA6	LPDDR5: C0 Rank A CA6 LPDDR4/x:C0 Rank A CS0	O	VDDQ_LPDDR
C0_LPDDRA_LP4CS1	LPDDR5: Not used LPDDR4/x:C0 Rank A CS1	O	VDDQ_LPDDR
C0_LPDDRA_WCK0_C	C0 Rank A Byte0 Data clock negative for LPDDR5, not used for LPDDR4/x	O	VDDQ_LPDDR
C0_LPDDRA_WCK0_T	C0 Rank A Byte0 Data clock positive for LPDDR5, not used for LPDDR4/x	O	VDDQ_LPDDR
C0_LPDDRA_WCK1_C	C0 Rank A Byte1 Data clock negative for LPDDR5, not used for LPDDR4/x	O	VDDQ_LPDDR
C0_LPDDRA_WCK1_T	C0 Rank A Byte1 Data clock positive for LPDDR5, not used for LPDDR4/x	O	VDDQ_LPDDR
C0_LPDDRB_CA0	C0 Rank B CA0	I/O	VDDQ_LPDDR

Signal Name	Description	Type	Power Supply
C0_LPDDRB_CA1	C0 Rank B CA1	I/O	VDDQ_LPDDR
C0_LPDDRB_CA2	C0 Rank B CA2	I/O	VDDQ_LPDDR
C0_LPDDRB_CA3	C0 Rank B CA3	I/O	VDDQ_LPDDR
C0_LPDDRB_CA4	C0 Rank B CA4	I/O	VDDQ_LPDDR
C0_LPDDRB_CA5	C0 Rank B CA5	I/O	VDDQ_LPDDR
C0_LPDDRB_CLK_C	C0 Rank B differential clock output negative.	I/O	VDDQ_LPDDR
C0_LPDDRB_CLK_T	C0 Rank B differential clock output positive.	I/O	VDDQ_LPDDR
C0_LPDDRB_DMI0	C0 Rank B DM0 and DBI0	I/O	VDDQ_LPDDR
C0_LPDDRB_DMI1	C0 Rank B DM1 and DBI1	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ0	C0 Rank B Data0	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ1	C0 Rank B Data1	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ2	C0 Rank B Data2	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ3	C0 Rank B Data3	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ4	C0 Rank B Data4	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ5	C0 Rank B Data5	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ6	C0 Rank B Data6	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ7	C0 Rank B Data7	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ8	C0 Rank B Data8	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ9	C0 Rank B Data9	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ10	C0 Rank B Data10	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ11	C0 Rank B Data11	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ12	C0 Rank B Data12	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ13	C0 Rank B Data13	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ14	C0 Rank B Data14	I/O	VDDQ_LPDDR
C0_LPDDRB_DQ15	C0 Rank B Data15	I/O	VDDQ_LPDDR
C0_LPDDRB_DQS0_C	C0 Rank B Byte0 Data strobe negative	I/O	VDDQ_LPDDR
C0_LPDDRB_DQS0_T	C0 Rank B Byte0 Data strobe positive	I/O	VDDQ_LPDDR
C0_LPDDRB_DQS1_C	C0 Rank B Byte1 Data strobe negative	I/O	VDDQ_LPDDR
C0_LPDDRB_DQS1_T	C0 Rank B Byte1 Data strobe positive	I/O	VDDQ_LPDDR
C0_LPDDRB_LP4CKE0_ LP5CS0	LPDDR5: C0 Rank B CS0 LPDDR4/x:C0 Rank B CKE0	I/O	VDD2H_LPDDR
C0_LPDDRB_LP4CKE1_ LP5CS1	LPDDR5: C0 Rank B CS1 LPDDR4/x:C0 Rank B CKE1	I/O	VDD2H_LPDDR
C0_LPDDRB_LP4CS0_L P5CA6	LPDDR5: C0 Rank B CA6 LPDDR4/x:C0 Rank B CS0	I/O	VDDQ_LPDDR
C0_LPDDRB_LP4CS1	LPDDR5: Not used LPDDR4/x:C0 Rank B CS1	I/O	VDDQ_LPDDR
C0_LPDDRB_WCK0_C	C0 Rank B Byte0 Data clock negative for LPDDR5, not used for LPDDR4/x	I/O	VDDQ_LPDDR
C0_LPDDRB_WCK0_T	C0 Rank B Byte0 Data clock positive for LPDDR5, not used for LPDDR4/x	I/O	VDDQ_LPDDR
C0_LPDDRB_WCK1_C	C0 Rank B Byte1 Data clock negative for LPDDR5, not used for LPDDR4/x	I/O	VDDQ_LPDDR
C0_LPDDRB_WCK1_T	C0 Rank B Byte1 Data clock positive for LPDDR5, not used for LPDDR4/x	I/O	VDDQ_LPDDR
LPDDR Chanel1			
C1_LPDDR_ATO	C1 Analog test output	O	VDDQ_LPDDR
C1_LPDDR_DTO	C1 Digital test output	I/O	VDD2H_LPDDR
C1_LPDDR_RESET_N	C1 DRAM reset signal output, active low.	O	VDD2H_LPDDR

Signal Name	Description	Type	Power Supply
C1_LPDDR_ZN	C1 Calibration reference, connected to VDDQ through a 240Ω ±1% resistor.	I/O	VDDQ_LPDDR
C1_LPDDRA_CA0	C1 Rank A CA0	O	VDDQ_LPDDR
C1_LPDDRA_CA1	C1 Rank A CA1	O	VDDQ_LPDDR
C1_LPDDRA_CA2	C1 Rank A CA2	O	VDDQ_LPDDR
C1_LPDDRA_CA3	C1 Rank A CA3	O	VDDQ_LPDDR
C1_LPDDRA_CA4	C1 Rank A CA4	O	VDDQ_LPDDR
C1_LPDDRA_CA5	C1 Rank A CA5	O	VDDQ_LPDDR
C1_LPDDRA_CLK_C	C1 Rank A differential clock output negative.	O	VDDQ_LPDDR
C1_LPDDRA_CLK_T	C1 Rank A differential clock output positive.	O	VDDQ_LPDDR
C1_LPDDRA_DMI0	C1 Rank A DM0 and DBI0	O	VDDQ_LPDDR
C1_LPDDRA_DMI1	C1 Rank A DM1 and DBI1	O	VDDQ_LPDDR
C1_LPDDRA_DQ0	C1 Rank A Data0	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ1	C1 Rank A Data1	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ2	C1 Rank A Data2	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ3	C1 Rank A Data3	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ4	C1 Rank A Data4	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ5	C1 Rank A Data5	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ6	C1 Rank A Data6	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ7	C1 Rank A Data7	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ8	C1 Rank A Data8	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ9	C1 Rank A Data9	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ10	C1 Rank A Data10	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ11	C1 Rank A Data11	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ12	C1 Rank A Data12	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ13	C1 Rank A Data13	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ14	C1 Rank A Data14	I/O	VDDQ_LPDDR
C1_LPDDRA_DQ15	C1 Rank A Data15	I/O	VDDQ_LPDDR
C1_LPDDRA_DQS0_C	C1 Rank A Byte0 Data strobe negative	I/O	VDDQ_LPDDR
C1_LPDDRA_DQS0_T	C1 Rank A Byte0 Data strobe positive	I/O	VDDQ_LPDDR
C1_LPDDRA_DQS1_C	C1 Rank A Byte1 Data strobe negative	I/O	VDDQ_LPDDR
C1_LPDDRA_DQS1_T	C1 Rank A Byte1 Data strobe positive	I/O	VDDQ_LPDDR
C1_LPDDRA_LP4CKE0_ LP5CS0	LPDDR5: C1 Rank A CS0 LPDDR4/x:C1 Rank A CKE0	O	VDD2H_LPDDR
C1_LPDDRA_LP4CKE1_ LP5CS1	LPDDR5: C1 Rank A CS1 LPDDR4/x:C1 Rank A CKE1	O	VDD2H_LPDDR
C1_LPDDRA_LP4CS0_L P5CA6	LPDDR5: C1 Rank A CA6 LPDDR4/x:C1 Rank A CS0	O	VDDQ_LPDDR
C1_LPDDRA_LP4CS1	LPDDR5: Not used LPDDR4/x:C1 Rank A CS1	O	VDDQ_LPDDR
C1_LPDDRA_WCK0_C	C1 Rank A Byte0 Data clock negative for LPDDR5, not used for LPDDR4/x	O	VDDQ_LPDDR
C1_LPDDRA_WCK0_T	C1 Rank A Byte0 Data clock positive for LPDDR5, not used for LPDDR4/x	O	VDDQ_LPDDR
C1_LPDDRA_WCK1_C	C1 Rank A Byte1 Data clock negative for LPDDR5, not used for LPDDR4/x	O	VDDQ_LPDDR
C1_LPDDRA_WCK1_T	C1 Rank A Byte1 Data clock positive for LPDDR5, not used for LPDDR4/x	O	VDDQ_LPDDR
C1_LPDDRB_CA0	C1 Rank B CA0	I/O	VDDQ_LPDDR
C1_LPDDRB_CA1	C1 Rank B CA1	I/O	VDDQ_LPDDR

Signal Name	Description	Type	Power Supply
C1_LPDDRB_CA2	C1 Rank B CA2	I/O	VDDQ_LPDDR
C1_LPDDRB_CA3	C1 Rank B CA3	I/O	VDDQ_LPDDR
C1_LPDDRB_CA4	C1 Rank B CA4	I/O	VDDQ_LPDDR
C1_LPDDRB_CA5	C1 Rank B CA5	I/O	VDDQ_LPDDR
C1_LPDDRB_CLK_C	C1 Rank B differential clock output negative.	I/O	VDDQ_LPDDR
C1_LPDDRB_CLK_T	C1 Rank B differential clock output positive.	I/O	VDDQ_LPDDR
C1_LPDDRB_DMI0	C1 Rank B DM0 and DBI0	I/O	VDDQ_LPDDR
C1_LPDDRB_DMI1	C1 Rank B DM1 and DBI1	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ0	C1 Rank B Data0	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ1	C1 Rank B Data1	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ2	C1 Rank B Data2	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ3	C1 Rank B Data3	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ4	C1 Rank B Data4	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ5	C1 Rank B Data5	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ6	C1 Rank B Data6	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ7	C1 Rank B Data7	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ8	C1 Rank B Data8	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ9	C1 Rank B Data9	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ10	C1 Rank B Data10	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ11	C1 Rank B Data11	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ12	C1 Rank B Data12	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ13	C1 Rank B Data13	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ14	C1 Rank B Data14	I/O	VDDQ_LPDDR
C1_LPDDRB_DQ15	C1 Rank B Data15	I/O	VDDQ_LPDDR
C1_LPDDRB_DQS0_C	C1 Rank B Byte0 Data strobe negative	I/O	VDDQ_LPDDR
C1_LPDDRB_DQS0_T	C1 Rank B Byte0 Data strobe positive	I/O	VDDQ_LPDDR
C1_LPDDRB_DQS1_C	C1 Rank B Byte1 Data strobe negative	I/O	VDDQ_LPDDR
C1_LPDDRB_DQS1_T	C1 Rank B Byte1 Data strobe positive	I/O	VDDQ_LPDDR
C1_LPDDRB_LP4CKE0_LP5CS0	LPDDR5: C1 Rank B CS0 LPDDR4/x:C1 Rank B CKE0	I/O	VDD2H_LPDDR
C1_LPDDRB_LP4CKE1_LP5CS1	LPDDR5: C1 Rank B CS1 LPDDR4/x: C1 Rank B CKE1	I/O	VDD2H_LPDDR
C1_LPDDRB_LP4CS0_LP5CA6	LPDDR5: C1 Rank B CA6 LPDDR4/x: C1 Rank B CS0	I/O	VDDQ_LPDDR
C1_LPDDRB_LP4CS1	LPDDR5: Not used LPDDR4/x: C1 Rank B CS1	I/O	VDDQ_LPDDR
C1_LPDDRB_WCK0_C	C1 Rank B Byte0 Data clock negative for LPDDR5, not used for LPDDR4/x	I/O	VDDQ_LPDDR
C1_LPDDRB_WCK0_T	C1 Rank B Byte0 Data clock positive for LPDDR5, not used for LPDDR4/x	I/O	VDDQ_LPDDR
C1_LPDDRB_WCK1_C	C1 Rank B Byte1 Data clock negative for LPDDR5, not used for LPDDR4/x	I/O	VDDQ_LPDDR
C1_LPDDRB_WCK1_T	C1 Rank B Byte1 Data clock positive for LPDDR5, not used for LPDDR4/x	I/O	VDDQ_LPDDR
EMMC			
EMMC_RSTN_OUT	EMMC Flah reset output	O	VDDIO_EMMCSD/ VDDIO33_EMMCSD
EMMC_DS	EMMC Data strobe: Generated by EMMC flash and used for data O and CRC status response O in HS400 mode.	I	
EMMC_CLK	EMMC Clock O	O	

Signal Name	Description	Type	Power Supply
EMMC_CMD	EMMC Command O	O	
EMMC_D0	EMMC data line0	I/O	
EMMC_D1	EMMC data line1	I/O	
EMMC_D2	EMMC data line2	I/O	
EMMC_D3	EMMC data line3	I/O	
EMMC_D4	EMMC data line4	I/O	
EMMC_D5	EMMC data line5	I/O	
EMMC_D6	EMMC data line6	I/O	
EMMC_D7	EMMC data line7	I/O	
SDIO			
SDIO0_CLK	SDIO0 Clock Output	O	VDDIO_EMMCSD/ VDDIO33_EMMCSD
SDIO0_CMD	SDIO0 Command Output	O	
SDIO0_D0	SDIO0 data line0	I/O	
SDIO0_D1	SDIO0 data line1	I/O	
SDIO0_D2	SDIO0 data line2	I/O	
SDIO0_D3	SDIO0 data line3	I/O	
SDIO0_DETECT	SDIO0 SD card detect Input	I	
SDIO0_WRITE_PROT	SDIO0 SD card write protect Input	I	
SDIO1_CLK	SDIO1 Clock Output	O	VDDIO_EMMCSD/ VDDIO33_EMMCSD
SDIO1_CMD	SDIO1 Command Output	O	
SDIO1_D0	SDIO1 data line0	I/O	
SDIO1_D1	SDIO1 data line1	I/O	
SDIO1_D2	SDIO1 data line2	I/O	
SDIO1_D3	SDIO1 data line3	I/O	
SDIO1_DETECT	SDIO1 SD card detect Input	I	
SDIO1_WRITE_PROT	SDIO1 SD card write protect Input	I	
SATA			
SATA_RXM	SATA Receiver Lane minus terminal	I	VP_SATA
SATA_RXP	SATA Receiver Lane positive terminal	I	VP_SATA
SATA_TXM	SATA Transmitter Lane minus terminal	O	VPTX_SATA
SATA_TXP	SATA Transmitter Lane positive terminal	O	VPTX_SATA
SATA_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_SATA
USB			
USB0_SSRX_M	USB 3.0 SuperSpeed receiver Lane minus terminal	I	VP_USB
USB0_SSRX_P	USB 3.0 SuperSpeed receiver positive terminal	I	VP_USB
USB0_SSTX_M	USB 3.0 SuperSpeed transmitter Lane minus terminal	O	VPTX_USB
USB0_SSTX_P	USB 3.0 SuperSpeed transmitter positive terminal	O	VPTX_USB
USB0_DM	USB2.0 Data minus	I/O	VDDIO33_USB
USB0_DP	USB2.0 Data positive	I/O	VDDIO33_USB
USB0_VBUS	30K 1% resistor connector to VBUS	I	VDDIO33_USB
USB0_ID	Test pin, do not connect anything	O	VDDIO33_USB
USB0_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VDDIO33_USB
USB0_PWREN	USB Power enable	O	VDDIO18
USB1_SSRX_M	USB 3.0 SuperSpeed receiver Lane minus terminal	I	VP_USB
USB1_SSRX_P	USB 3.0 SuperSpeed receiver positive terminal	I	VP_USB

Signal Name	Description	Type	Power Supply
USB1_SSTX_M	USB 3.0 SuperSpeed transmitter Lane minus terminal	O	VPTX_USB
USB1_SSTX_P	USB 3.0 SuperSpeed transmitter positive terminal	O	VPTX_USB
USB1_DM	USB2.0 Data minus	I/O	VDDIO33_USB
USB1_DP	USB2.0 Data positive	I/O	VDDIO33_USB
USB1_VBUS	30K 1% resistor connector to VBUS	I	VDDIO33_USB
USB1_ID	Test pin, do not connect anything	O	VDDIO33_USB
USB1_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VDDIO33_USB
USB1_PWREN	USB Power enable	O	VDDIO18
PCIE			
PCIE_CLKN	PCIE reference clock lane negative terminal	I	VP_PCIE
PCIE_CLKP	PCIE reference clock lane positive terminal	I	VP_PCIE
PCIE_RX0N	PCIE RX lane0 negative terminal	I	VP_PCIE
PCIE_RX0P	PCIE RX lane0 positive terminal	I	VP_PCIE
PCIE_RX1N	PCIE RX lane1 negative terminal	I	VP_PCIE
PCIE_RX1P	PCIE RX lane1 positive terminal	I	VP_PCIE
PCIE_RX2N	PCIE RX lane2 negative terminal	I	VP_PCIE
PCIE_RX2P	PCIE RX lane2 positive terminal	I	VP_PCIE
PCIE_RX3N	PCIE RX lane3 negative terminal	I	VP_PCIE
PCIE_RX3P	PCIE RX lane3 positive terminal	I	VP_PCIE
PCIE_TX0N	PCIE TX lane0 negative terminal	O	VP_PCIE
PCIE_TX0P	PCIE TX lane0 positive terminal	O	VP_PCIE
PCIE_TX1N	PCIE TX lane1 negative terminal	O	VP_PCIE
PCIE_TX1P	PCIE TX lane1 positive terminal	O	VP_PCIE
PCIE_TX2N	PCIE TX lane2 negative terminal	O	VP_PCIE
PCIE_TX2P	PCIE TX lane2 positive terminal	O	VP_PCIE
PCIE_TX3N	PCIE TX lane3 negative terminal	O	VP_PCIE
PCIE_TX3P	PCIE TX lane3 positive terminal	O	VP_PCIE
PCIE_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_PCIE
PCIE_CLKREQ_N	PCIE clock requirement	I/O	VDDIO18
PCIE_PERST_N	PCIE Reset signal	O	VDDIO18
PCIE_WAKE_N	PCIE Link Reactivation	I/O	VDDIO18
HDMI			
HDMI_TX0N	HDMI data lane0 negative terminal	O	VP_HDMI
HDMI_TX0P	HDMI data lane0 positive terminal	O	VP_HDMI
HDMI_TX1N	HDMI data lane1 negative terminal	O	VP_HDMI
HDMI_TX1P	HDMI data lane1 positive terminal	O	VP_HDMI
HDMI_TX2N	HDMI data lane2 negative terminal	O	VP_HDMI
HDMI_TX2P	HDMI data lane2 positive terminal	O	VP_HDMI
HDMI_TXCKN	HDMI reference clock lane negative terminal	O	VP_HDMI
HDMI_TXCKP	HDMI reference clock lane positive terminal	O	VP_HDMI
HDMI_HPD	HDMI Hot Plug Detect	I	VPH_HDMI
HDMI_REXT	Calibration reference, 1.62K 1% resistor connector to ground	Reference	VPH_HDMI
HDMI_CEC	Consumer Electronics Control, internal pull-up	I/O	VDDIO18
HDMI_SCL	HDMI SCL (DDC lock)	O	VDDIO18
HDMI_SDA	HDMI SDA (DDC data)	I/O	VDDIO18
MIPI DSI			
MIPI_DSI_CKN	MIPI DSI clock lane negative terminal	O	VPH_DSI
MIPI_DSI_CKP	MIPI DSI clock lane positive terminal	O	VPH_DSI

Signal Name	Description	Type	Power Supply
MIPI_DSI_D0N	MIPI DSI data lane0 negative terminal	O	VPH_DSI
MIPI_DSI_D0P	MIPI DSI data lane0 positive terminal	O	VPH_DSI
MIPI_DSI_D1N	MIPI DSI data lane1 negative terminal	O	VPH_DSI
MIPI_DSI_D1P	MIPI DSI data lane1 positive terminal	O	VPH_DSI
MIPI_DSI_D2N	MIPI DSI data lane2 negative terminal	O	VPH_DSI
MIPI_DSI_D2P	MIPI DSI data lane2 positive terminal	O	VPH_DSI
MIPI_DSI_D3N	MIPI DSI data lane3 negative terminal	O	VPH_DSI
MIPI_DSI_D3P	MIPI DSI data lane3 positive terminal	O	VPH_DSI
MIPI_DSI_ATB	MIPI DSI test Output	O	VPH_DSI
MIPI_DSI_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_DSI
MIPI CSI D-PHY			
MIPI_CSI0_CKN	MIPI CSI0 DPHY clock lane negative terminal	I	VPH_CSI01
MIPI_CSI0_CKP	MIPI CSI0 DPHY clock lane positive terminal	I	VPH_CSI01
MIPI_CSI0_D0N	MIPI CSI0 DPHY data lane0 negative terminal	I	VPH_CSI01
MIPI_CSI0_D0P	MIPI CSI0 DPHY data lane0 positive terminal	I	VPH_CSI01
MIPI_CSI0_D1N	MIPI CSI0 DPHY data lane1 negative terminal	I	VPH_CSI01
MIPI_CSI0_D1P	MIPI CSI0 DPHY data lane1 positive terminal	I	VPH_CSI01
MIPI_CSI0_ATB	MIPI CSI0 test Output	O	VPH_CSI01
MIPI_CSI0_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI01
MIPI_CSI0_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI0_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI0_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI1_CKN	MIPI CSI1 DPHY clock lane negative terminal	I	VPH_CSI01
MIPI_CSI1_CKP	MIPI CSI1 DPHY clock lane positive terminal	I	VPH_CSI01
MIPI_CSI1_D0N	MIPI CSI1 DPHY data lane0 negative terminal	I	VPH_CSI01
MIPI_CSI1_D0P	MIPI CSI1 DPHY data lane0 positive terminal	I	VPH_CSI01
MIPI_CSI1_D1N	MIPI CSI1 DPHY data lane1 negative terminal	I	VPH_CSI01
MIPI_CSI1_D1P	MIPI CSI1 DPHY data lane1 positive terminal	I	VPH_CSI01
MIPI_CSI1_ATB	MIPI CSI1 test Output	O	VPH_CSI01
MIPI_CSI1_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI01
MIPI_CSI1_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI1_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI1_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI2_CKN	MIPI CSI2 DPHY clock lane negative terminal	I	VPH_CSI23
MIPI_CSI2_CKP	MIPI CSI2 DPHY clock lane positive terminal	I	VPH_CSI23
MIPI_CSI2_D0N	MIPI CSI2 DPHY data lane0 negative terminal	I	VPH_CSI23
MIPI_CSI2_D0P	MIPI CSI2 DPHY data lane0 positive terminal	I	VPH_CSI23
MIPI_CSI2_D1N	MIPI CSI2 DPHY data lane1 negative terminal	I	VPH_CSI23
MIPI_CSI2_D1P	MIPI CSI2 DPHY data lane1 positive terminal	I	VPH_CSI23
MIPI_CSI2_ATB	MIPI CSI2 test Output	O	VPH_CSI23
MIPI_CSI2_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI23
MIPI_CSI2_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI2_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI2_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI3_CKN	MIPI CSI3 DPHY clock lane negative terminal	I	VPH_CSI23
MIPI_CSI3_CKP	MIPI CSI3 DPHY clock lane positive terminal	I	VPH_CSI23
MIPI_CSI3_D0N	MIPI CSI3 DPHY data lane0 negative terminal	I	VPH_CSI23

Signal Name	Description	Type	Power Supply
MIPI_CSI3_D0P	MIPI CSI3 DPHY data lane0 positive terminal	I	VPH_CSI23
MIPI_CSI3_D1N	MIPI CSI3 DPHY data lane1 negative terminal	I	VPH_CSI23
MIPI_CSI3_D1P	MIPI CSI3 DPHY data lane1 positive terminal	I	VPH_CSI23
MIPI_CSI3_ATB	MIPI CSI3 test Output	O	VPH_CSI23
MIPI_CSI3_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI23
MIPI_CSI3_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI3_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI3_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI4_CKN	MIPI CSI4 DPHY clock lane negative terminal	I	VPH_CSI45
MIPI_CSI4_CKP	MIPI CSI4 DPHY clock lane positive terminal	I	VPH_CSI45
MIPI_CSI4_D0N	MIPI CSI4 DPHY data lane0 negative terminal	I	VPH_CSI45
MIPI_CSI4_D0P	MIPI CSI4 DPHY data lane0 positive terminal	I	VPH_CSI45
MIPI_CSI4_D1N	MIPI CSI4 DPHY data lane1 negative terminal	I	VPH_CSI45
MIPI_CSI4_D1P	MIPI CSI4 DPHY data lane1 positive terminal	I	VPH_CSI45
MIPI_CSI4_ATB	MIPI CSI4 test Output	O	VPH_CSI45
MIPI_CSI4_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI45
MIPI_CSI4_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI4_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI4_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI5_CKN	MIPI CSI5 DPHY clock lane negative terminal	I	VPH_CSI45
MIPI_CSI5_CKP	MIPI CSI5 DPHY clock lane positive terminal	I	VPH_CSI45
MIPI_CSI5_D0N	MIPI CSI5 DPHY data lane0 negative terminal	I	VPH_CSI45
MIPI_CSI5_D0P	MIPI CSI5 DPHY data lane0 positive terminal	I	VPH_CSI45
MIPI_CSI5_D1N	MIPI CSI5 DPHY data lane1 negative terminal	I	VPH_CSI45
MIPI_CSI5_D1P	MIPI CSI5 DPHY data lane1 positive terminal	I	VPH_CSI45
MIPI_CSI5_ATB	MIPI CSI5 test Output	O	VPH_CSI45
MIPI_CSI5_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI45
MIPI_CSI5_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI5_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI5_XVS	Vertical sync signal	O	VDDIO18
MIPI CSI C-PHY			
MIPI_CSI0_A0	MIPI CSI0 CPHY trio 0 wire A	I	VPH_CSI01
MIPI_CSI0_B0	MIPI CSI0 CPHY trio 0 wire B	I	VPH_CSI01
MIPI_CSI0_C0	MIPI CSI0 CPHY trio 0 wire C	I	VPH_CSI01
MIPI_CSI0_A1	MIPI CSI0 CPHY trio 1 wire A	I	VPH_CSI01
MIPI_CSI0_B1	MIPI CSI0 CPHY trio 1 wire B	I	VPH_CSI01
MIPI_CSI0_C1	MIPI CSI0 CPHY trio 1 wire C	I	VPH_CSI01
MIPI_CSI0_ATB	MIPI CSI0 test Output	O	VPH_CSI01
MIPI_CSI0_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI01
MIPI_CSI0_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI0_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI0_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI1_A0	MIPI CSI1 CPHY trio 0 wire A	I	VPH_CSI01
MIPI_CSI1_B0	MIPI CSI1 CPHY trio 0 wire B	I	VPH_CSI01
MIPI_CSI1_C0	MIPI CSI1 CPHY trio 0 wire C	I	VPH_CSI01
MIPI_CSI1_A1	MIPI CSI1 CPHY trio 1 wire A	I	VPH_CSI01
MIPI_CSI1_B1	MIPI CSI1 CPHY trio 1 wire B	I	VPH_CSI01

Signal Name	Description	Type	Power Supply
MIPI_CSI1_C1	MIPI CSI1 CPHY trio 1 wire C	I	VPH_CSI01
MIPI_CSI1_ATB	MIPI CSI1 test Output	O	VPH_CSI01
MIPI_CSI1_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI01
MIPI_CSI1_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI1_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI1_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI2_A0	MIPI CSI2 CPHY trio 0 wire A	I	VPH_CSI23
MIPI_CSI2_B0	MIPI CSI2 CPHY trio 0 wire B	I	VPH_CSI23
MIPI_CSI2_C0	MIPI CSI2 CPHY trio 0 wire C	I	VPH_CSI23
MIPI_CSI2_A1	MIPI CSI2 CPHY trio 1 wire A	I	VPH_CSI23
MIPI_CSI2_B1	MIPI CSI2 CPHY trio 1 wire B	I	VPH_CSI23
MIPI_CSI2_C1	MIPI CSI2 CPHY trio 1 wire C	I	VPH_CSI23
MIPI_CSI2_ATB	MIPI CSI2 test Output	O	VPH_CSI23
MIPI_CSI2_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI23
MIPI_CSI2_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI2_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI2_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI3_A0	MIPI CSI3 CPHY trio 0 wire A	I	VPH_CSI23
MIPI_CSI3_B0	MIPI CSI3 CPHY trio 0 wire B	I	VPH_CSI23
MIPI_CSI3_C0	MIPI CSI3 CPHY trio 0 wire C	I	VPH_CSI23
MIPI_CSI3_A1	MIPI CSI3 CPHY trio 1 wire A	I	VPH_CSI23
MIPI_CSI3_B1	MIPI CSI3 CPHY trio 1 wire B	I	VPH_CSI23
MIPI_CSI3_C1	MIPI CSI3 CPHY trio 1 wire C	I	VPH_CSI23
MIPI_CSI3_ATB	MIPI CSI3 test Output	O	VPH_CSI23
MIPI_CSI3_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI23
MIPI_CSI3_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI3_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI3_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI4_A0	MIPI CSI4 CPHY trio 0 wire A	I	VPH_CSI45
MIPI_CSI4_B0	MIPI CSI4 CPHY trio 0 wire B	I	VPH_CSI45
MIPI_CSI4_C0	MIPI CSI4 CPHY trio 0 wire C	I	VPH_CSI45
MIPI_CSI4_A1	MIPI CSI4 CPHY trio 1 wire A	I	VPH_CSI45
MIPI_CSI4_B1	MIPI CSI4 CPHY trio 1 wire B	I	VPH_CSI45
MIPI_CSI4_C1	MIPI CSI4 CPHY trio 1 wire C	I	VPH_CSI45
MIPI_CSI4_ATB	MIPI CSI4 test Output	O	VPH_CSI45
MIPI_CSI4_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI45
MIPI_CSI4_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI4_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI4_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI5_A0	MIPI CSI5 CPHY trio 0 wire A	I	VPH_CSI45
MIPI_CSI5_B0	MIPI CSI5 CPHY trio 0 wire B	I	VPH_CSI45
MIPI_CSI5_C0	MIPI CSI5 CPHY trio 0 wire C	I	VPH_CSI45
MIPI_CSI5_A1	MIPI CSI5 CPHY trio 1 wire A	I	VPH_CSI45
MIPI_CSI5_B1	MIPI CSI5 CPHY trio 1 wire B	I	VPH_CSI45
MIPI_CSI5_C1	MIPI CSI5 CPHY trio 1 wire C	I	VPH_CSI45
MIPI_CSI5_ATB	MIPI CSI5 test Output	O	VPH_CSI45
MIPI_CSI5_REXT	Calibration reference, 200R 1% resistor connector to ground	Reference	VPH_CSI45

Signal Name	Description	Type	Power Supply
MIPI_CSI5_MCLK	Sensor master clock Output	O	VDDIO18
MIPI_CSI5_XHS	Horizontal sync signal	O	VDDIO18
MIPI_CSI5_XVS	Vertical sync signal	O	VDDIO18
MIPI_CSI_XTRIG0	Sensor slave mode, XVS/XHS trigger mode I 0	I	VDDIO18
MIPI_CSI_XTRIG1	Sensor slave mode, XVS/XHS trigger mode I 1	I	VDDIO18
DVP			
DVP0_D[13:0]	DVP0 data	I	VPH_CSI
DVP0_PCLK	DVP0 Pixel CLK	O	VPH_CSI
DVP0_MCLK	DVP0 Sensor master clock Output	O	VPH_CSI
DVP0_XHS	DVP0 Horizontal sync signal	O	VPH_CSI
DVP0_XVS	DVP0 Vertical sync signal	O	VPH_CSI
DVP1_D[13:0]	DVP1 data	I	VPH_CSI
DVP1_PCLK	DVP1 Pixel CLK	O	VPH_CSI
DVP1_MCLK	DVP1 Sensor master clock Output	O	VPH_CSI
DVP1_XHS	DVP1 Horizontal sync signal	O	VPH_CSI
DVP1_XVS	DVP1 Vertical sync signal	O	VPH_CSI
GRMII			
RGMII0_CLK_125	External 125MHz Reference Clock I, this pin should be tie low if not used	I	VDDIO33_RGMII
RGMII0_INTB	Interrupt Input	I	VDDIO33_RGMII
RGMII0_MDC	Management Data Clock	O	VDDIO33_RGMII
RGMII0_MDIO	Management Data	I/O	VDDIO33_RGMII
RGMII0_RXCLK	Receive reference clock.It will be 125MHz, 25MHz, or 2.5MHz	I	VDDIO33_RGMII
RGMII0_RXD[4:0]	Receive Data. Data is transmitted from PHY to MAC via RXD[3:0]	I	VDDIO33_RGMII
RGMII0_RXDV	Receive Control Signal from PHY	I	VDDIO33_RGMII
RGMII0_TXCLK	Transmit reference clock.It will be 125MHz, 25MHz, or 2.5MHz	O	VDDIO33_RGMII
RGMII0_TXD[4:0]	Transmit Data	O	VDDIO33_RGMII
RGMII0_TXEN	Transmit Control Signal to PHY	O	VDDIO33_RGMII
RGMII1_CLK_125	External 125MHz Reference Clock I, this pin should be tie low if not used	I	VDDIO33_RGMII
RGMII1_INTB	Interrupt Input	I	VDDIO33_RGMII
RGMII1_MDC	Management Data Clock	O	VDDIO33_RGMII
RGMII1_MDIO	Management Data	I/O	VDDIO33_RGMII
RGMII1_RXCLK	Receive reference clock.It will be 125MHz, 25MHz, or 2.5MHz	I	VDDIO33_RGMII
RGMII1_RXD[4:0]	Receive Data. Data is transmitted from PHY to MAC via RXD[3:0]	I	VDDIO33_RGMII
RGMII1_RXDV	Receive Control Signal from PHY	I	VDDIO33_RGMII
RGMII1_TXCLK	Transmit reference clock.It will be 125MHz, 25MHz, or 2.5MHz	O	VDDIO33_RGMII
RGMII1_TXD[4:0]	Transmit Data	O	VDDIO33_RGMII
RGMII1_TXEN	Transmit Control Signal to PHY	O	VDDIO33_RGMII
System			
CHIP_MODE	Chip mode selection I,internal pull-down 0: Function mode 1: Test mode	I	VDDIO18
BOOT_SEL0	Boot selection	I	VDDIO18
BOOT_SEL1		I	VDDIO18
BOOT_SEL2		I	VDDIO18

Signal Name	Description	Type	Power Supply
BOOT_SEL3		I	VDDIO18
S_MODE	single die or dual die package selection 0: Dual die 1: Single die	I	VDDIO18
POR_SEL	Internal or external POR selection 0: Function mode 1: DFT mode	I	VDDIO18
POR_TIME_SEL0	Internal POR keep low voltage time delay selction 0	I	VDDIO18
POR_TIME_SEL1	Internal POR keep low voltage time delay selction 1		VDDIO18
DDR_REFCLK_SEL	DDR reference clock selection, internal PLL or external clock source. 0: External buck up reference clock 1: Internal pll clock	I I/O	VDDIO18
KEY_RESET_N	Reset I,active low,internal pull-up	I	VDDIO18
RST_OUT_N	Reset O,active low,internal pull-up	O	VDDIO18
XIN_24M	24MHz clk Input	I	VDDIO18
XOUT_24M	24MHz clk Output	O	VDDIO18
JTAG			
JTAG0_TCK	JATG0 Clock	I	VDDIO18
JTAG0_TDI	JATG0 Data In	I	VDDIO18
JTAG0_TDO	JATG0 Data Out	O	VDDIO18
JTAG0_TMS	JTAG0 Mode Select	I	VDDIO18
JTAG1_TCK	JATG1 Clock	I	VDDIO18
JTAG1_TDI	JATG1 Data In	I	VDDIO18
JTAG1_TDO	JATG1 Data Out	O	VDDIO18
JTAG1_TMS	JTAG1 Mode Select	I	VDDIO18
JTAG2_TCK	JATG2 Clock	I	VDDIO18
JTAG2_TDI	JATG2 Data In	I	VDDIO18
JTAG2_TDO	JATG2 Data Out	O	VDDIO18
JTAG2_TMS	JTAG2 Mode Select	I	VDDIO18
JTAG2_TRST	JTAG2 Reset	I	VDDIO18
SPI			
SPI0_CLK	SPI0 Serial Clock	O	VDDIO18
SPI0_D0	SPI0 data0	I/O	VDDIO18
SPI0_D1	SPI0 data1	I/O	VDDIO18
SPI0_D2	SPI0 data2	I/O	VDDIO18
SPI0_D3	SPI0 data3	I/O	VDDIO18
SPI0_CS_N	SPI0 Chip Select	O	VDDIO18
SPI1_CLK	SPI1 Serial Clock	O	VDDIO18
SPI1_D0	SPI1 data0	I/O	VDDIO18
SPI1_D1	SPI1 data1	I/O	VDDIO18
SPI1_D2	SPI1 data2	I/O	VDDIO18
SPI1_D3	SPI1 data3	I/O	VDDIO18
SPI1_CS0_N	SPI1 Chip Select 0	O	VDDIO18
SPI1_CS1_N	SPI1 Chip Select 1	O	VDDIO18
SPI2_CLK	SPI2 Serial Clock	O	VDDIO18
SPI2_D0	SPI2 data0	I/O	VDDIO18
SPI2_D1	SPI2 data1	I/O	VDDIO18
SPI2_D2	SPI2 data2	I/O	VDDIO18

Signal Name	Description	Type	Power Supply
SPI2_D3	SPI2 data3	I/O	VDDIO18
SPI2_CS0_N	SPI2 Chip Select 0	O	VDDIO18
SPI2_CS1_N	SPI2 Chip Select 1	O	VDDIO18
SPI3_CLK	SPI3 Serial Clock	I	VDDIO18
SPI3_DI	SPI3 data in	I	VDDIO18
SPI3_DO	SPI3 data out	O	VDDIO18
SPI3_CS_N	SPI3 Chip Select	I	VDDIO18
PWM			
FAN_TACH	Fan Speed Detection	I	VDDIO18
FAN_PWM	Fan Speed Control	O	VDDIO18
PWM1	PWM Output0	O	VDDIO18
PWM2	PWM Outpur1	O	VDDIO18
GPIO			
GPIO[31:0]	GPIO, With Interrupt	I/O	VDDIO18
GPIO[111:32]	GPIO, Without Interrupt	I/O	VDDIO18
LED			
SD0_LED_CONTROL	SD0 Data transmission indicator light	I/O	VDDIO18
SD1_LED_CONTROL	SD1 Data transmission indicator light	I/O	VDDIO18
EMMC_LED_CONTROL	EMMC Data transmission indicator light	I/O	VDDIO18
SATA_ACT_LED	SATA Data transmission indicator light	I/O	VDDIO18
I2S			
I2S_MCLK	I2S Master Clock, It is shared for I2S0, I2S1, I2S2	O	VDDIO18
I2S0_BCLK	Bit clock(BCLK), Or Serial SCLK	I/O	VDDIO18
I2S0_SDI	Audio Data Input	I	VDDIO18
I2S0_SDO	Audio Data Output	O	VDDIO18
I2S0_WCLK	WCLK (LRCLK) ， Switch of Data from left and right channels，0:left channel，1:right channel	I/O	VDDIO18
I2S1_BCLK	Bit clock(BCLK), Or Serial SCLK	I/O	VDDIO18
I2S1_SDI	Audio Data Input	I	VDDIO18
I2S1_SDO	Audio Data Output	O	VDDIO18
I2S1_WCLK	WCLK (LRCLK) ， Switch of Data from left and right channels，0:left channel，1:right channel	I/O	VDDIO18
I2S2_BCLK	Bit clock(BCLK), Or Serial SCLK	I/O	VDDIO18
I2S2_SDI	Audio Data Input	I	VDDIO18
I2S2_SDO	Audio Data Output	O	VDDIO18
I2S2_WCLK	WCLK (LRCLK) ， Switch of Data from left and right channels，0:left channel，1:right channel	I/O	VDDIO18
I2C			
I2C0_SDA	I2C0 serial data/address。	B	VDDIO18
I2C0_SCL	I2C0 serial clock。	B	VDDIO18
I2C1_SDA	I2C1 serial data/address。	B	VDDIO18
I2C1_SCL	I2C1 serial clock。	B	VDDIO18
I2C2_SDA	I2C2 serial data/address。	B	VDDIO18
I2C2_SCL	I2C2 serial clock。	B	VDDIO18
I2C3_SDA	I2C3 serial data/address。	B	VDDIO18
I2C3_SCL	I2C3 serial clock。	B	VDDIO18
I2C4_SDA	I2C4 serial data/address。	B	VDDIO18

Signal Name	Description	Type	Power Supply
I2C4_SCL	I2C4 serial clock。	B	VDDIO18
I2C5_SDA	I2C5 serial data/address。	B	VDDIO18
I2C5_SCL	I2C5 serial clock。	B	VDDIO18
I2C6_SDA	I2C6 serial data/address。	B	VDDIO18
I2C6_SCL	I2C6 serial clock。	B	VDDIO18
I2C7_SDA	I2C7 serial data/address。	B	VDDIO18
I2C7_SCL	I2C7 serial clock。	B	VDDIO18
I2C8_SDA	I2C8 serial data/address。	B	VDDIO18
I2C8_SCL	I2C8 serial clock。	B	VDDIO18
I2C9_SDA	I2C9 serial data/address。	B	VDDIO18
I2C9_SCL	I2C9 serial clock。	B	VDDIO18
I2C10_SDA	I2C10 serial data/address。	B	VDDIO18
I2C10_SCL	I2C10 serial clock。	B	VDDIO18
I2C11_SDA	I2C11 serial data/address。	B	VDDIO18
I2C11_SCL	I2C11 serial clock。	B	VDDIO18
UART			
UART0_RXD	UART0 receiver.	I	VDDIO18
UART0_TXD	UART0 transmitter.	O	VDDIO18
UART1_RXD	UART1 receiver.	I	VDDIO18
UART1_TXD	UART1 transmitter.	O	VDDIO18
UART1_RTSN	UART1 Request To Send.	O	VDDIO18
UART1_CTSN	UART1 Clear To Send.	I	VDDIO18
UART2_RXD	UART2 receiver.	I	VDDIO18
UART2_TXD	UART2 transmitter.	O	VDDIO18
UART3_RXD	UART3 receiver.	I	VDDIO18
UART3_TXD	UART3 transmitter.	O	VDDIO18
UART4_RXD	UART4 receiver.	I	VDDIO18
UART4_TXD	UART4 transmitter.	O	VDDIO18

2.4 Pin Multiplexing Table

Multiple usage pins are used to conserve pin consumption for different features. The EIC7700X devices can be used in many different applications but each application will not utilize all the on chip features. As a result, some of the features share the same pin. Most of the multiple usage pins can be used as a GPIO
pin as well.

Table 2-4 GPIOZ_x Multi-Function Pin

Ball Number	Ball Name	FUNC0:Default	FUNC1	FUNC2	FUNC3
D11	BOOT_SEL0	BOOT_SEL0		GPIO107	
E11	BOOT_SEL1	BOOT_SEL1		GPIO108	
F11	BOOT_SEL2	BOOT_SEL2		GPIO109	
F12	BOOT_SEL3	BOOT_SEL3		GPIO110	
G11	CHIP_MODE	CHIP_MODE			
AN3	POR_SEL	POR_SEL			

G15	S_MODE	S_MODE		GPIO94	
AM2	GPIO29	POR_TIME_SEL0	EMMC_LED_CONTROL	GPIO29	
AL3	GPIO34	POR_TIME_SEL1	SD0_LED_CONTROL	GPIO34	
AK2	GPIO95	LPDDR_REFCLK_SEL		GPIO95	
AD2	FAN_PWM	FAN_PWM		GPIO68	
AD1	FAN_TACH	FAN_TACH		GPIO69	
AP3	KEY_RESET_N	KEY_RESET_N			
AN2	RST_OUT_N	RST_OUT_N			
H27	LPDDR_REF_CLK	LPDDR_REF_CLK			
AM4	GPIO0	GPIO0			
AL2	GPIO106	GPIO106			
AH3	GPIO11	GPIO11			
AL4	GPIO111	GPIO111			
AL1	GPIO27	GPIO27	SATA_ACT_LED		
AM1	GPIO28	GPIO28			
AG1	GPIO5	GPIO5	SPI2_D3		
AK4	GPIO92	I2C8_SCL	MIPI_CSI_XTRIG0	GPIO92	UART3_TX
AK1	GPIO93	I2C8_SDA	MIPI_CSI_XTRIG1	GPIO93	UART3_RX
AJ21	HDMI_CEC	HDMI_CEC			
AL21	HDMI_SCL	HDMI_SCL			
AK21	HDMI_SDA	HDMI_SDA			
AE5	I2C0_SCL	I2C0_SCL		GPIO44	
AF5	I2C0_SDA	I2C0_SDA		GPIO45	
AE3	I2C1_SCL	I2C1_SCL		GPIO46	
AE4	I2C1_SDA	I2C1_SDA		GPIO47	
AE1	I2C10_SCL	I2C10_SCL		GPIO102	
AE2	I2C10_SDA	I2C10_SDA		GPIO103	
AF4	I2C11_SCL	I2C11_SCL		GPIO104	
AF3	I2C11_SDA	I2C11_SDA		GPIO105	
AK8	I2C2_SCL	I2C2_SCL		GPIO48	
AL8	I2C2_SDA	I2C2_SDA		GPIO49	
AM8	I2C3_SCL	I2C3_SCL		GPIO50	
AL9	I2C3_SDA	I2C3_SDA		GPIO51	
AN8	I2C4_SCL	I2C4_SCL		GPIO52	
AP8	I2C4_SDA	I2C4_SDA		GPIO53	

AM9	I2C5_SCL	I2C5_SCL		GPIO54	
AL10	I2C5_SDA	I2C5_SDA		GPIO55	
AL5	I2S_MCLK	I2S_MCLK		GPIO22	
AN6	I2S0_BCLK	I2S0_BCLK		GPIO18	
AP7	I2S0_SDI	I2S0_SDI		GPIO20	
AP6	I2S0_SDO	I2S0_SDO		GPIO21	
AN7	I2S0_WCLK	I2S0_WCLK		GPIO19	
AP4	I2S1_BCLK	I2S1_BCLK		GPIO30	
AP5	I2S1_SDI	I2S1_SDI		GPIO32	
AN5	I2S1_SDO	I2S1_SDO		GPIO33	
AN4	I2S1_WCLK	I2S1_WCLK		GPIO31	
AK5	I2S2_BCLK	I2S2_BCLK		GPIO23	
AM5	I2S2_SDI	I2S2_SDI		GPIO25	
AM6	I2S2_SDO	I2S2_SDO		GPIO26	
AL6	I2S2_WCLK	I2S2_WCLK		GPIO24	
AF1	JTAG0_TCK	JTAG0_TCK	SPI2_CLK	GPIO1	
AF2	JTAG0_TDI	JTAG0_TDI	SPI2_D1	GPIO3	
AG2	JTAG0_TDO	JTAG0_TDO	SPI2_D2	GPIO4	
AH1	JTAG0_TMS	JTAG0_TMS	SPI2_D0	GPIO2	
AG4	JTAG1_TCK	JTAG1_TCK		GPIO7	
AG5	JTAG1_TDI	JTAG1_TDI		GPIO9	
AH5	JTAG1_TDO	JTAG1_TDO		GPIO10	
AH4	JTAG1_TMS	JTAG1_TMS		GPIO8	
AJ2	JTAG2_TCK	JTAG2_TCK		GPIO64	
AJ3	JTAG2_TDI	JTAG2_TDI		GPIO66	
AJ1	JTAG2_TDO	JTAG2_TDO		GPIO67	
AJ4	JTAG2_TMS	JTAG2_TMS		GPIO65	
AH2	JTAG2_TRST	JTAG2_TRST		GPIO17	
AJ12	MIPI_CSI0_MCLK	MIPI_CSI0_MCLK		GPIO72	
AJ11	MIPI_CSI0_XHS	MIPI_CSI0_XHS		GPIO71	
AK10	MIPI_CSI0_XVS	MIPI_CSI0_XVS		GPIO70	
AM11	MIPI_CSI1_MCLK	MIPI_CSI1_MCLK		GPIO75	
AL11	MIPI_CSI1_XHS	MIPI_CSI1_XHS		GPIO74	
AK11	MIPI_CSI1_XVS	MIPI_CSI1_XVS		GPIO73	
AM12	MIPI_CSI2_MCLK	MIPI_CSI2_MCLK		GPIO78	

AP12	MIPI_CSI2_XHS	MIPI_CSI2_XHS		GPIO77	
AN12	MIPI_CSI2_XVS	MIPI_CSI2_XVS		GPIO76	
AM13	MIPI_CSI3_MCLK	MIPI_CSI3_MCLK		GPIO81	
AL13	MIPI_CSI3_XHS	MIPI_CSI3_XHS		GPIO80	
AL12	MIPI_CSI3_XVS	MIPI_CSI3_XVS		GPIO79	
AM14	MIPI_CSI4_MCLK	MIPI_CSI4_MCLK		GPIO84	
AN13	MIPI_CSI4_XHS	MIPI_CSI4_XHS		GPIO83	
AP13	MIPI_CSI4_XVS	MIPI_CSI4_XVS		GPIO82	
AL14	MIPI_CSI5_MCLK	MIPI_CSI5_MCLK		GPIO87	
AK13	MIPI_CSI5_XHS	MIPI_CSI5_XHS		GPIO86	
AK14	MIPI_CSI5_XVS	MIPI_CSI5_XVS		GPIO85	
F14	MODE_SET0	SDIO0_DETECT		GPIO13	
G12	MODE_SET1	SDIO0_WRITE_PROT		GPIO14	
G13	MODE_SET2	SDIO1_DETECT		GPIO15	
G14	MODE_SET3	SDIO1_WRITE_PROT		GPIO16	
AH14	PCIE_CLKREQ_N	PCIE_CLKREQ_N			
AJ14	PCIE_PERST_N	PCIE_PERST_N			
AH17	PCIE_WAKE_N	PCIE_WAKE_N			
H4	RGMII0_CLK_125	RGMII0_CLK_125			
J2	RGMII0_INTB	RGMII0_INTB			
N2	RGMII0_MDC	RGMII0_MDC			
N1	RGMII0_MDIO	RGMII0_MDIO			
J5	RGMII0_RXCLK	RGMII0_RXCLK			
J3	RGMII0_RXD0	RGMII0_RXD0			
K3	RGMII0_RXD1	RGMII0_RXD1			
K4	RGMII0_RXD2	RGMII0_RXD2			
K5	RGMII0_RXD3	RGMII0_RXD3			
J4	RGMII0_RXDV	RGMII0_RXDV			
K1	RGMII0_TXCLK	RGMII0_TXCLK			
L1	RGMII0_TXD0	RGMII0_TXD0			
L2	RGMII0_TXD1	RGMII0_TXD1			
M1	RGMII0_TXD2	RGMII0_TXD2			
M2	RGMII0_TXD3	RGMII0_TXD3			
K2	RGMII0_TXEN	RGMII0_TXEN			
P1	RGMII1_CLK_125	RGMII1_CLK_125			

P2	RGMII1_INTB	RGMII1_INTB			
T4	RGMII1_MDC	RGMII1_MDC			
T5	RGMII1_MDIO	RGMII1_MDIO			
P3	RGMII1_RXCLK	RGMII1_RXCLK			
P5	RGMII1_RXD0	RGMII1_RXD0			
R5	RGMII1_RXD1	RGMII1_RXD1			
R4	RGMII1_RXD2	RGMII1_RXD2			
R3	RGMII1_RXD3	RGMII1_RXD3			
P4	RGMII1_RXDV	RGMII1_RXDV			
R1	RGMII1_TXCLK	RGMII1_TXCLK			
T1	RGMII1_TXD0	RGMII1_TXD0			
T2	RGMII1_TXD1	RGMII1_TXD1			
U1	RGMII1_TXD2	RGMII1_TXD2			
U2	RGMII1_TXD3	RGMII1_TXD3			
R2	RGMII1_TXEN	RGMII1_TXEN			
V1	SPI0_CLK	SPI0_CLK		GPIO97	
W1	SPI0_CS_N	SPI0_CS_N		GPIO96	
V2	SPI0_D0	SPI0_D0		GPIO98	
Y1	SPI0_D1	SPI0_D1		GPIO99	
W2	SPI0_D2	SPI0_D2		GPIO100	
Y2	SPI0_D3	SPI0_D3		GPIO101	
AA1	SPI1_CLK	SPI1_CLK		GPIO36	
AC1	SPI1_CS0_N	SPI1_CS0_N		GPIO35	
AB2	SPI1_CS1_N	SPI1_CS1_N	PWM2	GPIO41	
AA2	SPI1_D0	SPI1_D0	I2C9_SCL	GPIO37	UART4_TX
AC3	SPI1_D1	SPI1_D1	I2C9_SDA	GPIO38	UART4_RX
AC2	SPI1_D2	SPI1_D2	SD1_LED_CONTRO L	GPIO39	
AB1	SPI1_D3	SPI1_D3	PWM1	GPIO40	
AJ5	SPI2_CS0_N	SPI2_CS0_N		GPIO6	
AJ6	SPI2_CS1_N	SPI2_CS1_N		GPIO12	
AC4	SPI3_CLK	SPI3_CLK		GPIO89	
AC5	SPI3_CS_N	SPI3_CS_N		GPIO88	
AD4	SPI3_DI	SPI3_DI		GPIO90	
AD5	SPI3_DO	SPI3_DO		GPIO91	
Y4	UART0_RX	UART0_RX		GPIO57	

Y3	UART0_TX	UART0_TX		GPIO56	
AA4	UART1_CTS	UART1_CTS	I2C6_SCL	GPIO60	
AB5	UART1_RTS	UART1_RTS	I2C6_SDA	GPIO61	
Y5	UART1_RX	UART1_RX		GPIO59	
AA5	UART1_TX	UART1_TX		GPIO58	
AB4	UART2_RX	UART2_RX	I2C7_SDA	GPIO63	
AB3	UART2_TX	UART2_TX	I2C7_SCL	GPIO62	
H6	USB0_PWREN	USB0_PWREN		GPIO42	
H5	USB1_PWREN	USB1_PWREN		GPIO43	
AP10	XIN_24M	XIN_24M			
AN10	XOUT_24M	XOUT_24M			

2.5 ESD Ratings

Table 2-5 ESD Ratings

			VALUE	UNIT
$V_{(ESD)}$	Elecrogstatic discharge	Human body model(HBM) ESD stress voltage	±2000	V
		Charged device model(CDM) ESD stress voltage	±500	V

2.6 Soldering Process Recommendations

2.6.1 Requirements on Parameters of the Lead-Free Reflow Soldering Process

Figure 2-6 shows the curve of the lead-free reflow soldering process.

Figure 2-6 Curve of the lead-free reflow soldering process

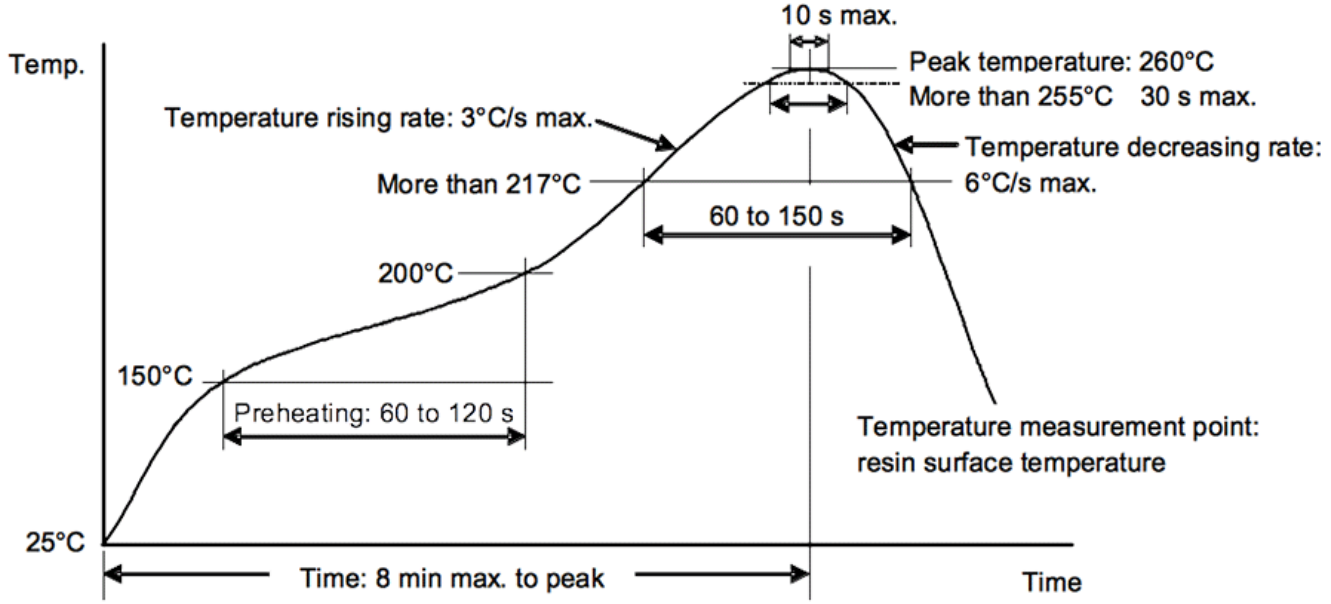


Table 2-6 Describes parameters of the lead-free reflow soldering process.

Table 2-6 Parameters of the lead-free reflow soldering process

Zone	Duration	Heating Rate	Peak Temperature	Cooling Rate
Preheat zone (40 ℃–150 ℃ or 104 ℉–302 ℉)	60s–150s	≤ 2.0ºC/s (≤ 36ºF/s)	-	-
Uniform temperature zone (150 ℃–200 ℃ or 302 ℉–392 ℉)	60s–120s	< 1.0 ℃/s (< 34 ℉/s)	-	-
Reflow zone (greater than 217 ℃ or 423 ℉)	60s–90s	-	230ºC–260ºC (446ºF–500ºF)	-
Cooling zone (Tmax to 180 ℃ or 356 ℉)	-	-	-	1.0ºC/s (34ºF/s) ≤ Slope ≤ 4.0ºC/s (39ºF/s)

NOTE

- Preheat zone: The temperature range is 40 ℃–150 ℃ (104 ℉–302 ℉), the heating rate is about 2 ℃/s (36 ℉/s), and the duration of this temperature zone is 60s–150s.
- Uniform temperature zone: The temperature range is 150 ℃–200 ℃ (302 ℉–392 ℉), the temperature is increased steadily, the heating rate is less than 1 ℃/s (34 ℉/s), and the zone duration is 60s–120s. (Note: Slow heating is required for this zone. Otherwise, improper soldering easily occurs.)
- Reflow zone: The temperature increases from 217 ℃ (423 ℉) to Tmax, and then decreases from Tmax to 217 ℃ (423 ℉). The zone duration is 60s–90s.
- Cooling zone: The temperature decreases from Tmax to 180 ℃ (356 ℉). The maximum cooling rate is 4 ℃/s (39 ℉/s).
- It should take no more than 6 minutes for the temperature to increase from the 25 ℃ (77 ℉) ambient temperature to 250 ℃ (482 ℉).
- The values on the reflow soldering curve shown in Figure 2-6 are recommended values. The values need to be adjusted on the client as required.
- Typically, the duration of the reflow zone is 60s–90s. For the boards with great heat capacity, the duration can be prolonged to 120s. For details about the requirements on package thermal resistance, see the IPC/JEDEC J-STD-020D standard. For details about the method of measuring the package temperature, see the JEP 140 standard.

Table 2-7 describes the temperature resistance standard for the lead-free package according to IPC/JEDEC 020D

Table 2-7 Temperature resistance standard for the lead-free package according to IPC/JEDEC 020D

Package Thickness	Temperature 1 (Package Volume < 350 mm^3 or 0.02 $in.^3$)	Temperature 2 (Package Volume= 350–2000 mm^3 or 0.02–0.12 $in.^3$)	Temperature 3 (Package Volume > 2000 mm^3 or 0.12 $in.^3$)
< 1.6 mm (0.06in.)	260 ℃ (500 ℉)	260 ℃ (500 ℉)	260 ℃ (500 ℉)
1.6 mm to 2.5 mm(0.06 in. to 0.10in.)	260 ℃ (500 ℉)	250 ℃ (482 ℉)	245 ℃ (473 ℉)
> 2.5 mm (0.10in.)	250 ℃ (482 ℉)	245 ℃ (473 ℉)	245 ℃ (473 ℉)

The component soldering terminals (such as the solder ball and pin) and external heat sinks are not considered for volume calculation.

The method of measuring the reflow soldering process curve is as follows:

According to the JEP140 standard, to measure the package temperature, you are advised to place the temperature measuring probe of the thermocouple close to the chip surface if the chip package is thin, or to drill a hole on the package surface and place the temperature measuring probe of the thermocouple into the hole if the chip package is thick. The second method is recommended for all components because of the requirement on quantizing the component thickness. However, this method is not applicable if the chip package is too thin to drill a hole.

NOTE

To measure the temperature of the QFP-packaged chip, place the temperature measuring probe close to pins.

2.6.2 Requirements of Mixing Reflow Soldering

Lead-free components must be properly soldered during mixing reflow soldering. Table 2-4 describes mixing reflow soldering parameters

Table 2-8 Mixing reflow soldering parameters

Zone		Lead BGA	Lead-free BGA	Other Components
Preheat zone (40 ℃–150 ℃ or 104 ℉–302 ℉)	Duration	60s–150s		
	Heating up slope	< 2.5 ℃/s (37 ℉/s)		
Uniform temperature zone (150 ℃–183 ℃ or 302 ℉–361 ℉)	Duration	30–90s		
	Heating up slope	< 1.0 ℃/s (34 ℉/s)		
Reflow zone (greater than 183 ℃ or 361 ℉)	Peak temperature	210 ℃–240 ℃ (410 ℉– 464 ℉)	220 ℃–240 ℃ (428 ℉– 464 ℉)	210 ℃–245 ℃ (410 ℉–473 ℉)
	Duration	30s–120s	60s–120s	30s–120s
Cooling zone (Tmax to 150 ℃ or 302 ℉)	Cooling down slope	1.0 ℃/s (34 ℉/s) ≤ Slope ≤ 4.0 ℃/s (39 ℉/s)		

When the soldering curve is adjusted, the package thermal resistance requirements on the board components must be met. For details about the requirements on package thermal resistance, see the IPC/JEDEC J-STD-020D standard. For

details about the method of measuring the package temperature, see the JEP 140 standard.

Table 2-9 describes the thermal resistance standard for the lead package according to the IPC/JEDEC 020D standard.

Table 2-9 Thermal resistance standard for the lead package

Package Thickness	Temperature 1 (Package Volume < 350 mm^3 or 0.02 $in.^3$)	Temperature 1 (Package Volume ≥ 350 mm^3 or 0.02 $in.^3$)
< 2.5 mm (0.10 in.)	235 ℃ (455 ℉)	220 ℃ (428 ℉)
≥ 2.5 mm (0.10 in.)	220 ℃ (428 ℉)	220 ℃ (428 ℉)

The component soldering terminals (such as the solder ball and pin) and external heat sinks are not considered for volume calculation.

According to the JEP 140 standard, the method of measuring the temperature of the package soldered with the mixing technology is the same as that for measuring the temperature of the package soldered with the lead-free technology. For details, see section 2.3.1 "Requirements on Parameters of the Lead-Free Reflow Soldering Process."

2.7 Moisture-Sensitive Specifications

This chapter defines the usage principles for moisture-sensitive ICs to ensure that ICs are properly used. Related terms are explained as follows:

- Floor life: longest period during which a HiSilicon chip can be stored in the workshop below 30 ℃ (86 ℉) and 60% relative humidity (RH), that is, the time from moisture barrier bag (MBB) unpacking to reflow soldering
- Desiccant: a material for absorbing moisture and keeping the product dry
- Humidity indicator card (HIC): a card that indicates the humidity status
- Moisture sensitivity level (MSL): a level for measuring the moisture degree. The MSL of this product is 3.
- MBB: a vacuum bag for protecting products against moisture
- Solder reflow: reflow soldering
- Shelf life: normal storage period of a product with the MBB

2.7.1 Moisture-Proof Packaging

2.7.1.1 Basic Information

The vacuum packaging materials include:

- An HIC
- An MBB
- Desiccant

Figure 2-7 Vacuum packaging materials

2.7.2 Incoming Inspection

When the vacuum bag is unpacked before SMT in the factories of customers or outsourcing vendors:

- If the largest indicator dot of the HIC is not in blue or khaki, rebake the product by referring to Table 2-7.
- If the 10% RH dot of the HIC is in blue or khaki, the product is dry. In this case, replace the desiccant and pack the product into a vacuum bag.
- If the 10% RH dot of the HIC is not in blue or khaki and the 5% RH dot is in red or light green, the product has become damp. In this case, rebake the product according to Table 2-7.

2.7.3 Storage and Usage

[Storage Environment]

You are advised to use vacuum packaging for the product and store it below 30 ℃ (86 ℉) and 60% RH.

[Shelf Life]

Normal storage period of a product with the MBB

Below 30 ℃ (86 ℉) and 60% RH, the shelf life of the product with vacuum packaging is less than or equal to 12 months.

[Floor Life]

Table 2-10 describes the floor life below 30 ℃ (86 ℉) and 60% RH.

Table 2-10 Floor life

MSL	Floor life(out of bag) at factory ambient ≤ 30 ℃/60% RH or as stated
1	Unlimited at ≤ 30 ℃/85% RH
2	1 year
2a	4 weeks
3	168 hours
4	72 hours
5	48 hours
5a	24 hours
6	Mandatory bake before use, must be reflowed within the time limit specified on the label

[Usage of Moisture-Sensitive Products]

- If the product has been exposed to air for more than 2 hours at 30 ℃ (86 ℉) or lower and at most 60% RH, rebake it and pack it into a vacuum bag.
- If the product has been exposed to air for no more than 2 hours at 30 ℃ (86 ℉) or lower and at most 60% RH, replace the desiccant and pack the chip into a vacuum bag.

For details about other storage and usage rules, see the JEDEC J-STD-033A standard.

2.7.4 Rebaking

[Application Scope]

All moisture-sensitive ICs that need to be rebaked. Following lists the rebaking reference.

Table 2-11 Rebaking reference

Body Thickness	Level	Bake@125 ℃	Bake@90 ℃ ≤ 5% RH	Bake@40 ℃ ≤ 5% RH
≤ 1.4 mm	2a	3 hours	11 hours	5 days
	3	7 hours	23 hours	9 days
	4	7 hours	23 hours	9 days
	5	7 hours	24 hours	10 days

Body Thickness	Level	Bake@125 ℃	Bake@90 ℃ ≤ 5% RH	Bake@40 ℃ ≤ 5% RH
	5a	10 hours	24 hours	10 days
≤ 2.0 mm	2a	16 hours	2 days	22 days
	3	17 hours	2 days	23 days
	4	20 hours	3 days	28 days
	5	25 hours	4 days	35 days
	5a	40 hours	6 days	56 days
≤ 4.5 mm	2a	48 hours	7 days	67 days
	3	48 hours	8 days	67 days
	4	48 hours	10 days	67 days
	5	48 hours	10 days	67 days
	5a	48 hours	10 days	67 days

NOTE

- Table 2-11 lists the minimum rebaking time required for damp products.
- Low-temperature rebaking is recommended.
- For details, see the JEDEC standard.

2.8 Electrical Specifications

2.8.1 Temperature

Table 2-12 Operating environment parameters and requirements on the junction temperatures of EIC7700X

Table 2-12 Operating environment parameters and requirements

Chip	Junction Temperature for Working (℃)		Destructive Junction Temperature (℃)
	Min(℃)	Max(℃)	
EIC7700X	-20	105	125

2.8.2 Operating Conditions

2.8.2.1 Absolute Ratings

The table below gives the absolute maximum ratings. Exposure to stresses beyond those listed in this table may result in permanent device damage, unreliability or both.

Table 2-13 Absolute maximum rated voltage

Parameters	Related Power Group	Min	Max	Unit
VDD_CPU	CPU Core Power	-0.3	0.88	V
VDD_NPU	NPU Core Power	-0.3	1.15	V
VDD_SOC	SOC Core Power	-0.3	0.88	V
VDD_LPDDR	LPDDR Core Power	-0.3	0.88	V
VDDQ_LPDDR	LPDDR IO Power	-0.3	1.17	V
VDD2H_LPDDR	LPDDR VDD2H Power	-0.3	1.17	V
VAA_VDD2H_LPDDR0 VAA_VDD2H_LPDDR1	LPDDR PLL power（1.8V）	-0.3	1.98	V
VP_CSI01, VP_CSI23, VP_CSI45	MIPI CSI0, CSI1 Analog Power（0.8V）	-0.3	0.88	V
VP_DSI	MIPI DSI Analog Power（0.8V）	-0.3	0.88	V
VP_HDMI	HDMI Analog Power（0.8V）	-0.3	0.88	V
VP_PCIE	PCIE Analog Power（0.8V）	-0.3	0.88	V
VPTX_SATA	SATA Analog Power（0.8V）	-0.3	0.88	V
VP_SATA	SATA Analog Power（0.8V）	-0.3	0.88	V
VP_USB0	USB Analog Power（0.8V）	-0.3	0.88	V
VPTX_USB0	USB Analog Power（0.8V）	-0.3	0.88	V
DVDD_USB0	USB Analog Power（0.8V）	-0.3	0.88	V
VP_USB1	USB Analog Power（0.8V）	-0.3	0.88	V
VPTX_USB1	USB Analog Power（0.8V）	-0.3	0.88	V
DVDD_USB1	USB Analog Power（0.8V）	-0.3	0.88	V
VPH_CSI01, VPH_CSI23, VPH_CSI45	MIPI CSI0, CSI1 Analog Power（1.8V）	-0.3	1.98	V
VPH_DSI	MIPI DSI Analog Power（1.8V）	-0.3	1.98	V
VPH_HDMI	HDMI Analog Power（1.8V）	-0.3	1.98	V
VPH_PCIE	PCIE Analog Power（1.8V）	-0.3	1.98	V
VPH_SATA	SATA Analog Power（1.8V）	-0.3	1.98	V
VDDIO_EMMCSD	EMMC and SDIO 1.8V Power	-0.3	1.98	V
VDDIO18_RGMII	RGMII Internal logic 1.8V Power	-0.3	1.98	V
VDDIO18	1.8V IO Power	-0.3	1.98	V
AVDDHV_PLL	1.8V PLL Power	-0.3	1.98	V
VDD18_POR	Internal POR 1.8V Power	-0.3	1.98	V
VDD18_PVT_DDR	1.8V PVT Power For DDR	-0.3	1.98	V
VDD18_PVT_SOC	1.8V PVT Power For SOC	-0.3	1.98	V
VDDIO_OTP	OTP IO Voltage(1.8V)	-0.3	1.98	V
VDDIO33_RGMII	RGMII 1.8V IO Power	-0.3	3.63	V
VDDIO33_USB	USB Analog Power（3.3V）	-0.3	3.63	V
VDDIO33_EMMCSD	EMMC and SDIO 3.3V Power	-0.3	3.63	V

2.8.2.2 Recommended Operating Condition

Following table describes the recommended operating condition.

Table 2-14 Operating conditions for the power supply under normal voltage

Power Name	Description	Voltage			
		Min	Typ	Max	Unit
VDD_CPU	CPU Core Power	0.76	0.8	0.84	V
VDD_NPU	NPU Core Power	0.76	-	1.10	V
VDD_SOC	SOC Core Power	0.76	0.8	0.84	V
VDD_LPDDR	LPDDR Core Power	0.76	0.8	0.84	V
VDDQ_LPDDR	IO power For LPDDR4	0.76	0.8	0.84	V
	IO power For LPDDR4x	0.57	0.6	0.63	V
	IO power For LPDDR5	0.48	0.5	0.53	V
VDD2H_LPDDR	VDD2H For LPDDR4/LPDDR4x	1.05	1.1	1.16	V
	VDD2H For LPDDR5	1.00	1.05	1.10	V
VAA_VDD2H_LPDDR0 VAA_VDD2H_LPDDR1	LPDDR PLL power（1.8V）	1.71	1.8	1.89	V
VP_CSI01, VP_CSI23, VP_CSI45	MIPI CSI0, CSI1 Analog Power（0.8V）	0.76	0.8	0.84	V
VP_DSI	MIPI DSI Analog Power（0.8V）	0.76	0.8	0.84	V
VP_HDMI	HDMI Analog Power（0.8V）	0.76	0.8	0.84	V
VP_PCIE	PCIE Analog Power（0.8V）	0.76	0.8	0.84	V
VPTX_SATA	SATA Analog Power（0.8V）	0.76	0.8	0.84	V
VP_SATA	SATA Analog Power（0.8V）	0.76	0.8	0.84	V
VP_USB0	USB Analog Power（0.8V）	0.76	0.8	0.84	V
VPTX_USB0	USB Analog Power（0.8V）	0.76	0.8	0.84	V
DVDD_USB0	USB Analog Power（0.8V）	0.76	0.8	0.84	V
VP_USB1	USB Analog Power（0.8V）	0.76	0.8	0.84	V
VPTX_USB1	USB Analog Power（0.8V）	0.76	0.8	0.84	V
DVDD_USB1	USB Analog Power（0.8V）	0.76	0.8	0.84	V
VPH_CSI01, VPH_CSI23, VPH_CSI45	MIPI CSI0, CSI1 Analog Power（1.8V）	1.71	1.8	1.89	V
VPH_DSI	MIPI DSI Analog Power（1.8V）	1.71	1.8	1.89	V
VPH_HDMI	HDMI Analog Power（1.8V）	1.71	1.8	1.89	V
VPH_PCIE	PCIE Analog Power（1.8V）	1.71	1.8	1.89	V
VPH_SATA	SATA Analog Power（1.8V）	1.71	1.8	1.89	V
VDDIO_EMMCSD	EMMC and SDIO 1.8V Power	1.71	1.8	1.89	V
VDDIO18_RGMII	RGMII Internal logic 1.8V Power	1.71	1.8	1.89	V
VDDIO18	1.8V IO Power	1.71	1.8	1.89	V
AVDDHV_PLL	1.8V PLL Power	1.71	1.8	1.89	V
VDD18_POR	Internal POR 1.8V Power	1.71	1.8	1.89	V
VDD18_PVT_DDR	1.8V PVT Power For DDR	1.71	1.8	1.89	V
VDD18_PVT_SOC	1.8V PVT Power For SOC	1.71	1.8	1.89	V
VDDIO_OTP	OTP IO Voltage(1.8V)	1.71	1.8	1.89	V
VDDIO33_RGMII	RGMII 1.8V IO Power	1.71	1.8	1.89	V
	RGMII 3.3V IO Power	3.135	3.3	3.465	V

Power Name	Description	Voltage			
VDDIO33_USB	USB Analog Power（3.3V）	3.135	3.3	3.465	V
VDDIO33_EMMCSD	EMMC and SDIO 3.3V Power	3.135	3.3	3.465	V

2.8.3 Power-On and Power-Off Sequences

Figure 2-8 describe power-on sequences, and the the power-off sequences

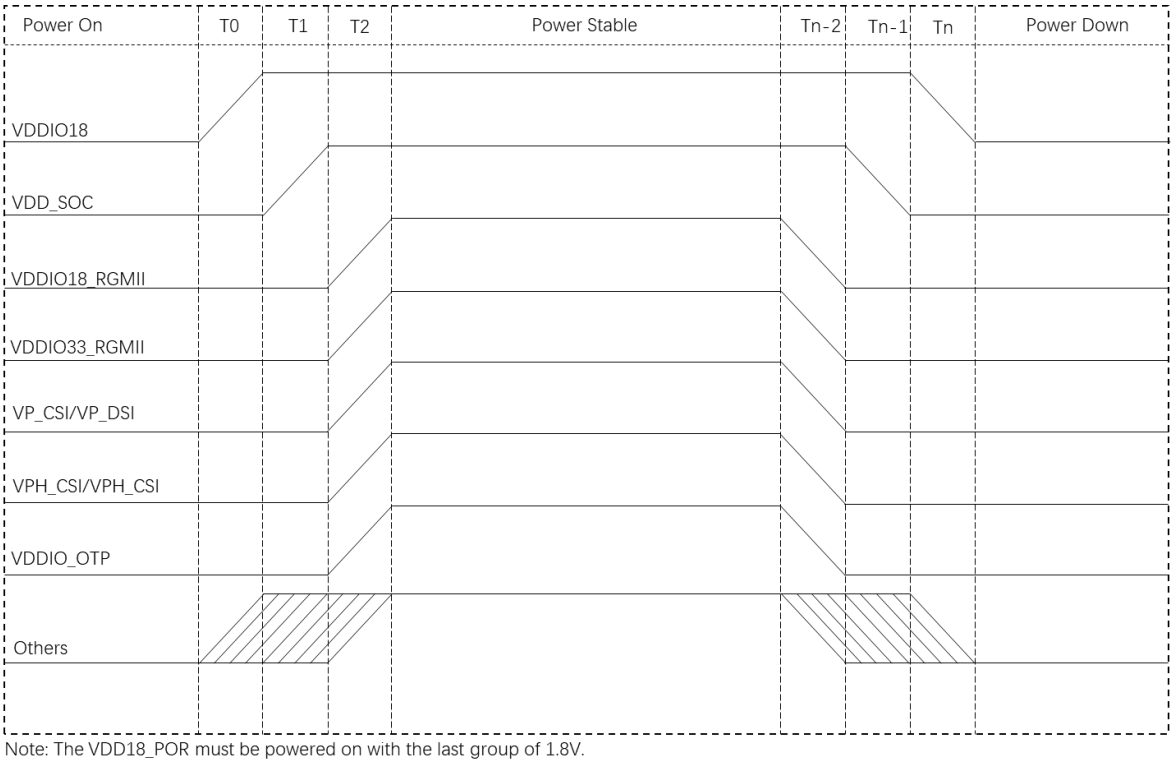


Figure 2-8 power-on sequences, and the power-off sequences

2.8.4 Normal GPIO Electrical Parameters

Table 2-16 Normal GPIO DC Electrical Specifications (VDDIO18 = 1.8 V)

Table 2-15 GPIO DC Electrical Specifications

Symbol	Description	Min	Typ	Max	Unit	Remarks
VDDIO18	Interface voltage	1.71	1.8	1.89	V	
V_{IH}	High-level input voltage	1.27	-	1.98	V	
V_{IL}	Low-level input voltage	–0.3	-	0.58	V	
I_L	Input leakage current	-	-	±10	μA	
I_{OZ}	Tristate output leakage current	-	-	±10	μA	
V_{OH}	High-level output voltage	1.4	-	-	V	
V_{OL}	Low-level output voltage	-	-	0.45	V	
R_{SPU}	Internal Srtong pull-up resistor	2.8	3.5	4.5	kΩ	

Symbol	Description	Min	Typ	Max	Unit	Remarks
R_{PU}	Internal pull-up resistor	18	25	38	kΩ	
R_{PD}	Internal pull-down	17	22	33	kΩ	
I_{OL}	High Level Output Current @VOH(max)	2.1	3.1	4.1	mA	DS[3:0] = ’0000’
		4.6	6.7	8.9	mA	DS[3:0] = ’0001’
		6.6	9.6	12.8	mA	DS[3:0] = ’0010’
		8.8	12.9	17.2	.mA	DS[3:0] = ’0011’
		12.2	18.0	24.0	mA	DS[3:0] = ’0100’
		14.2	20.9	28	mA	DS[3:0] = ’0101’
		15.7	23.2	31.1	mA	DS[3:0] = ’0110’
		17.5	25.9	34.8	mA	DS[3:0] = ’0111’
I_{OH}	High Level Output Current @VOH(max)	1.8	2.8	3.8	mA	DS[3:0] = ’0000’
		3.9	5.9	8.1	mA	DS[3:0] = ’0001’
		5.5	8.4	11.6	mA	DS[3:0] = ’0010’
		7.4	11.3	15.6	mA	DS[3:0] = ’0011’
		10.2	15.7	21.6	mA	DS[3:0] = ’0100’
		11.9	18.3	25.1	mA	DS[3:0] = ’0101’
		13.1	20.2	27.9	mA	DS[3:0] = ’0110’
		14.6	22.6	31.2	mA	DS[3:0] = ’0111’

2.8.5 LPDDR Electrical Parameters

Table 2-17 The different voltages of EIC7700X and LPDDR SDRAM in LPDDR4, LPDDR4X, LPDDR5 mode

Table 2-16 voltages of EIC7700X

	EIC7700X			SDRAM		
	VDDQ(V)	VDD2H(V)	VAA_VDD2H(V)	VDDQ(V)	VDD1(V)	VDD2H(V)

LPDDR4	0.8	1.1	1.8	1.1	1.8	1.1
LPDDR4X	0.6	1.1	1.8	0.6	1.8	1.1
LPDDR5	0.5	1.05	1.8	0.5	1.8	1.05

Table 2-17 DC electrical parameters for the signals which powered by VDDQ

Symbol	Parameter	Minimum	Typical	Maximum	Unit
IIZ	BP_DAT input leakage current			221.15	uA
Vref	Referenc evoltage	0.49*VDDQ	0.5*VDDQ	0.51*VDDQ	V
VIH-DC	Input high voltage for BP_DAT	Vref +0.020			V
VIL-DC	Input low voltage for BP_DAT			Vref – 0.020	V
VID-DC	Differential input voltage for abs(DQS_t – DQS_c)	0.1			V

Table 2-18 DC electrical parameters for the signals which powered by VDD2H

Symbol	Parameter	Minimum	Typical	Maximum	Unit
VIH-DC	Input high voltage	0.7*VDD2H			V
VIL-DC	Input low voltage			0.3*VDD2H	V

2.8.6 SDIO Electrical Parameters

Table 2-19 SDIO electrical parameters (3.3 V)

Symbol	Description	Min	Typ	Max	Unit
VDDIO	Power voltage	3.135	3.3	3.465	V
VOH	High-level output	0.75 x DVDD33	-	-	V
VOL	Low-level output	-	-	0.125 x VDDIO33	V
VIH	High-level input	1.68	-	-	V
VIL	Low-level input	-0.3	-	0.9	V

Table 2-20 SDIO electrical parameters (1.8 V)

Symbol	Description	Min	Typ	Max	Unit
DVDD	Power voltage	1.71	1.8	1.89	V
VOH	High-level output	VDDIO-0.45	-	-	V
VOL	Low-level output	-	-	0.45	V
VIH	High-level input	1.07	-	-	V
VIL	Low-level input	–0.3	-	0.68	V

2.8.7 RGMII Electrical Parameters

Table 2-21 RGMII electrical parameters (3.3 V)

Symbol	Description	Min	Typ	Max	Unit	Remarks
VDDIO33_RGMII	Interface voltage	3.135	3.3	3.3465	V	
V_{IH}	High-level input voltage	2	-	VDDIO+0.3	V	
V_{IL}	Low-level input voltage	–0.3	-	0.8	V	
I_L	Input leakage current	-	-	±10	μA	
I_{OZ}	Tristate output leakage current	-	-	±10	μA	
V_{OH}	High-level output voltage	2.4	-	-	V	
V_{OL}	Low-level output voltage	-	-	0.4	V	
R_{PU}	Internal pull-up resistor	24		34	kΩ	
R_{PD}	Internal pull-down	24		34	kΩ	
I_{OL}	High Level Output Current @VOH(max)	1.1	1.6	2.1	mA	DS[3:0] = ’0000’
		2.1	3.1	4.1	mA	DS[3:0] = ’0001’
		3.2	4.7	6.2	mA	DS[3:0] = ’0010’
		4.2	6.2	8.2	mA	DS[3:0] = ’0011’
		5.3	7.8	10.3	mA	DS[3:0] = ’0100’
		6.4	9.3	12.3	mA	DS[3:0] = ’0101’
		7.4	10.8	14.3	mA	DS[3:0] = ’0110’
		8.4	12.2	16.2	mA	DS[3:0] = ’0111’
I_{OH}	High Level Output Current @VOH(max)	2.1	3.6	5.2	mA	DS[3:0] = ’0000’
		4.1	6.9	10.1	mA	DS[3:0] = ’0001’
		6.4	10.7	15.7	mA	DS[3:0] = ’0010’
		8.1	13.5	19.8	mA	DS[3:0] = ’0011’
		10.4	17.3	25.4	mA	DS[3:0] = ’0100’
		12.4	20.6	30.3	mA	DS[3:0] = ’0101’
		14.4	23.9	35.1	mA	DS[3:0] = ’0110’
		16.1	26.8	39.3	mA	DS[3:0] = ’0111’

Table 2-22 GRMII electrical parameters (1.8 V)

Symbol	Description	Min	Typ	Max	Unit	Remarks
VDDIO33_RGMII	Interface voltage	1.71	1.8	1.89	V	
V_{IH}	High-level input voltage	1.27	-	1.98	V	
V_{IL}	Low-level input voltage	–0.3	-	0.58	V	
I_L	Input leakage current	-	-	±10	μA	
I_{OZ}	Tristate output leakage current	-	-	±10	μA	

V_{OH}	High-level output voltage	1.4	-	-	V	
V_{OL}	Low-level output voltage	-	-	0.45	V	
R_{PU}	Internal pull-up resistor	24		34	kΩ	
R_{PD}	Internal pull-down	24		34	kΩ	

2.8.8 Oscillator Electrical Parameters

EIC7700X requires the 24MHz oscillator for generating the main clock source.

Table 2-23 Oscillator electrical parameters (1.8 V)

Symbol	Description	Min.	Typ.	Max.	Unit
Fo	Nominal Frequency		24		MHz
Δf/fo	Frequency Tolerance	-30		30	ppm
CL	Load Capacitance	7.5	12	12.5	pF
ESR	Equivalent Series Resistance			100	oHm

2.9 Interface Timing

2.9.1 Ethernet MAC Port Timings

2.9.1.1 RGMII Timings

Figure 2-9 shows the 1000 Mbit/s RX timing of the RGMII.

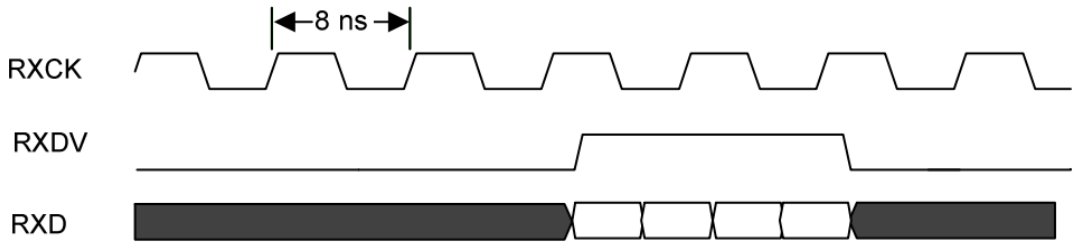


Figure 2-9 1000 Mbit/s RX timing of the RGMII

Figure 2-10 shows the 1000 Mbit/s TX timing of the RGMII.

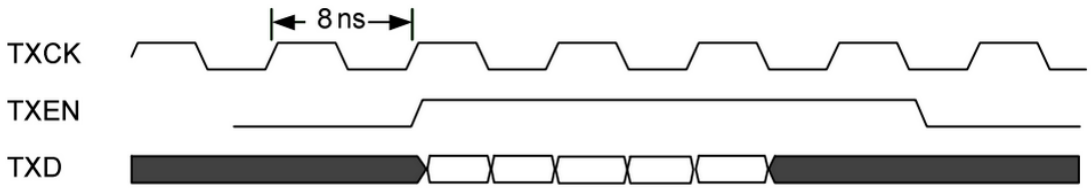


Figure 2-10 1000 Mbit/s TX timing of the RGMII

Table 2-25 describes the timing parameters of the RGMII.

Table 2-24 Timing parameters of the RGMII

Parameter	Symbol	Signal	Min	Max	Unit
RGMII clock cycle	T	RXCK, TXCK	8	8	ns
RGMII signal setup time	Tsu (RX)	RXER, RXDV, RXD[3:0]	1	N/A	ns
RGMII signal hold time	Thd (RX)	RXER, RXDV, RXD[3:0]	1	N/A	ns
RGMII output signal delay	Tov (TX)	TXD[3:0], TXEN	-0.5	0.5	ns

2.9.1.2 MDIO Interface Timing

Figure 2-11 shows the read timing of the MDIO interface.

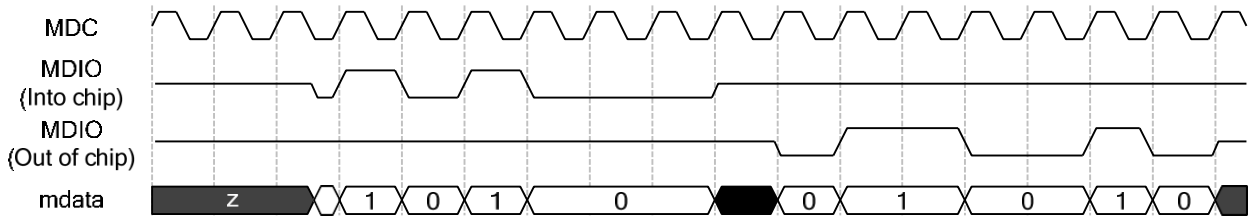


Figure 2-11 Read timing of the MDIO interface

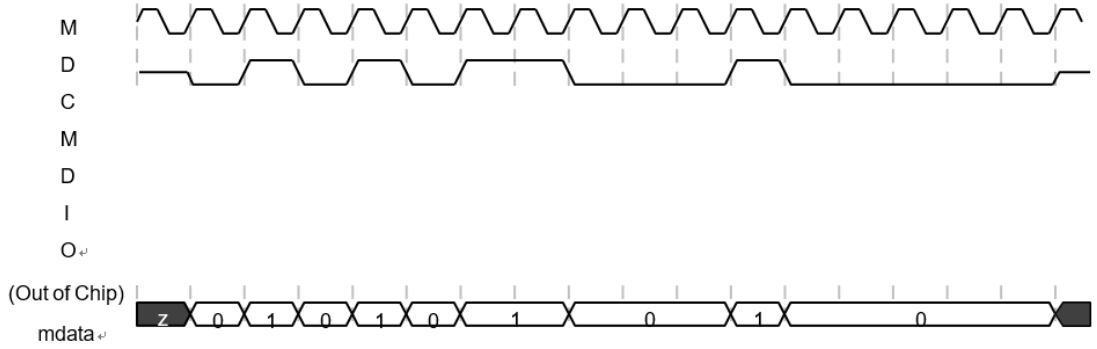


Figure 2-12 Write timing of the MDIO interface

Figure 2-13 illustrates the RX timing parameters of the MDIO interface.

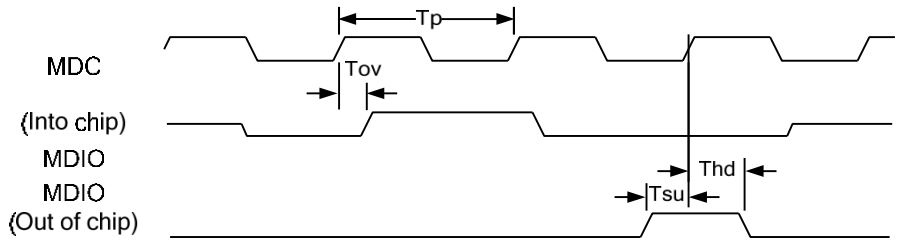


Figure 2-13 Timing parameters of the MDIO interface

Table 2-25 The timing parameters of the MDIO interface

Parameter	Symbol	Signal	Min	Max	Unit
MDIO data RX delay	Tov	MDIO	0	300	ns
MDIO clock cycle	Tp	MDCK	400	400	ns
MDIO data TX setup time	Tsu	MDIO	10	N/A	ns
MDIO data TX hold time	Thd	MDIO	10	N/A	ns

2.9.2 SPI Timings

In Figure 2-14 to Figure 2-16, the conventions are as follows:

- MSB = most significant bit
- LSB = least significant bit
- SPI_CK(0): spo = 0
- SPI_CK(1): spo = 1

Figure 2-14 shows the SPI clock (SPICK) timing.

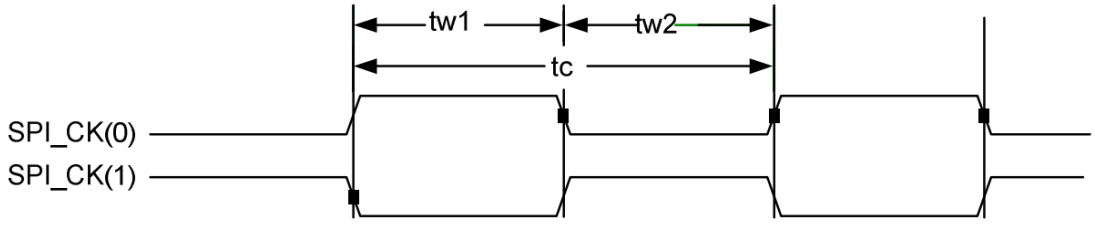


Figure 2-14 SPICK timing

Figure 2-15 and Figure 2-16 show the SPI timings in master mode.

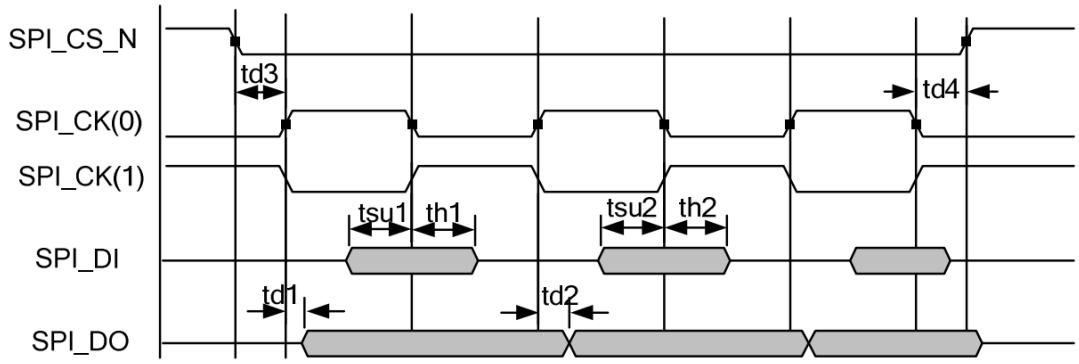


Figure 2-15 SPI timing in master mode (sph = 1)

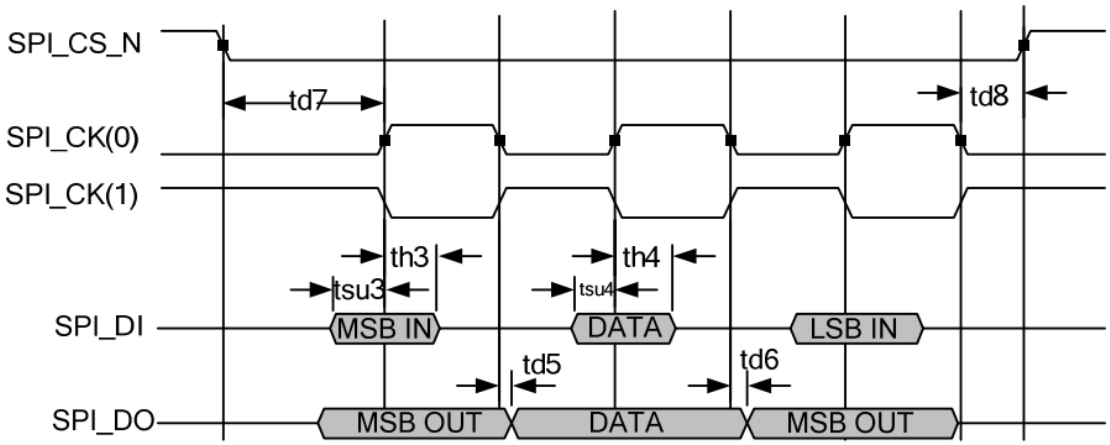


Figure 2-16 SPI timing in master mode (sph = 0)

3 System

3.1 Reset

3.1.1 Sources Of Reset

Table 3-1 description of the sources of reset

Sources of reset	description
internal por	You can use the POR_SEL pad to select internal por or external reset from KEY_RESET_N.
KEY_RESET_N	1. POR_SEL=1: select the internal por, EIC7700X has debounce logic and the debounce time is 10ms 2. POR_SEL=0: select the external por, debounce function is disabled 3. After por reset is released, EIC7700X will check whether the reset signal of KEY_RESET_N is deasert. If true, EIC7700X will continue to release the other reset after the period of debounce
watchdog	This signal is also output to the pad(RST_OUT_N). You can use this signal to reset other peripherals for system synchronization.
sw reset	This reset is generated by the software configuration 0x1AC0_FFE6. It is also output to the pad(RST_OUT_N).
debug_ndreset	For details about the debug_ndreset reset applications of MCPU, LPCPU, SCPU, and NPU, see the corresponding CPU description
lowpwr_rst_n[:]	Low-power reset for power domain. After powering on, the default value is 1, and all domains are powered on. It is output to other modules after AND with the reset from system reset from CSR.

3.1.2 Reset Requirements

A reset required for modules with bus reset, core logic reset and configuration interface reset, unless otherwise specified, is shown below:

1. When reset, the corresponding logic should be in the IDLE state, and there is no data transmission in progress, otherwise the module may not be able to return a response signal to the other party after being reset, resulting in the bus hanging.

2. It is recommended to enable the clock before releasing the reset of the module, so as to avoid the failure of some synchronous reset modules to reset.
3. For the released sequence, you can release the configuration interface first, and then release the bus interface and core logic. Make sure that all logic has been reset before starting the configuration and start-up work, so as to avoid the module being accessed by the pre-master before it is ready.

3.2 clock

Address space：0x5182_8000~0x5182_FFFF，32k。

3.2.1 Overview

A 24M clock serves as a pll reference clock input for the entire chip. There are a total of 7 PLLs, of which one is for the CPU and one for the DDR, and the rest are in the clock reset subsystem.

The following features are supported:

- Compatible with APB3.0 interface, bit width 32bit
- Contains 7 PLLs
- clock gating is Suport
- Clock division is supported
- Clock MUX is supported
- Soft reset of other subsystems is supported

3.2.2 Block diagram

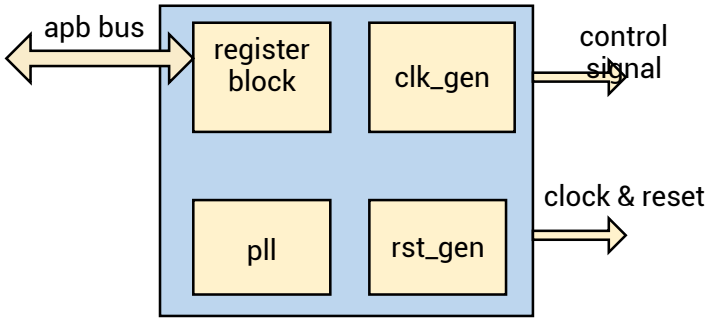


Figure 3-1 Clock and Reset block diagram

3.2.3 System clock

3.2.3.1 Clock structure

Each clock can be composed of any basic unit, and each basic unit contains at least one of PLL, MUX, DIV, and GATE

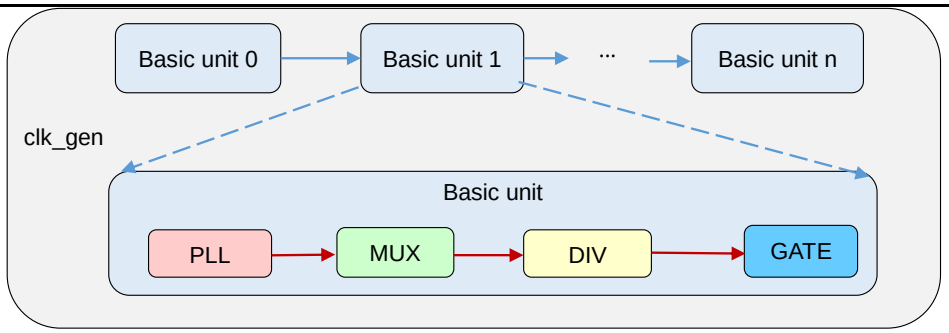


Figure 3-2 Clock structure

3.2.3.2 Clock solution

- N – NO, There is no such base unit
- Y – YES
- GM – glitch free mux
- M – static mux

“gate” Column:：

- 1: Gate is enabled by default
- 0: Gate is disenabled by default

class	clkname	max_f	dflt_f	source0	source1	source2	mux	dflt_sel	min_div	dflt_div	max_div	gate
source	clk_xtal_24m	24	24									
source	clk_spll0_fout1	1600	1600									
source	clk_spll0_fout2	800	800									1
source	clk_spll0_fout3	400	400									
source	clk_spll1_fout1	1500	1500									
source	clk_spll1_fout2	300	300									
source	clk_spll1_fout3	250	250									
source	clk_spll2_fout1	2080	2080									
source	clk_spll2_fout2	1040	1040									
source	clk_spll2_fout3	416	416									
source	clk_vpll_fout1	1188	1188									
source	clk_vpll_fout2	594	594									
source	clk_vpll_fout3	49.5	49.5									
source	clk_apll_fout1	983.04	983.04									
source	clk_apll_fout2	0	0									
source	clk_apll_fout3	0	0									
source	ext_mclk											

class	clkname	max_f	dflt_f	source0	source1	source2	mux	dflt_sel	min_div	dflt_div	max_div	gate
ddr	lpddr_ref_clk_bak	125	-									
internal	clk_1m	1	1	clk_xtal_24m			N	N	24	24	24	N
mcpu	clk_sys_cfg	200	200	clk_spll0_fout3			N	N	2	2	7	N
mipi dphy, hdmi	clk_mipi_txesc	20	20	clk_sys_cfg			N	N	10	10	10	N
noc	noc_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	N
noc	noc_nsp_clk	1040	1040	clk_spll2_fout1			N	N	2	2	7	1
mcpu	clk_u84_core_lp	400	400	clk_spll0_fout2			N	N	2	2	2	N
scpu	clk_scpu_core	800	800	clk_spll0_fout1			N	N	2	2	15	1
scpu	clk_scpu_bus	400	400	clk_scpu_core			N	N	2	2	2	1
lpcpu	clk_lpcpu_core	800	800	clk_spll0_fout1			N	N	2	2	15	1
lpcpu,mcpu, npu,secure	cpu_rtc_toggle	1	1	clk_1m			N	N	N	N	N	N
lpcpu	clk_lpcpu_bus	400	400	clk_lpcpu_core			N	N	2	2	2	1
3Dgpu	gpu_aclk	800	800	clk_spll0_fout1			N	N	2	2	15	0
3Dgpu	gpu_gray_clk	24	24	clk_xtal_24m			N	N	N	N	N	0
3Dgpu	gpu_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	0
internal	clk_dsp_aclk_st1	1040	1040	clk_spll2_fout1	clk_spll0_fout1		GM	0	2	2	15	N
dsp	dspt_aclk	1040	1040	clk_dsp_aclk_st1			N	N	N	N	N	0
dsp	dspt_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	0
internal	clk_d2ddr_aclk	1040	1040	clk_spll2_fout1	clk_spll0_fout1		GM	0	2	2	15	N
tcu	tcu_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	1
internal	clk_ddr_aclk_st0	2080	1600	clk_spll2_fout1	clk_spll0_fout1		GM	1	N	N	N	N
internal	clk_ddr_aclk	1040	800	clk_ddr_aclk_st0			N	N	2	2	15	N
tcu	tcu_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	1

class	clkname	max_f	dflt_f	source0	source1	source2	mux	dflt_sel	min_div	dflt_div	max_div	gate
ddr	ddrt0_p0_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt0_p1_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt0_p2_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt0_p3_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt0_p4_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt0_trace_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt1_trace_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt1_p0_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt1_p1_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt1_p2_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt1_p3_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
ddr	ddrt1_p4_aclk	1040	800	clk_ddr_aclk			N	N	N	N	N	1
tcu	tcu_aclk_free	1040	800	clk_ddr_aclk			N	N	N	N	N	N
hsp	hsp_aclk	800	800	hsp_aclk_free			N	N	N	N	N	1
hsp	hsp_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	1
internal	sata_phy_ref_clk	50	50	clk_spll1_fout2			N	N	6	6	15	N
internal	sata_phy_ref_clk_gated	125	50	sata_phy_ref_clk	lpddr_ref_clk_bak		M	0	N	N	N	1
sata	hsp_sata_phy_ref_clk	125	50	sata_phy_ref_clk_gated			N	N	N	N	N	N
usb	hsp_ref_in_clk	24	24	clk_xtal_24m			N	N	N	N	N	N
gmac	hsp_eth_app_clk	200	200	clk_sys_cfg			N	N	N	N	N	N
gmac	hsp_eth_csr_clk	200	200	clk_sys_cfg			N	N	N	N	N	N
gmac	hsp_eth0_core_clk	125	125	clk_spll1_fout3			N	N	2	2	127	N
gmac	hsp_eth1_core_clk	125	125	clk_spll1_fout3			N	N	2	2	127	N

class	clkname	max_f	dflt_f	source0	source1	source2	mux	dflt_sel	min_div	dflt_div	max_div	gate
internal	rmii_ref_clk_mux	125	50	rmii_ref_clk	lpddr_ref_clk_bak		M	0	6	6	6	N
gmac	hsp_rmii0_ref_clk	125	50	rmii_ref_clk_mux			N	N	N	N	N	1
gmac	hsp_rmii1_ref_clk	125	50	rmii_ref_clk_mux			N	N	N	N	N	1
emmc	hsp_mshc0_core_clk	200	200	clk_spll0_fout3	clk_spll2_fout3		M	0	2	2	63	1
emmc	hsp_mshc1_core_clk	200	200	clk_spll0_fout3	clk_spll2_fout3		M	0	2	2	63	1
emmc	hsp_mshc2_core_clk	200	200	clk_spll0_fout3	clk_spll2_fout3		M	0	2	2	63	1
emmc	hsp_mshc0_tmr_clk	1	1	clk_1m			N	N	N	N	N	N
emmc	hsp_mshc1_tmr_clk	1	1	clk_1m			N	N	N	N	N	N
emmc	hsp_mshc2_tmr_clk	1	1	clk_1m			N	N	N	N	N	N
usb	hsp_usb0_suspend_clk	1	1	clk_1m			N	N	N	N	N	N
usb	hsp_usb1_suspend_clk	1	1	clk_1m			N	N	N	N	N	N
pcie	pciet_aclk	520	520	clk_spll2_fout2			N	N	2	2	15	1
pcie	pciet_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	0
pcie	pciet_cr_clk	100	100	clk_sys_cfg			N	N	2	2	2	0
internal	clk_npu_aclk_st1	800	800	clk_spll0_fout1			N	N	2	2	15	N
npu	npu_aclk	800	800	clk_npu_aclk_st1			N	N	N	N	N	0
npu	npu_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	0
internal	clk_npu_llc_src0	800	800	clk_spll0_fout1			N	N	2	2	15	N
internal	clk_npu_llc_src1	1040	1040	clk_spll2_fout1			N	N	2	2	15	N
npu	npu_llc_aclk	1188	1188	clk_npu_llc_src0	clk_npu_llc_src1	clk_vpll_fout1	GM	2	N	N	N	0
internal	clk_npu_core_st1	1500	1500	clk_spll1_fout1	clk_vpll_fout1	clk_spll2_fout2	GM	0	1	1	15	N
npu	npu_clk	1500	1500	clk_npu_core_st1			N	N	N	N	N	0

class	clkname	max_f	dflt_f	source0	source1	source2	mux	dflt_sel	min_div	dflt_div	max_div	gate
noc	npu_noc_aclk	800	800	clk_npu_aclk_st1			N	N	N	N	N	N
internal	clk_vi_aclk_st1	1040	800	clk_spll0_fout1	clk_spll2_fout1		GM	0	2	2	15	N
video input	vi_aclk	1040	800	clk_vi_aclk_st1			N	N	N	N	N	0
video input	vi_dig_dw_clk	800	594	clk_vpll_fout1	clk_spll0_fout1		M	0	2	2	15	0
video input	vi_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	0
video input	vi_dvp_clk	800	594	clk_vpll_fout1	clk_spll0_fout1		GM	0	2	2	15	0
video input	vi_dig_isp_clk	800	594	clk_vpll_fout1	clk_spll0_fout1		GM	0	2	2	15	0
shutter	vi_ref_clk[0]	74.25	74.25	clk_vpll_fout2			N	N	8	8	127	0
shutter	vi_ref_clk[1]	74.25	74.25	clk_vpll_fout2			N	N	8	8	127	0
shutter	vi_ref_clk[2]	74.25	74.25	clk_vpll_fout2			N	N	8	8	127	0
shutter	vi_ref_clk[3]	74.25	74.25	clk_vpll_fout2			N	N	8	8	127	0
shutter	vi_ref_clk[4]	74.25	74.25	clk_vpll_fout2			N	N	8	8	127	0
shutter	vi_ref_clk[5]	74.25	74.25	clk_vpll_fout2			N	N	8	8	127	0
Combo PHY	vi_phy_txclkesc	20	20	clk_mipi_txesc			N	N	N	N	N	0
Combo PHY	vi_phy_cfg_clk	24	24	clk_xtal_24m			N	N	N	N	N	0
video output	vo_aclk	1040	800	clk_vi_aclk_st1			N	N	N	N	N	0
video output	vo_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	1
video output	vo_hdmi_iesmclk	200	200	clk_spll0_fout3			N	N	2	2	15	0
video output	vo_pixel_clk	594	594	clk_vpll_fout1	clk_spll2_fout2		M	0	2	2	63	1
video output	clk_vo_mclk_st1	98.3	98.3	clk_apll_fout1			N	N	10	10	255	N
video output	vo_i2s_mclk	98.3	98.3	clk_vo_mclk_st1	ext_mclk		M	0	N	N	N	0
video output	vo_cr_clk	20	20	clk_mipi_txesc			N	N	N	N	N	0
video output	vo_cec_clk	0.032	0.032	clk_vpll_fout2			N	N	18128	18128	18128	N
video output	vo_phy_txclkesc	20	20	clk_mipi_txesc			N	N	N	N	N	0
video codec	vc_aclk	1040	1040	clk_spll0_fout1	clk_spll2_fout1		GM	1	2	2	15	0
internal	pad_aud_mclk_o	98.3	98.3	clk_vo_mclk_st1			N	N	N	N	N	N

class	clkname	max_f	dflt_f	source0	source1	source2	mux	dflt_sel	min_div	dflt_div	max_div	gate
video codec	vc_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	0
internal	clk_vc_cdec_root	2080	1600	clk_spll0_fout1	clk_spll2_fout2		GM	0	N	N	N	N
video codec	vc_je_clk	1040	800	clk_vc_cdec_root			N	N	2	2	15	0
video codec	vc_jd_clk	1040	800	clk_vc_cdec_root			N	N	2	2	15	0
video codec	vc_ve_clk	693.33	533.33	clk_vc_cdec_root			N	N	3	3	15	0
video codec	vc_vd_clk	1040	800	clk_vc_cdec_root			N	N	2	2	15	0
video codec	clk_g2d_st1	1040	1040	clk_dsp_root			N	N	2	2	15	N
video codec	g2d_clk	1040	1040	clk_g2d_st1			N	N	N	N	N	0
video codec	g2d_aclk	1040	1040	clk_g2d_st1			N	N	N	N	N	0
internal	clk_pvt_inner	1.2	1.2	clk_xtal_24m			N	N	20	20	20	N
system	pvt_clk[0]	1.2	1.2	clk_pvt_inner			N	N	N	N	N	1
ddr	pvt_clk[1]	1.2	1.2	clk_pvt_inner			N	N	N	N	N	1
dma	clk_aondma_cfg	200	200	clk_sys_cfg			N	N	N	N	N	1
internal	clk_aondma_axi_st3	800	24	clk_spll0_fout1	clk_xtal_24m		M	1	2	2	15	N
dma	aondma_aclk	800	24	clk_aondma_axi_st3			N	N	N	N	N	1
secure, lpcpu	aon_aclk	800	24	clk_aondma_axi_st3			N	N	N	N	N	1
TIMER0	timer_clk[0]	24	24	clk_xtal_24m			N	N	N	N	N	1
TIMER1	timer_clk[1]	24	24	clk_xtal_24m			N	N	N	N	N	1
TIMER2	timer_clk[2]	24	24	clk_xtal_24m			N	N	N	N	N	1
TIMER3	timer_clk[3]	24	24	clk_xtal_24m			N	N	N	N	N	1
TIMER0	timer_pclk[0]	200	200	clk_sys_cfg			N	N	N	N	N	1
TIMER1	timer_pclk[1]	200	200	clk_sys_cfg			N	N	N	N	N	1
TIMER2	timer_pclk[2]	200	200	clk_sys_cfg			N	N	N	N	N	1
TIMER3	timer_pclk[3]	200	200	clk_sys_cfg			N	N	N	N	N	1
rtc	clk_rtc_cfg	200	200	clk_sys_cfg			N	N	N	N	N	1

class	clkname	max_f	dflt_f	source0	source1	source2	mux	dflt_sel	min_div	dflt_div	max_div	gate
rtc	clk_rtc	0.5	0.15625	clk_1m			N	N	2	64	64	N
i2c10	i2c10_pclk	200	200	clk_sys_cfg			N	N	N	N	N	1
i2c11	i2c11_pclk	200	200	clk_sys_cfg			N	N	N	N	N	1
secure	clk_pka_cfg	200	200	clk_sys_cfg			N	N	N	N	N	1
secure	clk_spacc_cfg	200	200	clk_sys_cfg			N	N	N	N	N	1
clk_crypto	clk_crypto	400	400	clk_spll0_fout1			N	N	4	4	15	1
clk_trng_cfg	clk_trng_cfg	200	200	clk_sys_cfg			N	N	N	N	N	1
otp	clk_otp_cfg	200	200	clk_sys_cfg			N	N	N	N	N	1
clmm	clmm_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	N
gpio	clmm_deb_clk	1	1	clk_1m			N	N	N	N	N	N
lsp	lsp_cfg_clk	200	200	clk_sys_cfg			N	N	N	N	N	N
noc	noc_wd_refclk	24	24	clk_sys_cfg			N	N	1	1	2	1
noc	rnoc_nsp_clk	1040	800	clk_vi_aclk_st1			N	N	N	N	N	1
TIMER3-timer8	timer3_clk8	50	50	clk_vpll_fout3			N	N	N	N	N	1

3.2.3.3 PLL

3.2.3.3.1 features

- Input frequency range: 12MHz ~ 1200MHz
- Output frequency range: 9MHz ~ 2500MHz
- 24 bit fractional accuracy
- The lock signal is used to indicate that the output frequency has been successfully locked at the set value

3.2.3.3.2 configuration

In integer mode, the frequency is calculated by the following formula:

$$\mathrm{FOUT}* = \frac{FREF \times FBDIV}{4 \times REFDIV \times （POSTDIV_A + 1） \times （POSTDIV_B + 1）}$$

In fractional mode, the frequency is calculated by the following formula:

$$\mathrm{FOUT}* = \frac{FREF}{4 \times REFDIV} \times \frac{FBDIV + \frac{FRAC}{2^{24}}}{（POSTDIV_A + 1） \times （POSTDIV_B + 1）}$$

Note: POSTDIV_A should be bigger than POSTDIV_B

Configuration process:

1. Configure pllen = 0
2. Configure PLL working mode, fraction mode (dsmen=1) or integer mode (dsmen=0)
3. Configure clock frequency/bypass, and other information
4. Configure pllen = 1 and wait for lock = 1

3.2.3.3.3 PLL reference clock

Table 3-2 PLL reference clock

pll name	refclk name	refclk freq (MHz)	remarks
spll0	clk_xtal_24m	24	Input from external source
spll1	clk_xtal_24m	24	Input from external source
spll2	clk_xtal_24m	24	Input from external source
vpll	clk_xtal_24m	24	Input from external source
apll	clk_xtal_24m	24	Input from external source
cpll	clk_xtal_24m	24	Input from external source
dpll	clk_xtal_24m	24	Input from external source，default
	pinmux_lpddr_ref_clk_bak	-	Input directly by pad LPDDR_REF_CLK. The frequency depends on the input frequency

3.2.3.3.4 PLL spread-spectrum

The spll0/spll1/spll2/vpll/apll/cpll/dpll in the system have spread spectrum modulator (SSMOD) module, which can spread the output clock of the pll and reduce the EMI impact of the peak frequency. SSMOD is not recommended for clocks with long-term jitter requirements. Turning on ssmod causes a maximum of 1.56% skew.

When using ssmod, the PLL must be configured to fractional mode:

1. Initialize the modulator with **RESET**=1'b1, **DISABLE_SSCG**=1'b1, and **RESETPTR**=1'b1
2. Initialize the PLL with **PD**=1'bl
3. Provide valid settings to **INTIN**(fbdiv), **FRACIN**, **DIVVAL**(ss_div), **SPREAD**, and **DOWNSPREAD**
4. Set the PLL to fration mode，**DSMPD** = 0，**DACPD** = 0（**DSMEN** = 1, **DACEN** = 1）
5. Keep **pllen**=0 at least 1us
6. Set **RESET**=1'b0 and **DISABLE_SSCG**=1'b0 to activate the modulator.
7. After the 1us delay, set **PD**=1'b0 and wait for the PLL to establish **LOCK**=1'b1
8. Once the PLL has asserted its LOCK signal, set **RESETPTR**=1'b0 and after 2 **CLKSSCG** periods, the modulator will begin to update the PLL divide settings.

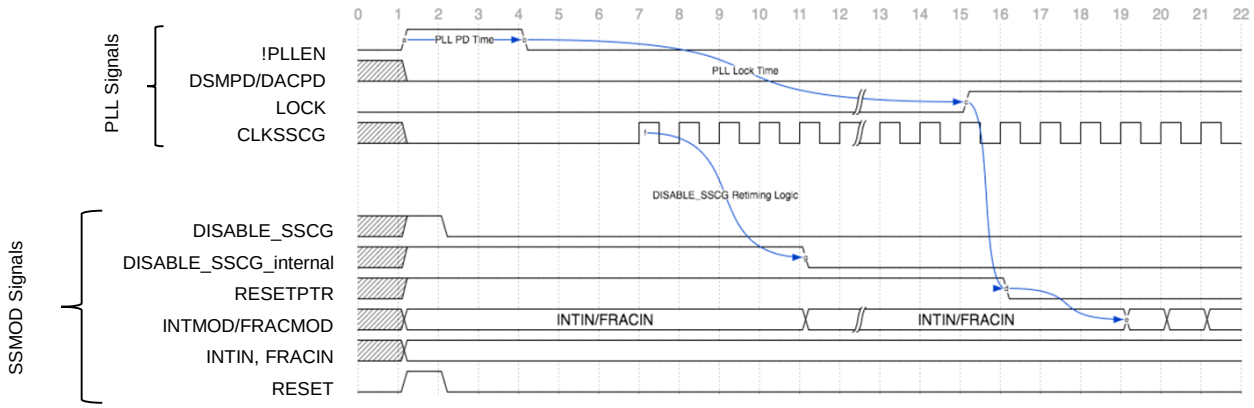


Figure 3-3 Timing diagram for programming SSMOD

3.2.3.4 Clock switching

The clock is switched as follows:

1. Static switching

Clock switching with ordinary MUX can only be configured before the subsystem reset is released, cannot be switched during work

2. Dynamic switching

The clock with glitch-free mux can be switched dynamically without glitches during the switching. If the clock has clock gating, it is recommended to gating first and then switching

3. Dynamic divider

The clock of dynamic frequency division supports dynamic configuration during operation, but the following requirements need to be followed:

- Gating the clock first
- Configure the dynamic divider
- Wait 32 cycles to release gating

3.3 Processor

3.3.1 MCPU

3.3.1.1 Overview

The EIC7700X MCPU includes 4 64-bit RISC-V cores, along with the necessary functional units required to support the cores. These units include a Core-Local Interruptor (CLINT) to support local interrupts, a Platform-Level Interrupt Controller (PLIC) to support platform interrupts, physical memory protection, a Debug unit to support a JTAG-based debugger host connection, Nexus 5001 compliant instruction trace for non-invasive debug and performance profiling, and a local crossbar that integrates the various components together.

The EIC7700X MCPU memory system consists of a Data Cache and Instruction Cache, with coherent L1 caches, private L2 Cache, shared L3 Cache, and a directory based coherence manager. The EIC7700X MCPU also includes a Front Port, which allows external masters to be coherent with the L1 memory system and access to the TIMs, thereby removing the need to maintain coherence in software for any external agents. All memories, including caches and TIMs, support Single Error Correction, Double Error Detection (SECDED) ECC to provide improved reliability and address safety critical applications.

The EIC7700X MCPU is guaranteed to be compatible with all applicable RISC-V standards, and this document should be read together with the official RISC-V user-level, privileged, and external debug architecture specifications.

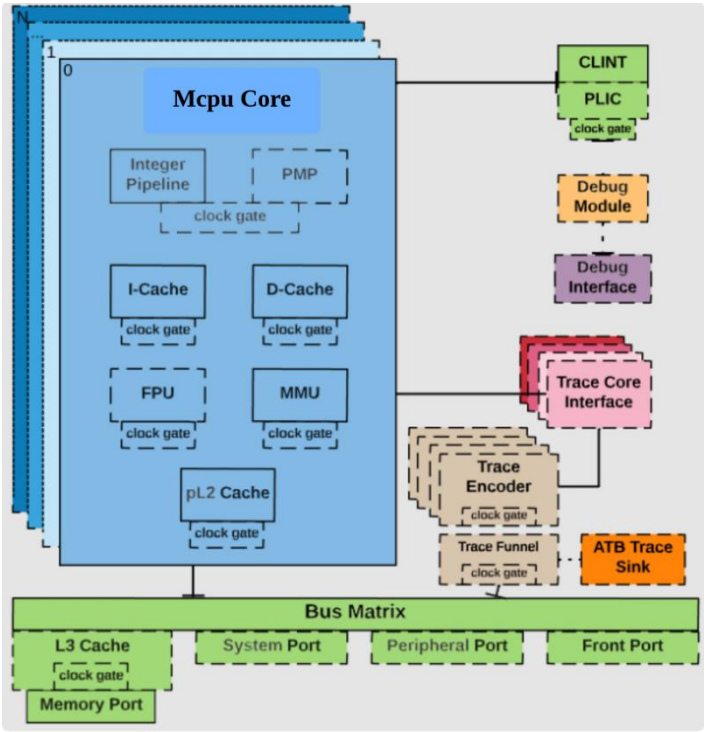


Figure 3-4 MCPU

3.3.1.2 Feature

Table 3-3 MCPU feature summary

Feature	Description
ISA	RV64GC_Zba_Zbb_Sscofpmf
Custom Instruction Extension (SCIE)	Present
Privilege Modes	M, U, HS, VU, VS
L1 Instruction Cache	32 KiB 4-way instruction cache
L1 Data Cache	32 KiB 4-way data cache
Private L2 (pL2) Cache	256 KiB 8-way pL2 cache with 2 banks

L3 Cache	4 MiB 16-way L3 cache with 4 banks
ECC Support	Single error correction, double error detection on the L1 data cache, pL2 cache, and L3 cache
Physical Memory Protection	8 regions with a granularity of 4096 bytes
Memory Management Unit	Sv48 virtual memory support with fully-associative 32-entry L1 data and instruction TLBs, and a 4-way 512-entry unified TLB

- EIC7700X RISC-V Core

The EIC7700X MCPU includes four 64-bit EIC7700X RISC-V cores, which each have a triple-issue, out-of-order, 13 stage RISC-V processor. The EIC7700X core is guaranteed to be compatible with all applicable RISC-V standards.

Each EIC7700X core is configured to support the RV64I base ISA, as well as the Multiply (M), Atomic(A), Single-Precision Floating Point (F), Double-Precision Floating Point (D), Compressed (C), CSR Instructions (Zicsr), Instruction-Fetch Fence (Zifencei), Address Calculation (Zba), Basic Bit Manipulation (Zbb), and Count Overflow and Mode-Based Filtering (Sscofpmf) RISC-V extensions. This is captured by the RISC-V extension string: RV64GC_Zba_Zbb_Sscofpmf.

The EIC7700X MCPU also supports machine, hypervisor, user, virtual supervisor, and virtual user privilege modes, in conjunction with Physical Memory Protection (PMPs). The microarchitecture also incorporates a branch prediction unit that is composed of a 32-entry Branch Target Buffer (BTB), a 9.1 KiB-entry Branch History Table (BHT), a 16-entry Return Address Stack (RAS), 512-entry Indirect Jump Target Predictor (IJTP), and a 16-entry Return Instruction Predictor.

The EIC7700X MCPU includes an IEEE 754-2008 compliant Floating-Point Unit. The floating-point pipeline
sits in conjunction with the integer pipeline; however, it is longer, increasing the total pipeline stages to 7.

- Memory System

The EIC7700X MCPU memory system has a Level 1 memory system optimized for high performance. The instruction subsystem consists of a 32 KiB, 4-way instruction cache. The data subsystem is comprised of a high performance 32 KiB, 4-way L1 data cache. In addition to the L1 memory system, the EIC7700X MCPU has a per hart 256 KiB, 8-way private L2 cache.

- Interrupts

The EIC7700X MCPU provides the standard RISC-V M-mode timer and software interrupts via the Core-Local Interruptor (CLINT). The EIC7700X MCPU also includes a RISC-V standard Platform-Level Interrupt Controller (PLIC), which supports 520 global interrupts with 7 priority levels pre-integrated with the on core-complex peripherals.

- Debug Support

The EIC7700X MCPU provides external debugger support over an industry-standard JTAG port, including 4 hardware-programmable breakpoints per hart.

- Compliance

The EIC7700X MCPU is compliant to the following versions of the various RISC-V specifications:

Table 3-4 RISC-V Specification Compliance

ISA	Version	Status

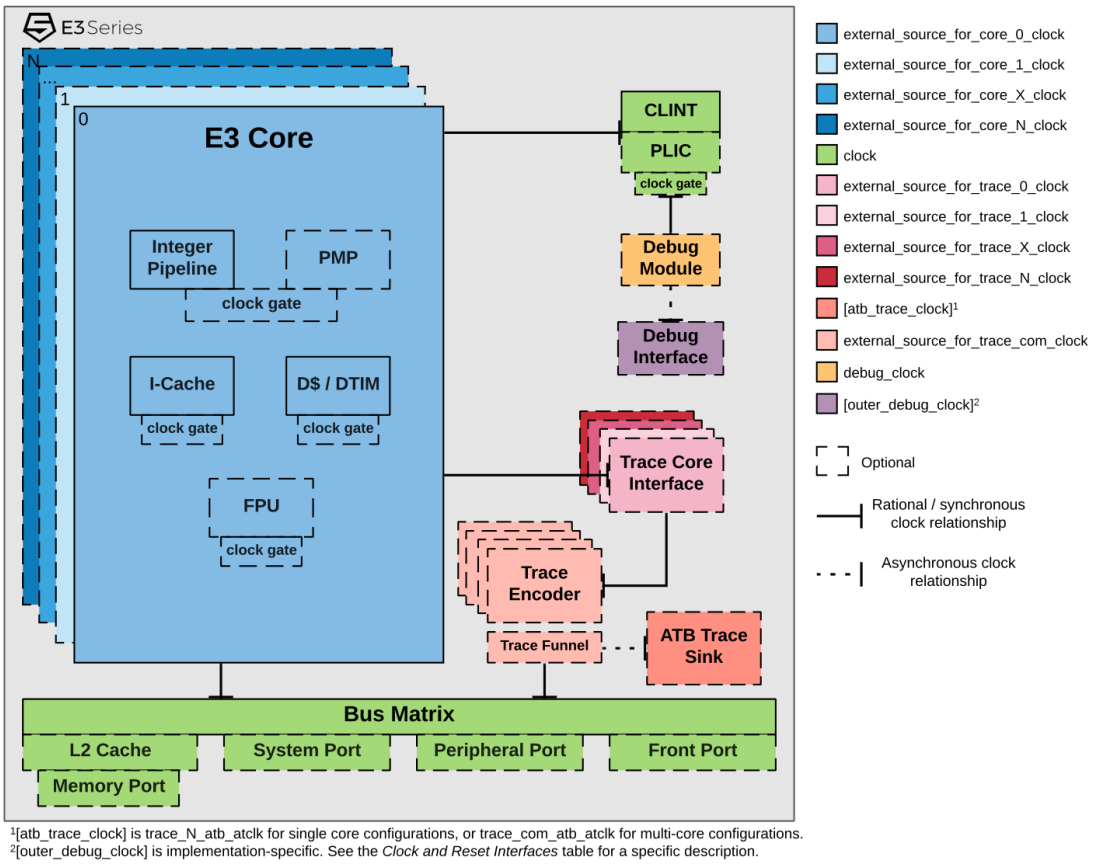
RV64I Base Integer Instruction Set	2.1	Ratified
Extensions	**Version**	**Status**
M Standard Extension for Integer Multiplication and Division	2.0	Ratified
A Standard Extension for Atomic Instruction	2.1	Ratified
F Standard Extension for Single-Precision Floating-Point	2.2	Ratified
D Standard Extension for Double-Precision Floating-Point	2.2	Ratified
C Standard Extension for Compressed Instruction	2.0	Ratified
Zicsr Standard Extension for Control and Status Register (CSR) Instructions	2.0	Ratified
Zifencei Standard Extension for Instruction-Fetch Fence	2.0	Ratified
Zba Standard Extension for Address Calculation	1.0	Ratified
Zbb Standard Extension for Basic Bit Manipulation	1.0	Ratified
Sscofpmf Standard Extension for Count Overflow and Mode-Based Filtering	0.1	Ratified
Privilege Mode	**Version**	**Status**
Machine-Level ISA	1.11	Ratified
User-Level ISA	1.11	Ratified
Supervisor-Level ISA	1.11	Ratified
Hypervisor-Level ISA	0.6	Frozen
Devices	**Version**	**Status**
The RISC-V Debug Specification	1.0	Frozen
RISC-V Platform-Level Interrupt Controller (PLIC) Specification	—	—

3.3.2 Co-processor

3.3.2.1 Overview

EIC7700X has two co-processors, both located in the aon subsystem, one for security boot(SCPU) and one for low Power control (LPCPU). The two Co-processors have same features,they are 32-bit RISC-V processors which is compatible with all applicable RISC-V standards.

The Co-processor block diagram is show as below.



[1][atb_trace_clock] is trace_N_atb_atclk for single core configurations, or trace_com_atb_atclk for multi-core configurations.
[2][outer_debug_clock] is implementation-specific. See the *Clock and Reset Interfaces* table for a specific description.

Figure 3-5 E31 block diagram

3.3.2.2 Feature

The Co-processor feature set is summarized in Table 3-5

Table 3-5 Co-processor feature set

Feature	Description
ISA	RV32IMAC
SiFive Custom Instruction Extension (SCIE)	Present
Modes	Machine mode, user mode
L1 Instruction Cache	64 KiB 8-way instruction cache
Instruction Tightly-Integrated Memory (ITIM)	Shared with instruction cache (max. 56 KiB)
Data Tightly-Integrated Memory (DTIM)	64 KiB DTIM
Physical Memory Protection	8 regions with a granularity of 4 bytes.

3.3.2.2.1 Supported modes

The Co-processor supports RISC-V user mode, providing two levels of privilege: machine (M) and user (U). U-mode provides a mechanism to isolate application processes from each other and from trusted code running in M-mode. See The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.10 for more information on the privilege modes.

3.3.2.2.2 execution pipeline

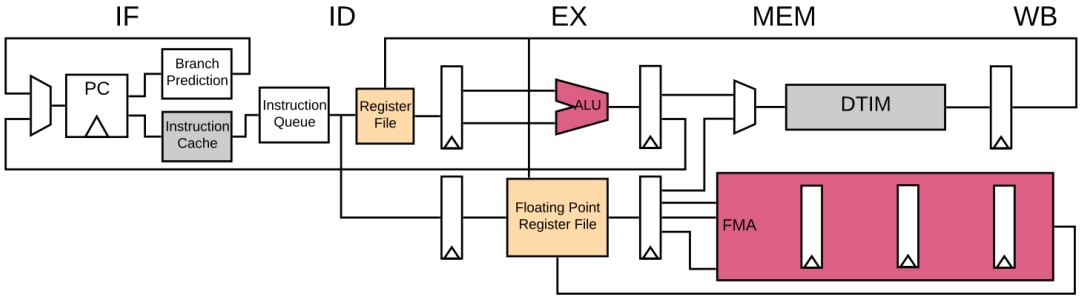


Figure 3-6 e31 pipeline

The E3 execution unit is a single-issue, in-order pipeline. The pipeline comprises five stages: instruction fetch (IF), instruction decode and register fetch (ID), execute (EX), data memory access (MEM), and register write-back (WB).

The pipeline has a peak execution rate of one instruction per clock cycle, and is fully bypassed such that most instructions have a one-cycle result latency. There are several exceptions, noted in Table 3-6.

Table 3-6 pipeline latency

Instruction	Latency
LW	Two-cycle result latency, assuming cache hit
LH, LHU, LB, LBU	Three-cycle result latency, assuming cache hit
CSR reads	Three-cycle result latency
MUL, MULH, MULHU, MULHSU	Three-cycle result latency
DIV, DIVU, REM, REMU	Between three-cycle to 35-cycle result latency, depending on operand values[1]
[1]The latency of DIV, DIVU, REM, and REMU instructions can be determined by calculating: `Latency = 2 cycles + log2(dividend) – log2(divisor) + 1 cycle` if the input is negative + 1 cycle if the output is negative	

The pipeline has some register dependencies, where it interlocks on read-after-write and write-after-write hazards, so instructions may be scheduled to avoid stalls. Otherwise, the processor can have multiple outstanding memory-mapped I/O accesses, even to the same address.

The E3 implements the standard Multiply (M) extension to the RISC-V architecture for integer multiplication and division. The E3 has a 32 bit per cycle hardware multiply and a 1 bit per cycle hardware divide. The multiplier is fully pipelined and can begin a new operation on each cycle,with a maximum throughput of one operation per cycle.

The hart will not abandon a divide instruction in flight. This means if an interrupt handler tries to use a register that is the destination register of a divide instruction, the pipeline stalls until the divide is complete.

Branch and jump instructions transfer control from the memory access pipeline stage. Correctly-predicted branches and jumps incur no penalty, whereas mispredicted branches and jumps incur a three-cycle penalty.

Most CSR writes result in a pipeline flush with a five-cycle penalty, so the results of the CSR write are observed on the next instruction.

3.3.2.2.3 memory map

Physical Memory Attributes: R—Read, W—Write, X—Execute, I—Instruction Cacheable, D—Data Cacheable, A—Atomics

table 3-7 Co-processor memory map

Base	Top	PMA	Description
0x0000_0000	0x0000_0FFF		Debug
0x0000_1000	0x0000_2FFF		Reserved
0x0000_3000	0x0000_3FFF	RWX A	Error Device
0x0000_4000	0x016F_FFFF		Reserved
0x0170_0000	0x0170_0FFF	RW A	Bus-Error Unit
0x0170_1000	0x017F_FFFF		Reserved
0x0180_0000	0x0180_FFFF	RWX A	ITIM
0x0181_0000	0x01FF_FFFF		Reserved
0x0200_0000	0x0200_FFFF	RW A	CLINT
0x0201_0000	0x0BFF_FFFF		Reserved
0x0C00_0000	0x0C3F_FFFF	RW A	PLIC
0x0C40_0000	0x1FFF_FFFF		Reserved
0x2000_0000	0x2FFF_FFFF	RWXI A	Peripheral Port (256 MiB)
0x3000_0000	0x3000_FFFF	RWX A	DTIM (64 KiB)
0x3001_0000	0x3FFF_FFFF		Reserved
0x4000_0000	0x7FFF_FFFF	RWXI	System Port (1 GiB)
0x8000_0000	0xFFFF_FFFF	RWXI	System Port (2 GiB)

3.3.2.2.4 Instruction Memory System

The regions of executable memory consist of all directly addressable memory in the system. The memory includes any volatile or non-volatile memory located off the Core Complex ports, and includes the on-core-complex DTIM and ITIM.

Table 3-8 excutable regions

Base	Top	Description
0x0180_0000	Up to 0x0180_FFFF	ITIM (optional)
0x2000_0000	0x2FFF_FFFF	Peripheral Port (256 MiB)
0x3000_0000	0x3000_FFFF	DTIM (64 KiB)
0x4000_0000	0x7FFF_FFFF	System Port (1 GiB)
0x8000_0000	0xFFFF_FFFF	System Port (2 GiB)

All executable regions, except the ITIM, are treated as instruction cacheable. There is no method to disable this behavior.

The ITIM is an optional region that repurposes a portion of the instruction cache, as described in Section 3.2.4.

Trying to execute an instruction from a non-executable address results in an instruction access trap.

3.3.2.2.5 Instruction Cache Reconfigurability

The instruction cache can be partially reconfigured into Instruction Tightly-Integrated Memory (ITIM), which occupies a fixed address range in the memory map. ITIM provides high-performance, predictable instruction delivery. Fetching an instruction from ITIM is as fast as an instruction cache hit, with no possibility of a cache miss. ITIM can hold data as well as instruct-tions, though loads and stores from a core to its ITIM are not as performant as loads and stores to its Data Tightly Integrated Memory (DTIM).

The ITIM region in the E31 memory map is represented by a fixed address range that includes both the maximum range that can be allocated to ITIM, or the ITIM Mem region; as well as the remaining region that must be reserved as instruction cache, or the ITIM Ctrl region.

The instruction cache can be configured as ITIM starting from address 0x0180_0000, in units of cache lines (64 bytes) up to a maximum size of 56 KiB, ending in address 0x0180_DFFF. A single instruction cache way, 8 KiB, must remain an instruction cache. The ITIM is allocated simply by writing to it. A store

to the n th byte of the ITIM memory map reallocates the first n+1 bytes of instruction cache as ITIM, rounded up to the next cache block. For determinism, software must clear the contents of ITIM after allocating it.

ITIM is deallocated by storing zero to the first byte after the maximum ITIM region, address 0x0180_E000. The deallocated ITIM space is automatically returned to the instruction cache. Returned cache lines are invalidated. It is unpredictable whether ITIM contents are preserved between deallocation and allocation.

A hart executing in user mode can reconfigure the cache. If this is not desired, then the Physical Memory Protection unit can be used to prevent writes to the ITIM region. Reads to the ITIM Mem region that are not allocated to the ITIM return 0x0.

Reads to the Ctrl region return unspecified data and are guaranteed not to have any side-effects. Writes to the Ctrl region beyond 0x0180_E000 have unspecified behavior and should be avoided.

3.3.2.2.6 Data Memory System

The data memory system consists of on-core-complex data and the ports in the E31 memory map, shown in Section 4.2. The on-core-complex data memory consists of a 64 KiB Data Tightly-Integrated Memory (DTIM). A design cannot have both data cache and DTIM, and the data cache is not reconfigurable like the instruction cache.

As no data cache is present, all data accesses are non-cacheable. Data accesses that are not targeted at the DTIM are also called memory-mapped I/O accesses, or MMIOs.

The E3 pipeline allows for multiple outstanding memory accesses. No store buffers are utilized in the Core Complex The number of outstanding MMIOs are implementation dependent. Mis- aligned accesses are not allowed to any memory region and result in a trap to allow for software emulation.

The DTIM provides deterministic access time, which is important for applications with hard realtime requirements. The access latency is two clock cycles for words and double-words, and three clock cycles for smaller quantities.

Stores are pipelined and commit on cycles where the data memory system is otherwise idle. Loads to addresses currently in the store pipeline result in a five-cycle penalty.

The DTIM region can be used to store instructions, but it has no lasting performance advantage over other memory regions. Fetching from the DTIM first results in an instruction cache line fill and execution occurs from the instruction cache.

The DTIM supports the RISC-V standard Atomic (A) extension, with the exception of the LR/SC instructions that are only supported in data cacheable regions. See Chapter 5 for more information on the instructions added by this extension.

3.4 Interrupt System

Table 3-9 RISC-V mcpu interrupt list

Interrupt source	Interrupt ID	Interrupt name	polarity
DDR0	5	ddr0_pmp0_intr	High level
	6	ddr0_pmp1_intr	High level
	7	ddr0_pmp2_intr	High level
	8	ddr0_pmp3_intr	High level
	9	ddr0_pmp4_intr	High level
	10	ddr0_ecc_intr	High level

Interrupt source	Interrupt ID	Interrupt name	polarity
	11	ddr0_derate_intr	High level
	12	ddr0_poison_intr	High level
	13	ddr0_phy_intr	High level
GPU	14	gpu_misc_intr	High level
	15	gpu_core_intr0	High level
NPU	16	npu_intr	High level
	17	npu_tbu_ras_intr	High level
	18	npu_tbu_pmu_intr	High level
Video IN	19	isp0_mi_intr	High level
	20	isp0_fe_intr	High level
	21	isp0_intr	High level
	22	isp1_mi_intr	High level
	23	isp1_fe_intr	High level
	24	isp1_intr	High level
	25	dwe_vse_intr	High level
	26	dwe_intr	High level
	27	dwe_fe_intr	High level
	28	vi_cmn_intr	High level
	29	mipi_csi0_intr	High level
	30	mipi_csi1_intr	High level
	31	mipi_csi2_intr	High level
	32	mipi_csi3_intr	High level
	33	mipi_csi4_intr	High level
	34	mipi_csi5_intr	High level
	35	vicap0_u84_intr	High level
	36	vicap1_u84_intr	High level
	37	vicap2_u84_intr	High level
	38	vicap3_u84_intr	High level
	39	vicap4_u84_intr	High level
	40	vicap5_u84_intr	High level
	41~56	Reserved	
DMA	57	hsp_dma0_u84_intr	High level
SATA	58	sata_intrq	High level
	59	sata_msi_req	High level
	60	sata_pme_req	High level
GMAC	61	eth0_sbd_intr	High level

Interrupt source	Interrupt ID	Interrupt name	polarity
	62	eth0_sbd_perch_tx_intr0	High level
	63	eth0_sbd_perch_tx_intr1	High level
	64	eth0_sbd_perch_tx_intr2	High level
	65	eth0_sbd_perch_tx_intr3	High level
	66	eth0_sbd_perch_rx_intr0	High level
	67	eth0_sbd_perch_rx_intr1	High level
	68	eth0_sbd_perch_rx_intr2	High level
	69	eth0_sbd_perch_rx_intr3	High level
	70	eth1_sbd_intr	High level
	71	eth1_sbd_perch_tx_intr0	High level
	72	eth1_sbd_perch_tx_intr1	High level
	73	eth1_sbd_perch_tx_intr2	High level
	74	eth1_sbd_perch_tx_intr3	High level
	75	eth1_sbd_perch_rx_intr0	High level
	76	eth1_sbd_perch_rx_intr1	High level
	77	eth1_sbd_perch_rx_intr2	High level
	78	eth1_sbd_perch_rx_intr3	High level
EMMC	79	emmc0_intr	High level
	80	emmc0_wakeup_intr	High level
SDIO	81	sd0_intr	High level
	82	sd0_wakeup_intr	High level
	83	sd1_intr	High level
	84	sd1_wakeup_intr	High level
USB	85	usb0_intr	High level
	86	usb1_intr	High level
WatchDog	87	lsp_wdt0_intr	High level
	88	lsp_wdt1_intr	High level
	89	lsp_wdt2_intr	High level
	90	lsp_wdt3_intr	High level
SPI	91	lsp_ssi0_intr	High level
	92	lsp_ssi1_intr	High level
PWM	93	lsp_timer_intr0	High level
	94	lsp_timer_intr1	High level
	95	lsp_timer_intr2	High level
	96~99	Reserved	High level
UART	100	lsp_uart0_intr	High level

Interrupt source	Interrupt ID	Interrupt name	polarity
	101	lsp_uart1_intr	High level
	102	lsp_uart2_intr	High level
	103	lsp_uart3_intr	High level
	104	lsp_uart4_intr	High level
I2C	105	lsp_i2c0_intr	High level
	106	lsp_i2c1_intr	High level
	107	lsp_i2c2_intr	High level
	108	lsp_i2c3_intr	High level
	109	lsp_i2c4_intr	High level
	110	lsp_i2c5_intr	High level
	111	lsp_i2c6_intr	High level
	112	lsp_i2c7_intr	High level
	113	lsp_i2c8_intr	High level
	114	lsp_i2c9_intr	High level
	115	Reserved	High level
Mailbox	116	u84_mbox0_intr	High level
	117	u84_mbox1_intr	High level
	118	u84_mbox2_intr	High level
	119	u84_mbox3_intr	High level
	120	u84_mbox4_intr	High level
	121	u84_mbox5_intr	High level
	122	u84_mbox6_intr	High level
	123	u84_mbox7_intr	High level
	124	u84_mbox8_intr	High level
	125	u84_mbox9_intr	High level
	126	u84_mbox10_intr	High level
	127	u84_mbox11_intr	High level
	128	u84_mbox12_intr	High level
	129	u84_mbox13_intr	High level
	130	u84_mbox14_intr	High level
	131	u84_mbox15_intr	High level
PCIe	132	pcie_dtim_err_tresp_oas_int	High level
	133	pcie_dtim_err_tresp_uid_int	High level
	134	pcie_dtim_err_icpl_uc_int	High level
	135	pcie_dtim_err_ireq_opertn_int	High level
	136	pcie_dtim_err_ireq_to_int	High level

Interrupt source	Interrupt ID	Interrupt name	polarity
	137	pcie_dtim_int	High level
	138	pcie_dtim_ireq_timeout_int	High level
	139~159	Reserved	High level
	160	pcie_hp_pme	High level
	161	pcie_hp_int	High level
	162	pcie_hp_msi	High level
	163	pcie_edma_int0	High level
	164	pcie_edma_int1	High level
	165	pcie_edma_int2	High level
	166	pcie_edma_int3	High level
	167	pcie_edma_int4	High level
	168	pcie_edma_int5	High level
	169	pcie_edma_int6	High level
	170	pcie_edma_int7	High level
	171	pcie_edma_int8	High level
	172	pcie_edma_int9	High level
	173	pcie_edma_int10	High level
	174	pcie_edma_int11	High level
	175	pcie_edma_int12	High level
	176	pcie_edma_int13	High level
	177	pcie_edma_int14	High level
	178	pcie_edma_int15	High level
	179	pcie_assert_inta	High level
	180	pcie_assert_intb	High level
	181	pcie_assert_intc	High level
	182	pcie_assert_intd	High level
	183	pcie_deassert_inta	High level
	184	pcie_deassert_intb	High level
	185	pcie_deassert_intc	High level
	186	pcie_deassert_intd	High level
	187	pcie_usp_eq_redo_executed_int	High level
	188~219	Reserved	High level
	220	pcie_msi_ctrl_int	High level
Video codec	221	vd_tbu_ras_intr	High level
	222	vd_tbu_pmu_intr	High level
	223	jd_tbu_ras_intr	High level

Interrupt source	Interrupt ID	Interrupt name	polarity
	224	jd_tbu_pmu_intr	High level
	225	ve_tbu_ras_intr	High level
	226	ve_tbu_pmu_intr	High level
	227	je_tbu_ras_intr	High level
	228	je_tbu_pmu_intr	High level
	229	ve_intr	High level
	230	ve_intr_norm	High level
	231	ve_intr_abn	High level
	232	je_intr	High level
	233	je_intr_norm	High level
	234	je_intr_abn	High level
	235	Reserved	High level
	236	vd_intr_vcd	High level
	237	jd_intr	High level
OSD	238	osd_intr	High level
	239~270	Reserved	High level
I2S	271	vo_i2s0_intr	High level
	272	vo_i2s1_intr	High level
	273	vo_i2s2_intr	High level
HDMI	274	hdmi_intr	High level
	275	hdcp_intr	High level
MIPI_DSI	276	dsi_intr	High level
MMU	277	tbu0_ras_intr	High level
	278	tbu0_pmu_intr	High level
	279	tbu2_ras_intr	High level
	280	tbu2_pmu_intr	High level
	281	tbu3_ras_intr	High level
	282	tbu3_pmu_intr	High level
	283	tbu4_ras_intr	High level
	284	tbu4_pmu_intr	High level
	285~288	Reserved	High level
TOP	289	aon_dma_intr	High level
	290	aon_i2c0_intr	High level
	291	aon_i2c1_intr	High level
	292	rtc_intr	High level
	293	aon_spic_intr	High level

Interrupt source	Interrupt ID	Interrupt name	polarity
DDR1	294	ddr1_pmp0_intr	High level
	295	ddr1_pmp1_intr	High level
	296	ddr1_pmp2_intr	High level
	297	ddr1_pmp3_intr	High level
	298	ddr1_pmp4_intr	High level
	299	ddr1_ecc_intr	High level
	300	ddr1_derate_intr	High level
	301	ddr1_poison_intr	High level
	302	ddr1_phy_intr	High level
GPIO	303	aon_u84_gpio_intr0	High level
	304	aon_u84_gpio_intr1	High level
	305	aon_u84_gpio_intr2	High level
	306	aon_u84_gpio_intr3	High level
	307	aon_u84_gpio_intr4	High level
	308	aon_u84_gpio_intr5	High level
	309	aon_u84_gpio_intr6	High level
	310	aon_u84_gpio_intr7	High level
	311	clmm_u84_gpio_intr0	High level
	312	clmm_u84_gpio_intr1	High level
	313	clmm_u84_gpio_intr2	High level
	314	clmm_u84_gpio_intr3	High level
	315	clmm_u84_gpio_intr4	High level
	316	clmm_u84_gpio_intr5	High level
	317	clmm_u84_gpio_intr6	High level
	318	clmm_u84_gpio_intr7	High level
	319	clmm_u84_gpio_intr8	High level
	320	clmm_u84_gpio_intr9	High level
	321	clmm_u84_gpio_intr10	High level
	322	clmm_u84_gpio_intr11	High level
	323	clmm_u84_gpio_intr12	High level
	324	clmm_u84_gpio_intr13	High level
	325	clmm_u84_gpio_intr14	High level
	326	clmm_u84_gpio_intr15	High level
	327	clmm_u84_gpio_intr16	High level
	328	clmm_u84_gpio_intr17	High level
	329	clmm_u84_gpio_intr18	High level

Interrupt source	Interrupt ID	Interrupt name	polarity
	330	clmm_u84_gpio_intr19	High level
	331	clmm_u84_gpio_intr20	High level
	332	clmm_u84_gpio_intr21	High level
	333	clmm_u84_gpio_intr22	High level
	334	clmm_u84_gpio_intr23	High level
NPU	335	npu_u84_llc0_c_intr	High level
	236	npu_u84_llc0_event_intr	High level
	237	npu_u84_llc0_uc_intr	High level
	238	npu_u84_llc1_c_intr	High level
	239	npu_u84_llc1_event_intr	High level
	240	npu_u84_llc1_uc_intr	High level
	341~342	Reserved	High level
Video In	343	shutter_u84_intr	High level
	344	Reserved	High level
TOP	345	aon_u84_timers_intr0	High level
	346	aon_u84_timers_intr1	High level
	347	aon_u84_timers_intr2	High level
	348	aon_u84_timers_intr3	High level
PVT	349	lsp_u84_pvt_intr0	High level
DDR0	350	lpddr_u84_pvt_intr	High level
	351~353	Reserved	High level
FAN_CTRL	354	lsp_u84_fanctrl_intr	High level
MMU	355	tcu_u84_event_q_irpt_s	High level
	356	tcu_u84_event_q_irpt_ns	High level
	357	tcu_u84_pri_q_irpt_ns	High level
	358	tcu_u84_cmd_sync_irpt_ns	High level
	359	tcu_u84_cmd_sync_irpt_s	High level
	360	tcu_u84_global_irpt_ns	High level
	361	tcu_u84_global_irpt_s	High level
	362	tcu_u84_ras_irpt	High level
	363	tcu_u84_pmu_irpt	High level
	359~367	Reserved	High level
Video IN	368	dvp2axi_u84_intr0	High level
	369	dvp2axi_u84_intr1	High level
	370	dvp2axi_u84_intr2	High level
	371	dvp2axi_u84_intr3	High level

Interrupt source	Interrupt ID	Interrupt name	polarity
	372	dvp2axi_u84_intr4	High level
	373	dvp2axi_u84_intr5	High level
	374	dvp2axi_u84_intr6	High level
	375	dvp2axi_u84_intr7	High level
PCIe	376	pciet_u84_hot_reset_int	High level
	377	pciet_u84_flr_int	High level
Video Out	378	vo_u84_cec_intr	High level
	379~386	Reserved	
NPU	387	npu_u84_nvdla_intr	High level
PCIe	388	pciet_u84_radm_fatal_err_int	High level
	389	pciet_u84_radm_nonfatal_err_int	High level
	390	pciet_u84_radm_correctable_err_int	High level
	391	Reserved	High level
NOC	392	cnoc_vo_timeout	High level
	393	cnoc_vi_timeout	High level
	394	cnoc_vc_timeout	High level
	395	cnoc_tcu_timeout	High level
	396	cnoc_pciet_x_timeout	High level
	397	cnoc_pciet_p_timeout	High level
	398	cnoc_npu_timeout	High level
	399	cnoc_mcput_d2d_timeout	High level
	400	cnoc_lsp_apb6_timeout	High level
	401	cnoc_lsp_apb4_timeout	High level
	402	cnoc_lsp_apb3_timeout	High level
	403	cnoc_lsp_apb2_timeout	High level
	404	cnoc_hsp_timeout	High level
	405	cnoc_gpu_timeout	High level
	406	cnoc_dspt_timeout	High level
	407	cnoc_ddrt1_phy_timeout	High level
	408	cnoc_ddrt1_ctrl_timeout	High level
	409	cnoc_ddrt0_phy_timeout	High level
	410	cnoc_ddrt0_ctrl_timeout	High level
	411	cnoc_aon_timeout	High level
	412	clmm_timeout	High level
	413	rnoc_ddrt1_p4_timeout	High level
	414	rnoc_ddrt0_p4_timeout	High level

Interrupt source	Interrupt ID	Interrupt name	polarity
	415	mnoc_ddr1_p3_timeout	High level
	416	mnoc_ddr0_p3_timeout	High level
	417	snoc_pciet_timeout	High level
	418	snoc_npu_timeout	High level
	419	snoc_dspt_timeout	High level
	420	snoc_ddrt1_p2_timeout	High level
	421	snoc_ddrt1_p1_timeout	High level
	422	snoc_ddrt0_p2_timeout	High level
	423	snoc_ddrt0_p1_timeout	High level
	424	snoc_aon_timeout	High level
	425	lnoc_ddrt1_p0_timeout	High level
	426	lnoc_ddrt0_p0_timeout	High level
PCIe	427	pcie_wake_n_in	High level
NOC	428	sys_trans2_statalarm	High level
	429	sys_trans1_statalarm	High level
	430	sys_trans0_statalarm	High level
	431	sys_observer_fault	High level
	432	Reserved	High level
	433	ddrt1_p2_reqpkt_statalarm	High level
	434	Reserved	High level
	435	ddrt1_p1_reqpkt_statalarm	High level
	436	Reserved	High level
	437	ddrt0_p2_reqpkt_statalarm	High level
	438	Reserved	High level
	439	ddrt0_p1_reqpkt_statalarm	High level
	440	llc_trans_statalarm	High level
	441	llc_observer_fault	High level
	442	Reserved	High level
	443	ddrt1_p0_reqpkt_statalarm	High level
	444	Reserved	High level
	445	ddrt0_p0_reqpkt_statalarm	High level
	446	cfg_observer_fault	High level
	447	realtime_trans_statalarm	High level
	448	realtime_observer_fault	High level
	449	Reserved	High level
	450	ddrt1_p4_reqpkt_statalarm	High level

Interrupt source	Interrupt ID	Interrupt name	polarity
	451	Reserved	High level
	452	ddrt0_p4_reqpkt_statalarm	High level
	453	media_trans_statalarm	High level
	454	media_observer_fault	High level
	455	Reserved	High level
	456	ddrt1_p3_reqpkt_statalarm	High level
	457	Reserved	High level
	458	ddrt0_p3_reqpkt_statalarm	High level

Note: Interrupt ID 0~4 of mcpu corresponds are reserved.

Table 3-10 SCPU interrupt list

Interrupt source	Interrupt ID	Interrupt name	Polarity
TOP	0	trng_edu_intr	High level
	1	trng_intr	High level
	2	pka_intr	High level
	3	spacc_intr	High level
Maibox	4	e21_0_mbox0_intr	High level
	5	e21_0_mbox1_intr	High level
	6	e21_0_mbox2_intr	High level
	7	e21_0_mbox3_intr	High level
	8	e21_0_mbox4_intr	High level
	9	e21_0_mbox5_intr	High level
	10	e21_0_mbox6_intr	High level
	11	e21_0_mbox7_intr	High level
	12	e21_0_mbox8_intr	High level
	13	e21_0_mbox9_intr	High level
	14	e21_0_mbox10_intr	High level
	15	e21_0_mbox11_intr	High level
	16	e21_0_mbox12_intr	High level
	17	e21_0_mbox13_intr	High level
	18	e21_0_mbox14_intr	High level
	19	e21_0_mbox15_intr	High level
DDR1	20	ddr1_pmp0_intr	High level
	21	ddr1_pmp1_intr	High level
	22	ddr1_pmp2_intr	High level
	23	ddr1_pmp3_intr	High level

Interrupt source	Interrupt ID	Interrupt name	Polarity
	24	ddr1_pmp4_intr	High level
DDR0	25	ddr0_pmp0_intr	High level
	26	ddr0_pmp1_intr	High level
	27	ddr0_pmp2_intr	High level
	28	ddr0_pmp3_intr	High level
	29	ddr0_pmp4_intr	High level
TOP	30	bootspi_int	High level
Timer	31	aon_timer0_intr_flag[0]	High level
	32	aon_timer0_intr_flag[1]	High level
	33	aon_timer0_intr_flag[2]	High level
	34	aon_timer0_intr_flag[3]	High level
	35	aon_timer0_intr_flag[4]	High level
	36	aon_timer0_intr_flag[5]	High level
	37	aon_timer0_intr_flag[6]	High level
	38	aon_timer0_intr_flag[7]	High level
	39	aon_timer1_intr_flag[0]	High level
	40	aon_timer1_intr_flag[1]	High level
	41	aon_timer1_intr_flag[2]	High level
	42	aon_timer1_intr_flag[3]	High level
	43	aon_timer1_intr_flag[4]	High level
	44	aon_timer1_intr_flag[5]	High level
	45	aon_timer1_intr_flag[6]	High level
	46	aon_timer1_intr_flag[7]	High level
	47	aon_timer2_intr_flag[0]	High level
	48	aon_timer2_intr_flag[1]	High level
	49	aon_timer2_intr_flag[2]	High level
	50	aon_timer2_intr_flag[3]	High level
	51	aon_timer2_intr_flag[4]	High level
	52	aon_timer2_intr_flag[5]	High level
	53	aon_timer2_intr_flag[6]	High level
	54	aon_timer2_intr_flag[7]	High level
	55	aon_timer3_intr_flag[0]	High level
	56	aon_timer3_intr_flag[1]	High level
	57	aon_timer3_intr_flag[2]	High level
	58	aon_timer3_intr_flag[3]	High level

Interrupt source	Interrupt ID	Interrupt name	Polarity
	59	aon_timer3_intr_flag[4]	High level
	60	aon_timer3_intr_flag[5]	High level
	61	aon_timer3_intr_flag[6]	High level
	62	aon_timer3_intr_flag[7]	High level
TOP	63	otp_int	High level
	64	hblock_accvio	High level
	65	edu_vtrng_irq	High level
	66	scpu_rtc_intr	High level

Table 3-11 LPCPU interrupt list

Interrupt source	Interrupt ID	Interrupt name	Polarity
TOP	0	bootspi_int	High level
	1	rtc_int	High level
	2	aon_i2c1_intr	High level
	3	aon_i2c0_intr	High level
	4	aon_dma_intr	High level
Maibox	5	lpcpu_mbox0_intr	High level
	6	lpcpu_mbox1_intr	High level
	7	lpcpu_mbox2_intr	High level
	8	lpcpu_mbox3_intr	High level
	9	lpcpu_mbox4_intr	High level
	10	lpcpu_mbox5_intr	High level
	11	lpcpu_mbox6_intr	High level
	12	lpcpu_mbox7_intr	High level
	13	lpcpu_mbox8_intr	High level
	14	lpcpu_mbox9_intr	High level
	15	lpcpu_mbox10_intr	High level
	16	lpcpu_mbox11_intr	High level
	17	lpcpu_mbox12_intr	High level
	18	lpcpu_mbox13_intr	High level
	19	lpcpu_mbox14_intr	High level
	20	lpcpu_mbox15_intr	High level
	21	Reserved	High level
Timer	22	aon_timer0_intr_flag[0]	High level
	23	aon_timer0_intr_flag[1]	High level
	24	aon_timer0_intr_flag[2]	High level

Interrupt source	Interrupt ID	Interrupt name	Polarity
	25	aon_timer0_intr_flag[3]	High level
	26	aon_timer0_intr_flag[4]	High level
	27	aon_timer0_intr_flag[5]	High level
	28	aon_timer0_intr_flag[6]	High level
	29	aon_timer0_intr_flag[7]	High level
	30	aon_timer1_intr_flag[0]	High level
	31	aon_timer1_intr_flag[1]	High level
	32	aon_timer1_intr_flag[2]	High level
	33	aon_timer1_intr_flag[3]	High level
	34	aon_timer1_intr_flag[4]	High level
	35	aon_timer1_intr_flag[5]	High level
	36	aon_timer1_intr_flag[6]	High level
	37	aon_timer1_intr_flag[7]	High level
	38	aon_timer2_intr_flag[0]	High level
	39	aon_timer2_intr_flag[1]	High level
	40	aon_timer2_intr_flag[2]	High level
	41	aon_timer2_intr_flag[3]	High level
	42	aon_timer2_intr_flag[4]	High level
	43	aon_timer2_intr_flag[5]	High level
	44	aon_timer2_intr_flag[6]	High level
	45	aon_timer2_intr_flag[7]	High level
	46	aon_timer3_intr_flag[0]	High level
	47	aon_timer3_intr_flag[1]	High level
	48	aon_timer3_intr_flag[2]	High level
	49	aon_timer3_intr_flag[3]	High level
	50	aon_timer3_intr_flag[4]	High level
	51	aon_timer3_intr_flag[5]	High level
	52	aon_timer3_intr_flag[6]	High level
	53	aon_timer3_intr_flag[7]	High level
GPIO	54	lpcpu_gpio_intr0	High level
	55	lpcpu_gpio_intr1	High level
	56	lpcpu_gpio_intr2	High level
	57	lpcpu_gpio_intr3	High level
	58	lpcpu_gpio_intr4	High level
	59	lpcpu_gpio_intr5	High level

Interrupt source	Interrupt ID	Interrupt name	Polarity
	60	lpcpu_gpio_intr6	High level
	61	lpcpu_gpio_intr7	High level

3.5 System address mapping

The total physical address space of EIC7700X system is 2T. The following is the address space mapping diagram, from left to right, which is the global addr map, system space, and config space. The internal SRAM and TIM in the system are mapped in the low address space within 4G and the high address space above 4G. The Mem space is divided into coherency space and non-coherency space. The memory port address segment corresponds to the coherency address space, and the system port1 address segment corresponds to the non-coherency address space.

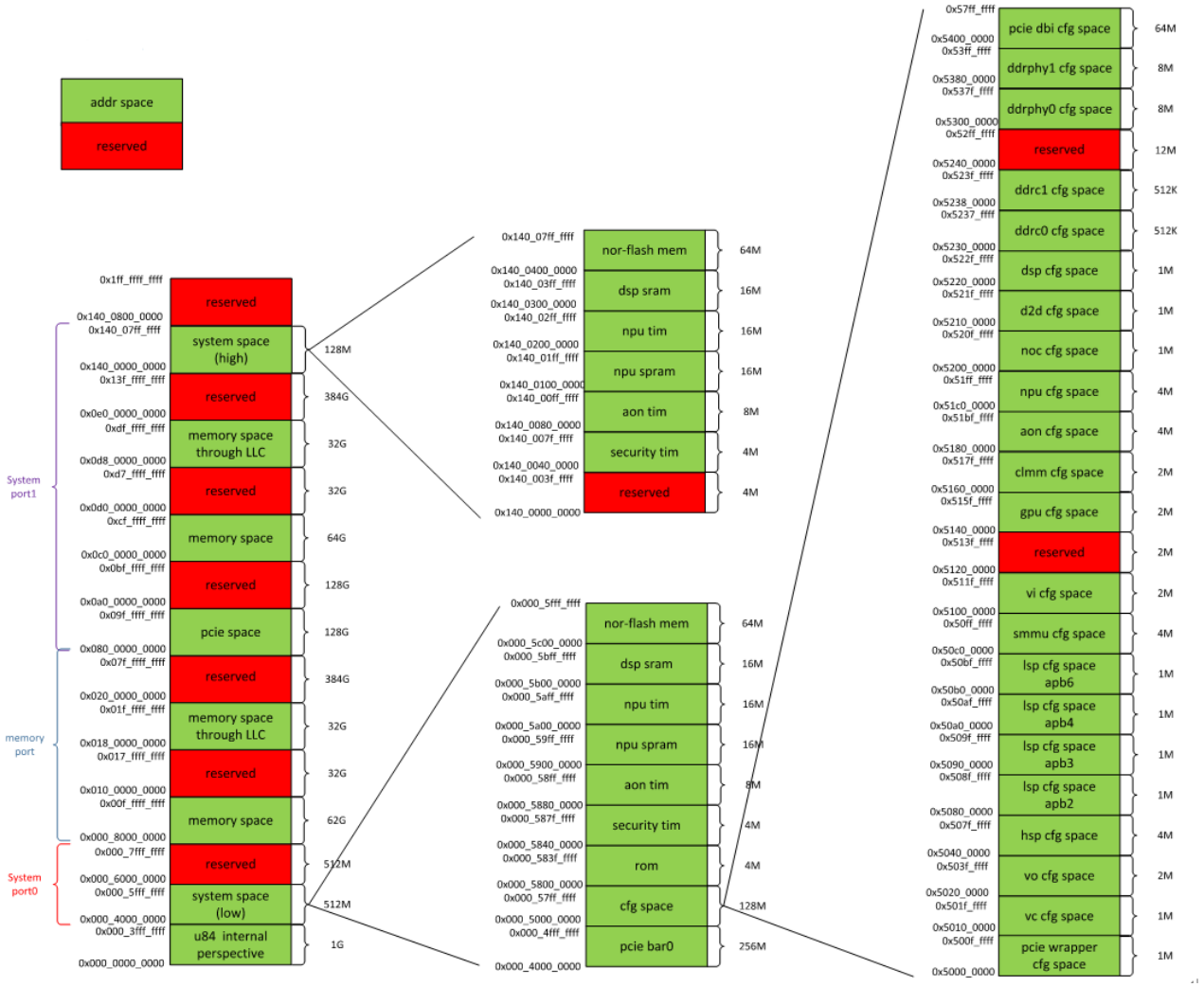


Figure 3-7 Address mapping

3.6 Interconnect

3.6.1 Overview

The EIC7700X system interconnect enables data interaction between various subsystems and DDR, SRAM, and Config Space.

The following are the main features supported by EIC7700X NoC：

- Supporting the standard amba protocol as the interface protocol between noc and generic ip. The related amba protocols are axi4, AXI4-Lite, ahb3, apb3, and apb4;
- Supporting the NSP protocol as an interface protocol for cascading NOCs;
- Use topology structures such as cross bar and distributed interconnection nodes;
- Support data network, each IP through the data network access memory, peripherals, register space;
- Supports monitoring networks, with monitoring points mainly including three types: transaction probe point, packet probe point, and error probe point. Through various monitoring points, latency, bandwidth, and error information are monitored;
- Support for a service network, which enables read and write access to the internal register space of the Noc through the service network;
- Support qos generator, including three modes: fixed, limiter, regulator;
- The Noc internal arbiter supports two levels of arbitration. The first level of arbitration is based on urgency, which can be converted through socket QoS maps or obtained through QoS generators. The second level of arbitration is based on age information;
- Support watch dog, monitor the response time of slave, report timeout interrupt to the timeout host, and generate response err to ensure the integrity of the response;
- Support for the sideband manager, which allows the user to query the idle status of each interface in the noc by querying the sideband manager;
- Support default error target. Accessing the error address issued by the host will be routed to the corresponding default error target, and the default err target will return the response err;
- Little-Endian；

3.6.2 Block diagram

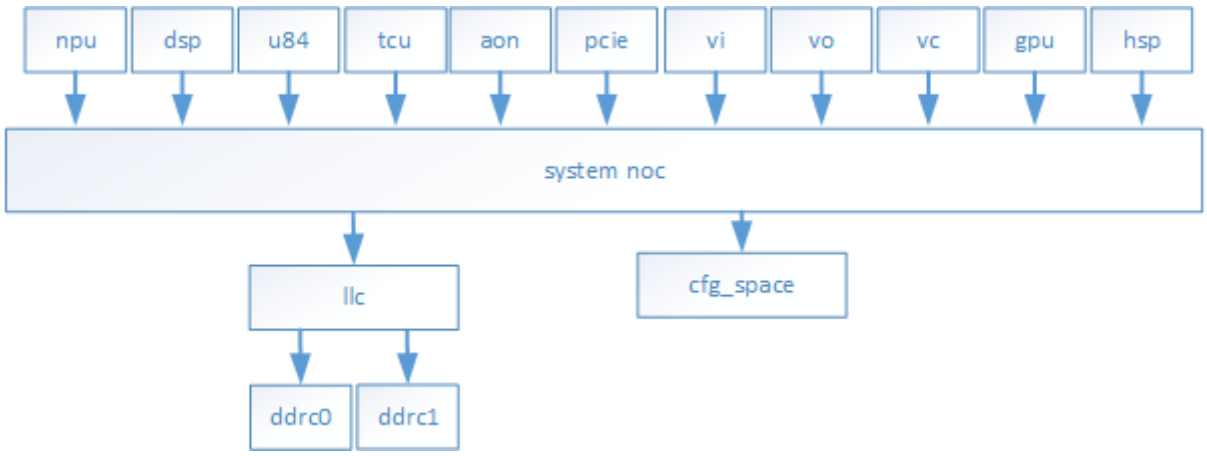


Figure 3-8 NoC interconnect diagram

3.7 DMA Controller

3.7.1 Overview

EIC7700X contains two DMA controllers for data interaction between memory and memory, and between memory and peripherals. After receiving the corresponding request, DMA starts the controller according to the system's configuration of the channel, sends the address and control signal to the memory or peripherals, and reports the transmission status to the system through the interrupt.

DMA0 supports the following features:

- Supports 12 programmable channels;channel1/2 fifo depth 256, used for m2m data transfer; channel3~12 fifo depth 64, used for m2p and p2m data transfer
- Programmable channel priority
- Programmable channel burst length
- Includes 16 group DMA request, interact with 8 peripherals
- Programmable multi-block transfer(llinked list/auto reload/shadow register/contiguous address)
- Supports one 32bit Master Interface
- Supports 8bit/16bit/32bit data bus transfer
- Supports channel locking

DMA0 request connection is shown in the following table.

Table 3-12 DMA0 Req

NUM	DMA_REQ	DMA_SINGLE	DESCRIPTION
0	uart1_dma_rx	uart1_dma_rx_single	valid when scu_lsp_uart0_dma_sel=0
1	uart1_dma_tx	uart1_dma_tx_single	valid when scu_lsp_uart0_dma_sel=0
2	i2c1_dma_rx	i2c1_dma_rx_single	valid when scu_lsp_i2c1_dma_sel=0
3	i2c1_dma_tx	i2c1_dma_tx_single	valid when scu_lsp_i2c1_dma_sel=0
4	i2c0_dma_rx	i2c0_dma_rx_single	valid when scu_lsp_i2c0_dma_sel=0
5	i2c0_dma_tx	i2c0_dma_tx_single	valid when scu_lsp_i2c0_dma_sel=0
6	ssi2_dma_rx	ssi2_dma_rx_single	valid when scu_lsp_ssi1_dma_sel=0
7	ssi2_dma_tx	ssi2_dma_tx_single	valid when scu_lsp_ssi1_dma_sel=0
8	ssi1_dma_rx	ssi1_dma_rx_single	valid when scu_lsp_ssi0_dma_sel=0
9	ssi1_dma_tx	ssi1_dma_tx_single	valid when scu_lsp_ssi0_dma_sel=0
10	i2s2_dma_rx	i2s2_dma_rx_single	valid when scu_vo_i2s2_dma_sel=0
11	i2s2_dma_tx	i2s2_dma_tx_single	valid when scu_vo_i2s2_dma_sel=0
12	i2s1_dma_rx	i2s1_dma_rx_single	valid when scu_vo_i2s1_dma_sel=0
13	i2s1_dma_tx	i2s1_dma_tx_single	valid when scu_vo_i2s1_dma_sel=0
14	i2s0_dma_rx	i2s0_dma_rx_single	valid when scu_vo_i2s0_dma_sel=0
15	i2s0_dma_tx	i2s0_dma_tx_single	valid when scu_vo_i2s0_dma_sel=0

DMA1 supports the following features:

- Supports 16 programmable channels;channel1/2 fifo depth 256, used for m2m data transfer; channel3~16 fifo depth 64, used for m2p and p2m data transfer
- Programmable channel priority
- Programmable channel burst length
- Includes 45 DMA request, interact with 23 peripherals
- Programmable multi-block transfer(llinked list/auto reload/shadow register/contiguous address)
- Supports two 64bit Master Interface
- Supports 8bit/16bit/32/64bit data bus transfer
- Supports channel locking

DMA1 request connection is shown in the following table.

Table 3-13 DMA1 req

NUM	DMA_REQ	DMA_SINGLE	DESCRIPTION
0	i2s2_dma_rx	i2s2_dma_rx_single	valid when scu_vo_i2s2_dma_sel=1
1	i2s2_dma_tx	i2s2_dma_tx_single	valid when scu_vo_i2s2_dma_sel=1
2	i2s1_dma_rx	i2s1_dma_rx_single	valid when scu_vo_i2s1_dma_sel=1
3	i2s1_dma_tx	i2s1_dma_tx_single	valid when scu_vo_i2s1_dma_sel=1
4	i2s0_dma_rx	i2s0_dma_rx_single	valid when scu_vo_i2s0_dma_sel=1
5	i2s0_dma_tx	i2s0_dma_tx_single	valid when scu_vo_i2s0_dma_sel=1
6	i2c9_dma_rx	i2c9_dma_rx_single	always valid
7	i2c9_dma_tx	i2c9_dma_tx_single	always valid
8	i2c8_dma_rx	i2c8_dma_rx_single	always valid
9	i2c8_dma_tx	i2c8_dma_tx_single	always valid
10	i2c7_dma_rx	i2c7_dma_rx_single	always valid
11	i2c7_dma_tx	i2c7_dma_tx_single	always valid
12	i2c6_dma_rx	i2c6_dma_rx_single	always valid
13	i2c6_dma_tx	i2c6_dma_tx_single	always valid
14	i2c5_dma_rx	i2c5_dma_rx_single	always valid
15	i2c5_dma_tx	i2c5_dma_tx_single	always valid
16	i2c4_dma_rx	i2c4_dma_rx_single	always valid
17	i2c4_dma_tx	i2c4_dma_tx_single	always valid
18	i2c3_dma_rx	i2c3_dma_rx_single	always valid
19	i2c3_dma_tx	i2c3_dma_tx_single	always valid
20	i2c2_dma_rx	i2c2_dma_rx_single	always valid
21	i2c2_dma_tx	i2c2_dma_tx_single	always valid
22	i2c1_dma_rx	i2c1_dma_rx_single	valid when scu_lsp_i2c1_dma_sel=1
23	i2c1_dma_tx	i2c1_dma_tx_single	valid when scu_lsp_i2c1_dma_sel=1
24	i2c0_dma_rx	i2c0_dma_rx_single	valid when scu_lsp_i2c0_dma_sel=1
25	i2c0_dma_tx	i2c0_dma_tx_single	valid when scu_lsp_i2c0_dma_sel=1
26	uart4_dma_rx	uart4_dma_rx_single	always valid
27	uart4_dma_tx	uart4_dma_tx_single	always valid
28	uart3_dma_rx	uart3_dma_rx_single	always valid
29	uart3_dma_tx	uart3_dma_tx_single	always valid

NUM	DMA_REQ	DMA_SINGLE	DESCRIPTION
30	uart2_dma_rx	uart2_dma_rx_single	always valid
31	uart2_dma_tx	uart2_dma_tx_single	always valid
32	uart1_dma_rx	uart1_dma_rx_single	valid when scu_lsp_uart0_dma_sel=1
33	uart1_dma_tx	uart1_dma_tx_single	valid when scu_lsp_uart0_dma_sel=1
34	uart0_dma_rx	uart0_dma_rx_single	always valid
35	uart0_dma_tx	uart0_dma_tx_single	always valid
36	ssi1_dma_rx	ssi1_dma_rx_single	valid when scu_lsp_ssi1_dma_sel=1
37	ssi1_dma_tx	ssi1_dma_tx_single	valid when scu_lsp_ssi1_dma_sel=1
38	ssi0_dma_rx	ssi0_dma_rx_single	valid when scu_lsp_ssi0_dma_sel=1
39	ssi0_dma_tx	ssi0_dma_tx_single	valid when scu_lsp_ssi0_dma_sel=1
40	-	bootspi_dma_req	always valid
41	aon_i2c0_dma_rx	aon_i2c0_dma_rx_single	always valid
42	aon_i2c0_dma_tx	aon_i2c0_dma_tx_single	always valid
43	aon_i2c1_dma_rx	aon_i2c1_dma_rx_single	always valid
44	aon_i2c1_dma_tx	aon_i2c1_dma_tx_single	always valid

3.7.2 Block Diagram

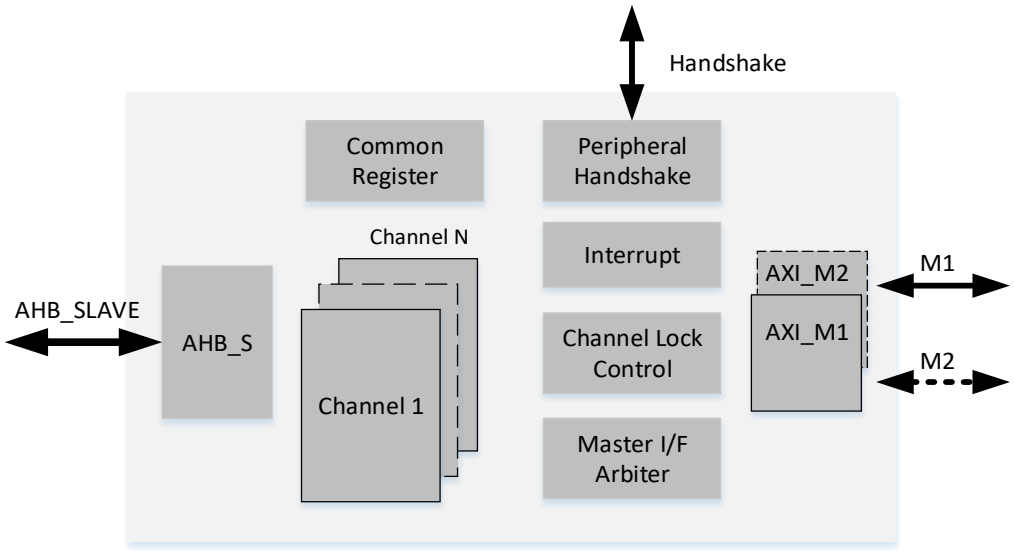


Figure 3-9 DMA Diagram

DMA Controller interact with system through axi bus, main blocks are shown as the following:

- Channel Contains the specific implementation of each channel to complete the data transmission of the system configuration
- Interrupt block generate interrupt according to the transmission status and inform CPU to process
- Peripheral Handshake handles the request of peripherals

- Channel Lock allows a channel to be locked to the arbiter
- Arbiter handles requests of all channels according to the configuration

3.7.3 Function Description

- Clock Gating
  Close DMA clock through system register
- Soft Reset
  Assert DMA reset through system register
- Flow for linked-list-based multi-block data transfer

  1.Software reads the DMAC channel enable register (DMAC_ChEnReg) to select an available (unused) channel.

  2. Software programs the CHx_CFG register with appropriate values for the DMA transfer.The SRC_MLTBLK_TYPE and/or DST_MLTBLK_TYPE bits must be set to 2'b11.

  3. Software programs the base address of the first linked list item and the master interface on which the linked list item is available in the CHx_LLP register.

  4. Software creates one or more linked list items in system memory. Software can create the entire linked list item in advance or dynamically extend the linked list using the CHx_CTL.ShadowReg_Or_LLI_Valid and CHx_CTL.LLI_Last fields of the LLI.

  5. Software enables the channel by writing 1 to the appropriate bit location in DMAC_ChEnReg register.

  6.dmac initiates the DMA block transfer operation based on the settings for the block transfer.The block transfer might start immediately or after the hardware or software handshaking request,depending on the settings of the TT_FC field in the CHx_CFG register. dmac copies the linked list contents to the registers used for executing the DMA block transfer (that is, the CHx_SAR and/or CHx_DAR, CHx_BLOCK_TS, and CHx_CTL registers) and initiates the DMA block transfer.

  7.During the linked list fetch phase:(a) if the current block is the final block in the transfer and completes the DMA transfer operation at the end of current block transfer (b)if there are one or more blocks to be transferred and goes to step 6 (c)if CHx_CTL.ShadowReg_Or_LLI_Valid bit of the fetched LLI as 0, dmac might generate ShadowReg_Or_LLI_Invalid_ERR Interrupt. dmac waits till software writes (any value) to CHx_BLK_TFR_ResumeReqReg to indicate valid LLI availability, before attempting another LLI read operation.

3.8 SMMU

3.8.1 Overview

This is a System-level Memory Management Unit (SMMU) that translates an input address to an output address. The SMMU can perform:

- support ARM SMMU v3.1
- Support configurable stream id, each AXI masters can have its dedicated stream id to have independent address space, and they can also have same stream id to share page table
- support 4KB, 16KB, 64KB translation granule size
- Adderss space covered by each entry in each level of page table, depending on translation granule size.
- Support configurable TBU priority
- support configurable AXI qos, higher qos for read page table is recommend.
- support global and per stream id bypass, address of bypassed transaction will never translated.
- Support prefetch: no prefetch, forward, backward.

3.8.2 Block diagram

AXI masters send AXI request to SMMU, then SMMU translate virtual address to physical address. If the needed page table is found in TLB cache, TBU translate the address with page table in the TLB cache. Otherwise, TBU send a page table request to tcu and translate with page table get from TCU. TCU checks its cache just like the TBU, but request page table from DDR while cache missed.

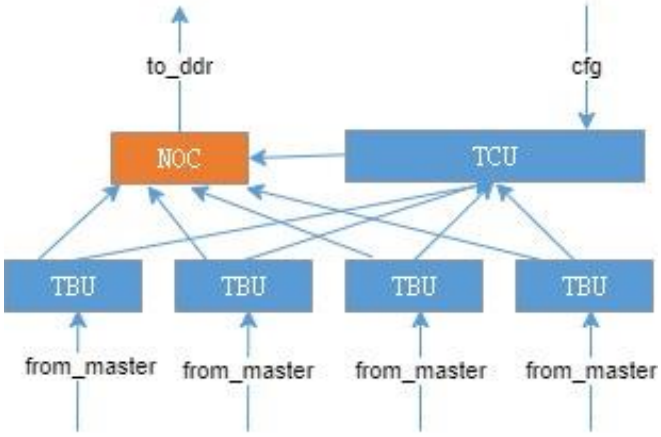


Figure 3-10 block diagram

3.8.3 Firmware flow

SMMU initialization:

1. allocate base address for command queue
2. allocate base address for event queue
3. configure stream table
4. initialize command queue
5. initialize event queue
6. Invalidate TLB and cache
7. configure context descriptor
8. configure stream table entry

Enable smmu：

1. set smmu enable bit in register
2. polling smmu status, until smmu enable request accepted

Disable TBU：

1. send power down request to the TBU to be disabled
2. polling power down status, until power down request accepted

Note: registers for disabling TBU are located in low power register of SOC not in smmu. Diable TBU should be performed before reset TBU, and release power down request before reset relase.

3.9 Mailbox

3.9.1 Function description

Mailbox is an interrupt module in the multi-core SOC system, because there may be different services between the cores, so the hardware design and allocation of one or two interrupts can no longer meet the needs of the system, and the software is also very difficult to expand, so mailbox can be understood as an interrupt module freely defined by the software.

There are MCPUS, SPU, NPU and DSP in the system, each module can generate interrupts for other modules through Mailbox, and after receiving the data, the module will carry out interrupt processing, and after completing the interrupt processing, the corresponding interrupt will be cleared for the next processing.

Mailbox hais following features:

- Interrupt registers, which are used to enable interrupts.
- Interrupt mask.
- FIFO data width is 64bit，fifo depth is 8.
- FIFO status register.
- Mailbox lock function.

3.9.2 Block diagram

The CPU accesses the mailbox through the APB bus, configures interrupt signals and interrupt enable signals, writes data to the FIFO, then generates an interrupt to the destination CPU after the data is written to the FIFO.The follow diagram is maibox function description.

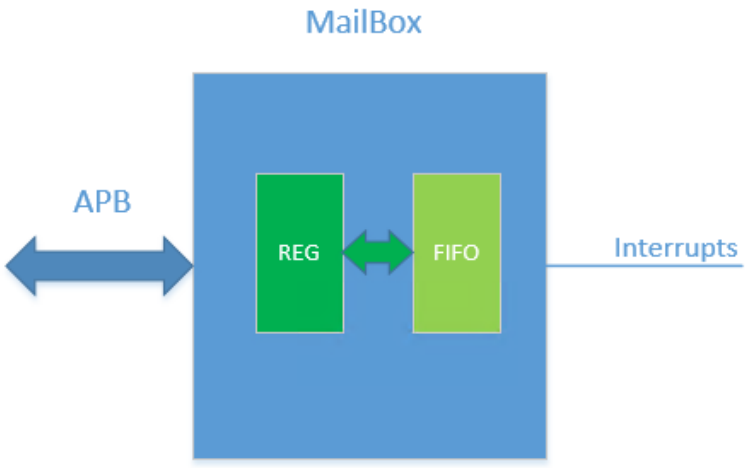


Figure 3-11 Mailbox block diagram

The system contains 16 mailboxes, before operating on the mailbox, you need to write WR_LOCK register, and then read the WR_LOCK register, judge whether the value of read and write is consistent, if the value of read and write is consistent, it means that the mailbox is not occupied by other devices and is in an idle state, otherwise the mailbox is occupied, the operation of the current mailbox should be stopped, and you can choose other mailbox to judge the state. If the MCPU needs to send an interrupt request to the SCPU, the MCPU first selects one of the mailboxes and sets the interrupt register bit1 to 1, which means that the interrupt is output to the first SCPU, and the data is written to the FIFO to generate an interrupt, so as to realize the interaction between the two CPU systems.

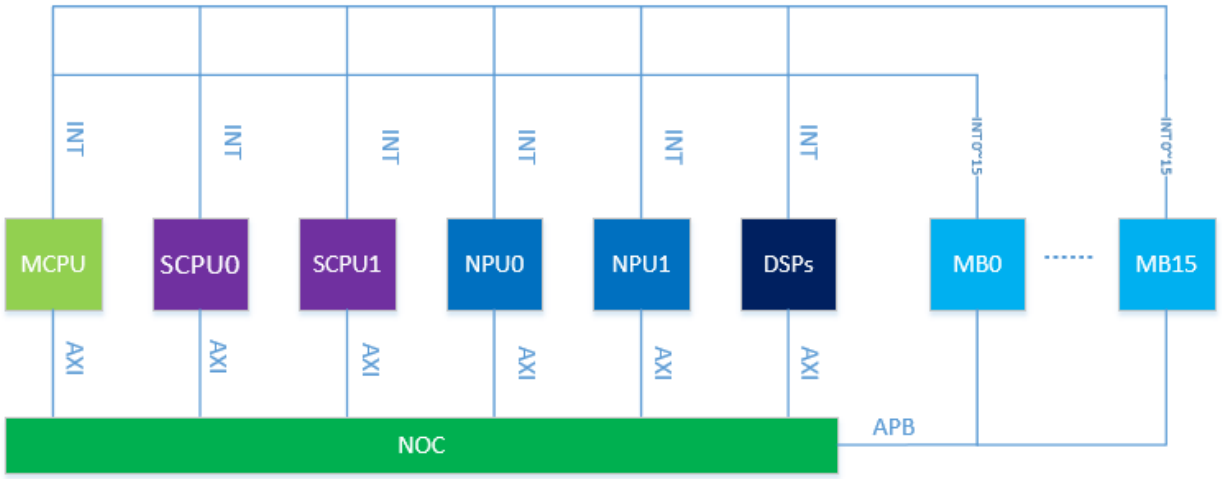


Figure 3-12 Interrupt interconnect

The following table describes the mapping between mailbox's interrupts and CPUs.

Table 3-14 Interrupt Mapping

Interrupt ID	CPU
INT0	MCPU
INT1	SCPU0
INT2	SCPU1
INT3	NPU0
INT4	NPU1

3.9.3 Initialization

Mailbox's Initialization as follow:

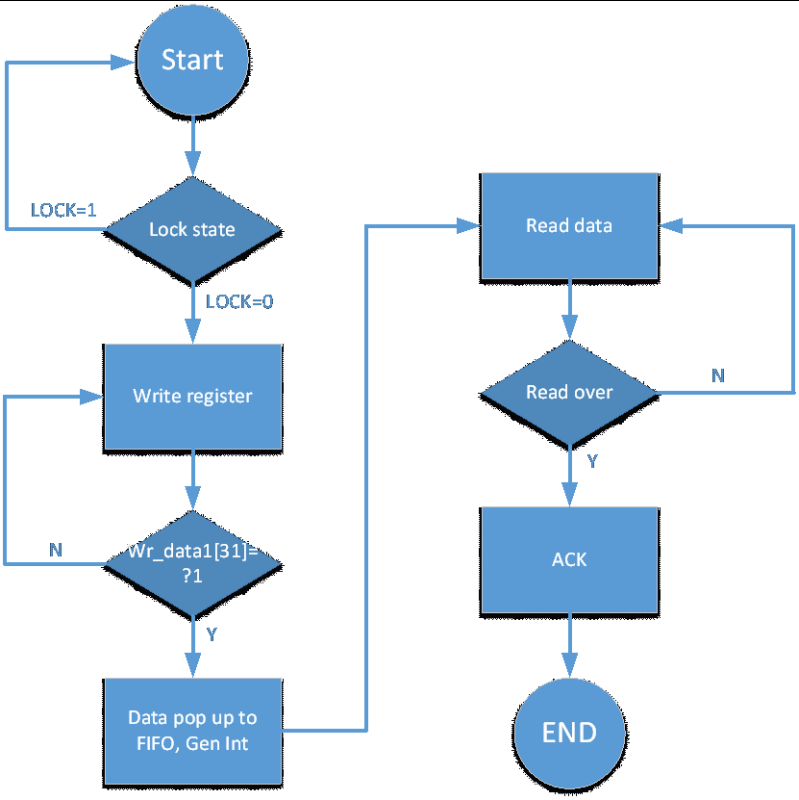


Figure 3-13 Flow of Mailbox Initial

3.10 WDT

3.10.1 Overview

Watchdog Timer (WDT) is an APB slave peripheral that can be used to prevent system lockup that caused by conflicting parts or programs in a SOC. When the counter reaches zero, depending on the output response mode selected, either a system reset or an interrupt occurs.

The following functions are supported：

- It supports APB 4.0 and has a bus width of 32 bits
- The counter bit width is 32 bits
- When system reset is released, it is diabled. The module is started according to the user's software configuration
- Counter counts down from a preset value to 0 to indicate the occurrence of a timeout
- If a timeout occurs the DW_apb_wdt can perform one of the following operations:
  - Generate a system reset
  - First generate an interrupt and even if it is cleared (or not) by the service routine by the time a
    second timeout occurs then generate a system reset.
- Programmable and hard coded reset pulse length
- Prevention of accidental restart of the WDT
- Prevention of accidental disabling of the WDT
- Outputs a high active interrupt

3.10.2 Block diagram

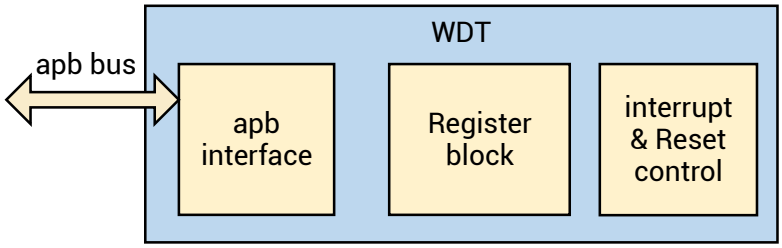


Figure 3-14 WDT block diagram

3.10.3 Function description

3.10.3.1 reset pulse length

The reset pulse width is the number of pclk cycles, which is determined by WDT_CR. RPL

3.10.3.2 Timeout Period Values

WDT has a fixed timeout period range，See Register WDT_TORR. TOP.

3.10.3.3 Counter Restart

As a safety feature to prevent accidental restarts, the value 0x76 must be written to the Current Counter Value Register (WDT_CRR).

3.10.4 Programming example

configured with the following configuration parameters:

- Single timeout period
- Response mode not hard coded
- Reset pulse length not hard coded
- WDT not always enabled
- Generates interrupt, and then system reset

3.11 RTC

1.1.1 Overview

The counter increments on successive positive edges of the input counter clock, rtc_clk. When the Counter Load Register (RTC_CLR) is programmed, the counter is loaded with a start value that allows the counter to increment.

3.11.1 block diagram

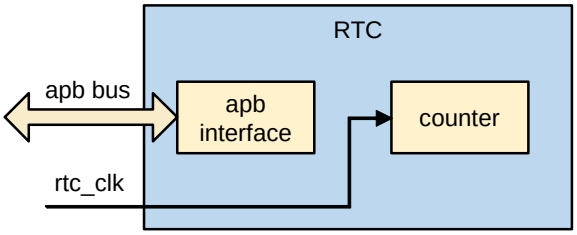


Figure 3-15 RTC block diagram

3.11.2 Function description

3.11.2.1 RTC Pre-scaler Counter

RTC supports preset counters, with the ability to update RTC counters at a rate lower than the RTC clock (rtc_clk). You can also enable/disable the RTC pre-scaler counter through the software using the rtc_psclr_en bit of the RTC_CCR register.

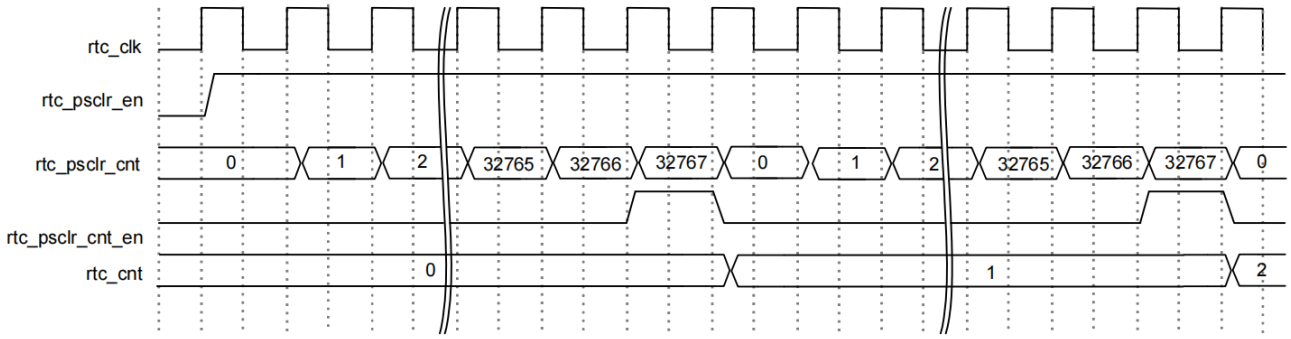


Figure 3-16 Timing Diagram for RTC Pre-scaler Counter with the RTC Counter

3.11.2.2 Match Register and Interrupt Generation

A match register, RTC_CMR, can be programmed and is compared to the internal counter. An interrupt is generated when the match register and the internal counter are equal, but only if interrupt generation is enabled (RTC_CCR, bit 0). This interrupt can be masked if the RTC_CCR register bit 1 is set to 1, which gives the user control of sending interrupts externally. The match register can be read any time.

The interrupt is kept active until it is cleared by an end-of-interrupt clear read access (RTC_EOI). The interrupt status bit is not polarity sensitive. The interrupt is active when the status is read as 1; otherwise, it is inactive. Programming the interrupt mask bit (rtc_mask) within the control register (RTC_CCR) masks the interrupt. The interrupt status can be read at any time. Even when the interrupt is masked, the raw interrupt status can be read.

3.11.3 Programming example

Make sure that the RTC clock reset is correctly configured, and then configure the RTC:

- Configure the initial count value，Counter_Load
- Configure Counter Match Register，Counter_Match
- Configure Pre-scaler Counter，RTC_CPSR
- Enable RTC，rtc_en

3.12 PVT

3.12.1 Overview

PVT sensors support the following features:

- Operating voltage 0.8V (digital), 1.8V(analog)；
- Operational temerature -40℃~125℃
- Temperature accuracy ±1℃（post triming），±6℃（untrimmed）
- Voltage accuracy（±3 sigma）±3%
- Sample rate 3000 samples/sec

3.12.2 Programming example

The software operation process is as follows:

- Confirm that the PVT clock is correct
- Configure REG_PVT_MODE to select the measurement mode
- Configure REG_PVT_ENA = 32'h1 to enable PVT
- Wait for the interrupt signal to be valid and read the REG_PVT_DATA
- Configure REG_PVT_INT = 32'h2 to clear interrupts

3.13 Low Power Scheme

3.13.1 Voltage Domain and Power Domain

3.13.1.1 Definition

The voltage domain defined as separate voltage supply from bump(external PMIC or component), the voltage are configured by external PMIC or component.

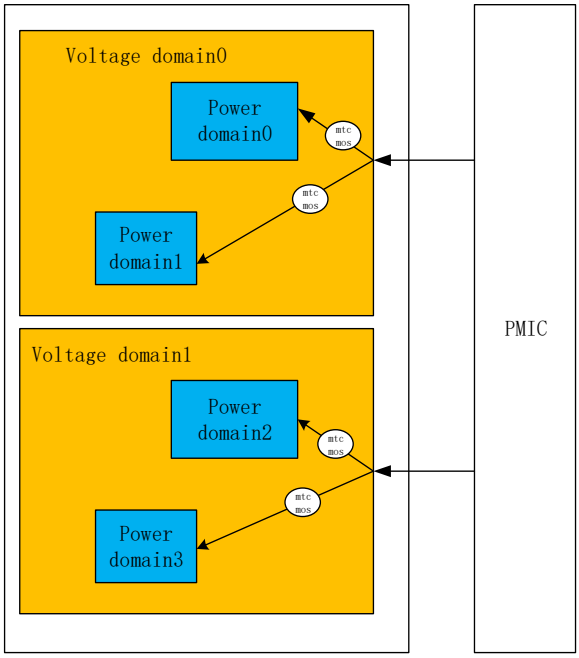


Figure 3-17 Voltage Domain and Power Domain

The power domain defined as independently controlled power on/off domain, normally it is inside one voltage domain. Power domain on/off normally controlled by internal power switch(MTCMOS).

3.13.2 Voltage Domain

Plan of the voltage domains as bellow.

- NPU has a separate 0.8v VDD_NPU. Supports step-up to 1.0v to meet 1GHz performance.
- CPU has a separate 0.8v VDD_CPU. Supports step-down to 0.75v.
- DDR has a separate 0.8v VDD_LPDDR. Supports step-up to 0.85v.
- Other digital logic share one 0.8v VDD_SOC.
- VDD_CPU, VDD_NPU and VDD_LPDDR's voltage can be changed outside DIE by PMIC(DC-DC).

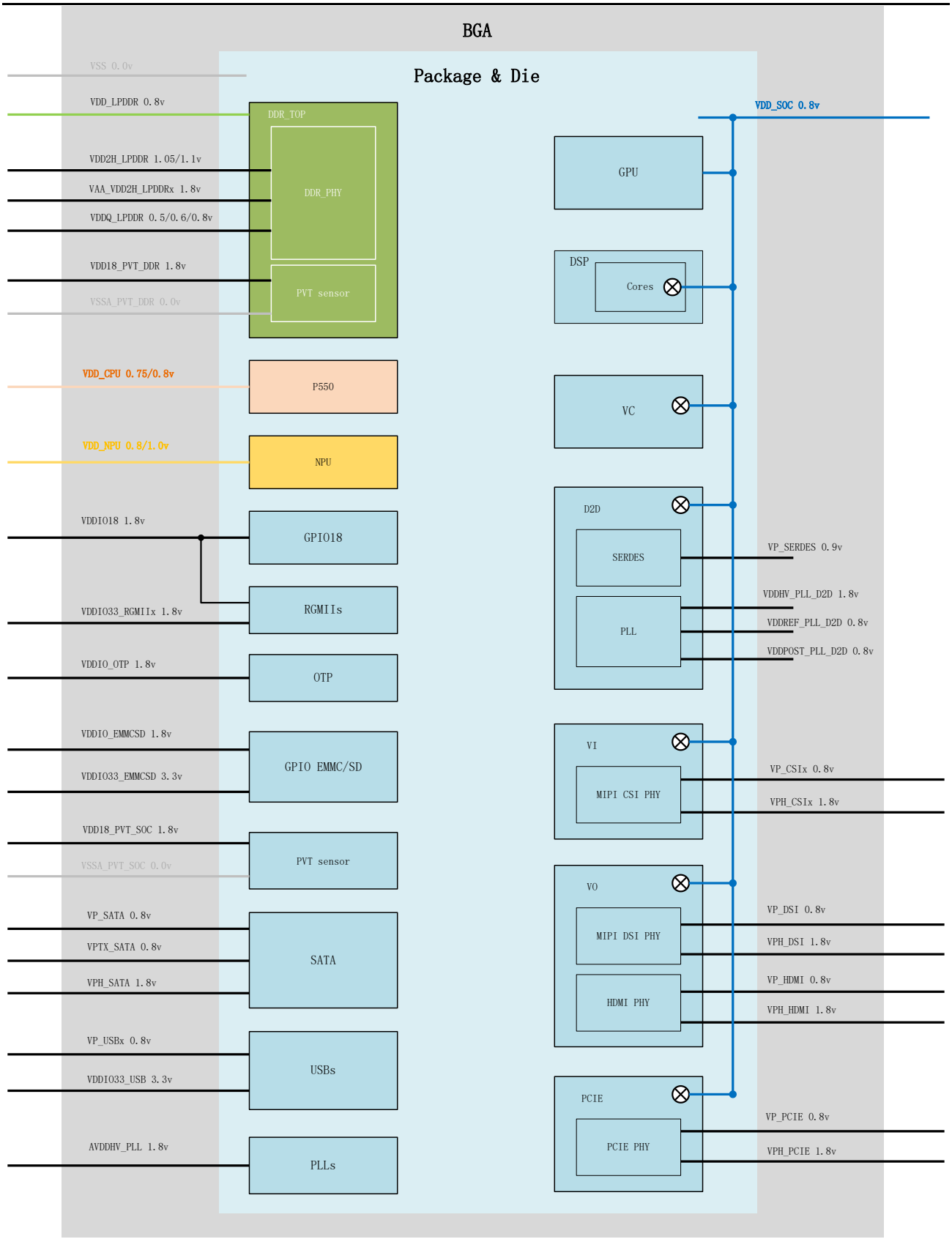


Figure 3-18 voltage domain and main power supplies

Please note that some power supplies are merged within the DIE, and some power supplies are merged in the package or board.

Combination of voltages for all supplies are shown as table below.

Table 3-15 combination of voltages

No.	VDD_SOC(V)	VDD_LPDDR	VDD_CPU(V)	VDD_NPU(V)	sign-off
1	0.8	0.8	0.8	0.8	Y
2	0.8	0.8	0.8	1.0	Y
3	0.8	0.8	0.75	0.8	Y
4	0.8	0.8	0.75	1.0	Y

3.13.3 Power Domain

The Power domain are defined mainly for VDD_NPU, VDD_CPU, VDD_LPDDR and VDD_SOC voltage domain. For other voltage domain, they are not power off, or can be partially power off by it's own control logic(such as PLL.etc)

3.13.3.1 Digital part power domain overview

As shown in the figure below, Isolation Cells, Levelshift Cells, and Enable-levelshift Cells are inserted between different power domains. VDD_SOC, VDD_CPU, and VDD_LPDDR are normally powered on and do not lose power during chip operation. The VDD_NPU power supply can be powered off by controlling the external power supply IC.

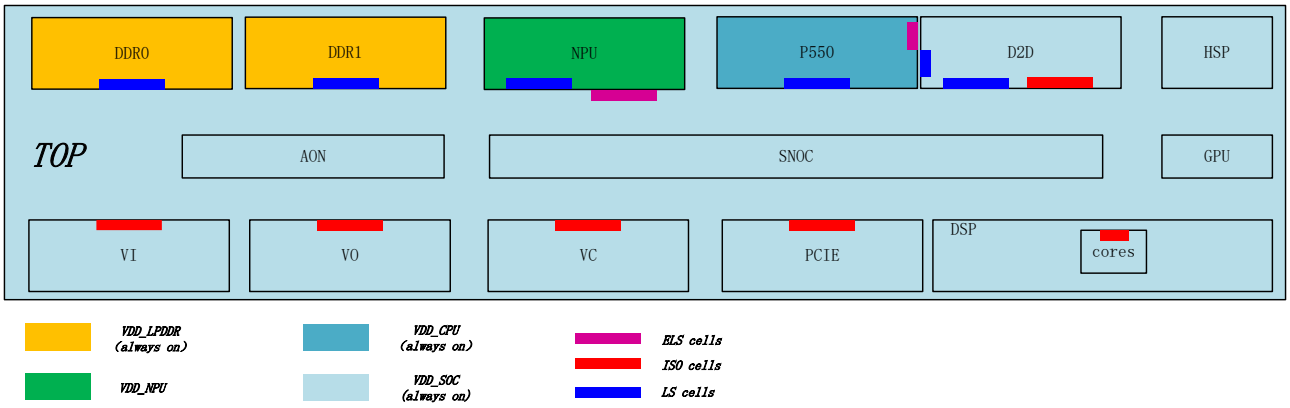


Figure 3-19 power domain and isolation/levelshift cells

4 Memory Interface

4.1 LPDDR

4.1.1 Overview

LPDDR memory system implements access control for 3 types of low-power DRAMs: LPDDR4，LPDDR4X and LPDDR5.

4.1.2 Features

DDR Controller：

- DDR controller supports 2 channels, each with 16-bit data width
- Support LPDDR4/4X/5 DRAM
- LPDDR4/4X supports maximum rate of 4267Mbps, and LPDDR5 supports maximum rate of 6400Mps
- Each DDR controller supports two AXI4 slave interfaces, one is connected to the NOC through PMP with a data bit width of 256 bits, the other is connected to the Trace master for collecting trace data with a data bit width of 64 bits

- The clock frequency ratio of the DDR controller core clock to the SDRAM clock is 1:4
- The DDR controller supports two ranks with two CS
- The AXI address can be mapped to row, column, bank, and rank by bit.
- Supports QOS, read transactions are divided into three levels (high priority, variable priority, low priority), write transactions are divided into two priority levels (normal priority, variable priority). Supports emergency and throttle control.
- Support calibration processor for training
- Support inline ECC

Memory PMP

- supports a maximum of 16 regions, region 0 covers all address space
- Region uses fixed priority, with priority decreasing from region15 to region0
- Each region is divided into 8 subregions on average, and each subregion can be disabled independently. When there is an overlap between regions, the higher priority region will be executed
- The security modes support 2 type: Standard and Exclusion. In the standard mode, security master can access non-security region. In the exclusion mode, security master can only access security region, any non-security access is denied.
- The minimum address range for each region is 32KB
- Read access supports speculative read. In speculative read mode, PMP is bypassed without the permission check
- Support secure_lock, preventing accidental operation
- Support illegal access interrupt

Trace master

- Collect transaction information from the DDR read command channel and write command channel, including ID, address, burst length, timestamp
- The collected information is buffered in the FIFO of the Trace Master, and after filling up one DDR burst size, the data will be written into the DDR SDRAM
- The collected data can be categorized by master or ID
- Trace Master is triggered by registers; it can be stopped by the time limit or automatically stopped after filling up the specified trace space.

4.1.3 Block Diagram

As shown in the figure below, the system includes two DDRs, a Memory PMP, and a Trace master. Two DDR controller system allocates a 64GB address space. The PMP is used to protect the DDR space and filter out non-secure accesses to the secure address. The Trace master extracts transaction access information from the signals between the PMP and DDR controller, and stores it inside the Trace master. After a certain number of transactions, the trace data is stored in the DDR SDRAM for subsequent recovery.

Each DDR controller and DDR PHY have an APB interface for configuring their registers. The DDR controller is connected to the DDR PHY through DFI 0 and DFI 1 interfaces. The DDR controller adopts a dual-channel architecture, with the controller outputting two DFI interfaces that connect to the DDR PHY. The interface signals between the two channels are mutually independent, with each channel having a data bit width of 16 bits.

The single-channel data interface of the DDR PHY has a 16-bit data bit width, while the dual-channel data interface has a 32-bit data bit width. Under the condition that the SDRAM clock frequency and controller core clock frequency are 4:1, the DFI data bit width is 8 times the SDRAM data bit width.

The Trace master is used to collect information related to read/write transactions accessing the AXI interface of the DDR controller for subsequent bandwidth analysis.

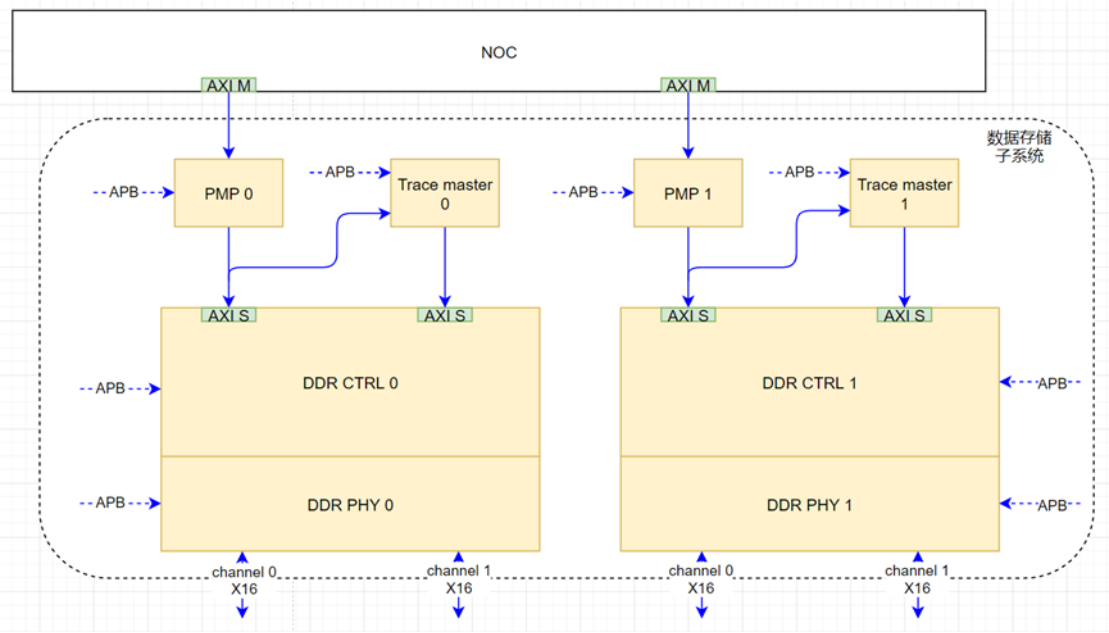


Figure 4-1 LPDDR subsystem block diagram

4.1.4 Initialization

4.1.4.1 DDR Controller Initialization

1. Assert the resets, configure the registers: ddr_rst_ctrl.sw_ddr_core_rstn and ddr_rst_ctrl.sw_ddr_prstn
2. Start the clocks (ddr_cfg_clk and ddr*_p*_aclk)
3. De-assert APB reset(ddr_rst_ctrl.sw_ddr_prstn) once the clocks are active and stable
4. Allow 128 APB clock cycles for synchronization of presetn to core_ddrc_core_clk and aclk domains and to permit initialization of end logic
5. Initialize the registers
6. Release the core resets(ddr_rst_ctrl.sw_ddr_core_rstn)

4.1.4.2 DDR PHY Initialization

1. Program the DWC_ddrctl registers, ensure that PHY is power up.
2. Disable auto-refreshes, self-refresh, powerdown and assertion of dfi_dram_clk_disable by setting RFSHCTL0.dis_auto_refresh = 1, PWRCTL.powerdown_en = 0 and PWRCTL.selfref_en = 0, PWRTL.en_dfi_dram_clk_disable = 0
3. release ddr_rst_ctrl.sw_ddr_core_rstn
4. Set DFIMISC.dfi_init_complete_en to '0' and Set DFIMISC.dfi_reset_n to '1'
5. Start PHY initialization and training by accessing relevant PUB registers
6. Poll the PUB register APBONLY.UctShadowRegs[0] =1'b0
7. Read the PUB Register APBONLY.UctWriteOnlyShadow for training status
8. Write the PUB Register APBONLY.DctWriteProt = 0
9. Poll the PUB register APBONLY.UctShadowRegs[0] =1'b1

10. Write the PUB Register APBONLY.DctWriteProt = 1
11. Poll the PUB register MASTER.CalBusy=0
12. Set DFIMISC.dfi_init_start to '1'
13. Poll DFISTAT.dfi_init_complete=1
14. Set DFIMISC.dfi_init_start to '0'
15. Updated registers: RANKTMG0.diff_rank_wr_gap, RANKTMG0.diff_rank_rd_gap, DRAMSET1TMG2.rd2wr, DRAMSET1TMG2.wr2rd
16. Set DFIMISC.dfi_init_complete_en to '1'
17. Set PWRCTL.selfref_sw to '0'
18. Wait for DWC_ddrctl to move to normal operating mode by monitoring STAT.operating_mode signal

Set back registers in step 2 to the original values if desired

4.1.4.3 Change Clock Frequencies with Frequency Set Point

1. During initialization, program the timing register-set REGB_FREQ(1/2/3)_CH, whichever you prefer, with the timing settings required for the alternative clock frequency.
2. Disable low-power activities by programming PWRCTL.selfref_en, PWRCTL.powerdown_en and HWLPCTL.hw_lp_en to '0'.
3. Ensure that STAT.operating_mode=1, which indicates that the controller is in normal mode. Continue to read this until it is read as '1' three times in a row.
4. Set PCTRL.port_en to '0' to block the AXI ports from taking the transactions. Poll PSTAT.rd_port_busy_n=0 and PSTAT.wr_port_busy_n=0. Wait until all AXI ports are idle.
5. Set OPCTRL1.dis_hif to '1', so that no new commands are accepted by the controller and poll
6. If DERATECTL0.derate_eanble is '1', then set DERATECTL0.derate_mr4_pause_fc to '1' to pause
7. If DFIUPD0.dis_auto_ctrlupd=0, then set DFIUPD0.dis_auto_ctrlupd to '1' to disable periodic
8. Set OPCTRLCMD.ctrlupd to '1' and poll OPCTRLSTAT.ctrlupd_busy=0 to issue DFI update request
9. If DFI PHY Master interface is active in the controller (DFIPHYMSTR.dfi_phymstr_en=1), then
10. Disable DQS Oscillator if it is enabled, by setting DQSOSCCTL0.dqsosc_enable to '0'. Poll DQSOSCSTAT0.dqsosc_state=0.
11. Disable automatic ZQ Calibration if it is enabled, by setting ZQCTL0.dis_auto_zq to '1'.
12. If DFILPCFG0.dfi_lp_en_sr is '1', set DFILPCFG0.dfi_lp_en_sr to '0' and poll DFISTAT.dfi_lp_ctrl_ack_stat=0
13. If DFILPCFG0.dfi_lp_data_req_en is '1', set DFILPCFG0.dfi_lp_data_req_en to '0' and poll DFISTAT.dfi_lp_data_ack_stat=0.
14. Set PWRCTL.en_dfi_dram_clk_disable to '0'.
15. Set PWRCTL.stay_in_selfref to '1' and then set PWRCTL.selfref_sw to '1'.
16. Poll STAT.selfref_state!=2'b00, indicating the controller entered into self-refresh state.
17. Read STAT.selfref_state and STAT.selfref_type. If STAT.selfref_state=2'b01 and STAT.selfref_type!=2'b01 ('Self Refresh 1' state caused solely by
18. Set OPCTRL1.dis_dq to '1' so that no CAM commands are de-queued. Poll OPCTRLCAM.wr_data_pipeline_empty=1, OPCTRLCAM.rd_data_pipeline_empty=1.

19. In LPDDR4, set PWRCTL.stay_in_selfref to '0' to enter 'Self-refresh power-down' and poll STAT.selfref_state=2'b10. This indicates the controller
20. Set DFIMISC.dfi_init_complete_en to '0' to ensure that the controller initialization state machine is not reset if the PHY needs to perform some
21. If DFI PHY update interface is active in the controller (DFIUPD0.dfi_phyupd_en=1), disable it by programming DFIUPD0.dfi_phyupd_en to '0'.
22. Program DFIMISC.dfi_frequency to the target_frequency value.
23. Toggle DFIMISC.dfi_freq_fsp[0] to select a different DRAM FSP for the target PState.
24. Program MSTR2.target_frequency to the target_frequency value, to select REGB_FREQ* registers.
25. Set DFIMISC.dfi_init_start to '1'. PHY performs internal sequences to cleanly stop pipelines and prepare for clock frequency change.
26. The PHY de-asserts dfi_init_complete. Poll DFISTAT.dfi_init_complete=0.
27. Change the clock frequency to the controller ensuring there are no glitches.
28. Toggle RFSHCTL0.refresh_update_level to allow the new refresh-related register values to propagate to the refresh logic.
29. Set DFIMISC.dfi_init_start to '0'. The PHY performs internal sequences to relock PLLs and calibrate ZQ/Delay-Lines.
30. If the DFI PHY update interface was active prior to frequency change, then enable PHY update Interface by setting DFIUPD0.dfi_phyupd_en to '1'.
31. The PHY asserts dfi_init_complete. Poll DFISTAT.dfi_init_complete=1.
32. Set DFIMISC.dfi_init_complete_en to '1', to indicate to the controller that PHY finished frequency change.
33. Set ZQCTL2.dis_srx_zqcl to '1', if ZQCTL2.dis_srx_zqcl=0.
34. In LPDDR4, set PWRCTL.stay_in_selfref to '1', and then set PWRCTL.selfref_sw to '0'. Poll STAT.selfref_state=2'b11. This indicates the controlle
35. If DVFSQ1 is not active (MR19.OP[3:2]=0), issue zq_calib_short command by programming OPCTRLCMD.zq_calib_short to '1' and poll OPCTRLSTAT.zq_cal
36. If DVFSQ2 is not active (MR19.OP[3:2]=0), set ZQCTL2.dis_srx_zqcl to '0', if it was set to '1' prior to frequency change.
37. In LPDDR5, set PWRCTL.selfref_sw to '0'.
38. Set PWRCTL.stay_in_selfref to '0'. Poll STAT.selfref_state=0. This indicates that the controller is not in self-refresh state.
39. After self-refresh exit, restore the values of registers DFILPCF0.dfi_lp_en_sr and DFILPCFG0.dfi_lp_data_req_en to its original value.
40. Set OPCTRL1.dis_dq back to '0', to allow read and write traffic to be sent to SDRAM.
41. Program the controller registers back to their default values, which were disabled prior to frequency change.
42. Enable HIF interface by setting OPCTRL1.dis_hif to '0'.
43. Reset DERATECTL0.derate_mr4_pause_fc to '0' if DERATECTL0.derate_mr4_pause_fc has been set to '1' prior to frequency change.
44. Reset DFIUPD0.dis_auto_ctrlupd to '0' if DFIUPD0.dis_auto_ctrlupd has been set to '1' prior to

frequency change.

45. If the DFI PHY Master interface was active prior to frequency change, then enable PHY Master interface by setting DFIPHYMSTR.dfi_phymstr_en to '
46. Enable DQS Oscillator if it was disabled prior to frequency change, by setting DQSOSCCTL0.dqsosc_enable to '1'.
47. If DVFSQ3 is not active (MR19.OP[3:2]=0), enable automatic ZQ Calibration if it was disabled prior to frequency change, by setting ZQCTL0.dis_au
48. Enable PWRCTL.selfref_en, PWRCTL.powerdown_en, HWLPCTL.hw_lp_en if it was disabled prior to frequency change.
49. Set PWRCTL.en_dfi_dram_clk_disable to '1' if it was programmed to '0' prior to frequency change.
50. Enable AXI ports by programming PCTRL.port_en to '1'.

4.1.4.4 Clock Removal and Recovery

Clock Removal

1. Write '0' to PCTRL.port_en
2. Poll PSTAT.rd_port_busy_n = 0, and poll PSTAT.wr_port_busy_n = 0
3. Write '1' to PWRCTL.selfref_sw
4. Poll STAT.selfref_type= 2'b10, and poll STAT.selfref_state = 3'b010
5. Remove ddr*_p*_aclk.

Clock Recovery

1. Enable ddr*_p*_aclk
2. Write '0' to PWRCTL.selfref_sw
3. Poll STAT.selfref_type = 2'b00
4. Write '1' to PCTRL.port_en

4.1.4.5 Power Removal and Recovery

Power Removal

1. Write '0' to PCTRL_n.port_en
2. Poll PSTAT.rd_port_busy_n = 0 and poll PSTAT.wr_port_busy_n = 0
3. Set ZQ Stop(MR28 OP[1]) to '1'. Send MRW command using MRCTRL0 and MRCTRL1. (LPDDR5 only)
4. Write '1' to PWRCTL.selfref_sw
5. Poll STAT.selfref_type= 2'b10 and poll STAT.selfref_state = 3'b010
6. Program DFIMISC.dfi_frequency as required
7. Set DFIMISC.dfi_init_start to '1'
8. The PHY de-asserts dfi_init_complete. The controller polls DFISTAT.dfi_init_complete=0.
9. Set DFIMISC.dfi_init_start to '0'.
10. The PHY asserts dfi_init_complete. The controller polls DFISTAT.dfi_init_complete=1.
11. Remove power

Re-enabling the Power

1. Enable power
2. Reset controller/PHY by driving ddr_rst_ctrl.sw_ddr_core_rstn and ddr_rst_ctrl.sw_ddr_prstn low
3. Release ddr_rst_ctrl.sw_ddr_prstn and reprogram the registers to pre-power removal values
4. Program RFSHCTL0.dis_auto_refresh =1'b1
5. Program INITTMG0.skip_dram_init = 2'b11
6. Program PWRCTL.selfref_sw = 1'b1
7. Program DFIMISC.dfi_init_complete_en to 1'b0
8. Release ddr_rst_ctrl.sw_ddr_core_rstn
9. Program RFSHCTL0.dis_auto_refresh =1'b0
10. Run PHY initialization/training as required,
11. Program DFIMISC.dfi_frequency as required.
12. Set DFIMISC.dfi_init_start to '1'
13. The PHY asserts dfi_init_complete. The controller polls DFISTAT.dfi_init_complete=1.
14. Set DFIMISC.dfi_init_start to '0'
15. Program DFIMISC.dfi_init_complete_en to 1'b1
16. Program PWRCTL.selfref_sw = 1'b0
17. Poll STAT.selfref_type = 2'b00
18. Poll STAT.operating_mode for Normal Mode entry
19. Set ZQ Stop(MR28 OP[1]) to '0'. Send MRW command using MRCTRL0 and MRCTRL1. (LPDDR5 only)
20. Write PCTRL.port_en = 1

4.1.4.6 Special Software Self-Refresh Sequence with Controller Initiated Retaining Followed by Burst PPT2

1. During initialization, PPT2 related registers are assumed to be set, including DFIUPDTMG2.ppt2_en=1
2. Disable low-power activities by programming PWRCTL.selfref_en, PWRCTL.powerdown_en and HWLPCTL.hw_lp_en to '0'.
3. Ensure that STAT.operating_mode=1, which indicates that the controller is in normal mode. Continue to read this until it is read as '1' three times in a row
4. Set PCTRL.port_en to '0' to block the AXI ports from taking transactions. Poll PSTAT.rd_port_busy_n=0 and PSTAT.wr_port_busy_n=0. Wait until all AXI ports are idle.
5. Set OPCTRL1.dis_hif to '1', so that no new commands are accepted by the controller and poll OPCTRLCAM.dbg_wr_q_empty=1, OPCTRLCAM.wr_data_pipeline_empty=1, OPCTRLCAM. dbg_rd_q_empty=1, OPCTRLCAM.rd_data_pipeline_empty=1.
6. If DERATECTL0.derate_eanble is '1', then set DERATECTL0.derate_mr4_pause_fc to '1' to pause automatic MRR to MR4.

7. If DFIUPD0.dis_auto_ctrlupd=0, then set DFIUPD0.dis_auto_ctrlupd to '1' to disable periodic DFI update request temporarily.
8. Set OPCTRLCMD.ctrlupd to '1' and poll OPCTRLSTAT.ctrlupd_busy=0 to issue DFI update request manually.
9. If DFILPCFG0.dfi_lp_en_sr is '1', set DFILPCFG0.dfi_lp_en_sr to '0' and poll
10. If DFILPCFG0.dfi_lp_data_req_en is '1', set DFILPCFG0.dfi_lp_data_req_en to '0' and poll DFISTAT.dfi_lp_data_ack_stat=0.
11. Set PWRCTL.en_dfi_dram_clk_disable to '0'.
12. Enter self-refresh mode by setting PWRCTL.selfref_sw to '1'. Poll operating_mode=3, which indicates that the controller has entered into self-refresh.
13. Ensure that self-refresh is due to software by checking that STAT.selfref_type[1:0]=2'b10.
14. Set OPCTRL1.dis_dq to '1' so that no CAM commands are de-queued. Poll OPCTRLCAM.wr_data_pipeline_empty=1, OPCTRLCAM.rd_data_pipeline_empty=1.
15. Set DFIMISC.dfi_init_complete_en to '0' to ensure that the controller initialization state machine is not reset if the PHY needs to perform some initialization while retraining.
16. Program DFIMISC.dfi_frequency to 5'h17, that is Retrain only.
17. Long term Idle.
18. Set DFIMISC.dfi_init_start to '1'. PHY performs internal sequences to cleanly stop pipelines.
19. The PHY de-asserts dfi_init_complete. Poll DFISTAT.dfi_init_complete=0.
20. Set DFIMISC.dfi_init_start to '0'. The PHY performs internal sequences to re-lock PLLs and calibrate ZQ/Delay-Lines.
21. The PHY asserts dfi_init_complete. Poll DFISTAT.dfi_init_complete=1.
22. Set DFIMISC.dfi_init_complete_en to '1', to indicate to the controller that the PHY has finished retraining.
23. Start the Burst PPT2 process. If ZQCTL0.dis_auto_zq is '0', set ZQCTL0.dis_auto_zq to '1'.
24. Program PPT2CTRL0.ppt2_burst to '1'.
25. Exit self-refresh by setting PWRCTL.selfref_sw to '0'. And wait until STAT.operating_mode!=2'b11, indicating that the controller has exited from the self-refresh state.
26. Once the selfref_sw is made, the Burst PPT2 process starts. Wait for Burst PPT2 to complete by polling PPT2STAT0.ppt2_burst_busy to '0'.
27. Reset ZQCTL0.dis_auto_zq to '0' if ZQCTL0.dis_auto_zq has been set to '1' prior to Burst PPT2.
28. Restore the values of DFILPCF0.dfi_lp_en_sr and DFILPCFG0.dfi_lp_data_req_en after the self-refresh exit.
29. Set OPCTRL1.dis_dq back to '0', to allow read and write traffic to be sent to SDRAM.
30. Program the controller registers back to their default values, which were disabled prior to retraining.
31. Enable HIF commands by setting OPCTRL1.dis_hif to '0'.
32. Reset DERATECTL0.derate_mr4_pause_fc to '0' if DERATECTL0.derate_mr4_pause_fc has been set to '1' prior to retraining.
33. Reset DFIUPD0.dis_auto_ctrlupd to '0' if DFIUPD0.dis_auto_ctrlupd has been set to '1' prior to retraining.

34. Enable PWRCTL.selfref_en, PWRCTL.powerdown_en, HWLPCTL.hw_lp_en if they were disabled prior to retraining.
35. Set PWRCTL.en_dfi_dram_clk_disable to '1' if it was programmed to '0' prior to retraining.
36. Enable AXI ports by programming PCTRL.port_en to '1'.

4.1.5 Memory Map

4.1.5.1 Application to HIF Address Mapping

The system address is a byte address. XPI maps the MSB bits of the application address to the HIF address (hif_cmd_addr) in the following ways:

1. hif_cmd_addr[32:4]=system_address[34:6]
2. hif_cmd_addr[3] is generated internally.
3. hif_cmd_addr[2:0]=0

4.1.5.2 HIF Address to SDRAM Address Mapping

The address mapper maps HIF word addresses to SDRAM addresses by selecting the HIF address bit that maps to each and every applicable SDRAM address bit. While it is possible to map HIF address bits to a SDRAM address in any desired manner, the full available address space is accessible only when no two SDRAM address bits are determined by the same HIF address bit. Each SDRAM address bit has an associated register vector to determine its source.

Registers ADDRMAPx (x=0 to 11) are used to program the address mapper. For more information on ADDRMAP registers, see REGB_ADDR_MAP0 in the “Register Descriptions" chapter.

The HIF address bit number is determined by adding the internal base of the ADDRMAPx (x=0 to 11) register to the programmed value for that register, as described in the following equation:

HIF address bit number = [internal base]+ [register value]

For example, for ADDRMAP5.addrmap_col_b7, the internal base is 7. When full data bus is in use, column bit 7 is determined by the following equation: 7 + [register value]

If this register is programmed to 2, the HIF address bit can be calculated by using following equation:[HIF address bit number] = 7 + 2 = 9

In other words, the column address bit 7 sent to SDRAM would always be equal to hif_cmd_addr[9] of the corresponding HIF source address.

The system address to physical address mapping can be done by choosing any one of the possible Combinations shown as follow. It explains the different possible ways to map the HIF address bits to the SDRAM rank/bank/bank group/row/column address.

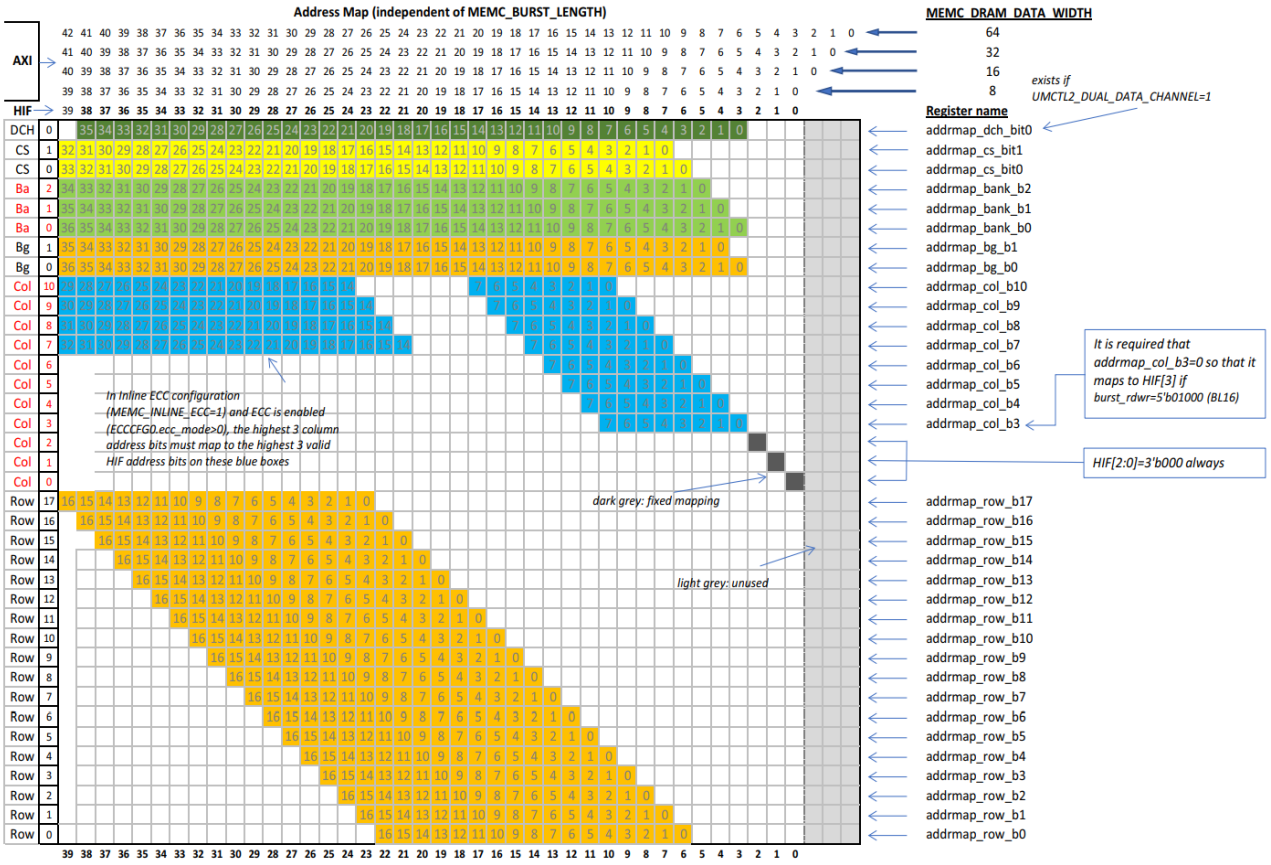


Figure 4-2 System Address to Physical Address Mapping

Register constraints are independent of ARB or HIF configuration and bus width mode.

SDRAM column address [2:0] does not exist and hence is not addressable. The HIF master must generate hif_cmd_addr[2:0]=000.

For any address bits which cannot be in use in all cases, all bits of the associated address map register field must be set to '1' when the associated SDRAM address bit is not in use.

4.1.5.3 Recommandation for Optimum SDRAM Utilization

Write or Masked Write to Masked Write have a tCCDMW penalty instead of the minimum tCCD. There is no performance hit for reads as tCCDMW is not related to them. These performance hits can be minimized/avoided by using address mapping recommendations in arbiter configurations and traffic recommendations in HIF configurations, as outlined as follow:

Table 4-1 Address Map Recommendations for Arbiter Configurations

Address Map	HIF Behavior	Performance
LPDDR5		
addrmap_col_b3=0(HIF[3]) addrmap_bg_b0=1(HIF[4]) addrmap_bg_b1=1(HIF[5]) addrmap_bank_b0=1(HIF[6])	Full writes, read, RMWs	SDRAM utilization: full Data of each CAM entry used:full
LPDDR4		
addrmap_col_b3=0(HIF[3])	Full writes, read,	SDRAM utilization: full

Address Map	HIF Behavior	Performance
addrmap_bg_b0=1(HIF[4]) addrmap_bg_b1=1(HIF[5])	RMWs	Data of each CAM entry used:full

Table 4-2 Address Map Recommendations for HIF Configurations

Traffic	Maximum HIF Burst Size	Performance
LPDDR5		
Rotate 2 bank group bits and 1 bank bits between HIF commands	Full writes, read, RMWs	SDRAM utilization: full Data of each CAM entry used:full
LPDDR4		
Rotate 2 bank bits between HIF commands	Full writes, read, RMWs	SDRAM utilization: full Data of each CAM entry used:full

It is recommended to perform bank or bank group rotation between back to back SDRAM commands. For LPDDR4, tCCDMW is equal to 4*tCCD, so two bank bits need to be rotated. For LPDDR5, tCCDMW in the case of BG mode is equal to 8*tCCD, so two bank group bits and one bank bit need to be rotated.

For setting the Address Map register, ADDRMAP6.addrmap_col_b3 must be set to '0'

4.1.5.4 Non-binary Device Densities

LPDDR4 3Gb/6Gb/12Gb per channel devices, LPDDR5 3Gb/6Gb/12Gb/24Gb per channel devices do not support addresses which have both MSB and MSB-1 row bits high. Any attempt to read or write to this address space results in memory misbehavior or system halts. When such devices are used, you must set the register ADDRMAP12.nonbinary_device_density accordingly.

When ADDRMAP12.nonbinary_device_density is set to non-zero, any write or RMW request with row[MSB:MSB-1]==2'b11 is discarded, while the HIF output hif_wdata_ptr_addr_err is asserted.

Also, any read request with row[MSB:MSB-1]==2'b11 is executed (spare read) by changing row[MSB:MSB-1] from 2'b11 to 2'b10. The return data for such reads are masked to zeros, while the HIF output hif_rdata_addr_err is asserted.

Table 4-3 Non-binary Device Densities

Non-binary device densities	Component type	Inaccessible region	
		LPDDR4	LPDDR5
3Gb	X16	[R14:R13]!=2'b11	[R13:R12]!=2'b11
3Gb	X8	[R15:R14]!=2'b11	[R14:R13]!=2'b11
6Gb	X16	[R15:R14]!=2'b11	[R14:R13]!=2'b11
6Gb	X8	[R16:R15]!=2'b11	[R15:R14]!=2'b11
12Gb	X16	[R16:R15]!=2'b11	[R15:R14]!=2'b11
12Gb	X8	[R17:R16]!=2'b11	[R16:R15]!=2'b11
24Gb	X16	N/A	[R16:R15]!=2'b11
24Gb	X8	N/A	[R17:R16]!=2'b11

Non-binary device densities	Component type	Inaccessible region	
		LPDDR4	LPDDR5
24Gb	X4	N/A	N/A

4.1.5.5 Bank Hashing

The bank hashing function in DDR controller performs fixed XOR functions on the SDRAM row and bank-BG bits. Based on the SDRAM, the number of bank and BG bits may vary, and the XOR function changes accordingly. Each bank is uniquely identified by concatenating the BG and bank bits. For each bank bit, a set of row bits are selected. An XOR operation is performed between the bank bit and the selected row bits. The output of this XOR is the final bank bit.

As shown below, the signals am_bg_bank and am_row indicate the mapped BG-bank bits and row bits from the address mapper respectively. These signals are propagated to bank hashing function. bh_bg_bank indicates the output from the bank hashing function. The function performs the following XOR operation on the am_bg_bank and am_row signals:

bh_bg_bank[0] = (am_bg_bank[0] ^ am_row[0] ^ am_row[3] ^ am_row[6] ^ am_row[9] ^

am_row[12] ^ am_row[15])；

bh_bg_bank[1] = (am_bg_bank[1] ^ am_row[1] ^ am_row[4] ^ am_row[7] ^ am_row[10] ^

am_row[13] ^ am_row[16]);

bh_bg_bank[2] = (am_bg_bank[2] ^ am_row[2] ^ am_row[5] ^ am_row[8] ^ am_row[11] ^

am_row[14] ^ am_row[17]);

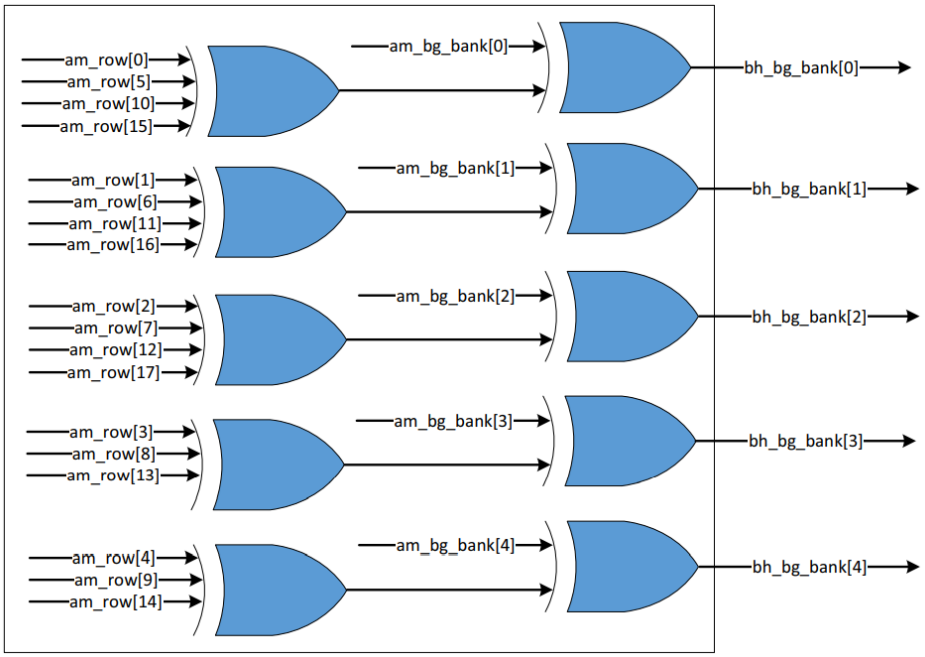


Figure 4-3 Bank Hashing Function Wth 5 Bank Bits

Bank hashing function is performed by the Bank Hashing Module which is instantiated after HIF to DRAM address mapper. Software register ADDRMAP12.bank_hash_en provides bypass path of the module. The module output bank/BG address (bh_bg_bank) is based on input bank/BG address (am_bg_bank) and row address (am_row)

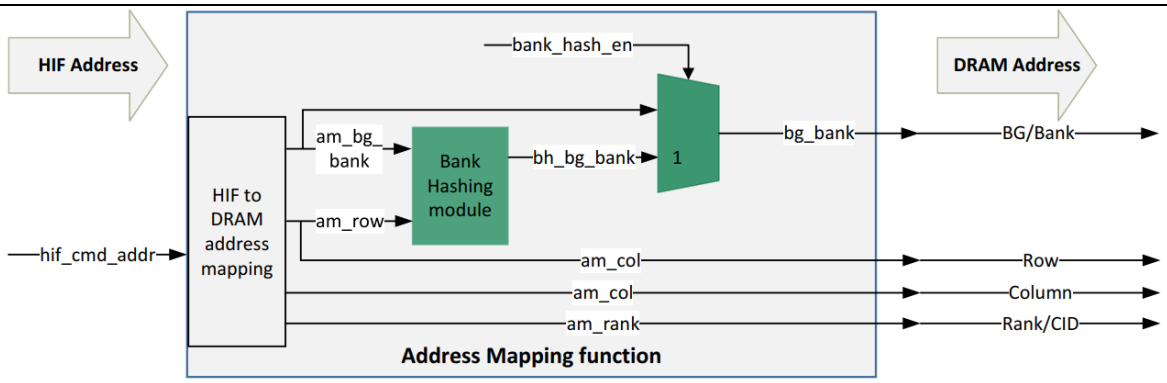


Figure 4-4 Address Mapping Function

4.1.5.6 Registers Related to Address Mapper

The following are the registers related to the Address Mapper:

- ADDRMAP0
- ADDRMAP1
- ADDRMAP3
- ADDRMAP4
- ADDRMAP5
- ADDRMAP6
- ADDRMAP7
- ADDRMAP8
- ADDRMAP9
- ADDRMAP10
- ADDRMAP11
- ADDRMAP12

For more information about these registers, refer to the “Register Descriptions" chapter

4.2 eMMC

4.2.1 Overview

The chip contains one eMMC controller to handle read and write operations to the emmc card. The main features are as follows:

- supports eMMC5.1
- supports CPU/SDMA/ADMA2/ADMA3 transfer
- supports Command Queuing Engine and EMMC CQ HCI
- supports legacy/high-speed SDR/high-speed DDR/HS200/HS400 speed mode
- supports 1/4/8bit data bus
- supports tuning
- supports boot and alternative boot

4.2.2 Block Diagram

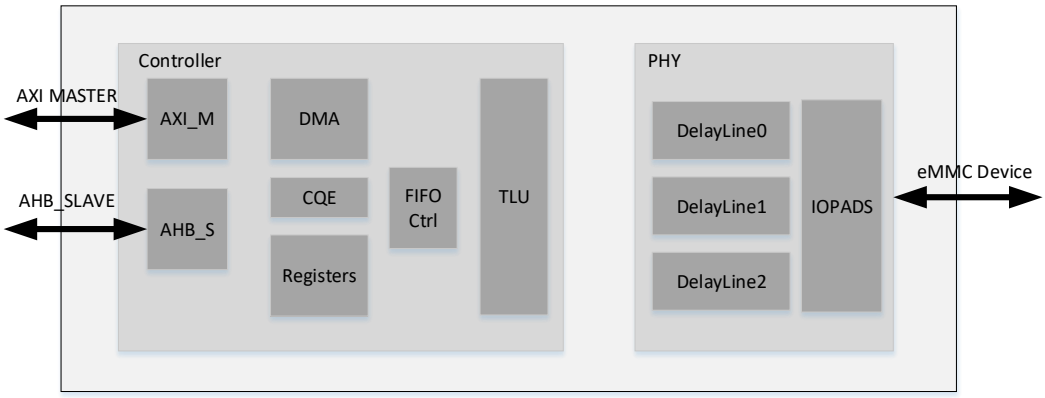


Figure 4-5 eMMC Diagram

The EMMC controller interacts with the system through the AXI bus and mainly consists of the following modules:

- DMA Controller handles data transfer between host controller and system memory
- CQE engine implements command queuing
- FIFO ctrl handles packet buffer access
- TLU Process read and write data to meet standard protocol requirements

4.2.3 Initialization

Before reading or writing data to an external EMMC card, initialize the controller to ensure that it works properly. The process is as follows:

- chip power up and reset
- release emmc controller reset
- initialize internal register(EMMC_CTRL_R/ HOST_CTRL2_R/ MSHC_CTRL_R)
- polling until PHY_CNFG.PWRGOOD=1
- assert PHY_CNFG.PHY_RSTN=1
- configure *PAD_CNFG
- switch clk_tx to 400kHz
- configure CLK_CTRL_R.SD_CLK_EN=1,enable output SD_CLK
- send card initilization and identification command
- start data transfer

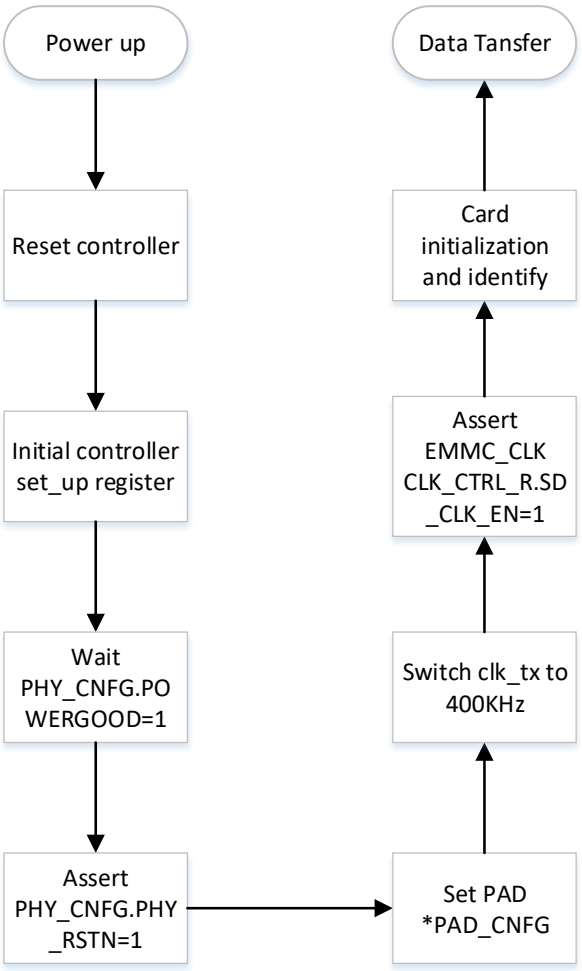


Figure 4-6 eMMC Initialization

4.2.4 Interface Timing

There are three delaylines (DL0~2) inside the controller. DL0 includes a phase detector and can be used in HS400 mode as a DLL master. DL1 is used to adjust the output phase of the EMMC_CLK, and DL2 is used to adjust the phase of the sample clock inside the controller. Each delayline contains one 128-level adjustable delay chain and one 128-level fixed delay chain. The fixed delay chain can be used to provide additional phase adjustment for the TX and RX clocks in low frequency operating mode. In high-frequency mode, there is no need to enable the fixed delay chain. Disable it by setting the corresponding extdlyen to 0.

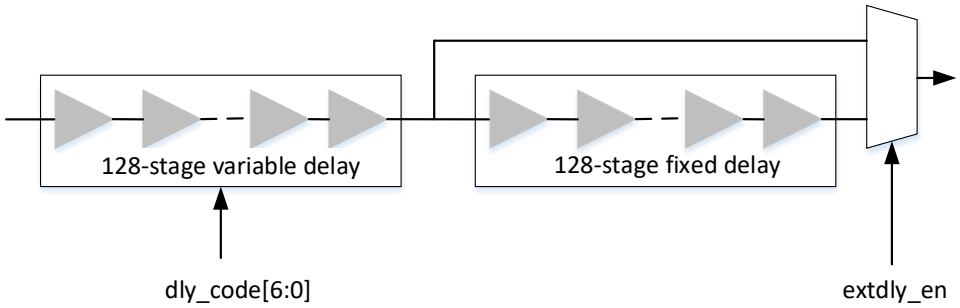


Figure 4-7 Delayline Diagram

Delay value of each delay unit is shown in the following table. If VDD_CORE is above nominal range, setting COMMDL_CNFG[DLSTEP_SEL] to 1 to increase delay value of delay unit, this provides additional phase delay.

Table 4-4 delay and step size

Parameter	dlstep_sel	Min	Max	Unit	Comment
VDD_CORE=0.8V(nominal)					
Total delay of variable delay chain	0	5	10.3	ns	All 128 stages enable,fixed delay disbled
	1	5.75	11.8	ns	All 128 stages enable,fixed delay disbled
Average step size	0	39	80	ps	Average delay per unit cell
	1	44	92	ps	Average delay per unit cell

By default, the DL1 variable delay chain delay_code is 0, and the fixed delay chain is disabled. During initialization, corresponding delay_code should be configured to adjust the EMMC_CLK phase to meet the timing requirements of external cards. The following figure shows the Delayline1 configuration process.

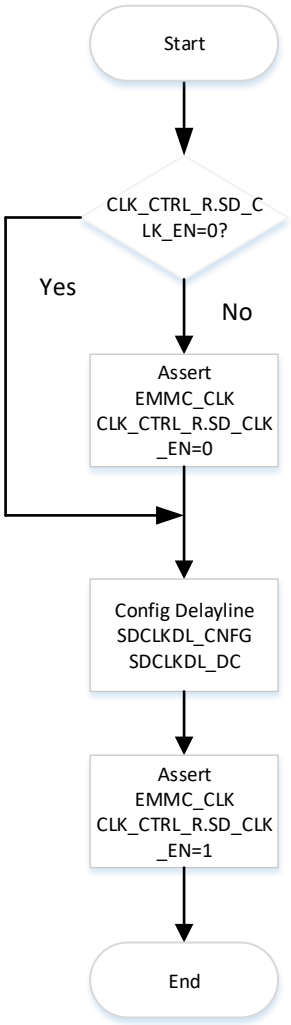


Figure 4-8 DL1 Config

The following shows the diagram for adjusting the clock in the output direction.

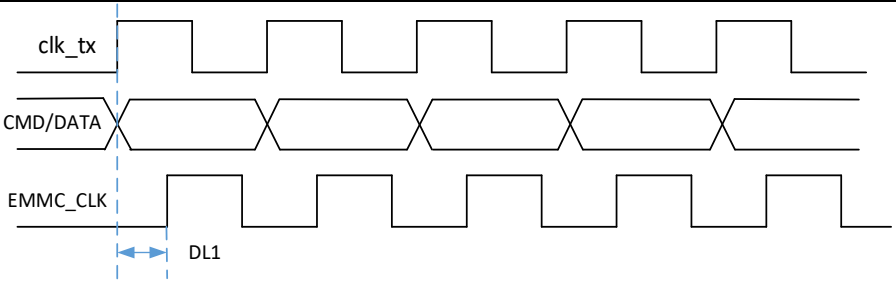


Figure 4-9 Output timing

In high frequency transfer mode, the controller can be tuned to obtain a suitable sampling range, and configure DL2 to adjust the phase of the sampling clock to ensure the correct sampling data. The tuning process is as follows.

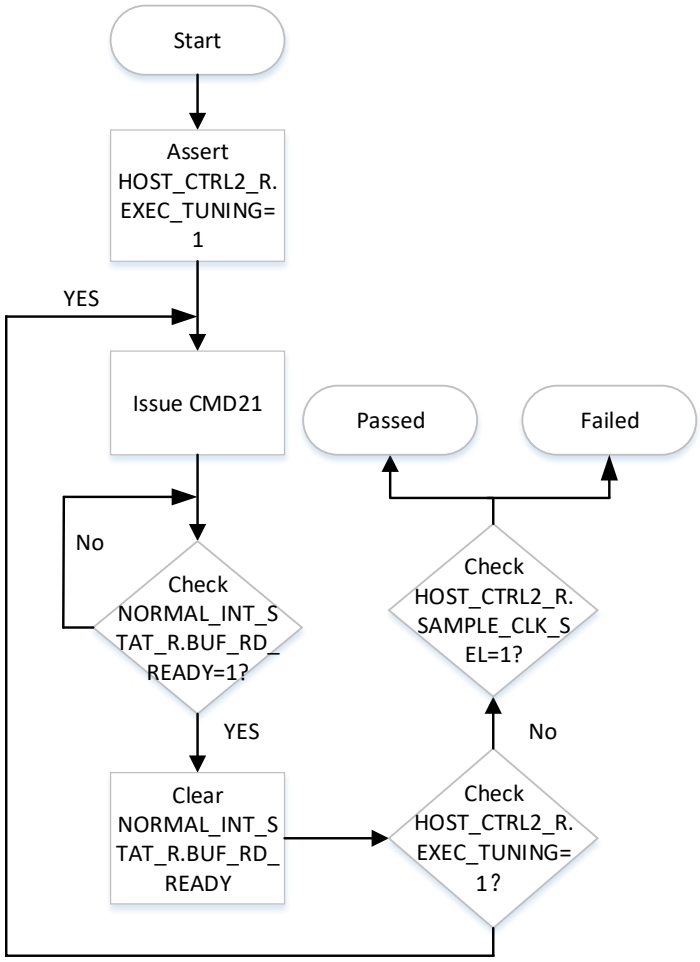


Figure 4-10 Tuning

The following shows the diagram for adjusting the clock in the input direction.

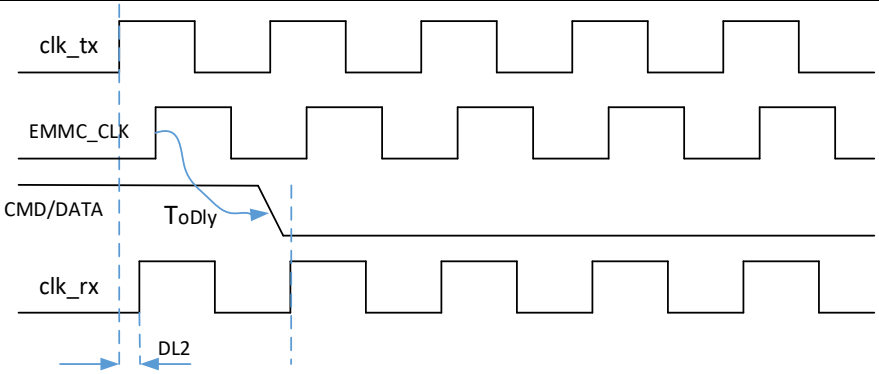


Figure 4-11 Input timing

4.3 SDIO

4.3.1 Overview

The chip contains two identical SD/SDIO controllers to handle read and write operations to the SD card, and supports extended peripherals through the SDIO protocol. The main features are as follows:

- supports SD Specifications Part A2 SD Host Controller Standard Specification Version 4.20
- supports SD Specifications Part 1 Physical layer Specification Version 6.00 February, 2017
- supports SD Specifications Part E1 SDIO Specification Version 4.10 Sept 2014
- supports CPU/SDMA/ADMA2/ADMA3 transfer
- supports tuning
- supports 1/4bit data bus
- supports UHS-I mode
- supports DS/HS/SDR12/SDR25/SDR50/DDR50/SDR104 speed mode
- supports SDIO read wait

4.3.2 Block Diagram

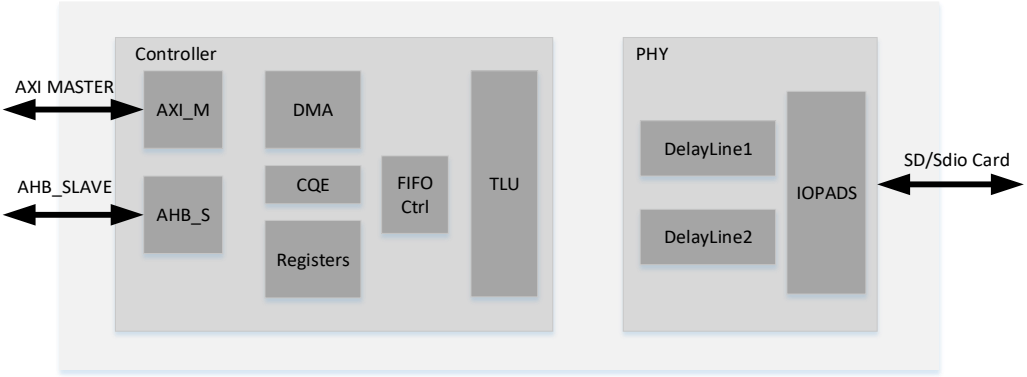


Figure 4-12 SD/SDIO Diagram

The SD/SDIO controller interacts with the system through the AXI bus and mainly consists of the following modules:

- DMA Controller handles data transfer between host controller and system memory
- CQE engine implements command queuing
- FIFO ctrl handles packet buffer access
- TLU Process read and write data to meet standard protocol requirements

4.3.3 Initialization

Before reading or writing data to an external SD card, initialize the controller to ensure that it works properly. The process is as follows:

- chip power up and reset
- release SD controller reset
- initialize internal register(HOST_CTRL2_R/ SDHC_CTRL_R)
- polling until PHY_CNFG.PWRGOOD=1
- assert PHY_CNFG.PHY_RSTN=1
- configure *PAD_CNFG
- switch clk_tx to 400kHz
- configure CLK_CTRL_R.SD_CLK_EN=1, enable output SD_CLK
- send card initilization and identification command
- start data transfer

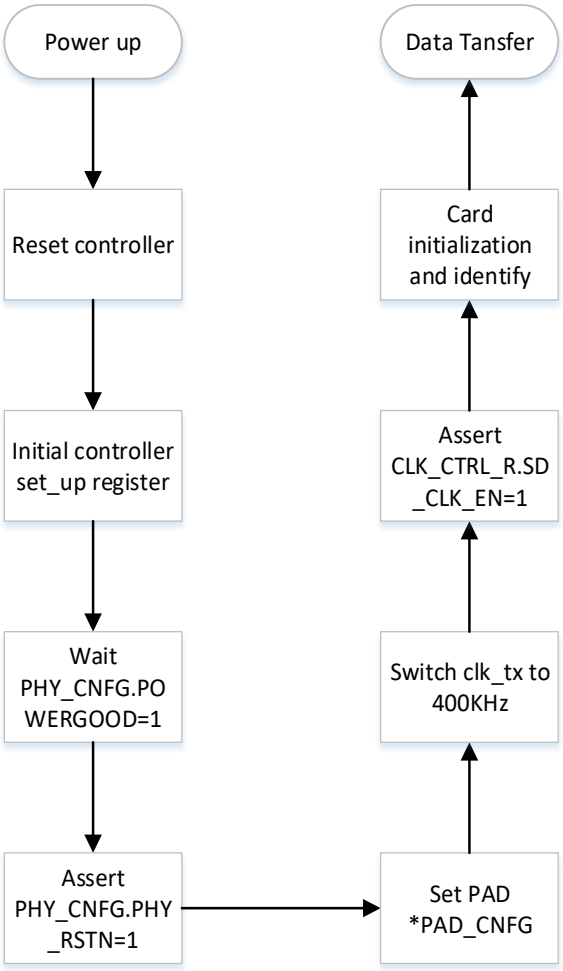


Figure 4-13 SD/SDIO Initialization

4.3.4 Interface Timing

There are two delaylines (DL1/2) inside the controller. DL1 is used to adjust the output phase of the SD_CLK clock, and DL2 is used to adjust the phase of the sample clock inside the controller. Each delayline contains one 128-level adjustable delay chain and one 128-level fixed delay chain. The fixed delay chain can be used to provide additional phase adjustment for the TX and RX clocks in low frequency operating mode. In high-frequency mode, there is no need to enable the fixed delay chain. Disable it by setting the corresponding extdlyen to 0.

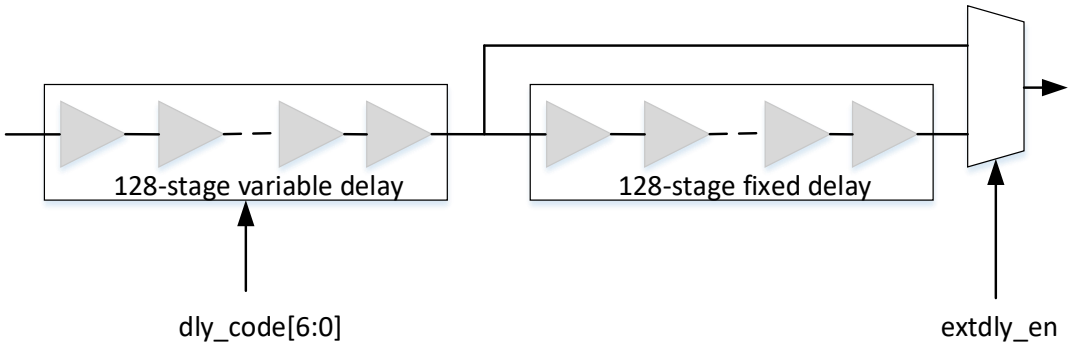


Figure 4-14 Delayline diagram

Delay value of each delay unit is shown in the following table. If VDD_CORE is above nominal range, setting COMMDL_CNFG[DLSTEP_SEL] to 1 to increase delay value of delay unit, this provides additional phase delay.

Table 4-5 delay and step size

Parameter	dlstep_sel	Min	Max	Unit	Comment
VDD_CORE=0.8V(nominal)					
Total delay of variable delay chain	0	5	10.3	ns	All 128 stages enable,fixed delay disbled
	1	5.75	11.8	ns	All 128 stages enable,fixed delay disbled
Average step size	0	39	80	ps	Average delay per unit cell
	1	44	92	ps	Average delay per unit cell

By default, the DL1 variable delay chain delay_code is 0, and the fixed delay chain is disabled. During initialization, corresponding delay_code should be configured to adjust the SD_CLK phase to meet the timing requirements of external cards. The following figure shows the Delayline1 configuration process.

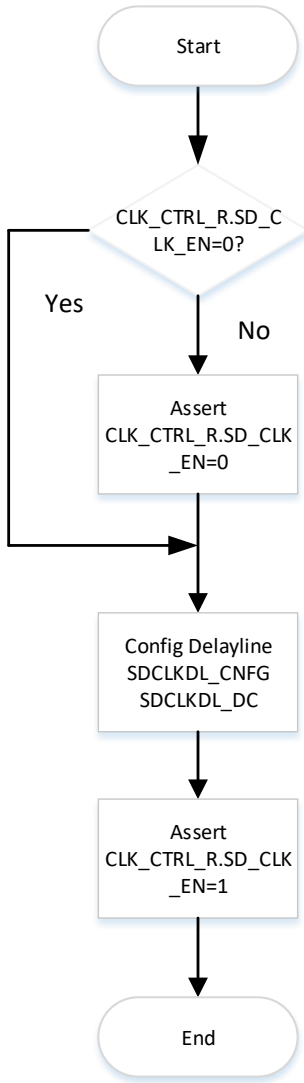


Figure 4-15 SD/SDIO DL1 Config

The following shows the diagram for adjusting the clock in the output direction.

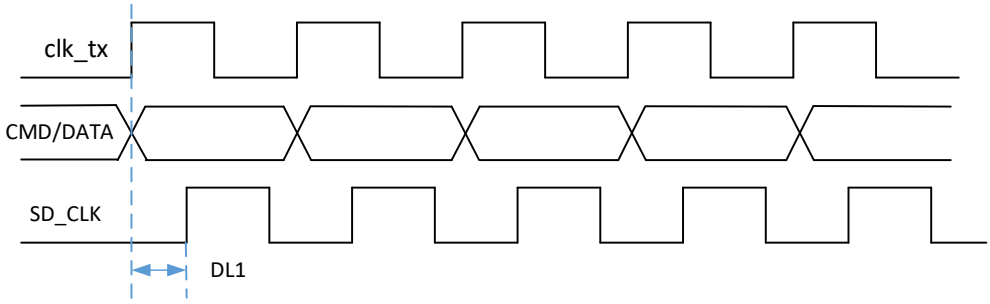


Figure 4-16 Output timing

In high frequency transfer mode, the controller can be tuned to obtain a suitable sampling range, and configure DL2 to adjust the phase of the sampling clock to ensure the correct sampling data. The tuning process is as follows.

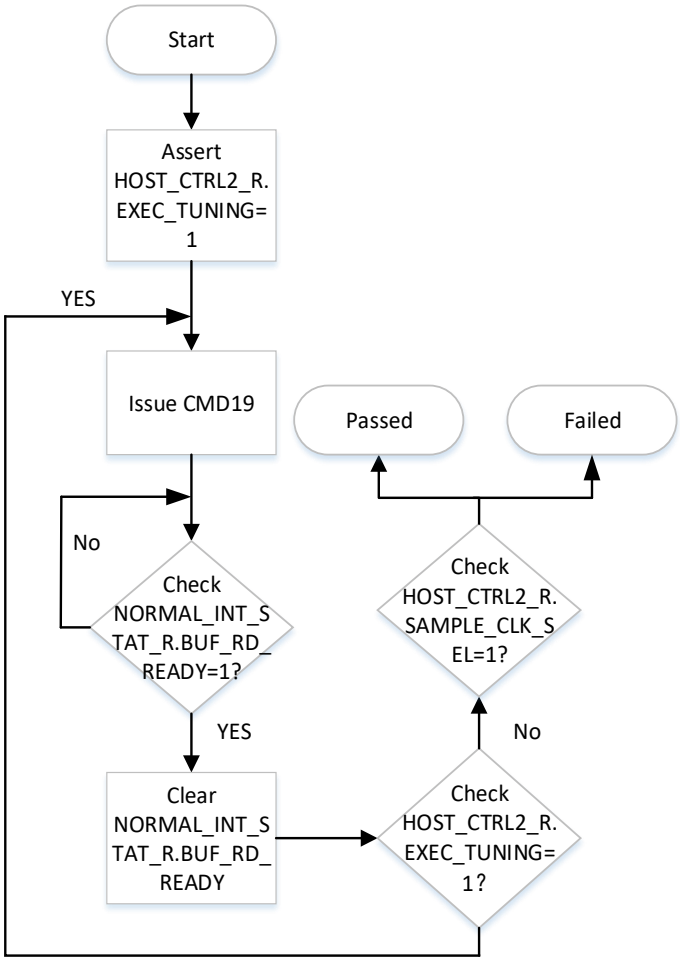


Figure 4-17 SD Tuning

The following shows the diagram for adjusting the clock in the input direction.

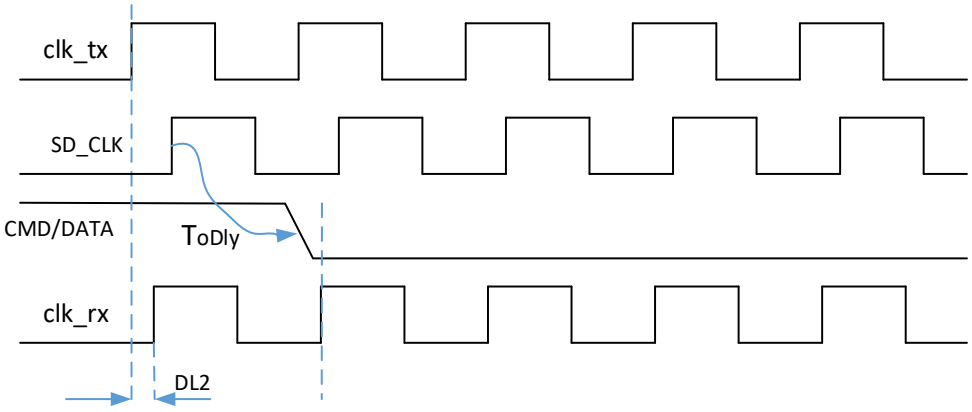


Figure 4-18 Input timing

4.4 SATA

4.4.1 Overview

SATA is hardened in high speed peripheral subsystem, which is intended to transfer data in between host and SATA SSD or hard drive.

SATA supports the following key features:

- AXI3 protocol
- 1 port
- 1.5Gb/s, 3.0Gb/s and 6.0Gb/s
- Internal DMA engine per port
- Command queuing for up to 32 entries.
- TX OOB detection and RX OOB detection (RX OOB clock frequency = 50MHz)
- 8b/10b encoding/decoding
- RX data buffer
- Data alignment circuit
- Command completion coalescing
- Low power mode: partial, slumber and devsleep.

4.4.2 Block Diagram

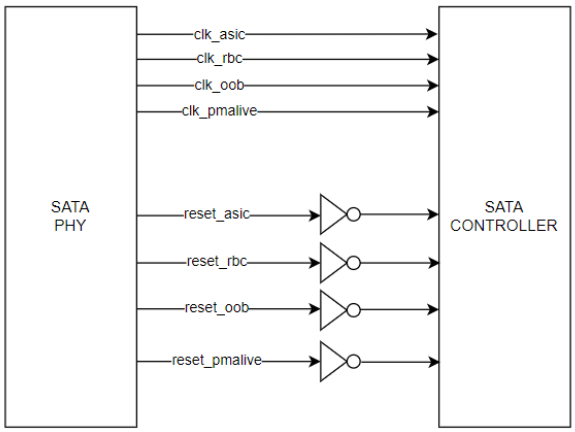


Figure 4-4-1 SATA host block diagram

SATA host integrates SATA 6G PHY and SATA controller. The clocks and resets going into controller are directly from PHY. Therefore, those clocks and resets are generated internally after successful power on

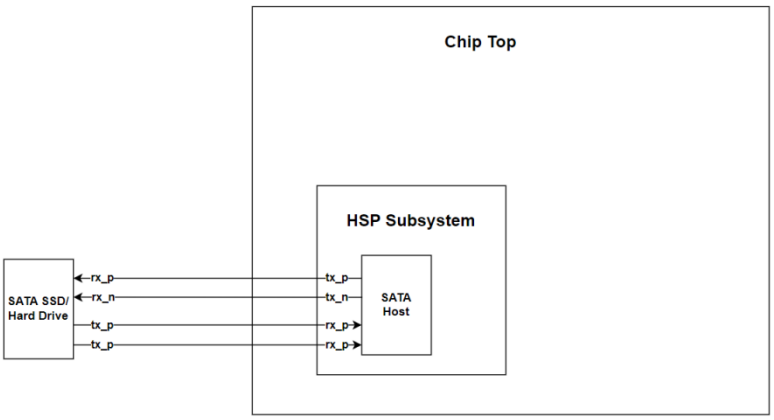


Figure 4-4-2 SATA in HSP subsystem and chip top level

4.4.3 SATA Initialization

SATA initialization flow can be mainly summarized as:

- Enable clocks and release corresponding resets.
- Configure SATA PHY, take PHY out of reset, and check PHY ready.
- Reset and initialization SATA controller, start OOB sequence and wait for linking up.
- Configure SATA controller.

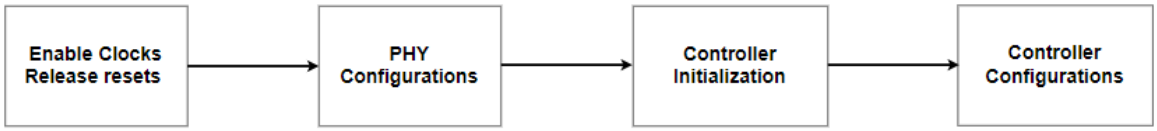


Figure 4-4-3 SATA reset release and initialization

4.4.3.1 Example Configurations

The following part is related configurations of SATA initialization flow, the corresponding registers' configurations of each step are shown below as well. After SATA initialization done, more configurations can be set to achieve specific usage intentions.

Enable chip and HSP subsystem clocks and release resets

Table 4-6 Enable Clocks and Release Resets

Register Address	Access	Value	Description
`SCU_BASE_ADDR + 0x41c	W	0x0800_0007	[27]: sw_sata_arstn
			[2]:sw_hsp_por_rstn
			[1]:sw_hsp_cfg_rstn
			[0]:sw_hsp_axi_rstn
`SCU_BASE_ADDR + 0x14c	W	0xc000_0000	[31]: hsp_cfg_clken
			[30]: scu_hsp_pclk_en

Transmit de-emphasis for SATA should be set to obtain the best eye diagram at the connector, therefore, these settings must be tuned based on the actual package and board in the application:

- tx_preamph_gen1[5:0]
- tx_preamph_gen2[5:0]

- tx_preamph_gen3[5:0]
- tx_amplitude_gen1[6:0]
- tx_amplitude_gen2[6:0]
- tx_amplitude_gen3[6:0]

The following parameters are technology-dependent and therefore must be set as constants:

- los_level[4:0]
- los_bias[2:0]

These parameters must be set prior to taking the PHY out of reset, and they must not be changed while the PHY is running.

Note: The values in this table are recommendations in databook and just for reference here.

Register Address	Access	Value	Description
`HSP_CSR_BASE_ADDR + 0x338	W	0x0073_4642	[22:16]: amplitude_gen3
			[14:8]: amplitude_gen2
			[6:0]: amplitude_gen1
` HSP_CSR_BASE_ADDR + 0x32c	W	0x0023_0505	[22:16]: preemph_gen3
			[14:8]: preemph_gen2
			[6:0]: preemph_gen1
` HSP_CSR_BASE_ADDR + 0x33c	W	0x0002_0009	[18:16]: los_bias
			[4:0]: los_level

Table 4-4-1 SATA PHY parameters control

Power management always alive clock(clk_pmalive) can be enabled to keep the power control module in SATA controller operating normally when TX and RX clocks are disabled in the low power modes. By setting address `HSP_CSR_BASE_ADDR + 0x334 bit[20] = 1, the always alive clock will be derived from reference clock automatically.

By setting `HSP_CSR_BASE_ADDR + 0x320 = ‘h003c_0000, The MPLL frequency factor shown below is the requirement in specification for 50MHz reference clock. The MPLL frequency can achieve 3GHz and the following clocks are available from the MPLL when the link is operating.

- mpll_word_clk = 600MHz
- mpll_dword_clk = 300MHz
- mpll_qword_clk = 150MHz

Table 4-7 Always alive clock and MPLL clocks

Register Address	Access	Value	Description
`HSP_CSR_BASE_ADDR + 0x334	W	0x0010_0001	[20]: ref_ssp_en enable refclk for phy
			[0]: ref_repeat_clk_en enable clock repeater buffer
`HSP_CSR_BASE_ADDR + 0x320	W	0x003c_0000	[22:16]: ‘b0111100, required MPLL frequency multiplication for 50MHz reference clock

Table 4-8 Take PHY out of reset and wait PHY ready

Register Address	Access	Value	Description
`HSP_CSR_BASE_ADDR + 0x340	W	0x0000_0000	[1]: sata_p0_reset
			[1]: sata_phy_reset
` HSP_CSR_BASE_ADDR + 0x324	R	0x0000_0001	[0]: 1 means phy ready

After PHY ready is read over register, bring SATA controller out of reset and start OOB sequence.

Table 4-9 SATA controller initialization

Register Address	Access	Value	Description
`SATA_BASE_ADDR + 0x4	W	0x0000_0001	GCSR_GHC.bit0=1, reset HBA
`SATA_BASE_ADDR + 0x0	W	0x0000_0000	GCSR_CAP.bit0=0 number of port=1
`SATA_BASE_ADDR + 0xc	W	0x0000_0001	GCSR_PI.bit0=1 the corresponding port is available for the software to use
`SATA_BASE_ADDR + 0x12c	W	0x0000_0001	PCSR_PSCTL.bit[2:0]=1 perform interface initialization sequence to establish communication
#100ns			
`SATA_BASE_ADDR + 0x12c	W	0x0000_0000	PCSR_PSCTL.bit[2:0]=0 send COMRESET OOB sequence

Once SATA is linking up, fill in descriptor and data fetching addresses related configurations, enable interrupts, and check if communication established between host and device. With a successful link, the command list will be processed and move further to data transferring stage.

Table 4-10 SATA Controller Ports Initialization

Register Address	Access	Value	Description
`SATA_BASE_ADDR + 0x100	W	0x8210_0000	PCSR_CLB port allocated command list area -- base physical address for the command list for this port
`SATA_BASE_ADDR + 0x108	W	0x8000_0000	PCSR_FB FIS receive area -- base physical address for received FISes
`SATA_BASE_ADDR + 0x118	W	[4]=1	PCSR_CMD bit4=1 FIS receive enable
`SATA_BASE_ADDR + 0x128	R	[3:0]=3 || [3:0]=1	PCSR_SSTS bit3:0 device presence detected
`SATA_BASE_ADDR + 0x114	W	0x0040_003f	PCSR_IE enable required interrupt
`SATA_BASE_ADDR + 0x110	W	0xffff_ffff	PCSR_IS clear interrupt status registers
`SATA_BASE_ADDR + 0x130	W	0xffff_ffff	PCSR_SERR clear error registers
`SATA_BASE_ADDR + 0x128	R	[3:0]=3	PCSR_SSTS bit3:0 device

Register Address	Access	Value	Description
			presence detected and phy communication established (phy ready is detected)
`SATA_BASE_ADDR + 0x120	R	[7]=0&[3]=0&[0]=0	PCSR_TFD bit7=0--interface is not busy, bit3=0--no data transfer requested, bit0=0--no error during transfer
`SATA_BASE_ADDR + 0x118	W	[0]=0	PCSR_CMD bit0=1 start to process the command list
`SATA_BASE_ADDR + 0x8	W	0xffff_ffff	GCSR_IS clear interrupt pending status
`SATA_BASE_ADDR + 0x4	W	[1]=1	GCSR_GHC bit1 enable interrupt

5 Video Codec

5.1 Video Encoder

5.1.1 Overview

VE(Video Encoder) is a hardware-based encoder that supports the H.264/H.265 video standards. It has the advantages of low CPU usage, low bus bandwidth consumption, low latency, and low power consumption

5.1.2 Features

VE has the following features：

- Support ITU-T H.264 Main Profile/Baseline Profile/High Profile/High 10 Profile/High 10 Intra Profile @Level6
- Support ITU-T H.265 Main Profile/Main 10 Profile/Main still Profile @Level6
- Support 1/4 MV accuracy
- Support H.264 Inter PU 16x16, 16x8, 8x16, H.265 Inter PU 2Nx2N, 2NxN, Nx2N
- Support H.264 Intra Prediction 4x4 mode, 8x8 mode, 16x16 mode, H.265 33 directional mode, DC and planar mode
- Support H.264 Transform Block 4x4 8x8, H.264 Transform Block 32x32, 16x16, 8x8, 4x4
- Support H.264 CABAC, CAVLC entropy encoding, H.265 CABAC entropy encoding
- Support tc_offset/beta_offset De-blocking filtering
- Support IPCM encoding
- Support 8/10-bit YCbCr 4:2:0 image input，support 8bit/10bit format conversion
- Support max 4K@60fps encoding
- Support H.264/H.265 the highest resolution of 8192x8192 encoding
- Support H.264 the minimum resolution: 144x128, H.265 minimum resolution 136x128
- H.264 image width align to16 pixels, H.265 image width align to 8 pixels, height align to 2 CTB
- Support ROI
- Support clip, 90°/180°/270° rotation
- Support interlaced input in H.265 mode
- Support rdo level
- Support GDR frame
- Support slice split
- Support SAO
- Support SEI
- Support constCb, constCr
- Support HDR10

5.1.3 Block Diagram

VE block diagram is shown as below

VE is composed of the following functional modules:

- Bus Interface: Access external memory and CPU through the bus interface.
- Pre-processor: Perform format conversion, cropping, rotation, and filling alignment on image input.
- Motion Estimation: Predict motion vectors and inter-frame motion vector variation
- RDO & Mode Decision: Performs rate-distortion optimization and determines the prediction mode during encoding（intra-frame and inter-frame）
- Transform & Quantization: Quantize and transform the input image before encoding.
- Entropy Coding: Replace frequent patterns with fewer bits to achieve lossless encoding
- Bitstream Formation: Generate encoded video bitstream.
- Reference Frame Compression/Decompression: Compressed reference frames are used for future frame prediction
- Inverse Transform: manipulations performed on the encoded macroblocks

- Deblock & SAO filtering: filter that reduces the visible blocking artifacts at block boundaries of decoded compressed video.

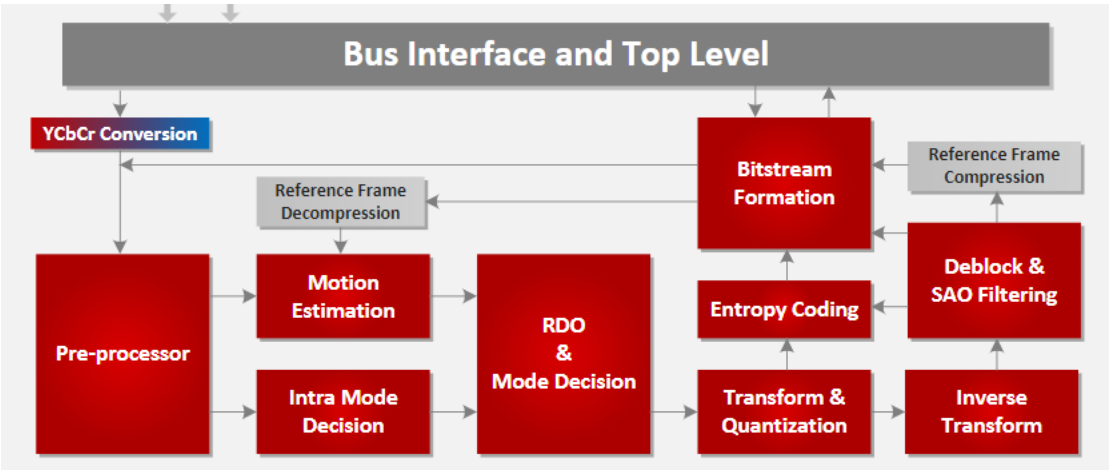


Figure 5-1 Video Encoder block diagram

5.2 Video Decoder

5.2.1 Overview

VD (Video Decoder) is a hardware-based decoder that supports the H.264/H.265 video standards. It has the advantages of low CPU usage, low bus bandwidth consumption, low latency, and low power consumption.

5.2.2 Features

VD features as follow：

- Support ITU-T H.264 Main Profile/Baseline Profile/High Profile/Constrained Baseline Profile/Progressive High Profile @Level6
- Support ITU-T H.265 Main Profile/Main 10 Profile/Main still Profile @Level5.1
- Support typical 16 lanes of 1080P@30 fps decoding
- Support maximum 8192 x 8192@30 fps decoding bitstream
- Support YUV420 semi/planar output format
- Support RGB output format
- Support flexible scale
- Support output picture buffer stride
- Support cropping

5.2.3 Block Diagram

VD block diagram is shown as below。

VD is composed of the following functional modules:

- Busifdtop: Access external memory and CPU through the bus interface.
- Bus Service: Manage and arbitrate all bus transactions required by the processor
- L2 Cache: Save bandwidth for the decoder and reduce DDR access.
- Reconshaper: For reconstructed fram write operation, the reconshaper is introduced to shape AXI write burst with a given alignment.
- Clkctrld: Perform clock management on decoding related clocks
- Decoding Cores: Support corresponding video format decoding
- Post Processor: Provides post processing abilities on reconstructed video, including crop, scaler, CSC, PP output burst alignment and support several output format conversions.

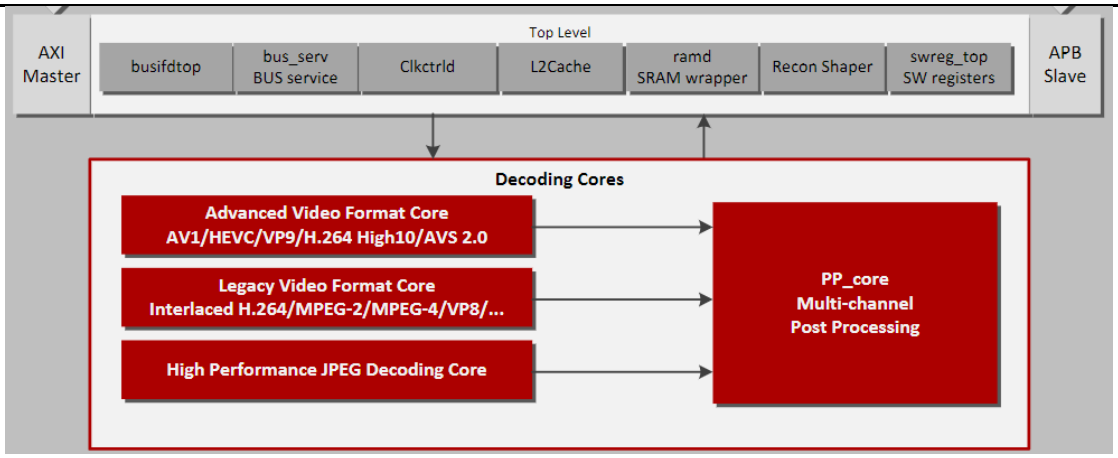


Figure 5-2 Video Decoder block diagram

5.3 JEPG

5.3.1 Overview

JE (JPEG Encoder) and JD (JPEG Decoder) are a set of high-performance JPEG codecs implemented in hardware, capable of encoding and decoding high-definition image MJPEG streams.

5.3.2 Features

JE features are shown as below：

- Support ITU-T JPEG Baseline encoding
- Support multiple input formats: YCbCr 4:2:0 planar/ YCbCr 4:2:0 semi-planar/ YCrCb 4:2:0 planar/YCbYCr 4:2:2 raster-scan/CbYCrY raster-scan/I010/P010
- Support multiple output formats: JFIF file format 1.02/Non-progressive JPEG 4:2:0/MJPEG format(T.81 Annex H)
- Support conversion from 8-bit YCbCr 4:2:2 to YCbCr 4:2:0
- Supports cropping to any encoding size up to a maximum of 32768x32768.， the minimum size is 32x32
- Support 90° /180° /270° rotation
- Support ROI
- Support whole/partial frame encoding
- Support single/multiple markers

JD features are shown as below：

- Support ITU-T JPEG Baseline decoding
- Support multiple input formats: YCbCr 4:0:0 4:2:0 4:2:2 4:4:0 4:1:1 4:4:4
- Support multiple output formats: YCbCr 4:2:0/4:0:0 semi-planar raster scan
- Support the minimum size of 48x48 pixels, maximum size of 32768x32768.
- Supports pixel processing capability up to 4K 160fps. (sequential baseline mode for 4:2:0 format)
- Supports image scaling from 1 down to 1/255

5.3.3 Block Diagram

JE block diagram is shown as below

JE is composed of the following functional modules:

- Bus Interface: Access external memory and CPU through the bus interface.
- Pre-processing：Perform format conversion, cropping, rotation, and filling alignment on image input
- JPEG CORE: Perform JPEG encoding

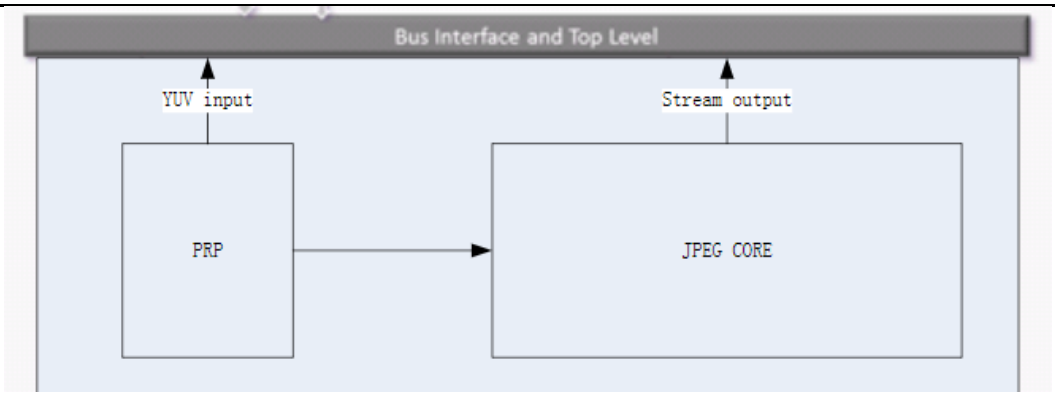


Figure 5-3 JPEG Encoder block diagram

6 Video and Image Processor

6.1 HAE

6.1.1 Overview

HAE is a hardware-based high-performance dual-core 2D raster image processor, designed to accelerate the display of diverse 2D images on the client side, while also supporting post-processing functions after video decoding, or preprocessing data required by AI engines.

6.1.2 Features

HAE features are shown as below:

- Supports image processing capability up to 8K @60fps.
- Support Bit Blit
- Support Stretch Blit
- Support Rectangle fill and clear
- Support One Pass Filter Blit
- Support AlphaBlending
- Support 32767 x 32767 coordinate system
- Support 90° /180° /270° rotation, clockwise and counter clockswise
- Support filp and mirror operation
- Support clipping and cropping
- Support Source/Destination ROP
- Support multi-source blending, maximum 4 lanes video input.
- Support multiple source blending and rotation
- Support 8-bit YUV 2 pixel aligned surface/ 10-bit YUV 4 pixel aligned surface/ RGB 1 pixel aligned surface
- Support separate U and V strides in multi-plane YUV format
- Support full multiple destination conversion from non-planar YUV to planar YUV
- Support YUV420 2-plane/YUV422 packed with alpha blending output
- Support conversion from UYVY/YUY2/YV12/NV12/NV16 to UYVY/YUY2 format
- Support conversion from YUV to RGB，from RGB to YUV BT2020
- Support multiple 10-bit formats: A2R10G10B10, YUV420 2-plane, 和 YUV422 2-plane
- Support FP16/FP32 normalization
- Support dither operarion
- Support Gamma
- Support Line operation
- Support ColorKey
- Support MaskBlit
- Support Monochrome

- Support Premultiply
- Support transparency
- Support Programmable CSC

6.1.3 Block Diagram

2D GPU single core block diagram is shown as below

2D GPU is composed of the following functional modules:

- AXI Front End: Lossless AXI trasactions compression and decompression for bandwidth optimization
- Host Interface: Access external memory and CPU through the bus interface.
- Memory Controller: intermal memory unit which serves as the block-to-host interface for memory requests
- Graphics Pipeline Front End: Insert high level primitives and commands into the graphics pipeline
- Draw Engine: Draws 2D graphics primitives and rasterizes 2D images
- Pixel Engine: Pixel manipulation and filtering on rendered images.
- PP Normalization: Data post processing for AI engine after blit or OPF filter operations.

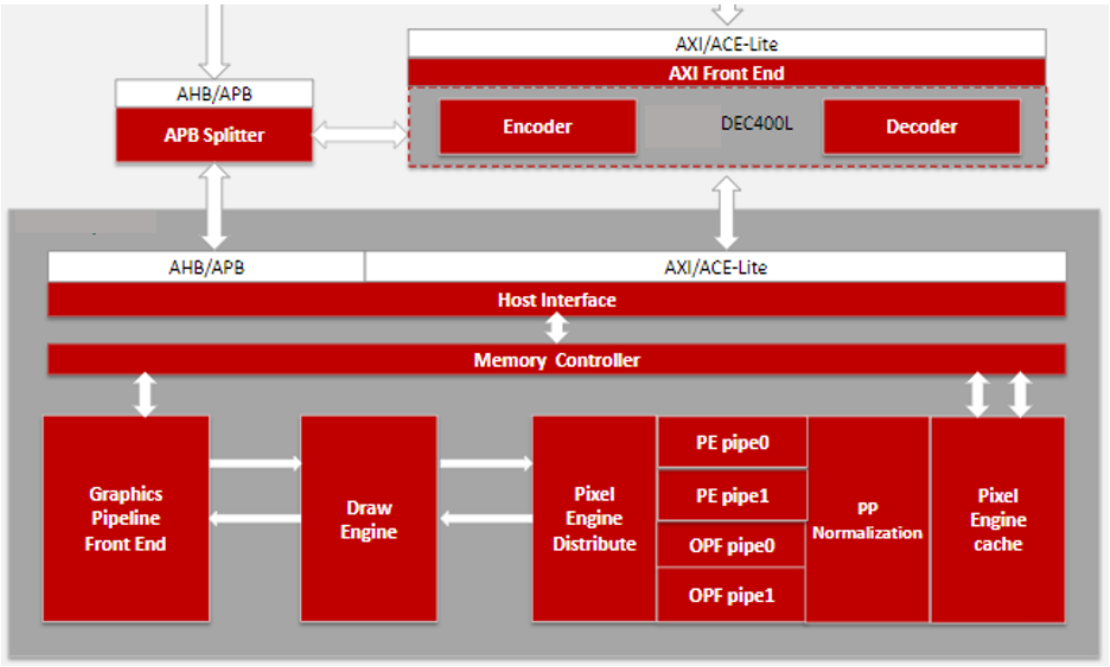


Figure 6-1 2D GPU block diagram

6.2 3D GPU

6.2.1 Introduction

The module function supports the processing of various load types, and can complete the processing of 3D scene vertex data and pixel data rendering 3D graphics; Also supports general purpose computing processing (GP-GPU); It also supports pixel data processing for rendering 2D objects.

6.2.2 Feature

The A-Series cores process a number of different workload types, namely:

• 3D Graphics Workload, which involves processing vertex data and pixel data for rendering of 3D scenes

• Compute Workload (GP-GPU), which involves general purpose data processing

• 2D Workload, which involves processing of pixel data for rendering 2D objects.

The 2D workload is structured as a series of 2D render packets by the driver, and hese are known as blits.

The GPU architecture is fully OpenGL ES 3.2, EGL 1.4, OpenCL 1.2 and 2.1 EP1, Vulkan 1.2 and Android NN HAL compliant.

6.2.2.1 GPU Key Features

This GPU core has the following features:

- Base architecture, fully compliant with the following APIs:
  - OpenGL ES 3.2
  - EGL 1.4
  - OpenCL 1.2 and 2.1 EP2 • Vulkan 1.2
  - Android NN HAL
- Tile-based deferred rendering architecture for 3D graphics workloads, with concurrent processing of multiple tiles
- Robust buffer access, ensuring that the hardware does not access data outside of the defined resource
- Tile processing time tracking for 3D fragment processing phase, which aids with performance debugging from a hardware perspective
- Programmable high quality image anti-aliasing
- Fine grain triangle culling
- Support for DRM security
- Support for GPU virtualization
  - up to 8 virtual GPUs
  - Support for IMG hyperlane technology, with 8 hyperlanes available
- Support for Imagination AI Synergy when paired with an Imagination NNA (Neural Network Accelerator) core
- Asynchronous Fast 2D Renders
- Multi-threaded Unified Shading Cluster (USC) engine incorporating pixel shader, vertex shader and GP-GPU (compute shader) functionality
- USC support for tessellation through hull shaders and domain shaders.
- USC incorporates an ALU architecture with high SIMD efficiency
- Fully virtualized memory addressing (up to 1 TB address space), supporting unified memory architecture
- Fine-grained task switching, workload balancing and power management
- Advanced DMA driven operation for minimum host CPU interaction
- System Level Cache (SLC) ,128KB
- Specialised Texture Cache Unit (TCU)
- Compressed Texture Decoding

- Lossless data compression (PVRGC) - The PowerVR's geometry compression, which is performed in the Geometry Processing phase of the 3D graphics workload.
- Dedicated processor for A-Series core firmware execution
    - Dual-threaded with a 16-KB instruction cache, a 2-KB data cache, and core memory.
- Full Data Master task removal
- Architecture is constructed of a number of SPUs (Scalable Processing Units) and one common system level module containing system level interfaces and shared modules (Jones)
- Separate Power Island for each SPU
- On-Chip Performance, Power and Statistics Registers.

6.2.2.2 Unified Shading Cluster Features

- 128 parallel instances per clock
- Local data and instruction caches
- Variable length instruction set encoding
- Full support for OpenCL™atomic operations (including compare exchange)
- Scalar and vector SIMD execution model
- Support for F16 data type in complex ALU
- 64 bit global atomics
- Flat addressing
- Split Shared and Coefficient Stores
- Bindless image/texture support

6.2.2.3 3D Graphics Features

Rasterisation

- Deferred Pixel Shading
- On-chip tile floating point depth buffer
- 8-bit stencil with on-chip tile stencil buffer
- Maximum tiles in flight (per ISP): 4
- 32 parallel depth/stencil tests per clock for MSAA renders and 16 parallel depth/ stencil tests per clock for non-MSAA renders
- One ISP per USC
- Context switching

Texture Lookups

- Load from source instruction support

Filtering

- Sample details: sample data and coefficient support
- Bilinear, volume and trilinear filtering
- Anisotropic filtering

- Corner filtering support for Cube Environment Mapped textures and filtering across faces
- F32/U32/S32 Border Colour support
- Chroma interpolation for YUV 420/422 formats

Texture Formats

- PVRTC I and II compressed texture formats
- ASTC LDR compressed texture format support
- ETC/EAC compressed texture format support
- Texture Array support - up to 2K layers
- Buffer texture types - up to 2 27 elements
- YUV planar support, 1, 2 and 3 planar - 420/422/444 formats 8-bit and 10-bit
- 10 bit sRGB and YUV format support

Resolution Support

- Frame buffer max size = 16K × 16K
- Texture max size = 16K × 16K

Anti-aliasing

- Maximum 8× multisampling

Primitive Assembly

- Early hidden object removal
- Vertex compression
- Tile acceleration

Render to Buffers

- Twiddled format support
- Multiple on-chip render targets (MRT)
- Lossless and/or visually lossless Frame Buffer Compression (and Decompression)
- Programmable Geometry Shader Support
- Direct Geometry Stream Out (Transform Feedback)
- Parallel Stream Out

6.2.2.4 Compute Features

- 1, 2 and 3 dimensional compute primitives
- Maximum workgroup size is 1024 work-items
- Per task input data DMA (to USC Unified Store)
- Conditional execution
- Execution fences

- Compute workload can be overlapped with any other workload
- Compute overlap uses barriers
- Round to zero
- Call/Return/Pre-Emption support

6.2.2.5 Performance

The performance characteristics of the GPU core are theoretical maximum performance with the architecture running at 100% efficiency.

Feature	Performance
Floating Point Operations (F32)	256 operations per clock
Floating Point Operations (F16)	256 operations per clock
Integer Operations	64 operations per clock
Geometry Performance	0.5 triangles per clock
Texture performance	8 texels per clock (@32 BPP)
Pixel performance	8 pixels per clock (@32 BPP)

Floating Point Operations (F32) = (Number of USCs) * (Number of Parallel FP Instance per cycle per USC) * (Number of F32 operations per instance)

- Number of Parallel FP Instance per cycle per USC = 128
- Number of F32 operations per instance = 2

Integer Operations = (Number of USCs) * (Number of Parallel INT Instance per cycle per USC) * (Number of Integer operations per instance)

- Number of Parallel INT Instance per cycle per USC = 32
- Number of Integer operations per instance = 2

6.2.3 Block Diagram

The chip integrates IMG's IMG_A-Series_AXM-8-256_30.V.408.101 3D-GPU IP. The System works cooperatively with IMG 3D-GPU through System Bus interface (SOCIF). IMG 3D-GPU reads and writes the system Memory using the Memory Bus (MEMIF).

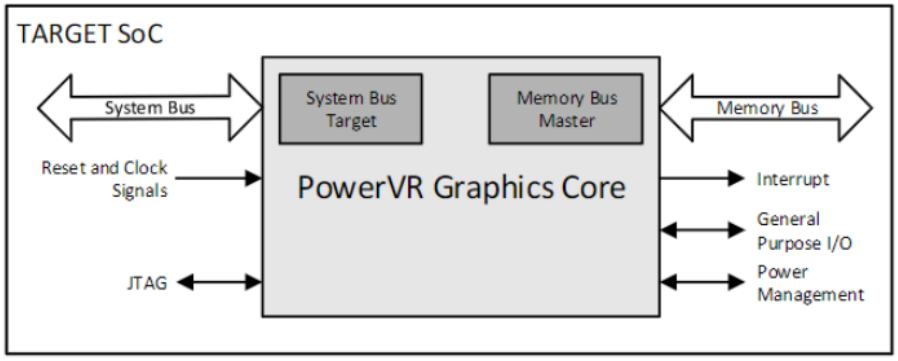


Figure 6-2 3D-GPU IP system integration diagram

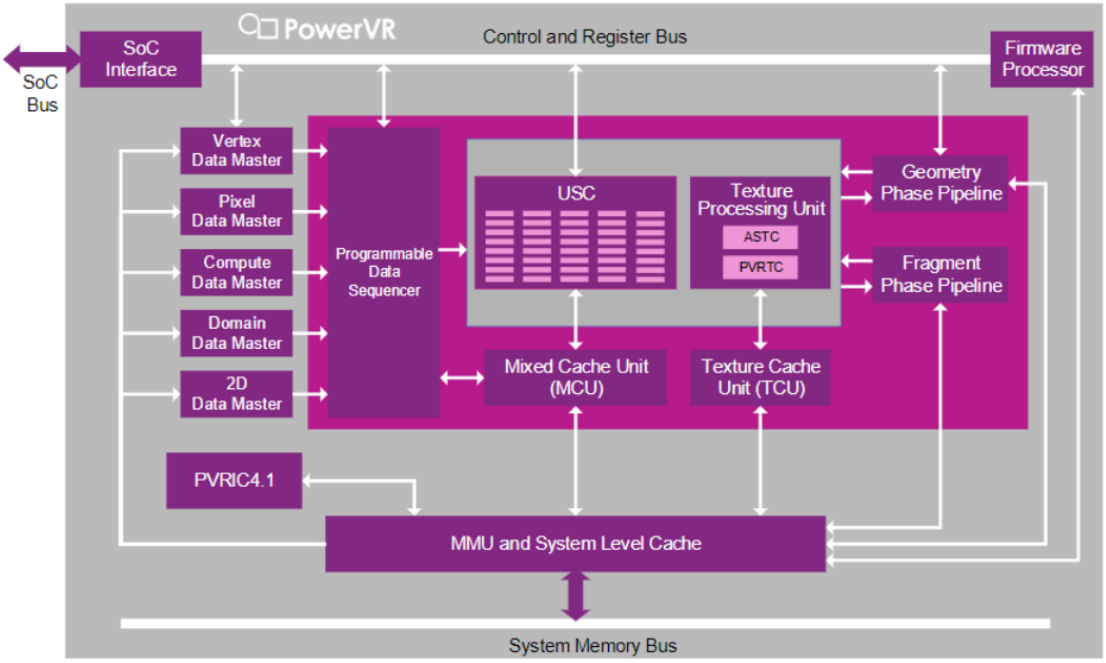


Figure 6-3 AXM-8-256 block diagram

6.2.3.1 Top-Level Interface Characteristics

SOCIF AXI Specification

The SOCIF is the AXI Slave interface through which the system accesses the GPU internal registers. The specifications of the interface are as follows:

Table 6-1 SOCIF AXI Specification

Feature Type	Feature Implementation
AXI type	AXI3
Master or Slave	Slave
Burst attribute	Bursts are not supported on the SOCIF
Address bus width	32 bits
Data bus width	32 bits
Tag ID width	AXIBUS_SOCIF_TAGIDS_WIDTH
Number of IDs	2^AXIBUS_SOCIF_TAGIDS_WIDTH
Max number of outstanding reads	24
Max number of outstanding writes	5
Interleaving	Write Interleaving is not supported
Sideband signal	N/A

MEMIF AXI Specification

MEMIF is the AXI Master interface through which the GPU reads and writes Memory data. The interface specifications are as follows:

Table 6-2 MEMIF AXI Specification

Feature Type	Feature Implementation
AXI type	ACE-Lite
Number of memory interfaces	1
Master or Slave	Master
Burst attribute	Max Burst: 4 beats Burst type: Incrementing (fixed and wrapped burst types are not used by the GPU)
Address bus width	40
Data bus width	256
Read Tag ID width	6 bits
Write Tag ID width	6 bits
Number of Read IDs	64
Number of Write IDs	64
Max number of outstanding reads	64
Max number of outstanding writes	64
Sideband signals	`axi_mst0_aruser[4:0]`: Requestor ID (read) `axi_mst0_awuser[4:0]`: Requestor ID (write) `axi_mst0_aruser[7:5]`: OSID (read) `axi_mst0_awuser[7:5]`: OSID (write)
Supported transaction types	The A-Series will issue the following transactions on the AR/AW channels: • ReadNoSnoop • ReadOnce • WriteNoSnoop • WriteUnique

6.3 ISP

6.3.1 Overview

ISP subsystem, include 2 processors, is responsible for the image processing.

Key feature are:

- Supports In-built Test Pattern Generator(TPG)
- Black level Compensation
- Defect Pixel Cluster Correction(DPCC)
- Green Equalizaiton, Lens Shade Correction(De-Vignetting)
- 2D/3D Noise Reduction
- Enhanced Color Interpolation(Bayer De-mosaic filter)
- Chromatic Aberration Correction(CAC), Color Correction(Xtalk) Matrix(CCM)
- Color Space Conversion(CSC), Programmable gamma correction for sensor adaptation and display correction
- Auto Focus measurement(AF), Auto White Balance measurement(AWB), Auto Exposure measurement(AE)
- Color Processing: Contrast, Saturation, Brightness, Hue(CPROC)
- Resize and Cropping

6.3.2 Block diagram

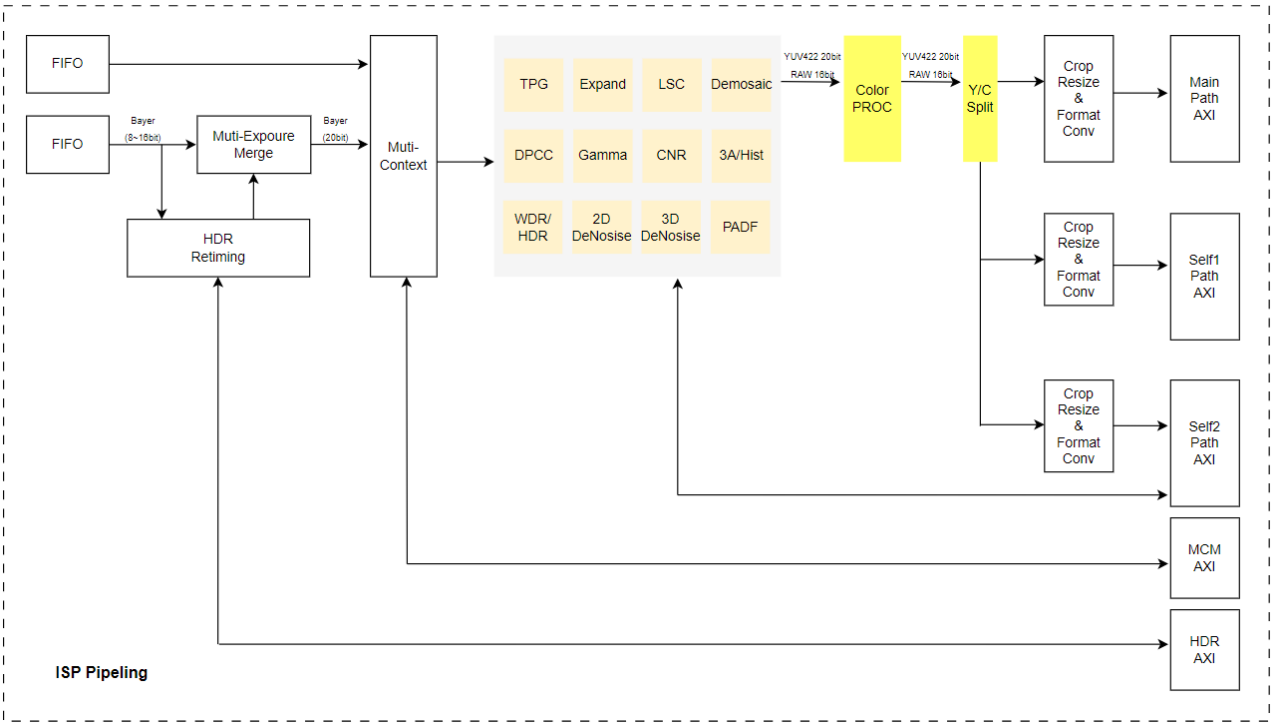


Figure 6-4 ISP Pipeline block diagram

6.3.3 Function description

6.3.3.1 Local address map

Table 6-3 ISP Local address mapping

Interface	Address region	description
AHB Slave	0x5100_0000~0x5100_ffff	ISP0 Csr register space
AHB Slave	0x5101_0000~0x5101_ffff	ISP0 Csr register space

6.3.3.2 Interrupt

ISP can send interrupt signals to host processor. Interrupts are distributed in different sub-modules, each of them is controller by a group of registers. The signal names are mi_irq and isp_irq, and they remain asserted until the host processor clears the interrupt by writing the interrupt clear register(XXX_ICR)

Table 6-4 ISP Interrupt Description

Signal Name	Width	I/O	Description	Clock Domain
isp_irq	1	OUT	ISP Interrupt	CLK
mi_irq	1	OUT	Memory Interface Interrupt	CLK
fe_irq	1	OUT	Fast register Configuration interrupt	CLK

All sub module masked interrupts are ORed together to drive the interrupt line, Five interrupt register for each module which can generate interrupts:

- Interrupt mask Register(IMSC)

- Raw interrupt status register(RIS)
- Masked interrupt clear register(MIS)
- Interrupt clear register(ICR)
- Interrupt set register(ISR)

6.3.3.3 Clock

There are four independent clock domains in the ISP, automatic localized clock gating is employed throughout the design to minimize the dynamic power consumption. Almost all the flip-flops are clock gated in the design. Block level clock gating is implemented in most of the blocks, if a block and the interface to the block are both idle, then the clock of that block will be gated automatically, this feature can be disabled by software.

6.3.3.4 Reset

ISP has a safe reset mechanism to make sure that the AXI transaction is completed before reset is asserted. The memory interface module could generate an interrupt to clarify that all axi AXI transaction are completed. After this interrupt is received, the system could assert a reset signal to reset all the logic in this module.

Figure 6-5 ISP Asynchronous Reset Mechanism

6.4 Distortion Correction

6.4.1 Overview

The Distortion Correction module include dewarp and scalar sub-module. The high performance dewarp processing module allows for the correction of the distortion that is introduced in image produced by fisheye and wide angle lenses, it is implement with a line/tile-cache based architecture, with configurable address mapping look up tables and per tile processing, it successfully generates a corrected output images. The Scalar processing provides high-performance scalar processing for yuv image.

Key feature are:

- Line based architecture and configurable map to support various lens distortions
- YUV422 and YUV420 inputs and outputs
- Configurable bilinear interpolation engine
- Grid map with 16x16 pixel macroblocks indexed using X,Y vertex coordinates
- Fisheye correciton
- Wide Field of View(FOV) correction
- Keystone correction
- Three scaling engines, with maximum resolution of: 4K,1080p and 1080p
- Supports YCbCr422 and YCbCr420 semi-planar inputs
- Supports YUV and RGB888 output
- Supports different output formats by choosing different scaling factors for luminance and chrominance component: YUV444,YUV422,YUV420,RGB888
- Separate scaling in horizontal and vertical directions, and for chrominance and luminance components
- Lookup table for programmable scaling characteristics

6.4.2 Block diagram

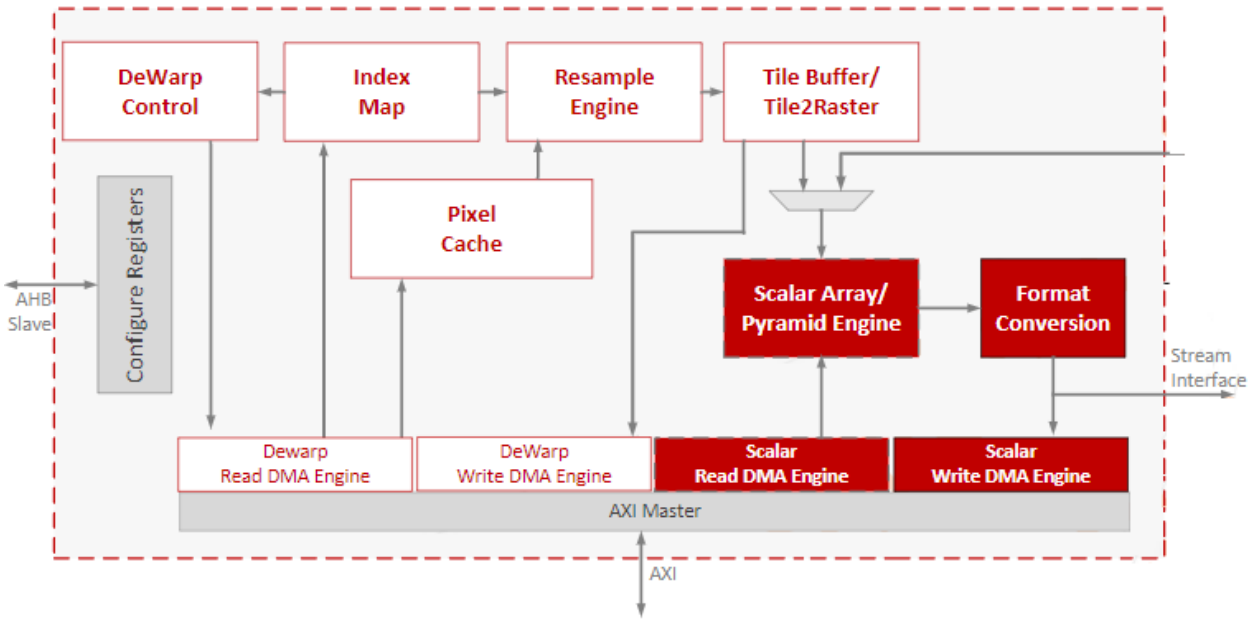


Figure 6-6 Distortion Correction block diagram

6.4.3 Function description

6.4.3.1 Local address map

Table 6-5 ISP Local address mapping

Interface	Address region	description
AHB Slave	0x5102_0000~0x5102_ffff	Distortion correction Csr register space

6.4.3.2 Interrupt

The Distortion Correction can send interrupt signals to the host process, the dwe_int and vse_int, once set interrupt will remain asserted until the host processor clears the interrupt by related interrupt clear registers(*_icr/icr1).

6.4.3.3 Start sequence

The diagram below illustrates the dewarp_start signal example timing.

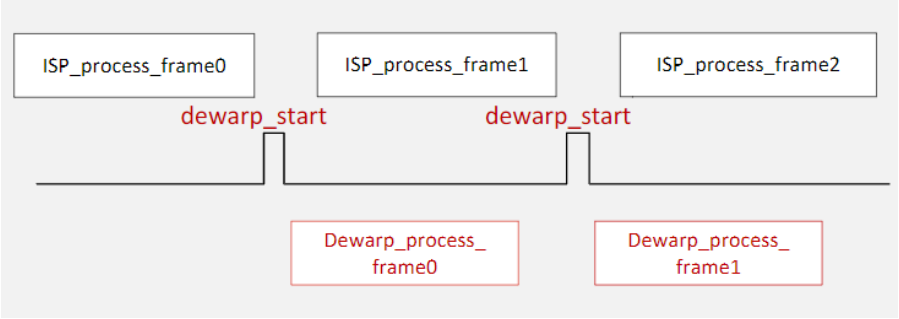


Figure 6-7 Timing for dewarp frame processing

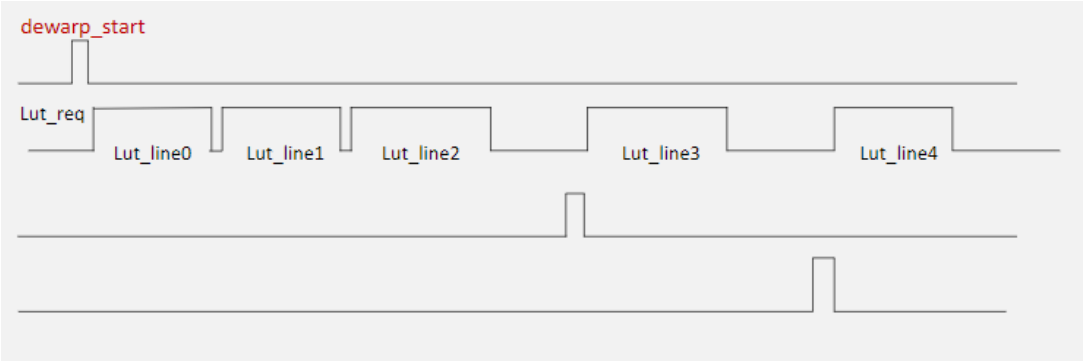


Figure 6-8 Timing for LUT address Requests

7 Intelligent Accelerator

7.1 DSP

Since we don't have permission from Cadence to release DSP compiler to users, thus we prefer not to publish too much information on DSP subsystem, only key highlights are being listed in this chapter. If users would like to know more about DSP subsystem including configurations and usages, please request information.

7.1.1 Overview

DSP subsystem is responsible for the acceleration of the computer vision and the DNN. DSP subsystem supports the following key features:

- 128 16-bit MAC, 512 8-bit MAC, 64 Half Precision Floating MAC, 32 Single Precision Floating MAC.
- Each core can work and reset independently, and running at configurable 520MHz or 1040MHz clock.
- Each core has 64KB ITCM(instruction TCM) and 256KB DTCM(data TCM).
- Each core has 16-entry embedded MPU.
- Each core supports external interrupt from mailbox and interrupt controller.
- Each core has 4 iDMA channel and supports trigger interface consists of TrigIn_iDMA and TrigOut_iDMA.
- Scatter Gather.

7.1.2 DSP Reset Sequence and Initialization

7.1.2.1 Set Static Registers

Certain registers have specific requirements with regard to reset, these registers are inputs to the processor and sampled by the processor during its bring-up. They must be held for 10 processor clock cycles after the deassertion of the breset of DSP cores.

7.1.2.2 Load Firmware

The reset vector is the address of the first instruction fetch when the processor reset is deasserted. This vector can be set through a register by software.

7.1.2.3 Take DSP subsystem out of reset

Taking DSP subsystem out of reset should follow the sequence below:

- Deassert DSP subsystem resets.
- Deassert DSP breset of DSP cores.
- Deassert DSP runstallonreset of DSP core (at least 10 processor clock cycles after the deassertion of corresponding breset)
- Deassert DSP dreset of DPS cores(optional, dreset is for debug usage)

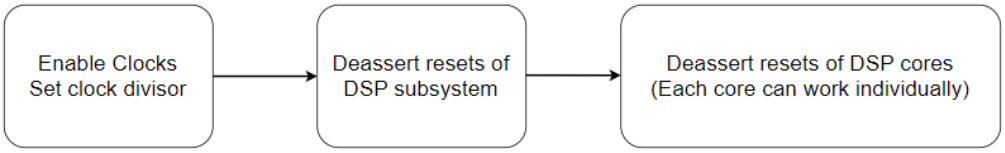


Figure 7-1 DSP subsystem taken out of reset

7.2 NPU

7.2.1 Overview

DNN accelerator is a specific function acceleration engine designed specifically for deep learning networks, which can accelerate the computation of image and speech neural networks. It can support multiple types of network layer operations such as convolution, deconvolution, fully connected, activation, pooling, batch normalization, matrix multiplication, and so on

The DNN accelerator supports the following features:

- Support arbitrary shape kernel convolution, deconvolution, hole convolution, matrix multiplication
- The hardware adopts flexible mapping to support high utilization of various network layers for computing.
- Support int8, uint8, int16, float16 data types
- 6144 MAD (multiply-add) int8 operations per cycle
- 3072MAD (multiply-add) int16/fp16 operations per cycle
- The maximum clock speed can reach 1.5Ghz, and can be dynamically reduced
- Support pooling (min, max, average)
- The maximum pooling kernel size is supported up to 8
- Support weight coefficient compression storage and compression calculation
- Support online compression calculation of feature maps
- Support any dimension padding
- Support for activation operations, including PreLU, ReLU, sigmod, tanh, and nonlinear activation
- Support addition, min, max, multiplication, batch norm in element units
- Support NCHW segmentation in various dimensions
- Support channel dimension segmentation and merging, and support output in NCHW format
- Support NCHWC0 data format, C0 dimension can be 1, 2, 4, 8, 16, 32, and the first layer supports C0=3 (RGB) in particular
- Support image batch processing

7.2.2 Block diagram

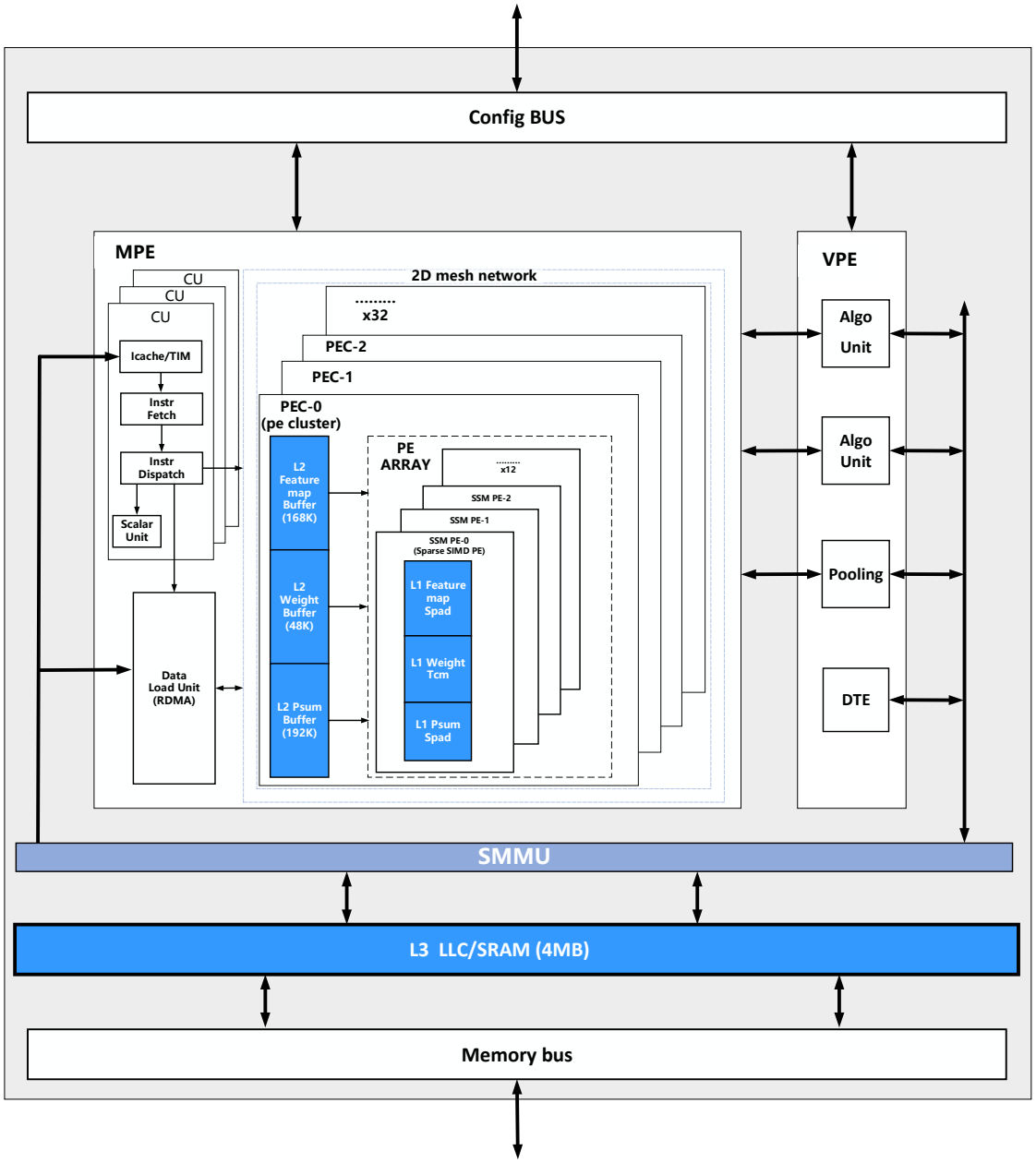


Figure 7-2 NPU Architecture

7.2.3 Function description

7.2.3.1 Config bus，Memory bus

The system CPU accesses the DNN accelerator registers through the Config bus to start and schedule DNN operations. The CU units within the MPE module can also access the register address space of each module within the DNN accelerator through the Config bus. Each functional module within the DNN accelerator accesses the system memory through the Memory bus.

7.2.3.2 MPE(Matrix Process element)

The MPE module is the main processing unit that undertakes neural network operations within the DNN accelerator. It consists of a CU unit, a RDMA unit, a PEC array unit, and a 2Dmesh network. It has flexible and efficient computing capabilities, with the CU unit as its master, scheduling the RDMA unit and PEC unit to work together. The PEC unit and RDMA unit can perform various types of

multiplication and addition operations with different granularities. The 2D Mesh network has the ability to broadcast, multicast, and unicast, and can be configured into suitable data broadcast modes according to different network shapes.

7.2.3.3 VPE(Vector Process element)

The VPE module performs vector-assisted operations within the DNN accelerator. Non-matrix operations such as activation operations, linear/non-linear operations, element-wise operations, pooling operations, and arbitrary data shape conversions in neural network operations are all performed by the VPE module.

7.2.3.4 L3 LLC/SRAM(last level cache 4MB)

The DNN accelerator is equipped with a SRAM that can be software-configured as a cache or SRAM, with a total size of 4MB.

7.2.4 Initialization sequence

- The main control CPU initializes the firmware of the CU unit
- Initialize the MPE unit in the CU unit
- Initialize the VPE unit in the CU unit
- MPE unit starts to work
- The VPE unit starts to work
- CU unit configures MPE unit for the next operation
- CU unit configures VPE unit for the next operation
- MPE completes an operation, triggers an interrupt to report to the CU, and starts the next operation
- The VPE completes an operation, triggers an interrupt to report to the CU, and begins the next operation.
- Repeat 6-9

8 Video Input

8.1 Overview

The VI subsystem includes six Combo-PHYs, six CSI2-Hosts, and VICAPs, as well as two ISP8000s, DW200s, DVP2AXI, and Shutter modules. The Combo-PHYs are used to de-serialize sensor data and generate parallel data. The PHYs can be configured in MIPI, LVDS, SubLVDS, and IO modes. The CSI2-Hosts serve as controllers for MIPI, while the VICAPs serve as controllers for LVDS/HiSPi/SubLVDS. The ISP8000s are image processing modules. The DW200s are distortion correction modules. The DVP2AXI is a DMA module for image data write-in. The Shutter modules serve as synchronization control modules for primary and secondary cameras.

8.2 Block Diagram

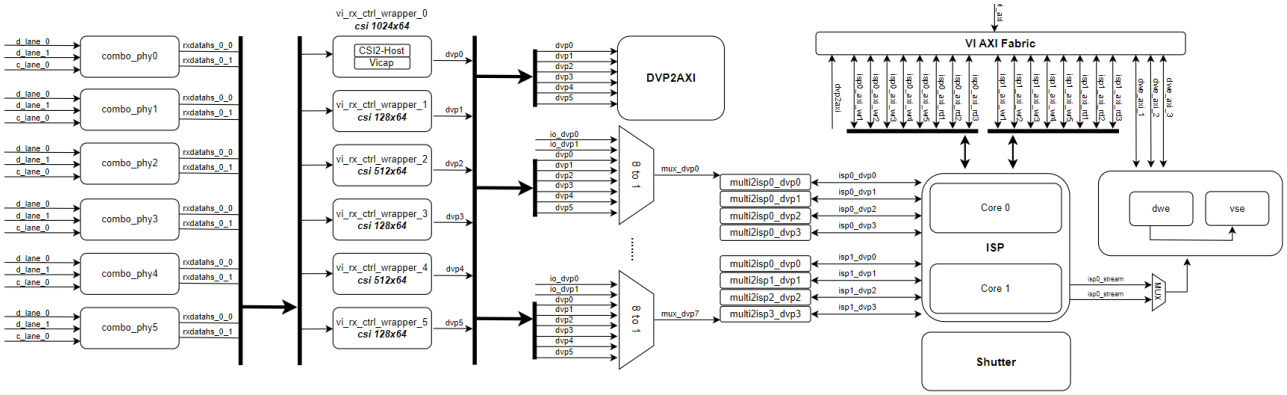


Figure 8-1 Video in block diagram

8.3 Function Description

8.3.1 Combo PHY

Video In subsystem include 6 Combo PHY for Camera data input.

Key feature are:

- MIPI Alliance Specification for D-PHY Version 2.1 support
- MIPI Alliance Specification for C-PHY Version 1.2 support
- Shared analog pins implementation (10 pins for 2 data lanes / 1 clock lane in D-PHY and 9 pins for 2
- trios in C-PHY)
- HS RX Passive Equalization and offset cancellation
- Flexible configuration clock (17-38.4 MHz)
- Advanced Peripheral Bus (APB) interface to access test control registers
- Mandatory external reference resistor (200 ohm) for calibration support
- PHY testability
- Power Collapsing
- SUB_LVDS/SLVDS/HiSPi
- SUB_LVDS/SLVDS/HiSPi Aggregation Mode (PHY can work as standalone or as aggregation mode:
- rx2l+rx2l=rx4l)
- GPIO

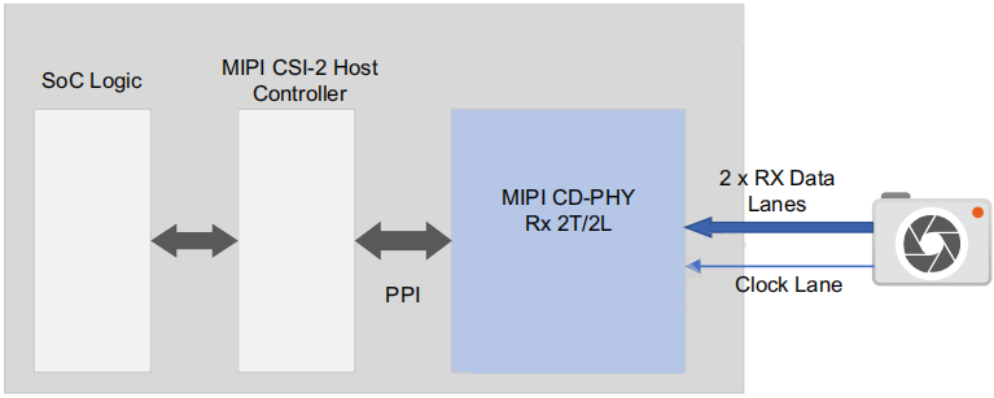


Figure 8-2 MIPI D-PHY System-Level Block Diagram

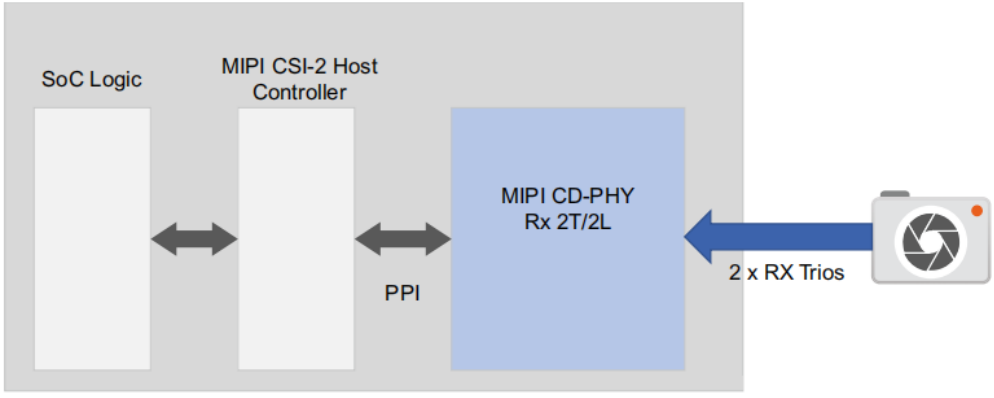


Figure 8-3 MIPI C-PHY System-Level Block Diagram

8.3.2 CSI2 Host

Video In subsystem include 6 CSI2 controller, it receives data from a CSI2 compliant camera sensor. An RX DPHY or Combo PHY acts as the physical layer.

Key feature are:

- Compliant with MIPI Alliance standards
- PHY-Protocol Interface(PPI) to MIPI DPHY, Supports PPI-8/PPI-16
- PHY-Protocol Interface(PPI) to MIPI CPHY, Supports PPI-8/PPI-16
- Dynamically configurable multi-lane merging
- Long and Short packet decoding
- Timing accurate signaling of Frame and Line synchronization packets
- 32-bit or 64-bit Image Data Interface(IDI) delivering data formatted as recommended in CSI-2 Specification
- Error detection and correction
- PPI Data Lanes De-Skew capability
- PPI Pattern Generator delivering data formatted
- Data scrambling for electromagnetic interference mitigation

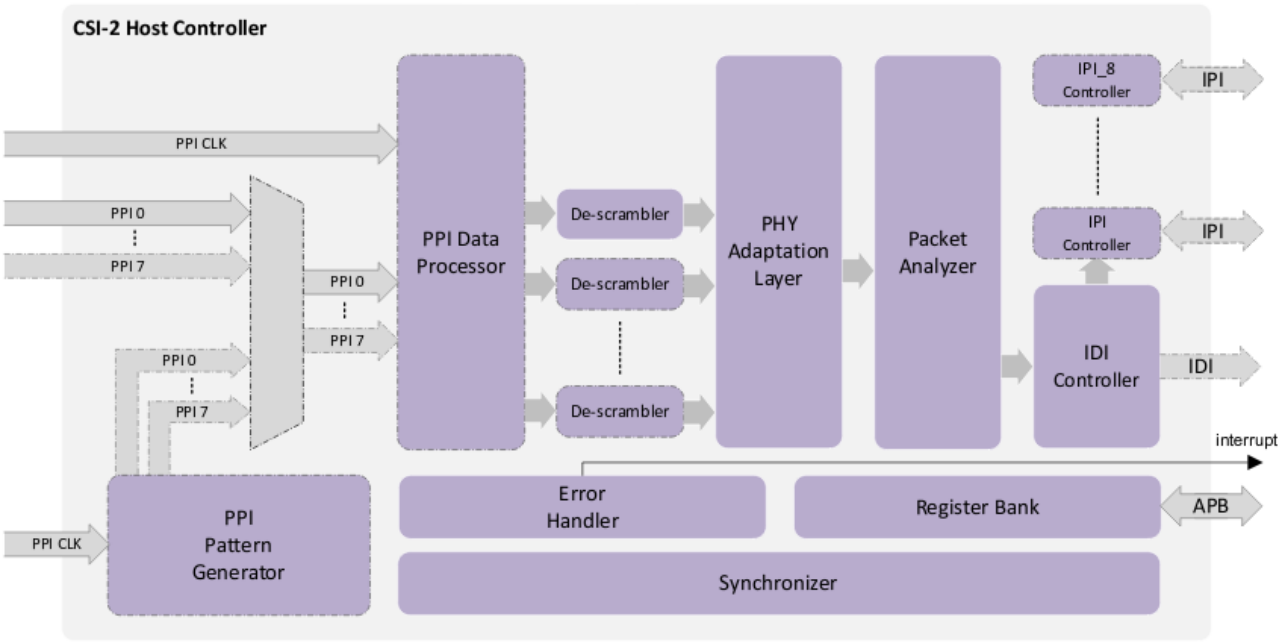


Figure 8-4 CSI-2 Host Block Diagram

8.3.3 VICAP

It converts the Combo-PHY serial video data received into 8-bit parallel data per clock for splitting and splicing. Then it extracts the synchronization code, decodes the video pixel data, and synchronizes the multi-Lane parallel word data, which contains all data of the HiSPi, SLVS, and SUB-LVDS protocols (including synchronization codes and other invalid information). The current module performs protocol analysis, separates the synchronization code and invalid information, and only sends the valid pixel and valid embedded data to the ISP end or writes them to memory through the DVP2AXI module. HiSPiRX specifically analyzes various sub-modes of the HiSPi protocol and outputs AIL (without crop and line tail invalid data discard); SLVS RX is used to parse the SLVS SUB-LVDS protocol and output pixel data and info data as well as OB data (without crop and line tail invalid data discard); crop dvp out mainly performs data serialization/deserialization conversion, performs crop and line tail invalid data discard,

and outputs effective data with ID through the DVP interface.

Key feature are:

- Supports SUB-LVDS, SLVS, and HiSPi protocols.
- Supports pixel data bit widths of 8bit/10bit/12bit/14bit/16bit configurable.
- Supports synchronization code MSB/LSB and pixel data MSB/LSB configurable.
- Supports 1Lane/2Lane/4Lane/8Lane modes.
- Supports 5-word synchronization code in SLVS and SUB-LVDS modes (requires Lane replication transmission)

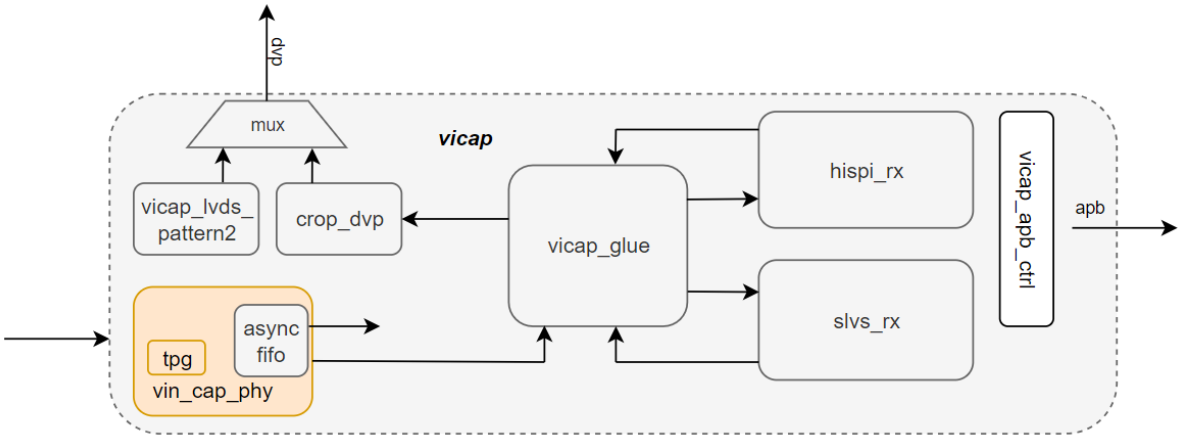


Figure 8-5 VICAP Block Diagram

8.3.4 DVP2AXI

Key feature are:

- Support for 6 simultaneous DVP interface inputs, of which 2 can be input from IO DVP
- Support for up to 3 virtual channels of multi-channel data input on 1 DVP, i.e. HDR mode
- Support for RAW8bit/10bit/12bit/14bit/16bit, RGB888, YUV8bit/10bit data input formats
- Support for YUV/RGB packed format stored in DDR
- Support for AXI outstanding with a maximum of 256 configurable options, AXI burst length with a maximum of 8/16, and data bit width of 128bit
- The starting address is aligned to 16 bytes, and support for writing embedded data to different address areas
- Support for virtual channel frame write complete interrupt and non-complete frame detection interrupt
- Support for exception handling, such as ensuring the integrity of AXI bus transmission during reset

8.3.5 Shutter

The Shutter module is mainly used in industrial inspection with two sensors for binocular vision. It is controlled by the SOC terminal to expose simultaneously and transmit data simultaneously to ensure synchronization between the two sensors.

Key feature are:

- XTRIG output mode supports external IO input Pulse/Level trigger, and the output pulse width of Pulse mode can be configured
- XTRIG output mode supports register configuration trigger XTRIG, and the output pulse width can be configured
- XVS and XHS output modes support VMAX and HMAX configurable, and XHS duty cycle can be configured
- XHS and XTRIG output modes support XHS features configurable, and XTRIG and XHS phase difference can be configured
- Supports loop count of 1 to 1024 configurable, or can configure infinite loop

- Supports two sets of configuration registers pipeline to generate corresponding output signals
- Supports polarity and phase half-cycle delay configuration for XTRIG and XVS output signals
- Supports XTRIG and XVS sharing the same PAD with selectable options to reduce chip peripheral connections
- Supports completion (sequential output or completion of all outputs) interrupt and Stop completion interrupt, idle status can be queried
- Supports rising edge/falling edge (configurable option) interrupt generation for XTRIG and XVS
- Supports software configuration of Stop end task, and ensures the current frame timing is intact

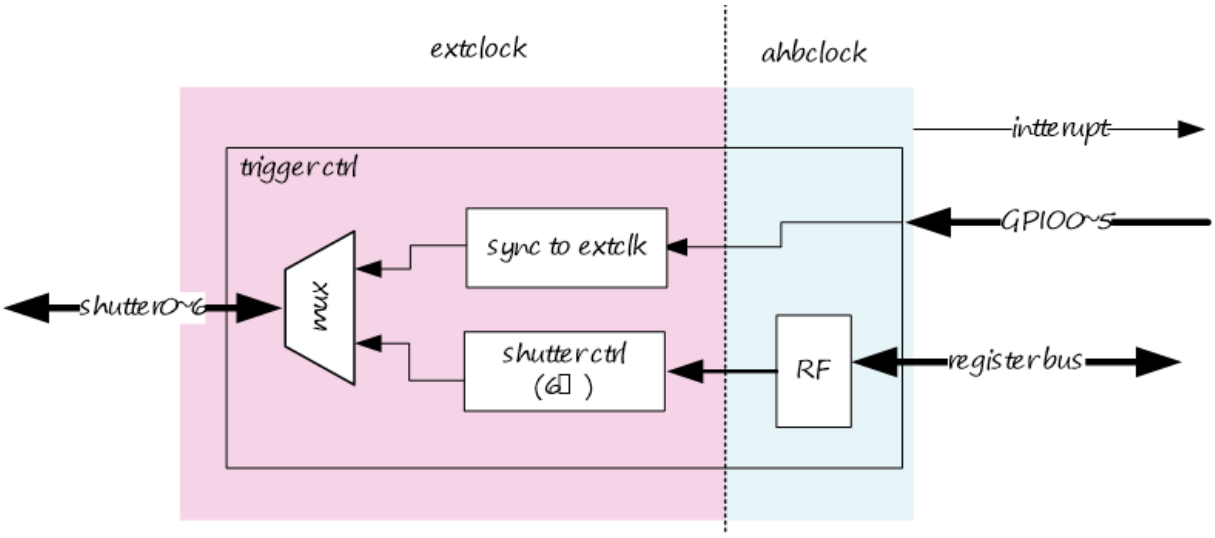


Figure 8-6 Shutter Block Diagram

It includes two parts. First, the shutter can be controlled by the GPIO of the chip through the upper computer, or it can be controlled by the AP inside the SOC. The specific shutter timing is determined by the register configuration. The hardware provides loop mode and free running mode, which outputs a fixed shutter control sequence according to the timing and frequency configured by the software. It can also be configured by the software to run freely. The shutter can be stopped by the software, and the hardware ensures the integrity of the shutter controlled by the current output frame.

8.4 Configure Sequence

8.4.1 Combo-PHY MIPI StartUp

Combo-PHY configuration sequence (the difference between CPHY and DPHY is mainly the phy_mode register):

1. Reset the entire subsystem.
2. Power up the required PHY.
3. Configure the phy interface control signals as shown in the figure below.

Register Name	Value	Bit	Address	
enable_dck	1	[4]	0x510c_c000	combo-phy0
enable_0	1	[5]	0x510c_c000	combo-phy0
enable_1	1	[6]	0x510c_c000	combo-phy0
phy_mode	0	[0]	0x5105_001c	csi2_host0
forcexnmode_0	1	[7]	0x510c_c000	combo-phy0
forcexnmode_1	1	[8]	0x510c_c000	combo-phy0
forcexnmode_dck	1	[9]	0x510c_c000	combo-phy0

4. Configure the relevant internal registers of the phy, see the phy datasheet for details.
5. Configure phy's shutdown_n = 1

Register Name	Value	Bit	Address	
shutdown_n	1	[0]	0x5105_0040	csi2_host0

6. Polling the phy_ready register until the signal goes high, indicating that the phy configuration is complete

Register Name	Value	Bit	Address	
phy_ready	/	[0]	0x510c_c004	combo-phy0

8.4.2 Combo-PHY LVDS StartUp

1. Configure the phy interface control signals as shown in the figure below

Register Name	Value	Bit	Address	
enable_dck	1	[4]	0x510c_c000	combo-phy0
enable_0	1	[5]	0x510c_c000	combo-phy0
enable_1	1	[6]	0x510c_c000	combo-phy0
phy_mode	0	[0]	0x5105_001c	csi2_host0
forcexnmode_0	1	[7]	0x510c_c000	combo-phy0
forcexnmode_1	1	[8]	0x510c_c000	combo-phy0
forcexnmode_dck	1	[9]	0x510c_c000	combo-phy0
shutdown_n	0	[0]	0x5105_0040	csi2_host0
rst	0	[0]	0x5105_0044	csi2_host0

2. Configure the relevant internal registers of the phy, see the phy datasheet for details.
3. Configure phy's shutdown_n = 1, configure phy's rst = 1

Register Name	Value	Bit	Address	
shutdown_n	1	[0]	0x5105_0040	csi2_host0
rst	1	[0]	0x5105_0044	csi2_host0

4. Poll the phy_ready register until the signal goes high, indicating the completion of phy configuration

Register Name	Value	Bit	Address	
phy_ready	/	[0]	0x510c_c004	combo-phy0

5. Configure the phy interface control signals as shown in the figure below

Register Name	Value	Bit	Address	
forcexnmode_0	1	[7]	0x510c_c000	combo-phy0
forcexnmode_1	1	[8]	0x510c_c000	combo-phy0
forcexnmode_dck	1	[9]	0x510c_c000	combo-phy0

6. Configure the relevant internal registers of the phy, please refer to the phy datasheet for details

8.4.3 Combo-PHY GPIO StartUp

1. Configure the phy interface control signals as shown in the figure below

Register Name	Value	Bit	Address	
phy_mode	0	[0]	0x5105_001c	csi2_host0

2. Configure the relevant internal registers of the phy, please refer to the phy datasheet for details

8.4.4 Combo-PHY Configuration Clock

The register configuration clock of CDPHY is pclk, sourced from the clock divider win2030_ck_div_dynm in the vi_crg module. The configuration operation of this standard cell needs to occur when the corresponding clock is in the off state. The configuration process is as follows:

1. Turn off the vi_ahb_clk clock by setting bit[30] of the register at offset address 0x188 in sys_crg_csr to 0.
2. Modify the clock division parameter of win2030_ck_div_dynm by configuring bits[3:0] of the register at offset address 0x68 in vi_common_top_reg.
3. Turn on the vi_ahb_clk clock by setting bit[30] of the register at offset address 0x188 in sys_crg_csr to 1.

9 Video Output

The video output is responsible for the display function, the OSD is responsible for the storage, processing, and output of the display data to the HDMI or MIPI interface,the follow is the function diagram：

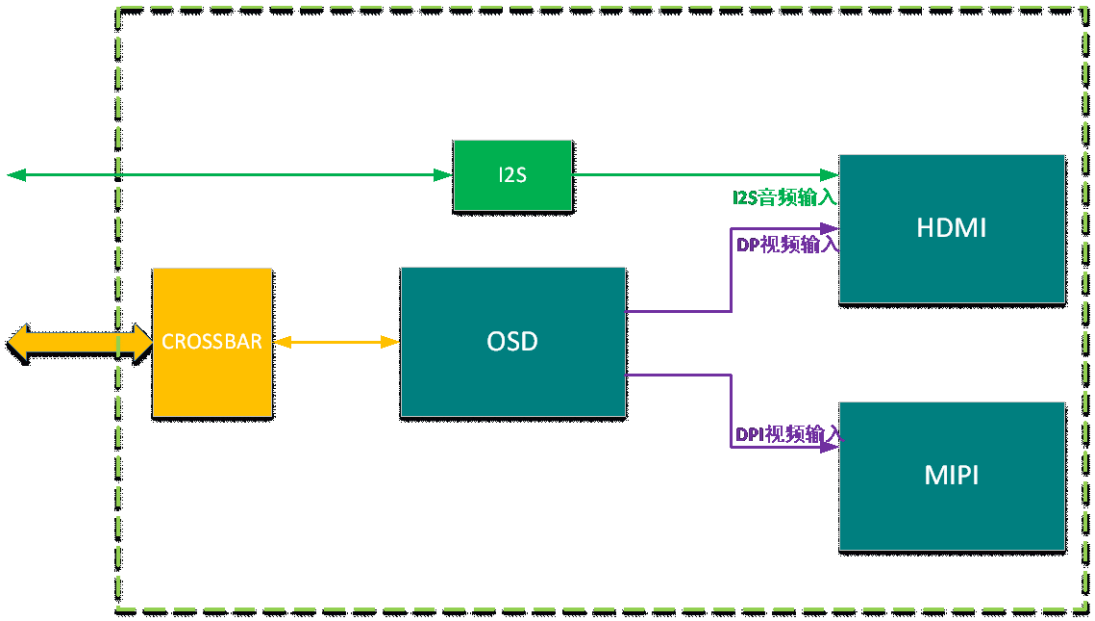


Figure 9-1 Video output

The following table describes the address mappings of each module in the video output system：

Table 9-1 Address mapping of video output

Name	Address range	Memory range size
Mipi_dsi	0x50270000~0x5027ffff	64K
Hdmi_hdcp	0x50290000~0x5029ffff	64K
Hdmi	0x502a0000~0x502bffff	128K
OSD	0x502c0000~0x502fffff	256K

9.1 OSD

9.1.1 Function description

OSD is a high-performance area-optimized display processor unit (DPU), that can be used for reading rendered images from the frame buffer to the display. In addition to providing hardware cursor patterns, the Display Controller performs format conversions, dithering, and gamma corrections. This controller supports parallel pixel output and is easily adapted to external serialization logic. The max definition is 4K, the max color depth is 10bit.

9.1.2 Feature

- Video Timing Generation。
  - HSYNC, VSYNC, and Data Enable (DE) signals
  - Programmable timers
- Display Capability。
  - Default display resolution: 1080
  - Maximum display resolution: 4K2K
  - Independent synchronization and blank signals for the display output panel
  - Independent gamma and dither tables for the display output panel
- Output Interface.
  - Parallel pixel output with 30-bit Data, HSync, VSync, and DE
  - Display pixel interface (DPI) with RGB output
  - DisplayPort (DP) with RGB output
  - Various DP and DPI output formats
  - Easy adaptation to external serialization logic
- Input Formats。
  - Various A/XRGB, YUV422, and YUV420 formats
  - Programmable color space conversions of YUV2RGB and RGB2RGB
  - Conversion between BT709 and BT2020
- Hardware Cursor.
- Decompression.
  - Supports lossless decompression
- Color.
  - A separate lookup table for dither
  - A separate lookup table for gamma
  - A separate lookup table for de-gamma
  - Alpha Blending: 8 Porter Duff Blending modes
- Filter and Scaling.

- Supported on both the video/graphic and overlay layers
- Vertical and horizontal scaling
- Horizontal 3-tap or 5-tap and vertical 3-tap filters
- Programmable filter order
- 15.16 fixed point scaling factor

9.1.3 Function Diagram

9.1.3.1 The main modules of the OSD

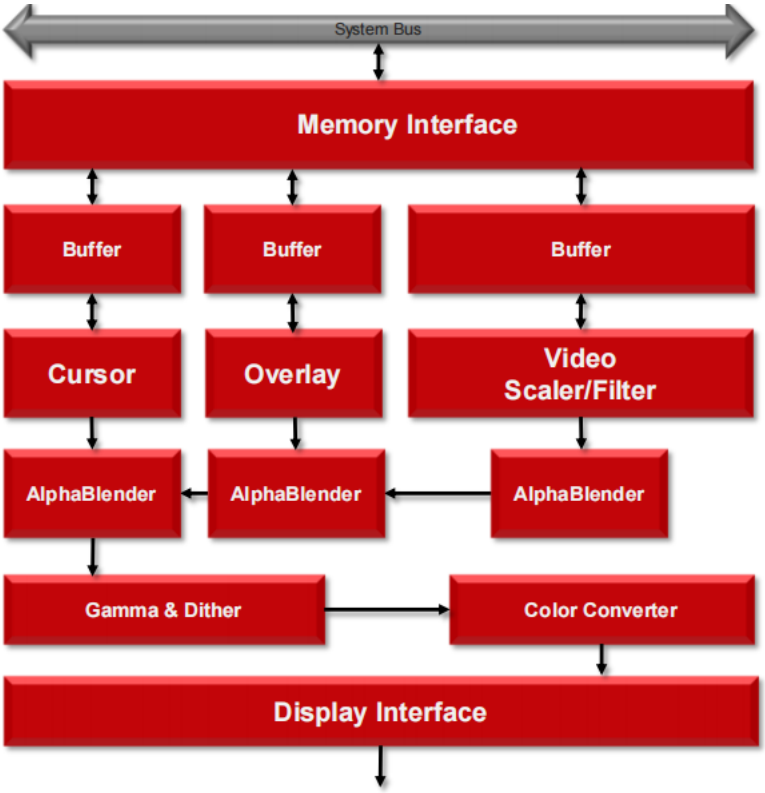


Figure 9-2 OSD function diagram

9.1.3.2 Data input

Swizzle is used to switch the position of components in pixel data. The same color format may be stored in different order in memory. To accommodate this possibility, the Vivante Display Controller will first swizzle an input pixel. Since the input pixel may have been input with four different format orders: ARGB, RGBA, ABGR or BGRA, this initial swizzle is just to convert the format into ARGB. For YUV formats, the U and V pixels may be placed in different positions, so UV-swizzle is used to convert them to a common format.

Note that the Vivante Display Controller assumes the initial input of ARGB data is in ARGB order, which means A is in the upper bits while B is in the lowest bits. If the user input is not in ARGB order, swizzle is needed to reshape the data. For example, if the input format is in BGRA order, the swizzle must be configured to handle the BGRA format.

For YUV formats, especially for UV mixed formats, the Display Controller assumes that for each UV pair, U is in the lower bits while V is in the upper bits. If the user input in not in this order, UV-swizzle reorders the input data.

32bit format	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A8R8G8B8	A	A	A	A	A	A	A	A	R	R	R	R	R	R	R	R	G	G	G	G	G	G	G	G	B	B	B	B	B	B	B	B
X8R8G8B8	X	X	X	X	X	X	X	X	R	R	R	R	R	R	R	R	G	G	G	G	G	G	G	G	B	B	B	B	B	B	B	B
A2B10G10R10	A	A	B	B	B	B	B	B	B	B	B	B	G	G	G	G	G	G	G	G	G	G	R	R	R	R	R	R	R	R	R	R
16 bit format	secend pixel																first pixel															
A1R5G5B5	A	R	R	R	R	R	G	G	G	G	G	B	B	B	B	B	A	R	R	R	R	R	G	G	G	G	G	B	B	B	B	B
X1R5G5B5	X	R	R	R	R	R	G	G	G	G	G	B	B	B	B	B	X	R	R	R	R	R	G	G	G	G	G	B	B	B	B	B
A4R4G4B4	A	A	A	A	R	R	R	R	G	G	G	G	B	B	B	B	A	A	A	A	R	R	R	R	G	G	G	G	B	B	B	B
X4R4G4B4	X	X	X	X	R	R	R	R	G	G	G	G	B	B	B	B	X	X	X	X	R	R	R	R	G	G	G	G	B	B	B	B
R5G6B6	R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B	R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B

Figure 9-3 Expected input organization for RGB data

The following table lists the input formats that the Display Controller supports for the video/graphic and overlay layers. The cursor uses the format of ARGB8888.OSD Video/graphic:

Table 9-2 OSD input data format

Format	Linear Input	Tile Input
ARGB2101010	Yes	Yes
ARBG8888/XRGB8888	Yes	Yes
ARBG1555/XRGB1555	Yes	Yes
RGB565	Yes	Yes
ARGB4444/XRGB4444	Yes	Yes
YUV422-YUY2	Yes	Yes
YUV422-UYVY	Yes	Yes
YUV420-YV12	Yes	No
YUV420-NV12	Yes	Yes
YUV420-P010_10bit	Yes	Yes
YUV422-NV16	Yes	No

9.1.3.3 Data output

The display pixel interface (DPI) supports the RGB output formats: DPI_D16CFG1, DPI_D16CFG2, DPI_D16CFG3, DPI_D18CFG1, DPI_D18CFG2, DPI_D24, and DPI_D30 (R10G10B10). The DisplayPort (DP) interface supports the RGB output formats: RGB101010, RGB888, RGB666, and RGB565. The output format is shown in the follow figure:

Configuration	Bits[2:0]	47	·····	43	42	41	40	39	38	37	·····	31	·····	27	26	25	24	23	22	·····	15	·····	11	10	9	8	7	6	·····	0
RGB101010	3	R	R	R	R	R	R	R	R			G	G	G	G	G	G	G	G		B	B	B	B	B	B	B	B		
RGB888	2	R	R	R	R	R	R					G	G	G	G	G	G				B	B	B	B	B	B				
RGB666	1	R	R	R	R							G	G	G	G						B	B	B	B						
RGB565	0	R	R	R								G	G	G							B	B	B							

Figure 9-4 DP port data format

The display pixel interface (DPI) supports the RGB output formats: DPI_D16CFG1, DPI_D16CFG2, DPI_D16CFG3, DPI_D18CFG1, DPI_D18CFG2, DPI_D24, and DPI_D30 (R10G10B10). The output format is shown in the follow figure:

config	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
30bitRGB	R	R	R	R	R	R	R	R	R	R	G	G	G	G	G	G	G	G	G	G	B	B	B	B	B	B	B	B	B	B
DPI 24bit							R	R	R	R	R	R	R	R	G	G	G	G	G	G	G	G	B	B	B	B	B	B	B	B
DPI 18bit 1													R	R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B	B
DPI 18bit 2									R	R	R	R	R	R			G	G	G	G	G	G			B	B	B	B	B	B
DPI 16bit 1															R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B
DPI 16bit 2										R	R	R	R	R			G	G	G	G	G	G				B	B	B	B	B
DPI 16bit 3									R	R	R	R	R				G	G	G	G	G	G			B	B	B	B	B	

Figure 9-5 DPIport data format

The DPI port is mipi dsi's upstream data port，The DP port is HDMI's upstream data port。

9.1.3.4 Color Keying for Video/Graphic and Overlay

OSD provide for overlay and transparency effects using color keys to blend pixels in images. A chosen color in a source image can be selected and filtered out so that it can be replaced with a different designated color in the destination target image.

The filter mechanism is controlled using the dcregOverlayColorKey0~15 and dcregOverlayKeyHigh0~15 registers. Each pixel color in the overlay or video/graphic layer is compared against the range specified in these registers. If the pixel color is in the range of [ColorKey, ColorKeyHigh], the Display Controller will regard it as a special color and replace it with another color as follows:

- If the source color is on the overlay layer, it is replaced with the corresponding pixel on the video/graphic layer.
- If the source color is on the video/graphic layer, it will be replaced with the configured background color.

9.1.3.5 Dither

In an image where color intensities change slowly, there may be noticeable jumps in intensity levels between pixels. Dithering can be used to diffuse the intensity across neighboring pixels. In dithering mode pixels are scanned in order, and errors in calculating a pixel's intensity are distributed (that is, diffused) to neighboring pixels to keep the overall intensity of the image closer to the input intensity.

Vivante's Display Controller dither implementation requires a lookup table of 4x4 entries of 4 bits. These 64-bit tables are located in the 32-bit registers gcregDisplayDitherTableLow0 and gcregDisplayDitherTableHigh0. The dither lookup table needs to be indexed by the two least significant bits x[1:0] and the two least significant bits y[1:0] of the displays.

Create a lookup table by setting registers dcregDisplayDitherTableLow0 and dcregDisplayDitherTableHigh0.

- The lookup table includes 16 entries, 4 bits for each.
- The lookup table provides a value **U[3:0]** through the indexes **X[1:0]** and **Y[1:0]**.

Example: Compare the color value **RedColor[3:0]** with this **U[3:0]**.

- If **RedColor[3:0]** > **U[3:0]** and **RedColor[7:4]** is not 4'b1111, then the final color value is **NewRedColor = RedColor[7:4]** + 1'b1.
- If RedColor[3:0] <= U[3:0], then NewRedColor = RedColor[7:4].

Figure 9-6 Dither algorithm

9.1.3.6 Alignment of Stride and Base Address

The following table provides the alignment of stride and base address.

Table 9-3 Alignment of Stride and Base Address

Linear/Tile	Format	Width x Height Alignment (Pixels)	Stride Alignment (Bytes)	Base Address Alignment (Bytes)
Linear	Planar YUV Semi-planar YUV	128x2	128	128
	RGB/Index1/2/4/8 Packed YUV	\	128	128
SuperTileX8x8	16bpp RGB/ YUY2/UYVY	64x64	128	128
SuperTileX8x4/ SuperTileY4x8	32bpp RGB	64x64	256	128
8x8Tile	16bpp RGB/ YUY2/UYVY	8x8	16	128
	NV12-Y planar	16x8	16	64
	P010-Y planar	16x8	32	128
8x4Tile	32bpp RGB	8x8	32	128
	NV12-UV planar	16x8	16	64
	P010-UV planar	16x8	32	128
Decompression				
Linear	NV12-Y planar NV12-UV planar	256x2	256	256
	P010-Y planar P010-UV planar	128x2	256	256
SuperTileX8x4/ SuperTileY4x8	32bpp RGB	64x64	256	128
8x8Tile	YUY2	8x8	16	128
	NV12-Y planar	32x8	32	256
	P010-Y planar	16x8	32	256
8x4Tile	NV12-UV planar	32x8	32	256

Linear/Tile	Format	Width x Height Alignment (Pixels)	Stride Alignment (Bytes)	Base Address Alignment (Bytes)
	P010-UV planar	16x8	32	256

9.1.4 Initialization

The following flow describes the basic steps of setting the display output panel and frame data. Set the timing parameters according to the specifications for the target panel specifications. Refer to the timing diagram for the key registers and register bits. The list below is not comprehensive.

- dcregHDisplay
- dcregHSync
- dcregVDisplay
- dcregVSync

Step 1. Set registers for the cursor, gamma, and dither functions (if needed).

- dcregCursor*
- dcregGamma*
- dcregDisplayDither*

Step 2. Set the frame buffer address and stride registers.

- dcregFrameBufferAddress
- dcregFrameBufferStride

Step 3. Configure panel requirements.

- dcregPanelConfig

Step 4. Configure the data format of the frame buffer.

- dcregFrameBufferConfig

Step 5. Start the signal transfer.

- dcregFrameBufferConfig.RESET=1

Step 6. Update parameters for the next frame.

Step 7. Use the interrupt registers to wait for the signal of which the transfer is complete, or check the interrupt status.

- dcregDisplayIntr
- dcregDisplayIntrEnable

Step 8. Update parameters for each succeeding frame as needed.

9.1.5 Timing

The synchronization pulse of the video signal output by the OSD can be configured with registers as either positive synchronization pulse or negative synchronization pulse. The negative pulse synchronization signal is active at a low level, and the timing diagram is shown below:

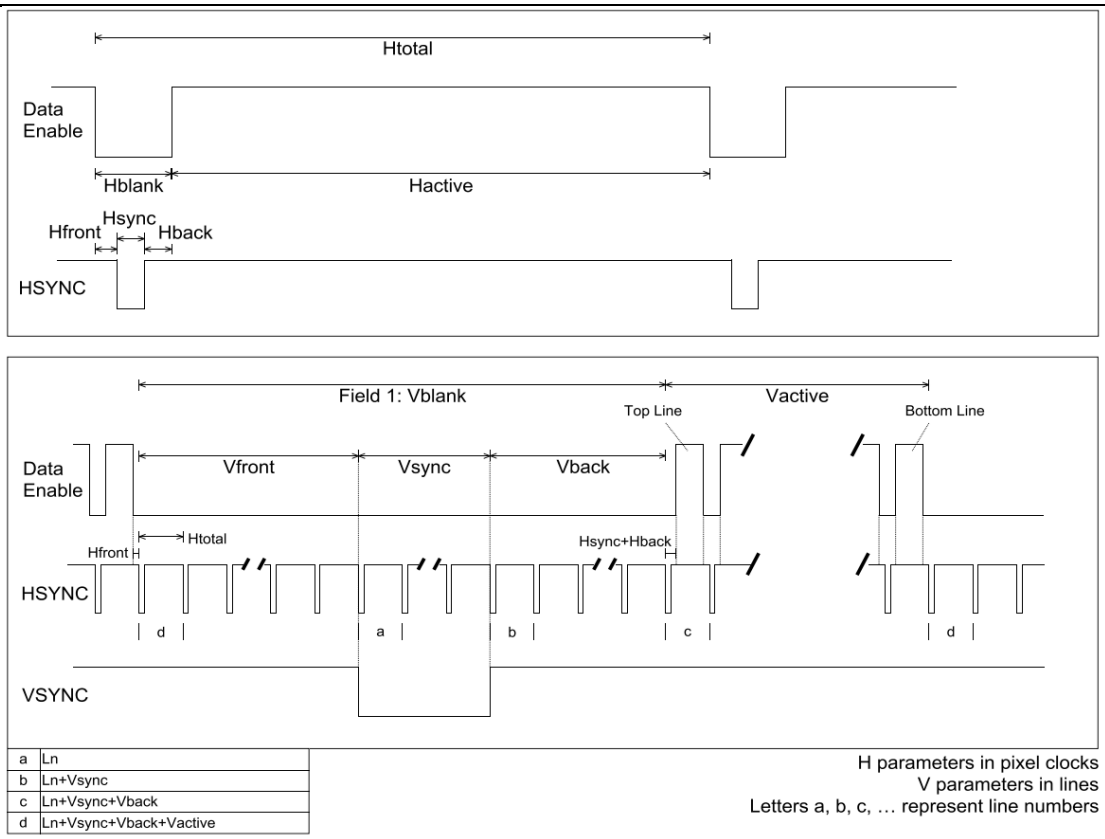


Figure 9-7 Negative pulse timing

The postive pulse synchronizer timing diagram as follow：

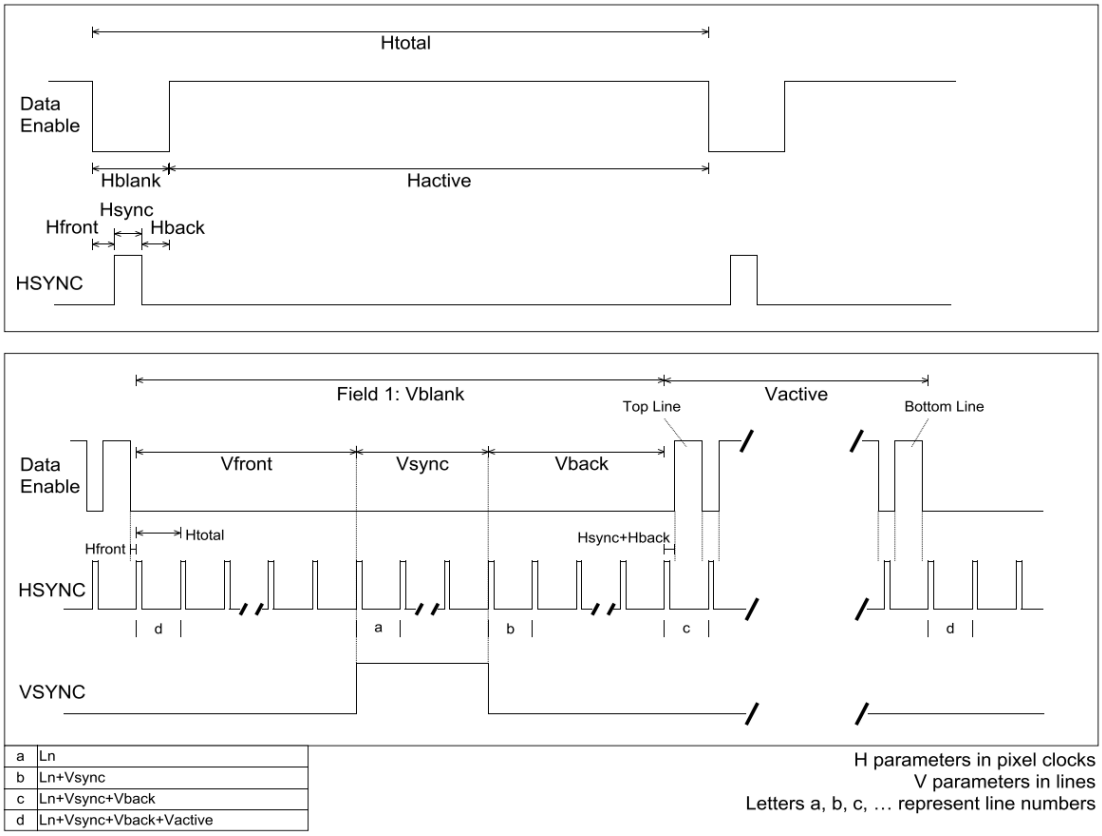


Figure 9-8 Postive pulse synchronizer timing diagram

The width of the pulse, the length of the blank, and the length of the effective area of image is in the

follow daigram, the relationship between each area and the register is shown in the figure below:

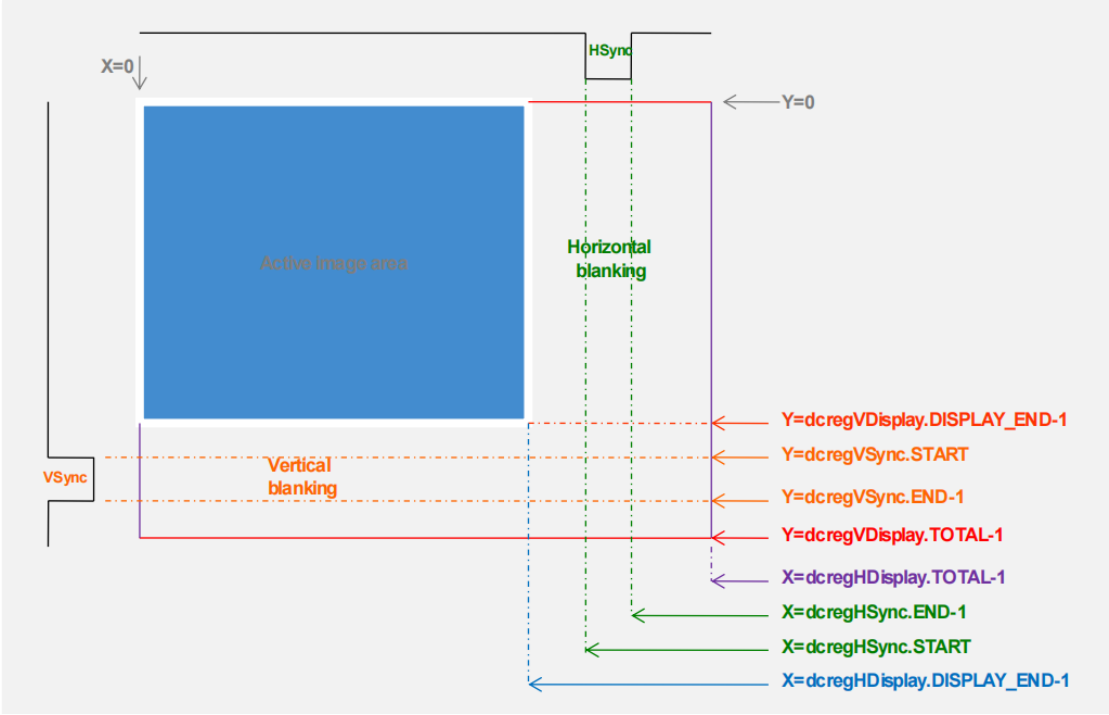


Figure 9-9 Register and frame parameter

9.2 MIPI TX

9.2.1 Function description

MIPI DSI is used to drive panel with MIPI interface, MIPI has four lanes, Lane operation ranging from 80 Mbps to 2.5 Gbps in forward in high speed mode, Maximum low power data rate supported of 10Mbps, the display data from OSD. The data is packaged into packets specified in the DSI protocol and then output to the DPHY, which serializes the parallel data and outputs it to an external display device. The main functional modules are shown in the figure below:

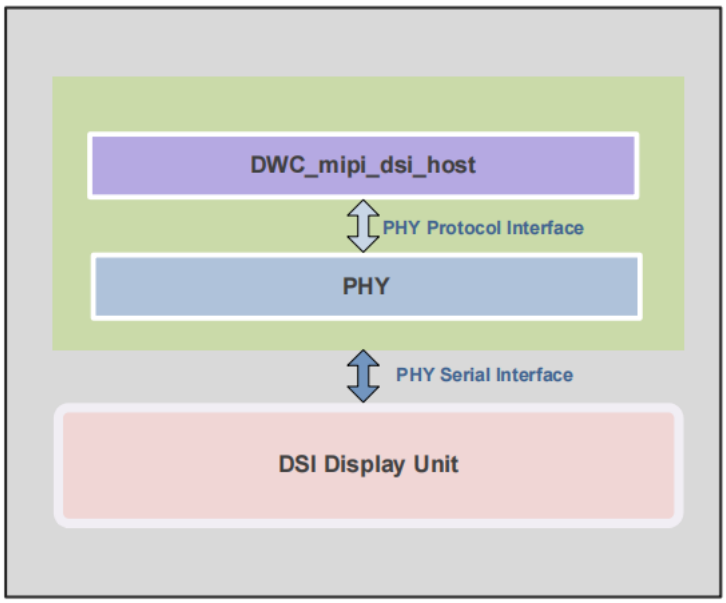


Figure 9-10 MIPI function diagram

9.2.2 Feature

Compliant with MIPI Alliance standards.

- Supports up to four lanes for data transfer.
- Supports up to 2.5 Gsps per lane in D-PHY.
- Low power data transfer rate up to 10Mbps.
- Bidirectional communication and escape mode support through data lane 0.
- DPI interface color coding mappings into 30-bit interface:
  - 16-bit RGB, configurations 1, 2, and 3
  - 18-bit RGB, configurations 1 and 2
  - 24-bit RGB
  - 30-bit RGB
  - 36-bit RGB (double clock rate required)
  - 24-bit YCbCr 4:2:2
  - 20-bit YCbCr 4:2:2 loosely packed
  - 16-bit YCbCr 4:2:2
  - 12-bit YCbCr 4:2:0
- Programmable polarity of all DPI interface signals.
- Supports Ultra Low-Power mode with PLL disabled.
- Support for End of Transmission Packet (EoTp).
- Video pattern generator:
  - Vertical and horizontal color bar generation without DPI stimuli
  - BER pattern without DPI stimuli

- Auto ULPS control scheme.

9.2.3 Function Diagram

The MIPI function diagram is as follow:

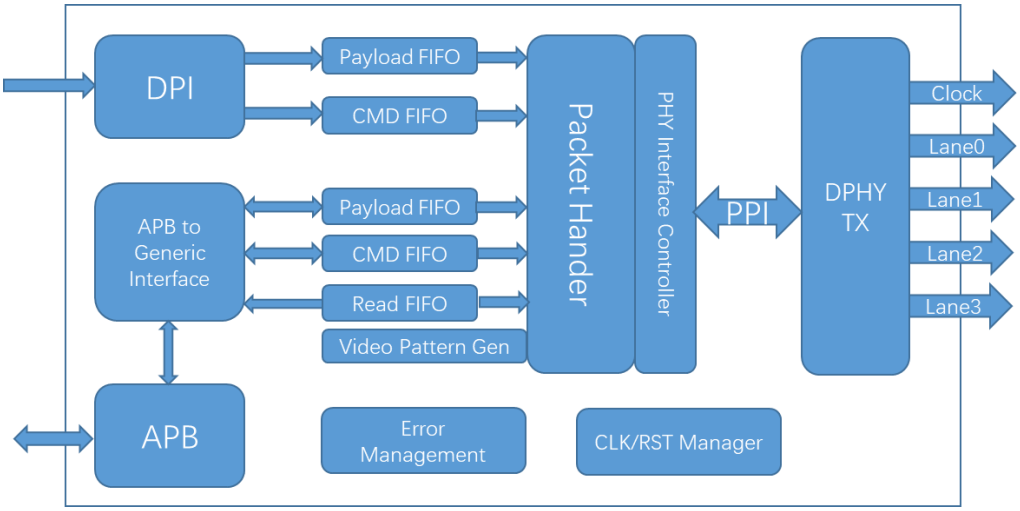


Figure 9-11 MIPI data stream

9.2.3.1 DPI

The DPI interface follows the MIPI DPI-2 specification with pixel data bus width up to 30 bits. It is used to transmit the information in Video mode in which the transfers from the OSD to the peripheral take the form of a real-time pixel stream. This interface allows sending ShutDown (SD) and ColorMode (CM) commands, which are triggered directly by video out register. To transfer additional commands (for example, to initialize the display), use another interface such as APB Slave Generic Interface to complement the DPI interface.

It is possible to update the DPI configuration on the fly without impacting the current frame. It is done with the help of shadow registers. This feature is controlled by the VID_SHADOW_CTRL register.

The new configuration is only used when the system requests for it. To update the DPI configuration during the transmission of a video frame, the configuration of that frame needs to be stored in the auxiliary registers. This way, the new frame configurations can be set through the APB interface without corrupting the current frame.

By default, this feature is disabled. To enable this feature, set the vid_shadow_en bit of the

VID_SHADOW_CTRL to 1. When this feature is enabled, the system supplies the configuration stored in the auxiliary registers. The follow diagram shows the necessary steps to update the DPI config:

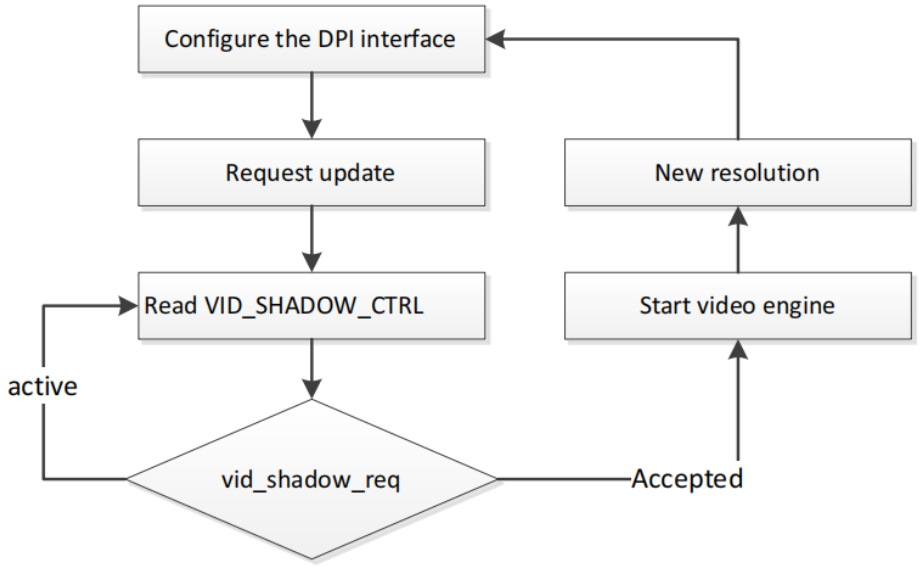


Figure 9-12 Shodow registers configuration

The different video transmission modes are as follows:

- Burst mode
- Non-Burst mode
  - Non-Burst mode with sync pulse
  - Non-Burst mode with sync event DSI

In Burst mode, the entire active pixel line is buffered into a FIFO and transmitted in a single packet with no interruptions. This transmission mode requires that the DPI Pixel FIFO has the capacity to store a full line of active pixel data inside it. This mode is optimally used if the difference between the pixel required bandwidth and DSI link bandwidth is very different. This enables the MIPI TX to quickly dispatch the entire active video line in a single burst of data and then return to low-power mode.

In NON-Burst mode, the MIPI DSI uses the partitioning properties of the MIPI TX to divide the video line transmission into several DSI packets. This is done to match the pixel required bandwidth with the DSI link bandwidth. With this mode, the MIPI TX does not require a full line of pixel data to be stored inside the DPI Pixel FIFO. It requires only the content of one video packet.

Selecting the Burst and Non-Burst mode is mainly dependent on the FIFO deepth and the device requirements. Choose the video transmission mode that suits the application scenario. The Burst mode is more beneficial because it increases the probability of the link spending more time in the low-power mode, decreasing power consumption. However, the following conditions should be met for availing the maximum benefits from the Burst mode of operation:

- The MIPI TX controller should have sufficient pixel memory to store an entire pixel line to avoid the overflow of the internal FIFOs.
- The display device should support receiving a full pixel line in a single packet burst to avoid the overflow on the reception buffer.
- The DSI output bandwidth should be higher than the DPI system interface input bandwidth in a relation that enables the link to go to low-power once per line.

If the system cannot meet these requirements, it is likely that the pixel data is lost causing the malfunctioning of the display device while using the Burst mode. These errors are related to the capabilities of the system to store the temporary pixel data.

If all the conditions for using the Burst mode cannot be met, use the Non-Burst mode to avoid the errors caused by the Burst mode. The Non-Burst mode provides a better matching of rates for pixel transmission, enabling:

- Only a certain amount of pixels to be stored in the memory and not requiring a full pixel line (lesser DPI RAM requirements in the MIPI TX)
- Operation with devices that support only a small amount of pixel buffering (less than a full pixel line)

The DSI Non-Burst mode should be configured in such way that the DSI output pixel ratio matches with the DPI input pixel ratio, reducing the memory requirements on both host and/or device side. This is achieved by dividing a pixel line into several chunks of pixels and optionally interleaving them with null packets.

The following equations show how the MIPI TX controller transmission parameters should be programmed in Non-Burst mode to match the DSI link pixel output ratio (left hand side of the "=" sign) and DPI pixel input (right hand side of the "=" sign).

- When the null packets are enabled:
  - lanebyteclkperiod * vid_num_chunks (vid_pkt_size * bytes_per_pixel + 12 + vid_null_size) / number_of_lanes = pixels_per_line * dpipclkperiod
- When the null packets are disabled:
  - lanebyteclkperiod * vid_num_chunks (vid_pkt_size * bytes_per_pixel + 6) / number_of_lanes = pixels_per_line * dpipclkperiod

In our system, the dpi pixel FIFO depth is 8192 byte.

9.2.3.2 APB interface

The APB Slave interface allows the transmission of generic information in Command mode. MIPI supports the transmission of write and read command mode packets as described in the DSI specification. These packets are built using the APB register access. The GEN_PLD_DATA register has two distinct functions based on the operation. Writing to this register sends the data as payload when sending a Command mode packet. Reading this register returns the payload of a read back operation. The GEN_HDR register contains the Command mode packet header type and header data. Writing to this register triggers the transmission of the packet implying that for a long Command mode packet, the packet's payload needs to be written in advance in the GEN_PLD_DATA register.

- Generic Write Short Packet 0 Parameters
- Generic Write Short Packet 1 Parameters
- Generic Write Short Packet 2 Parameters
- Generic Read Short Packet 0 Parameters
- Generic Read Short Packet 1 Parameters
- Generic Read Short Packet 2 Parameters
- Maximum Read Packet Configuration
- Generic Long Write Packet
- DCS Write Short Packet 0 Parameters
- DCS Write Short Packet 1 Parameters
- DCS Read Short Packet 0 Parameters
- DCS Write Long Packet.

For more information about of these packages, please refer to the official document of DCS Specification.

9.2.3.3 Transmission of Commands

The MIPI supports the transmission of commands, both in high-speed and low-power, while in Video mode. The MIPI uses Blanking or Low-Power (BLLP) periods to transmit commands inserted through the APB Generic interface. Those periods correspond to the shaded areas of follow diagram:

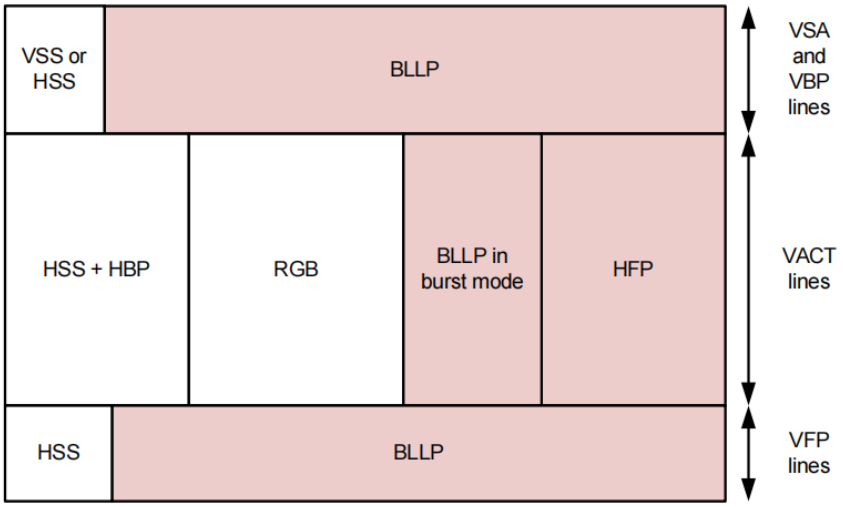


Figure 9-13 Command Transmission Periods within the Image Area

Commands are transmitted in the blanking periods after the following packets/states:

- Vertical Sync Start (VSS) packets, if the Video Sync pulses are not enabled
- Horizontal Sync End (HSE) packets, in the VSA, VBP, and VFP regions
- Horizontal Sync Start (HSS) packets, if the Video Sync pulses are not enabled in the VSA, VBP, and VFP regions
- Horizontal Active (HACT) state

Mipi dsi can be configured to send the low-power commands during the high-speed video mode transmission. To enable this feature, set the lp_cmd_en bit of the VID_MODE_CFG register to 1. In this case, it is necessary to calculate the time available, in bytes, to transmit a command in low-power mode to Horizontal Front Porch (HFP), Vertical Sync Active (VSA), Vertical Back Porch (VBP), and Vertical Front Porch (VFP) regions.

Bits 8 to 13 of the VID_MODE_CFG register indicates if MIPI can go to low-power when in idle. If the lp_cmd_en bit is set (1'b1) and non-video packets are in queue, mipi dsi ignores the low-power configuration and transmits low-power commands, even if it is not allowed to enter low-power in a specific region. After the low-power commands transmission, mipi dsi remains in lowpower until a sync event occurs.

For example, consider that the VFP is selected as high-speed region (lp_vfp_en = 1'b0) with lp_cmd_en set as a command to transmit in low-power in the VPF region. This command is transmitted in low-power, and the line stays in low-power until a new HSS arrives.

9.2.3.3.1 Calculating the Time to Transmit Commands in Low-Power Mode in the VSA, VBP, and VFP Regions

The outvact_lpcmd_time field of the DPI_LP_CMD_TIM register indicates the time available (in bytes) to transmit a command in low-power mode (based on the escape clock) on a line during the VSA, VBP, and the VFP regions.

Calculation of outvact_lpcmd_time depends on the Video mode that you use. The follow diagram illustrates the timing intervals for the Video mode in non-Burst with sync pulses and the timing intervals for the Video mode in Burst and non-Burst with sync events.

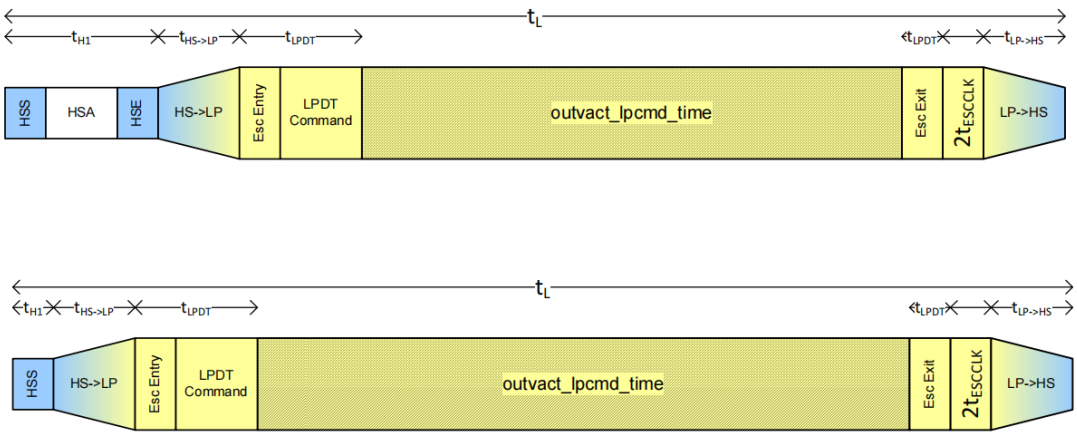


Figure 9-14 outvact_lpcmd_time for Non-Burst with Sync event and Sync Pulses

This time is calculated as follows:

outvact_lpcmd_time = (t_L - (t_{H1} + $t_{HS->LP}$ + $t_{LP->HS}$ + t_{LPDT} + $2t_{ESCCLK}$)) / (2 × 8 × t_{ESCCLK})

Where,

- t_L = Line time
- t_{H1} = Time of the HSA pulse for sync pulses mode or the time to send the HSS packet, including EoTp
- $t_{HS->LP}$ = Time to enter the low-power mode
- $t_{LP->HS}$ = Time to leave the low-power mode
- t_{LPDT} = PHY timing related with Escape Mode Entry, LPDT Command, and Escape Exit. According to the PHY specification, this value is always 11 bits in low-power (or 22 TX Escape clock cycles).
- t_{ESCCLK} = Escape clock period as programmed in the tx_esc_clk_division field of the CLKMGR_CFG register
- $2xt_{ESCCLK}$ = Delay imposed by the controller implementation

9.2.3.3.2 Calculating the Time to Transmit the Commands in Low-Power Mode in the HFP Region

The invact_lpcmd_time field of the DPI_LP_CMD_TIM register indicates the time available (in bytes) to transmit a command in low-power mode (based on the escape clock) in the Vertical Active (VACT) region. To calculate the value of invact_lpcmd_time, consider the video mode that you use. The follow diagram shows the timing intervals for video mode in Non-Burst with sync pulses, timing intervals for video mode in non-Burst with sync events and Burst video mode.

This time is calculated as follows:

invact_lpcmd_time = (t_L - (t_{HSA}+t_{HBP}+t_{HACT}+ $t_{HS->LP}$ + $t_{LP->HS}$ + t_{LPDT} + $2t_{ESCCLK}$))/(2×8×t_{ESCCLK})

Where

t_L = Line time

t_{HSA} = Time of the HSA pulse (VID_HSA_TIME)

t_{HBP} = Time of Horizontal back porch (VID_HBP_TIME)

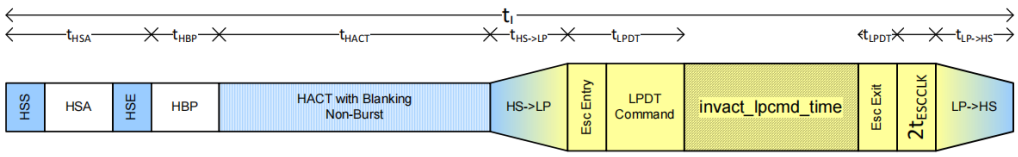
t_{HACT} = Time of Video active. For Burst mode, the Video active is time compressed and is calculated

as follows:

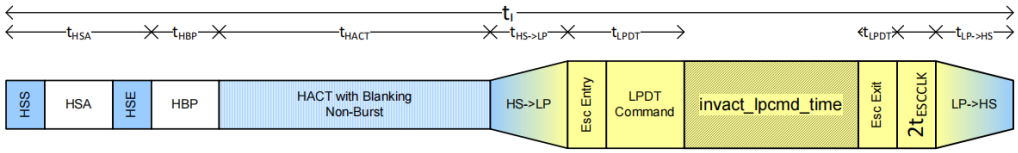
t_{HACT} = vid_pkt_size * Bytes_per_Pixel /Number_Lanes * tLane_byte_clk

t_{ESCCLK} = escape clock period as programmed in tx_esc_clk_division field of the CLKMGR_CFG register

invact_lpcmd_time for Non-Burst with Sync Pulses



invact_lpcmd_time for Non-Burst with Sync Events



invact_lpcmd_time for Burst Mode

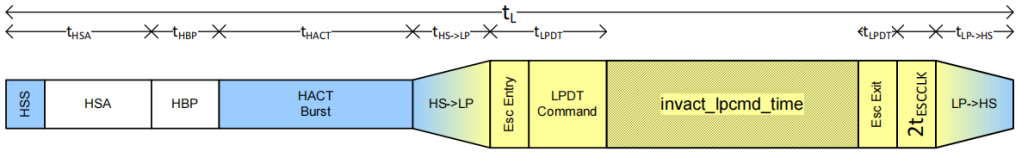


Figure 9-15 invact_lpcmd_time

The generation of ESCCLK is shown in the figure below, ESCCLK cannot be greater than 20MHz, and the frequency division coefficient is configured through the register, LANEBYTECLK/tx_clk_esc_division is the actual frequency of ESCCLK.

9.2.3.3.3 Transmission of Commands in High-Speed Mode

If the lp_cmd_en bit of the VID_MODE_CFG register is 0, the commands are sent in high-speed in Video Mode. In this case, the MIPI TX automatically determines the area where each command can be sent and no programming or calculation is required.

9.2.3.3.4 Read Command Transmission

The max_rd_time field of the PHY_TMR_RD_CFG register configures the maximum amount of time required to perform a read command in lane byte clock cycles. It is calculated as follows:

max_rd_time = Time to transmit the read command in low-power mode + Time to enter and leave low power mode + Time to return the read data packet from the peripheral device.

The time to return the read data packet from the peripheral depends on the number of bytes read and the escape clock frequency of the peripheral, not the escape clock of the host. The max_rd_time field is used in both high-speed and low-power mode to determine if there is time to complete a read command in a BLLP period.

In high-speed mode (lp_cmd_en = 0), max_rd_time is calculated as follows:

max_rd_time = ($t_{HS->LP}$ + $t_{LP->HS}$ + tread + 2x t_{BTA}) / lanebyteclkperiod

In low-power mode (lp_cmd_en = 1), max_rd_time is calculated as follows:

max_rd_time = ($t_{HS->LP}$ + $t_{LP->HS}$ + + t_{LPDT} + tlprd + tread + 2x t_{BTA}) / lanebyteclkperiod

Where,

- $t_{HS->LP}$ = Time to enter the low-power mode
- $t_{LP->HS}$ = Time to leave the low-power mode
- t_{LPDT} = PHY timing related to Escape mode entry, LPDT command, and Escape mode exit.
- According to the PHY specification, this value is always 11 bits in low-power (or 22 TX escape clock cycles).
- t_{lprd} = Read command time in low-power mode (64 * TX esc clock)
- t_{read} = Time to return the read data packet from the peripheral
- t_{BTA} = time to perform a bus turnaround (PHY dependent)

9.2.3.4 Video pattern genarator

The Video Mode Pattern Generator allows the transmission of horizontal/vertical color bar and PHY BER testing pattern without any stimuli.

The frame requirements must be defined in video registers that are listed in follow table:

Table 9-4 Register for pattern generator

Register Name	Description
VID_MODE_CFG	Video mode configuration
VID_PKT_SIZE	Video packet size
VID_NUM_CHUNKS	Number of chunks
VID_NULL_SIZE	Null packet size
VID_HSA_TIME	Horizontal sync active time
VID_HBP_TIME	Horizontal back porch time
VID_HLINE_TIME	Line time
VID_VSA_LINES	Vertical sync active period
VID_VBP_LINES	Vertical back porch period
VID_VFP_LINES	Vertical front porch period
VID_VACTIVE_LINES	Vertical resolution

The color bar pattern comprises eight bars of the colors white, yellow, cyan, green, magenta, red, blue, and black. Each color width is calculated by dividing the line pixel size (vertical pattern) or the number of lines (Horizontal pattern) by eight. In the vertical color bar mode, each single color bar has a width of the number of pixels in a line divided by eight. In case the number of pixels in a line is not divisible by eight, the last color (black) contains the remaining. In the horizontal color bar mode, each color line has a color width of the number of lines in a frame divided by eight. In case the number of lines in a frame is not divisible by eight, the last color (black) contains the remaining lines.:

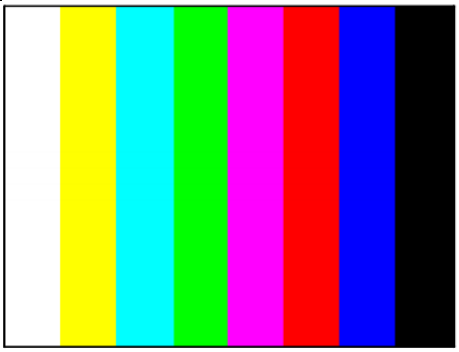


Figure 9-16 Vertical color bar

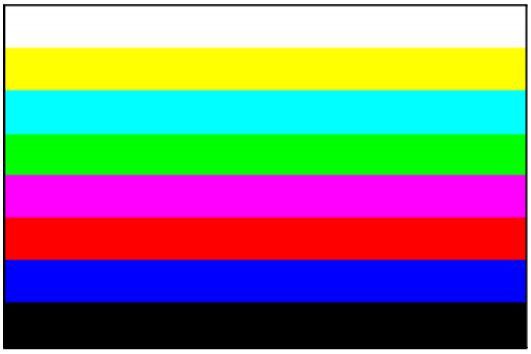


Figure 9-17 Horizontal Color Bar Mode

9.2.3.5 Error Control

The INT_ST0 and INT_ST1 registers are associated with error condition reporting. These registers can trigger an interrupt pin to inform the system about the occurrence of errors.

The MIPI TX controller has one interrupt pin that is set high when an error occurs in either the INT_ST0 or the INT_ST1 register.

The triggering of the interrupt pin can be masked by programming the mask registers INT_MSK0 and INT_MSK1. By default, all errors are masked. When any bit of these registers is set to 1, it enables the interrupt for a specific error. The error bit is always set in the respective INT_STn register. The INT_ST0 and INT_ST1 registers are always cleared after a read operation. The interrupt pin is cleared if all registers that caused the interrupt are read.

The interrupt force registers (INT_FORCE0 and INT_FORCE1) are used for test purposes, and they allow triggering the interrupt events individually without the need to activate the conditions that trigger the interrupt sources; this is because it is extremely complex to generate the stimuli for that purpose. This feature also facilitates the development and testing of the software associated with the interrupt events. Setting any bit of these registers to 1 triggers the corresponding interrupt.

9.2.3.6 PHY's config

The PHY's configuration is mainly the configuration of the PLL, generating lanebyte clock and serialized clock to ensure the serialize conversion of data, the frequency of reference clock Fin is 20MHz, and the output clock frequency of PLL should be half of the lane rate, for example, the lane rate is 2Gbps, and the output frequency of PLL should be 1GHz.

Since the VCO output frequency range is 320MHz~1250MHz, there are some differences in the frequency division coefficient of PLL in different frequency output ranges.

If the ouput of VCO in the range 320MHz~1250MHz, the relation between Fout and Fin as follow：

$$\text{Fout} = \text{Fvco} = \frac{M}{N} \times Fin$$

If the ouput of VCO in the range 160MHz~320MHz, the relation between Fout and Fin as follow:

$$\text{Fout} = \frac{\text{Fvco}}{2} = \frac{M}{2N} \times Fin$$

If the ouput of VCO in the range 80MHz ~ 160MHz，the relation between Fout and Fin as follow:

$$\text{Fout} = \frac{\text{Fvco}}{4} = \frac{M}{4N} \times Fin$$

If the ouput of VCO in the range 80MHz ~160MHz, the relation between Fout and Fin as follow:

$$\text{Fout} = \frac{\text{Fvco}}{8} = \frac{M}{8N} \times Fin$$

The value of parameter N can be calculated as follow formula, Since the frequency of fin is 20MHz, N can be taken as 10, The value of parameter M can be calculated upper formula.

$$2 \leq \frac{Fin}{N} \leq 8$$

M and N in the formula are not the actual configured register values, and the correspondence between M and N to the register value is shown in the following table：

Table 9-5 The relationship between the register m and coefficient M

Parameter	Description	M	m[9:0]
Minimum	M=m+2	64	10'h3E
Maximum		625	10'h26F

Table 9-6 The relationship between the register n and coefficient N

Parameter	Description	N	n[3:0]
Minimum	N=n+1	1	4'h0
Maximum		16	4'hF

All other PLL register's value VS the frequency as follow table:

Table 9-7 PLL output frequency vs register value

Output frequency [MHz]	vco_cntrl[5:0]	cpbias_cntrl[6:0]	gmp_cntrl[1:0]	int_cntrl[5:0]	prop_cntrl[5:0]
1150 – 1250	000001	0010000	01	000000	001110
1100 – 1152	000001	0010000	01	000000	001101
630 – 1149	000011	0010000	01	000000	001101
420 – 660	000111	0010000	01	000000	001101
320 – 440	001111	0010000	01	000000	001101
210 – 330	010111	0010000	01	000000	001101
160 – 220	011111	0010000	01	000000	001101
105 – 165	100111	0010000	01	000000	001101
80 – 110	101111	0010000	01	000000	001101
52.5 – 82.5	110111	0010000	01	000000	001101
40 – 55	111111	0010000	01	000000	001101

9.2.4 Initialization

The following figure shows the overall initialization process of MIPI:

Figure 9-18 MIPI initialization

- Reset MIPI DSI using the global reset.

- Configure n_lanes (PHY_IF_CFG) to define the number of lanes with which the controller can perform high-speed transmissions.
- Assert a bit corresponding to the error that is supposed to trigger an interrupt in INT_MSK_<group>.
- Configure DPI according to the programming sequence described in the panel databook.
- Initialize the PHY by following the programming sequence in the section "DPHY initialization".
- Assert the PWR_UP[0] bit to power-up the controller.

9.2.4.1 Configuring the Display Pixel Interface

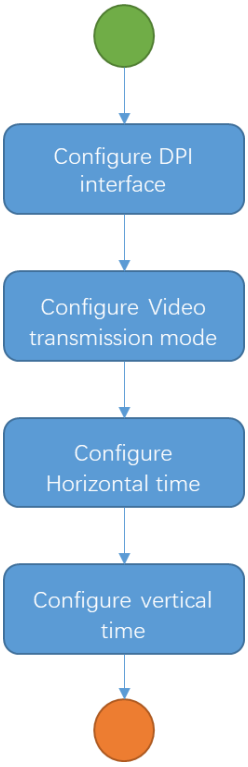


Figure 9-19 DPI configuration

- Configure the DPI interface to define how the DPI interface interacts with the controller.
  - **Configure dpi_vcid (DPI_VCID)**: This field configures the virtual channel that the packets generated by the DPI interface is indexed.
  - **Configure dpi_color_coding (DPI_COLOR_CODING)**: This field configures the bits per pixel that the interface transmits and also the variant configuration of each bpp.
  - **Configure dataen_active_low (DPI_CFG_POL)**: This bit configures the polarity of the dpidataen signal and sets it to be active low.
  - **Configure vsync_active_low (DPI_CFG_POL)**: This bit configures the polarity of the dpivsync signal and sets it to be active low.
  - **Configure hsync_active_low (DPI_CFG_POL)**: This bit configures the polarity of the dpihsync signal and sets it to be active low.
  - **Configure shutd_active_low (DPI_CFG_POL)**: This bit configures the polarity of the dpishutdn signal and sets it to be active low.
  - **Configure colorm_active_low (DPI_CFG_POL)**: This bit configures the polarity of the dpicolorm signal and sets it to be active low.

- Select the video transmission mode to define how the processor requires the video line to be transported through the DSI link.
  - **Configure the low-power transitions (VID_MODE_CFG)**: Defines the video periods which are permitted to go to low-power if there is time available to do so.
  - **Configure frame_bta_ack_en (VID_MODE_CFG)**: Specifies if MIPI TX should request the peripheral acknowledge message at the end of frames.
  - **Configure lp_cmd_en (VID_MODE_CFG)**: Specifies that commands are to be transmitted in low-power.
  - Burst mode:
    - Configure the register field vid_mode_type (VID_MODE_CFG) with value 2'b1x.
    - Configure vid_pkt_size (VID_PKT_SIZE) with the size of the active line period, measured in pixels.
    - The fields vid_num_chunks and vid_null_size are ignored by the MIPI TX.
  - Non-Burst mode:
    - Configure the vid_mode_type field (VID_MODE_CFG) with 2'b0x.
    - Configure the vid_mode_type field (VID_MODE_CFG) with 2'b00 to enable the transmission of sync pulses.
    - Configure the vid_mode_type field (VID_MODE_CFG) with 2'b01 to enable the transmission of sync events.
    - Configure the vid_pkt_size field (VID_MODE_CFG) with the number of pixels to be transmitted in a single packet. Selecting this value depends on the available memory of the attached peripheral.
    - Configure the vid_num_chunks field (VID_NUM_CHUNKS) with the number of packets to be transmitted per video line. The value of vid_pkt_size * vid_num_chunks is the number of pixels per line of video, except if vid_num_chunks is 0, which disables the multi-packets. If you set it to 1, there is still only one packet per line, but it can be part of a chunk, followed by a null packet.
    - Configure the vid_null_size field (VID_NULL_SIZE) with the size of null packets to be inserted as part of the chunks. Setting it to 0, disables null packets.
- Define the DPI horizontal timing configuration as follows:
  - Configure the vid_hline_time field (VID_HLINE_TIME) with the time taken by a DPI video line measured in cycles of **lane byte clock**. When the periods of DPI clock and lane byte clock are not multiples, the value to program the vid_hline_time needs to be **rounded up**. A timing mismatch is introduced between the lines due to the rounding of configuration values. If the DSI controller is configured not to go to low-power, then this timing divergence accumulates on every line, introducing a significant amount of mismatch towards the end of the frame. The reason for this is that the MIPI TX cannot re-synchronize on every new line because, it transmits the blanking packets when the horizontal Sync event occurs on the DPI interface. However, the accumulated mismatch should become extinct on the last line of a frame, where, according to the DSI specification, the link should always return to low-power regaining synchronization, when a new frame starts on a vertical sync event. If the accumulated timing mismatch is greater than the time in low-power on the last line, a malfunction occurs. This phenomenon can be avoided by configuring the MIPI TX to go to low-power once per line.
  - Configure the vid_hsa_time field (VID_HSA_TIME) with the time taken by a DPI Horizontal Sync Active period measured in cycles of **lane byte clock**.

- Configure the vid_hbp_time field (VID_HBP_TIME) with the time taken by the DPI Horizontal Back Porch period measured in cycles of **lane byte clock**. Special attention should be given to the calculation of this parameter.

- Define the vertical line configuration:
  - Configure the vsa_lines field (VID_VSA_LINES) with the **number of lines** existing in the DPI Vertical Sync Active period.
  - Configure the vbp_lines field (VID_VBP_LINES]) with the **number of lines** existing in the DPI Vertical Back Porch period.
  - Configure the vfp_lines field (VID_VFP_LINES) with the **number of lines** existing in the DPI Vertical Front Porch period.
  - Configure the v_active_lines field (VID_VACTIVE_LINES) with the **number of lines** existing in the DPI Vertical Active Period.

9.2.4.2 Initializing D-PHY

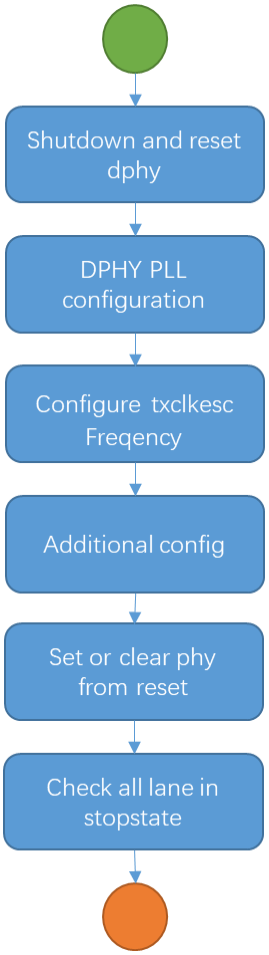


Figure 9-20 Initializing D-PHY

This process is based on the APB register interface access, hrough the accsess process, it can generate the test port timing.

- Power-Up and reset the PHY using “PHY_RSTZ” register.
- Configure the D-PHY PLL clock frequency through the Test interface.
  - Config PHY's register pll_m_ovr[9:0] and pll_m_ovr_en, the address of pll_m_ovr is 0x18, the address of pll_m_ovr_en is 0x19.pll_m_ovr need to be configured in two installments，when

bit7=1,it config pll_m_ovr[9:5]，when bit7=0, it config pll_m_ovr[4:0]。

Test Interface	Condition	Default value	Bits [7:0]							
			7	6	5	4	3	2	1	0
testdin	testdin[7] = 1'b1	8'b00001111	Reserved			pll_m_ovr[9:5]				
testdin	testdin[7] = 1'b0	8'b00001000	Reserved			pll_m_ovr[4:0]				
testdout	testdin[7] = 1'b0	8'b000XXXXX	Reserved			pll_m[4:0]				
testdout	testdin[7] = 1'b1	8'b000XXXXX	Reserved			pll_m[9:5]				

- Config PHY's register pll_n_ovr[3:0] and pll_n_ovr_e, the address of pll_n_ovr is 0x17,the address of pll_n_ovr_enis 0x19.

Test Interface	Condition	Default value	Bits [7:0]							
			7	6	5	4	3	2	1	0
testdin	No	8'b00001011	Reserved				pll_n_ovr			
testdout	No	8'b01101011	pll_lock_det_on	pll_gear_shift	pll_reset	pll_pwron	pll_n[3:0]			

Test Interface	Condition	Default value	Bits [7:0]							
			7	6	5	4	3	2	1	0
testdin	No	8'b00000000	Reserved		pll_m_ovr_en	pll_n_ovr_en	Reserved			
testdout	No	8'b00000000	Reserved		pll_m_ovr_en	pll_n_ovr_en	Reserved			

- Config PHY's register pll_vco_cntrl_ovr[5:0] and pll_vco_cntrl_ovr_en, the address is 0x12.

Test Interface	Condition	Default value	Bits [7:0]							
			7	6	5	4	3	2	1	0
testdin	No	8'b00111111	Reserved	pll_vco_cntrl_ovr_en	pll_vco_cntrl_ovr					
testdout	No	8'b00111111	Reserved	pll_vco_cntrl_ovr_en	pll_vco_cntrl_ovr					

- Config PHY's register cpbias_cntrl[6:0] and pll_cpbias_cntrl[6:0], the address is 0x1C.

Test Interface	Condition	Default value	Bits [7:0]							
			7	6	5	4	3	2	1	0
testdin	No	8'b00000000	Reserved	pll_cpbias_cntrl						
testdout	No	8'b00000000	Reserved	pll_cpbias_cntrl						

- Config PHY's register pll_gmp_cntrl[1:0], the address is 0x13.

Test Interface	Condition	Default value	Bits [7:0]							
			7	6	5	4	3	2	1	0
testdin	No	8'b00000000	Reserved		pll_gmp_cntrl		tstplldig			Reserved
testdout	No	8'b00000000	pll_testlock	Reserved	pll_gmp_cntrl		tstplldig			Reserved

- Config PHY's register pll_int_cntrl[5:0], the address is 0x0f.

Test Interface	Condition	Default value	Bits [7:0]							
			7	6	5	4	3	2	1	0
testdin	No	8'b00000000	Reserved		pll_int_cntrl					
testdout	No	8'b00000000	Reserved		pll_int_cntrl					

- Config PHY's register pll_prop_cntrl[5:0], the address is 0x0e.

Test Interface	Condition	Default value	Bits [7:0]							
			7	6	5	4	3	2	1	0
testdin	No	8'b00000000	Reserved		pll_prop_cntrl					
testdout	No	8'b00000000	Reserved		pll_prop_cntrl					

- TxClkEsc frequency must be configured between 2 to 20 MHz. This is done by writing into the tx_esc_clk_division field of the CLKMGR_CFG register. The tx_esc_clk_division field divides the byte clock, and generates a TX_ESC clock for the D-PHY. Follow diagram describes how txclkesc is obtained.

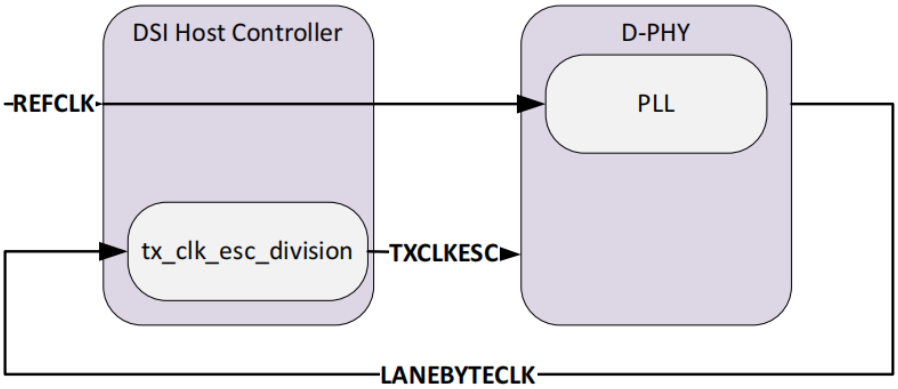


Figure 9-21 txclkesc generation

- Clear phy_shutdownz (PHY_RSTZ[0]) and phy_rstz (PHY_RSTZ[1]) to put the PHY in the reset state. Set phy_shutdownz (PHY_RSTZ[0]) and phy_rstz (PHY_RSTZ[1]) to remove the PHY from reset state.
- Verify that D-PHY active lanes are in Stop state:
  - PHY_STATUS[2] = 1'b1
  - PHY_STATUS[4] = 1'b1
  - PHY_STATUS[7] = 1'b1 – if Lane 2 active
  - PHY_STATUS[9] = 1'b1 – if Lane 3 active
  - PHY_STATUS[11] = 1'b1 – if Lane 4 active

9.2.4.3 ULPM（ultra low power mode）

The following are the two methods to enter into ultra low-power mode, one is entering and exiting the ULPM by the software, the other is entering and exiting the ULPM using Auto ULPS Control Scheme.

Entering the ULPM Using the Software:

- Set PHY_ULPS_CTRL[3:0] = 4'h5 to enter ULPM in the data and clock lanes.
- Verify that all the data lanes has entered in ULPM using PHY_STATUS register.

- The MIPI TX is now in ULPM. Perform next two steps to turn off the PHY PLL.
- Turn off the PHY PLL by setting PHY_RSTZ[3] = 1'b0.
- Verify that all the D-PHY PLL is unlocked: PHY_STATUS[0] = 1'b0.

Exiting the ULPM Using the Software:

- If the D-PHY PLL is already locked (PHY_STATUS[0] = 1'b1), go to step 4 else continue.
- Turn on the D-PHY PLL by setting PHY_RSTZ[3] = 1'b1.
- Verify that D-PHY PLL is turned on: PHY_STATUS[0] = 1'b1.
- Without deasserting the ULPM request bits, assert the Exit ULPM bits by setting PHY_ULPS_CTRL[3:0] = 4'hF.
- Verify that all D-PHY lanes are not in ULPM using PHY_STATUS register.
- Wait for 1 millisecond.
- De-assert the ULPM requests and the ULPM exit bits by setting PHY_ULPS_CTRL [3:0] = 4'h0.

Entering and Exiting the ULPM using Auto ULPS Control Scheme:

- Configure the idle time (in lanebyteclk cycles) before MIPI TX enters into ULPM, using AUTO_ULPS_ENTRY_DELAY register:

  $$AutoULPSEntryDelay = TimeIntended(s)/T_{lanebyteclk}(s)$$

- Configure PHY Wake-Up time (in pclk) using AUTO_ULPS_WAKEUP_TIME register. The default Wake-Up time specified by MIPI PHY specs is 1 millisecond. The AUTO_ULPS_WAKEUP_TIME register is composed by two fields:

  twakeup_clk_div [AUTO_ULPS_WAKEUP_TIME[15:0]: This configures the Twakeup counter (measured in pclk cycles). Minimum allowed value is 'd1.

  twakeup_cnt [AUTO_ULPS_WAKEUP_TIME[31:16] : This configures the Twakeup clock divider (measured in pclk cycles). Minimum allowed value is 'd1.

  $$Twakeup(s) = twakeup\_clk\_div * twakeup\_cnt * Tpclk(s)$$

- Configure the AutoULPS mode. Auto ULPS feature has two ways to enter and exit ULPM:
  - Turning off the PHY PLL - AUTO_ULPS_MODE[16] = 1'b1,Clock and data enters in ULPM.
  - Without Turning off the PHY PLL - AUTO_ULPS_MODE[16] = 1'b0,Data lanes enter in ULPM.
- Enable AutoULPS mechanism asserting bit [0] of AUTO_ULPS_MODE register.

9.2.5 Interface timing

The DPI port's timing is the same as the OSD's DPI port.

PHY's test port timing as follow:

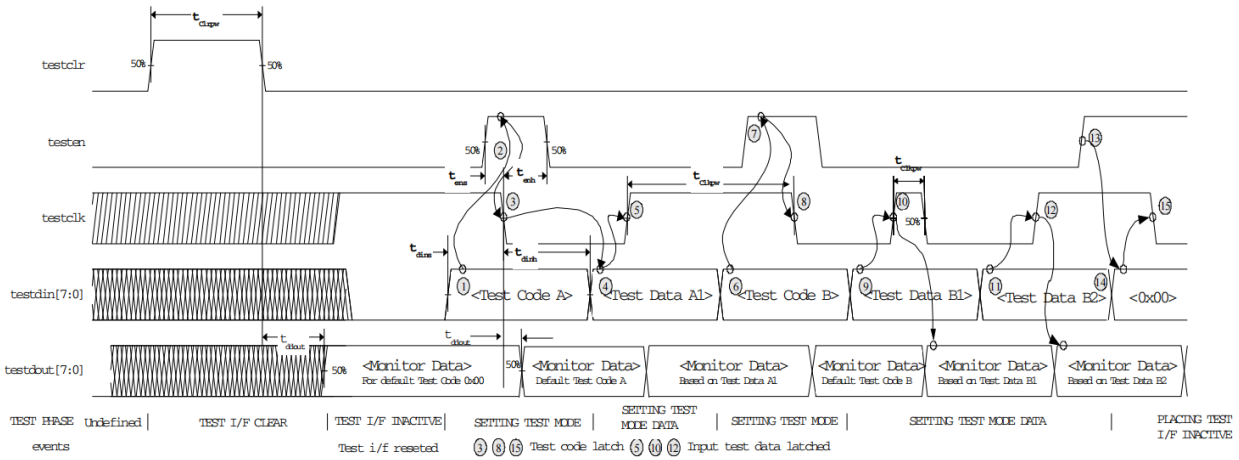


Figure 9-22 Test Port Timing

The follow diagram is test port initializing follow:

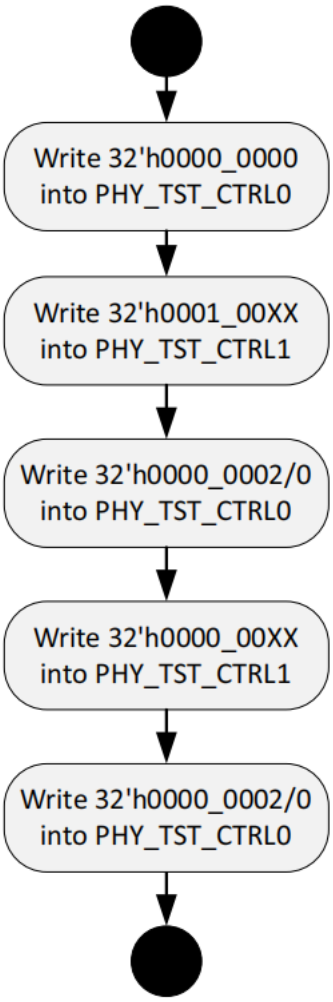


Figure 9-23 The process of PHY's register configuration

- Write 32'h00000000 into PHY_TST_CTRL0 to disable the testclr pin enabling the interface to write new values to the D-PHY internal registers.
- Write 32'h000100XX into PHY_TST_CTRL1 to enable the testen pin and configure the testdin[7:0] to 8'hXX. This operation initiates the configuration process of the test code number 0xXX.

- Write 32'h0000002 into PHY_TST_CTRL0 followed by writing 32'h00000000 into PHY_TST_CTRL0 to toggle testclk. The testdin[7:0] is sampled on the falling edge of testclk latching a new test code.
- Write 32'h000000XX into PHY_TST_CTRL1 to disable the testen pin and configure the testdin[7:0] to 8'hXX. This operation prepares the interface to load the test code 0xXX.
- Write 32'h00000002 into PHY_TST_CTRL0 followed by writing 32'h00000000 into PHY_TST_CTRL0 to toggle the testclk and the testdin[7:0] is sampled on the rising edge of testclk latching a new content data to the configured test code.

9.3 HDMI

9.3.1 Function description

HDMI receives audio data from the I2S, receives video data from the OSD, and outputs it to a display device with an HDMI interface. HDMI includes HDCP1.4 and HDCP2.2 functions, and the appropriate encryption method can be selected according to the support of the sink device

9.3.2 Feature

HDMI Tx supports the following features:

- Video formats:
  - All CEA-861-E video formats up to 1080p at 60 Hz and 720p/1080i at 120 Hz
  - HDMI 1.4b video formats
    - All CEA-861-E video formats up to 1080p at 120 Hz
    - HDMI 1.4b 4K x 2K video formats
    - HDMI 1.4b 3D video modes with up to 340 MHz (TMDS clock)
  - HDMI 2.0 video formats
    - All CEA-861-E video formats
- Colorimetry:
  - 24/30/36/48-bit RGB 4:4:4
  - 24/30/36/48-bit YCBCR 4:4:4
  - 16/20/24-bit YCBCR 4:2:2
  - xvYCBCR601
  - xvYCBCR709
  - HDMI 1.4b colorimetry:
    - sYCBCR601
    - Adobe RGB
    - Adobe YCBCR01
- Color space converter (CSC): RGB (4:4:4) to/from YCBCR (4:4:4 or 4:2:2)
- HDMI 1.4b supported Infoframes:
  - Audio InfoFrame packet extension to support LFE playback level information
  - AVI infoFrame packet extension to support YCBCR Quantization range (Limited Range, Full Range)
  - AVI infoFrame packet extension to support Content type (Graphics, Photo, Cinema, Game)

- NTSC VBI infoframe packet extension to support the carriage of SCTE 127 [29] payloads containing VBI data

- I2S andio interface
- Up to 192 kHz IEC60958 audio sampling rate, for IEC61937 compressed audio
  - HDMI 2.0b: Up to 1536 kHz
  - HDMI 1.4b: Up to 768 kHz
- Pixel clock from 13.5 MHz up to 600 MHz
- I2C, EDID block read mode
- SCDC I2C access
- TMDS Scrambler to enable support for 2160p@60 Hz with RGB/YCBCR 4:4:4 or YCBCR 4:2:2
- CEC hardware engine
- Single-channel DVI 1.0 backward compatibility

All HDMI sources are compatible with all DVI-compliant sinks and all HDMI sinks are compatible with DVI-compliant sources. All HDMI devices are compatible to the DVI 1.0 Specification, except some rules. For more information on these rules, see the *HDMI 1.4b Specification*.

9.3.3 Function Diagram

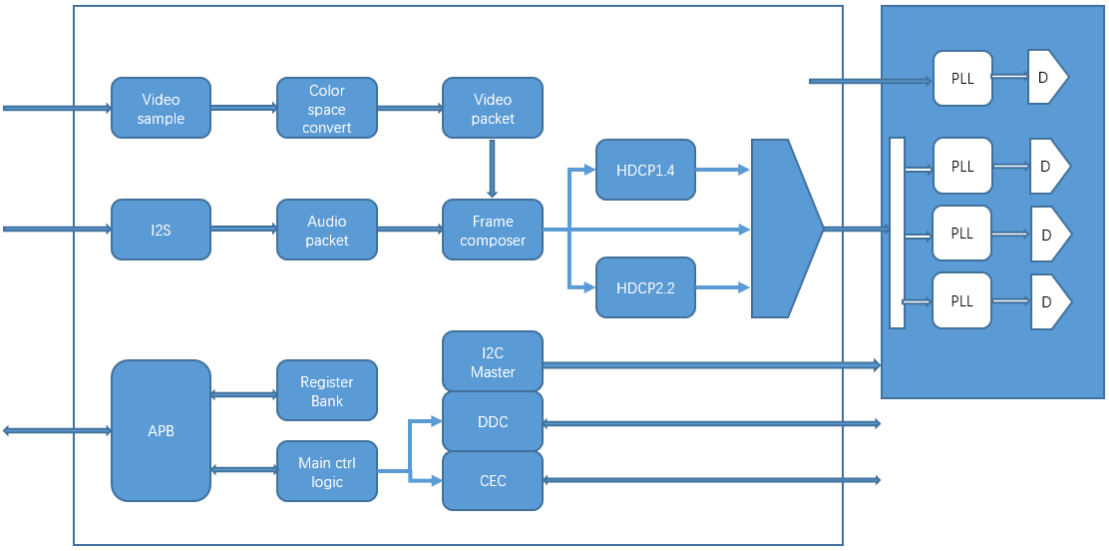


Figure 9-24 HDMI Block Diagram

9.3.3.1 Video Sample

The Video pixel sampler block synchronizes the video data, as per the video data input mapping defined by the Color Depth (Deep Color) and format configuration. Optionally, for YCBCR 4:2:2 format, data mapping can be performed to conform to ITU.601 and ITU.656 standards but without the support of the embedded synchronizers.

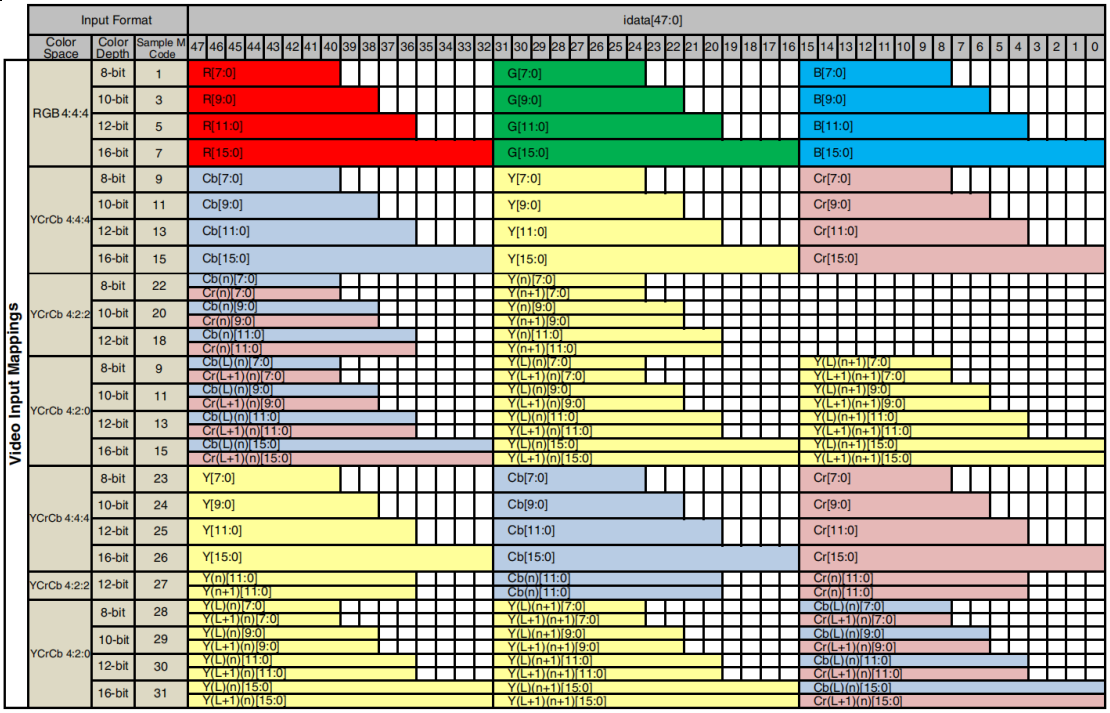


Figure 9-25 HDM Input Data Mapping

9.3.3.2 Color Space convert

This block is responsible for the following video color space conversion functions:

- RGB from/to YCbCr
- 422 from/to 444 up/down
- Limited to/from full quantization range

The mode diagram as follow:

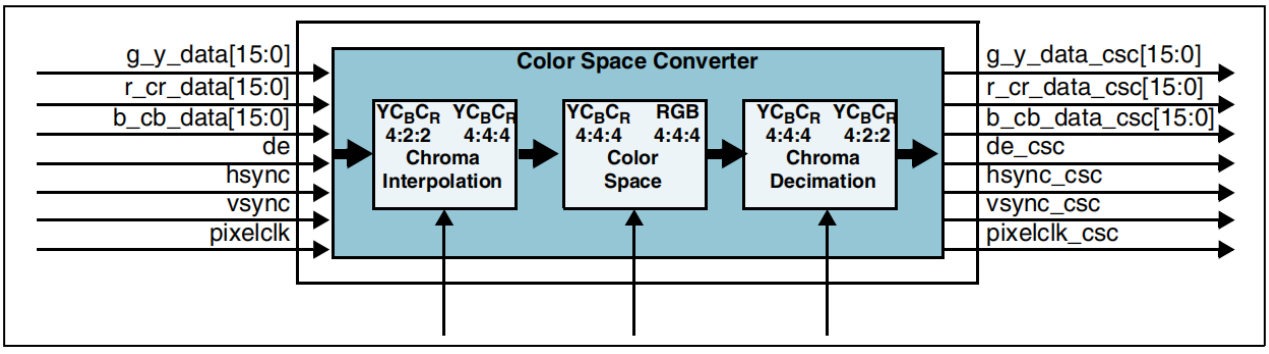


Figure 9-26 Color Space Convert

The color space conversion matrix is ruled by the following equations:

$$\begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix} = 2^{cscscale-14} \times \begin{bmatrix} A_1 & A_2 & A_3 \\ B_1 & B_2 & B_3 \\ C_1 & C_2 & C_3 \end{bmatrix} \begin{bmatrix} G \\ R \\ B \end{bmatrix} + 2^{cscscale-2} \times \begin{bmatrix} A_4 \\ B_4 \\ C_4 \end{bmatrix}$$

$$\begin{bmatrix} G \\ R \\ B \end{bmatrix} = 2^{cscscale-14} \times \begin{bmatrix} A_1 & A_2 & A_3 \\ B_1 & B_2 & B_3 \\ C_1 & C_2 & C_3 \end{bmatrix} \begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix} + 2^{cscscale-2} \times \begin{bmatrix} A_4 \\ B_4 \\ C_4 \end{bmatrix}$$

Figure 9-27 Color Space convert matrix

A_* B_* C_* is value of register, the color space conversion registers base address is 0x4100.

9.3.3.3 Video Packet

This block is responsible for the following:

- Pixel repetition
- 10-, 12-, and 16-bit packing when in deep color modes
- YCbCr 422 re-mapping according to the HDMI specification
- Clock rate transformation from pixel or repetition clock to the final TMDS clock domain

Depicts a functional diagram of the Video Packetizer block as follow diagram:

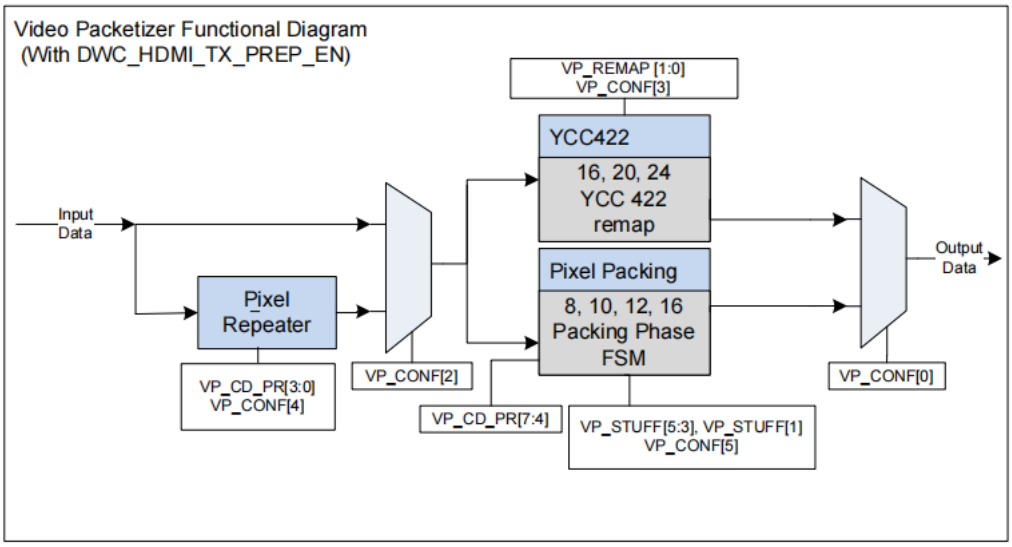


Figure 9-28 Video packet function

9.3.3.4 I2S audio interface

The I2S interface uses the I2S audio clock input to sample Linear-PCM input data and stores it in an input audio FIFO. There are one I2S data lines that support up to 192 kHz sampling rates (supports a maximum theoretical audio rate of 768 kHz or 1536 kHZ (supported in HDMI 2.0 only) for a two-channel standard I2S). The I2S interface is compliant with the I2S specification from NXP.

Each I2S interface supports two audio channels. The hdmi has one I2S interfaces that supports up to two audio channels simultaneously at 192 kHz. Each audio sample width can be configured to be from 16 bits up to 24 bits. The I2S_width field of the aud_conf1register selects the bit width for each right/left sample. Each right/left channel can carry 1 to N bits (N=16 to 24).

The i2slrck input signal must have the same frequency as the Audio Sampling Rate fs, that is 32 kHz to 192 kHz. The INPUTCLKFS setting must be consistent with the frequency of i2sclk.

For example:

If INPUTCLKFS = 128fs, and fs=32 kHz,

Then i2sclk = 128 x 32 kHz=4096 kHz

In I2S mode, the most significant bit of the sub-frame data (B bit) must be sent on the second active edge of ii2sclk following an ii2slrclk transition. The other bits up to the least significant bit (LSB) are then transmitted in order. Depending on word length, ii2sclk frequency, and sample rate, there may be unused ii2sclk cycles between the LSB of one sample and the MSB of the next.

The sub-frame can be as wide or larger than the sub-frame data size (such as, BPCUV+24 bits audio sample = 29 clock periods). In a typical scenario, the sub-frame can be 32 clocks wide, and all clock periods after the audio sample LSB are unused and i2sdata set to 0, regardless of audio sample width.

I²S Width	28	27	26	25	24	23	22	21	20	...	8	7	6	5	4	3	2	1	0
24	B	P	C	U	V	MSB													LSB
23	B	P	C	U	V	MSB												LSB	
22	B	P	C	U	V	MSB											LSB		
21	B	P	C	U	V	MSB										LSB			
20	B	P	C	U	V	MSB									LSB				
19	B	P	C	U	V	MSB								LSB					
18	B	P	C	U	V	MSB							LSB						
17	B	P	C	U	V	MSB						LSB							
16	B	P	C	U	V	MSB					LSB								

Figure 9-29 I2S Data Mapping

Because there is no audio clock carried through the HDMI link. Only the pixel clock is used. The CTS/N has to be set by software with value taken in the HDMI spec, the follow tables are recommended N and expected CTS value.

	TMDS Clock (MHz)													
	25.2		27		54		74.25		148.5		297		597	
Fs (kHZ)	N	CTS	N	CTS	N	CTS	N	CTS	N	CTS	N	CTS	N	CTS
32	4096	25200	4096	27000	4096	54000	4096	74250	4096	148500	3072	222750	3072	445500
44.1	6272	28000	6272	30000	6272	60000	6272	82500	6272	165000	4704	247500	9408	990000
48	6144	25200	6144	27000	6144	54000	6144	74250	6144	148500	5120	247500	6144	495000
64	8192	25200	8192	27000	8192	54000	8192	74250	8192	148500	8192	247500	8192	594000
88.2	12544	28000	12544	30000	12544	60000	12544	82500	12544	165000	9408	247500	18816	990000
96	12288	25200	12288	27000	12288	54000	12288	74250	12288	148500	10240	247500	12288	495000
128	16384	25200	16384	27000	16384	54000	16384	74250	16384	148500	16384	247500	16384	594000
176.4	25088	28000	25088	30000	25088	60000	25088	82500	25088	165000	18816	247500	37632	990000
192	24576	25200	24576	27000	24576	54000	24576	74250	24576	148500	20480	247500	24576	4950000
256	32768	25200	32768	27000	32768	54000	32768	74250	32768	148500	32768	297000	32768	594000
352.8	50176	28000	50176	30000	50176	60000	50176	82500	50176	165000	37632	247500	75264	990000
384	49152	25200	49152	27000	49152	54000	49152	74250	49152	148500	40960	247500	49152	594000

Figure 9-30 TMDS clock VS N and CTS

The TMDS clocks divided or multiplied by 1,001 coefficients are supported by configuring the aud_cts_dither register with the corresponding ratio. The CTS registers value configured are always the larger value of the interval. If the fractional portion of CTS values is:

- 0.50 – aud_cts_dither register must be configured with 8'h12 (1/2 duty cycle)
- 0.25 – aud_cts_dither register must be configured with 8'h14 (1/4 duty cycle)
- 0.75 – aud_cts_dither register must be configured with 8'h34 (3/4 duty cycle)
- 0.875 – aud_cts_dither register must be configured with 8'h7/8 (7/8 duty cycle, 7 with the larger value one with the smaller value).

	TMDS Clock (MHz)													
	25.2/1.001		27 54*1.001		54*1.001		74.25/1.001		148.5/1.001		297/1.001		594/1.001	
Fs (kHz)	N	CTS	N	CTS	N	CTS	N	CTS	N	CTS	N	CTS	N	CTS
32	4576	28125	4096	27027	4096	54054	11648	210938[a]	11648	421875	5824	421875	5824	843750
44.1	7007	31250	6272	30030	6272	60060	17836	234375	8918	234375	4459	234375	8918	937500
48	6864	28125	6144	27027	6144	54054	11648	140625	5824	140625	5824	281250	5824	562500
64	9152	28125	8192	27027	8192	54054	23296	210938[a]	23296	421875	11648	421875	11648	843750
88.2	14014	31250	12544	30030	12544	60060	35672	234375	17836	234375	8918	234375	17836	937500
96	13728	28125	12288	27027	12288	54054	23296	140625	11648	140625	11648	281250	11648	562500
128	18304	28125	16384	27027	16384	54054	46592	210938[a]	46592	421875	23296	421875	23296	843750
176.4	28028	31250	25088	30030	25088	60060	71344	234375	35672	234375	17836	234375	35672	937500
192	27456	28125	24576	27027	24576	54054	46592	140625	23296	140625	23296	281250	23296	562500
256	36608	28125	32768	27027	32768	54054	93184	210938[a]	93184	421875	46592	421875	46592	843750
352.8	56056	31250	50176	30030	50176	60060	142688	234375	71344	234375	35672	234375	71144	937500
384	54912	28125	49152	27027	49152	54054	93184	140625	46592	140625	46592	281250	46592	562500

Figure 9-31 TMDS clock frequency VS N and CTS

9.3.3.5 Frame Composer

This block assembles the video, audio, and data packets in a consistent frame that are streamed to the HDCP cipher and then finally to the HDMI TX PHY.

The HDMI standard describes the packet insertion timing and distribution that must be followed to correctly compose an HDMI TMDS (transition minimized differential signaling) stream. In this context, there are data island packets that—when available (ready for insertion in output stream)—have higher priority over others. Two packet descriptor queues are responsible for prioritizing packet insertion.

Table 9-8 High Priority Data Island Packets

Packet	Description
Audio Clock Regeneration (ACR)	Indicates to sink device the N/CTS values that should be used in the ACR process
Audio Sample (AUDS)	Transports L-PCM and IEC 61937 compressed audio
General Control (GCP)	Indicates Color Depth, Pixel Packing phase, and AV mute information to sink device

Table 9-9 Low Priority Data Island Packets

Packet	Classification
Audio Clock Regeneration (ACR)	Sent on data availability
Audio Sample (AUDS)	Sent on data availability (precede ACR if present)
Audio Content Protection (ACP)	On user request or automatic insertion
Audio InfoFrame (AUDI)	Once per two frames
Null (NULL)	On user request or automatic insertion to fill Data Island period
General Control (GCP)	Once per frame
International Standard Recording Code (ISRC1/ISRC2)	On user request
Vendor Specific (VSD) InfoFrame	On user request or automatic insertion
AVI infoFrame (AVI)	Once per frame
Source Data Product Descriptor (SPD) infoFrame	On user request or automatic insertion

9.3.3.6 HDCP

HDCP1.4 and HDCP2.2 are responsible for display data encryption, only one of the encryption methods can be selected to work, or both can be bypassed, whether the receiver supports HDCP, the specific support mode can query the receiving device through the DDC channel.

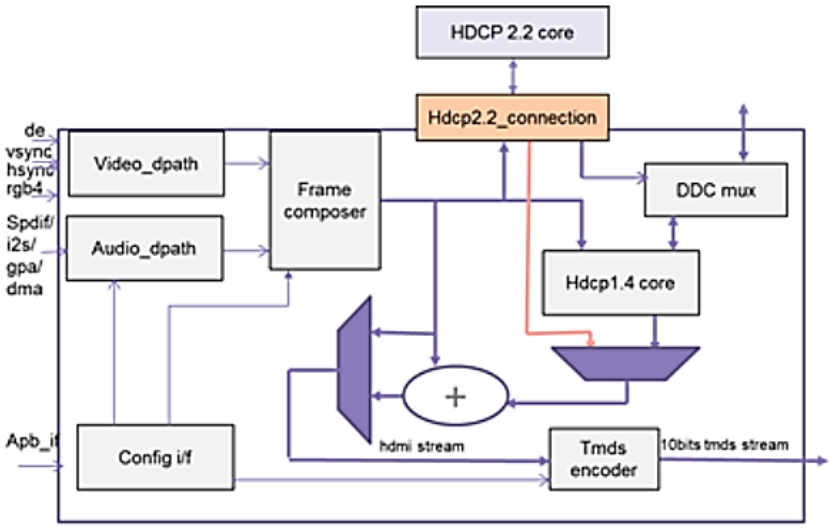


Figure 9-32 HDMI1.4 and HDMI 2.2

9.3.4 HDMI Initialization

Programming sequence is showed in the follow picture:

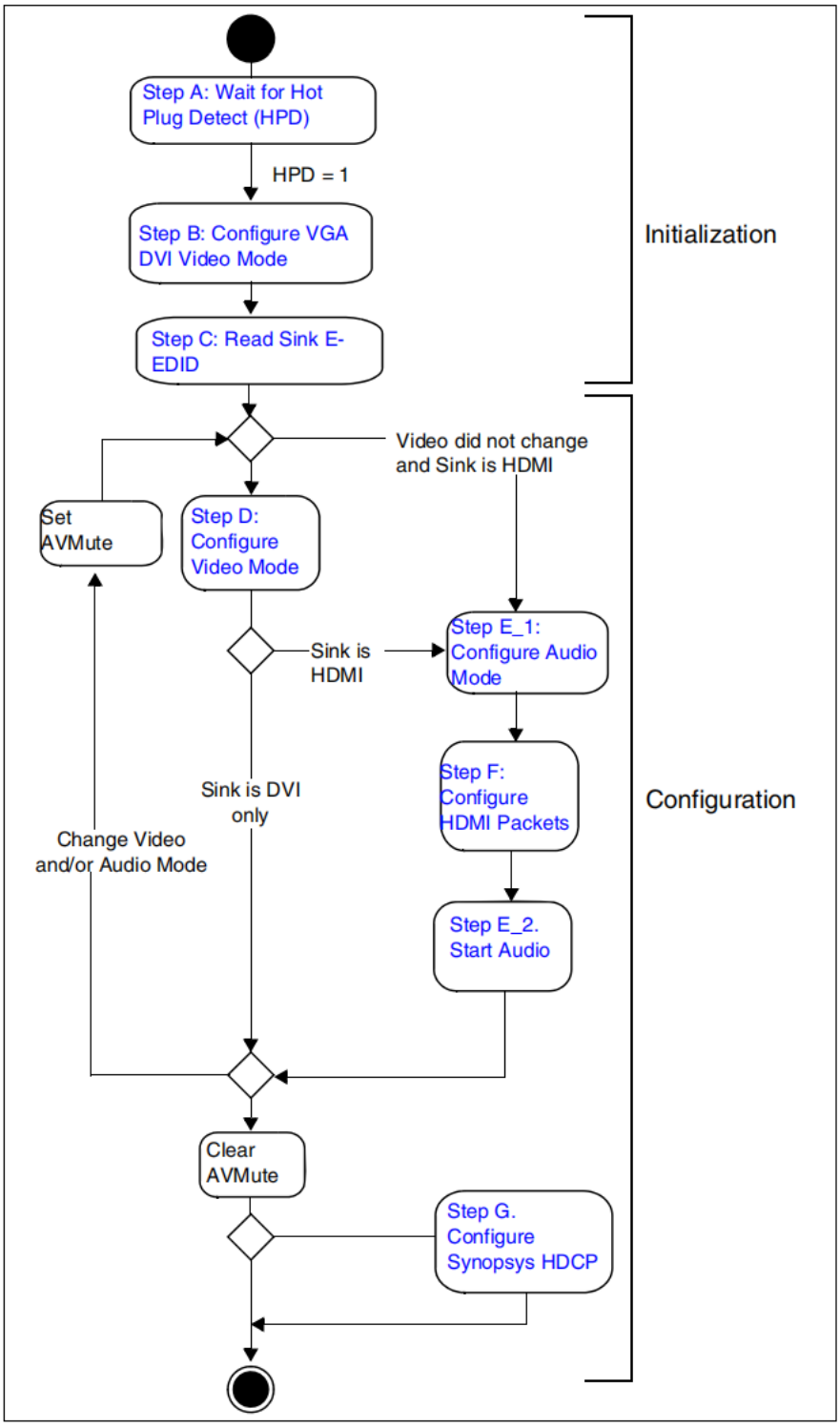


Figure 9-33 HDMI Initialization

9.3.4.1 Step A: Wait for Hot Plug Detect (HPD)

The Hot Plug Detect information notifies the HDMI controller when the cable is plugged into a Sink (Receiver) device and when the device is ready to receive video content.

To check the Hot Plug Detect status, the following steps have to be followed:

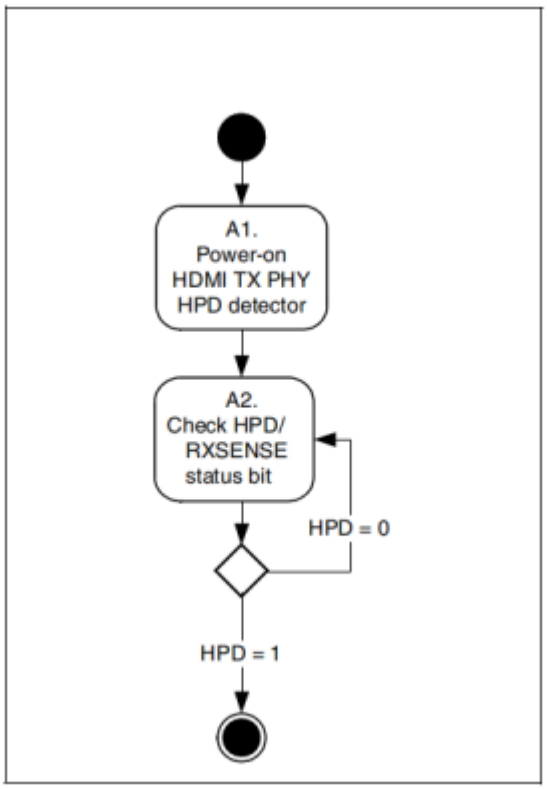


Figure 9-34 Hot Plug Detect

- Power-on the HDMI TX PHY HPD Detector, write 1'b1 in the phy_conf0.enhpdrxsense bit field register.
- Read the phy_stat0.HPD bit field register.
  - If HPD = 0, the Hot Plug signal is low (no Sink (Receiver) detected).
  - If HPD = 1, the Hot Plug signal is high (Sink (Receiver) detected).
- Read the phy_stat0.RX_SENSE_* bits field register.
  - If RX_SENSE_* = 0, the Sink (Receiver) is not detected by the PHY.
  - If RX_SENSE_* = 1, the Sink (Receiver) is detected by the PHY.

This status can be verified during normal operation to determine if the Sink (Receiver) is no longer connected and/or powered.

It is also possible to use interrupts instead of polling the phy_stat0 register by configuring the appropriate bit fields in registers phy_mask0, phy_pol0, and ih_mute_phy_stat0 during this step.

9.3.4.2 Step B: Configure VGA DVI Video Mode

Once a receiver is detected, it is necessary to start sending video content in DVI mode. Because some receivers need to receive a video signal (TMDS clock more precisely), it is important to clock all the digital logic and be able to reply on an E-EDID read access

The follow diagram is the configuration process:

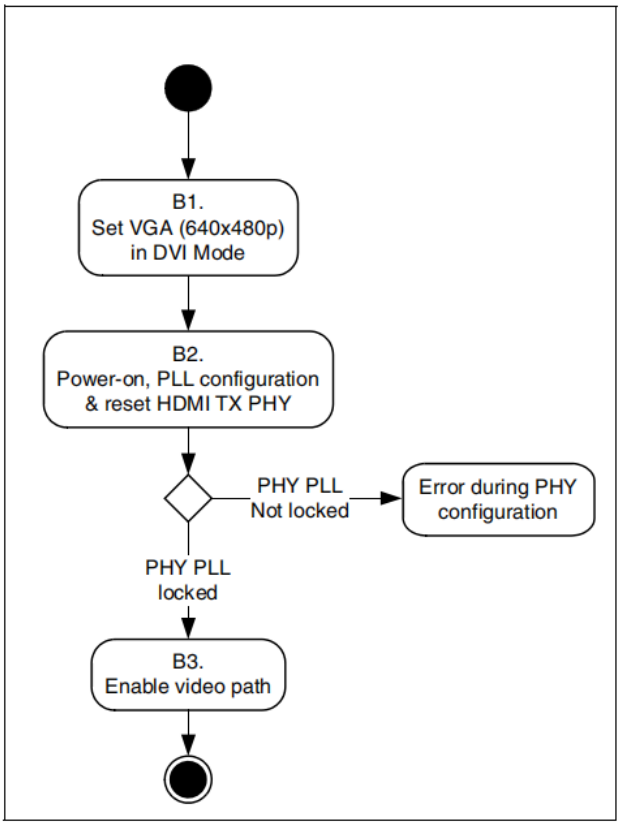


Figure 9-35 VGA configuration process

- **Set VGA (640x480p) in DVI mode.**
  - To select the Video Mapping input RGB444, 8-bits per color components (24-bits RGB), write "1" in the tx_invid0.video_mapping register.
  - To configure VGA timing information (CEA mode 1):
    - Write "0" in the fc_invidconf.vsync_in_polarity register.
    - Write "0" in the fc_invidconf.hsync_in_polarity register.
    - Write "1" in the fc_invidconf.de_in_polarity register.
    - Write "0" in the fc_invidconf.r_v_blank_in_osc register.
    - Write "0" in the fc_invidconf.in_I_P register.
    - 640 H active pixels: 0x280

      - Write "2" in the fc_inhactiv1.H_in_activ register.

      - Write "0x80" in the fc_inhactiv0.H_in_activ register.
    - 480 V active pixels: 0x1E0

      - Write "1" in the fc_invactiv1.V_in_activ register.

      - Write "0xE0" in the fc_invactiv0.V_in_activ register.
    - 160 H blanking pixels: 0xA0

      Write "0xA0" in thefc_inhblank0.H_in_blank.
    - 45 V blanking pixels: 0x2D

Write “0x2D” in the fc_invblank0.V_in_blank register.

- 16 Sync offset: 0x10

  Write “0x10” in the fc_hsyncindelay0.H_in_delay register.

- 10 Vsync offset: 0x0A

  Write “0x0A” in the fc_vsyncindelay0.V_in_delay register.

- 96 HSync pulse width: 0x60

  Write “0x60” in the fc_hsyncinwidth0.H_in_width register.

- 2 VSync pulse width: 0x02

  Write “0x02” in the fc_vsyncinwidth0.V_in_width register.

- To select the DVI mode:

  Write “0” in the fc_invidconf.DVI_modez register.

- **Power-on, configure the PLL and the HDMI TX PHY.**
  - Configure the HDMI PHY as described in the following sections:
  - At the end of the PHY configuration, it is recommended to check if the PHY PLL is locked.
    - Read the phy_stat0.TX_PHY_LOCK bit field register.

      - If TX_PHY_LOCK = 0, the PLL is not locked.

      - If TX_PHY_LOCK = 1, the PLL is locked.
    - Perform a reset to all clock domains by writing 0x00 in the mc_swrstzreq_1 register.
    - Re-Write the VSync pulse width in the fc_vsyncinwidth0.V_in_width bit field register.
- **Enable the video path.**
  - Set default parameters in hdmi tx, Control period duration: 12: 0x0C.
    - Write “0x0C” in the fc_ctrldur.ctrlperiodduration bit field register.
  - Set default parameters in hdmi tx, Extended Control period duration: 32: 0x20.
    - Write “0x20” in the fc_exctrldur.exctrlperiodduration bit field register.
  - Set default parameters in hdmi tx, Max spacing between extended Control period duration: 1: 0x1.
    - Write “0x01” in the fc_exctrlspac.exctrlperiodspacing bit field register.
  - Enable pixel clock data path:
    - Write “0” in the mc_clkdis.pixelclk_disable bit field register.
  - Enable TMDS clock data path:
    - Write “0” in the mc_clkdis.tmdsclk_disable bit field register.
    - Re-Write the VSync pulse width in the fc_vsyncinwidth0.V_in_width bit field register.

If you want to force a fixed color screen, please follow the next steps:

- Set the RGB color code using the following registers: fc_dbgtmds0, (Blue)fc_dbgtmds1, (Green) fc_dbgtmds2 (Red).
- Enable video fixed color forcing: Write “1” the fc_dbgforce.forcevideo register.

- **Configuring PHY**

To configure TX PHY, you are generally required to:

- Place the PHY in power down mode, starting by writing 1'b0 to phy_conf0.txpwron bit field register and 1'b1 to the phy_conf0.pddq bit field register.
- Use the PHY I2C interface, set I2C_JTAZ to 1.
- Reset the PHY by writing 0x01 in the mc_phyrstz register.
- Write the desired color depth and the pixel repetition in the vp_pr_cd register.
- After a PHY-dependent time, it is required to lift the reset by writing 0x00 to the mc_phyrstz register, which places the PHY in configuration mode.
- Using PHY I2C interface:
  - Write to jtag_phy_conf(0x3034) {3'b0, 1'b1,4'd0}.
  - Set the PHY slave address by writing 0x69 in the phy_i2cm_slave register.
  - According to the PLL register table, you are required to look up the configuration for your intended video mode and write those values to the PHY I2C interface.

    -Write the register address in the phy_i2cm_address register.

    - Write data in the phy_i2cm_datao_1 (MSB, [15:8]) and phy_i2cm_datao_0 (LSB, [7:0]) registers.

    - Initialize the write operation by writing 8'h10 in the phy_i2cm_operation register.

    - Wait for a done interruption from the I2C master.
- After all of the required PHY registers have been configured, you now need to place the PHY in power-on mode:
  - Write 1'b0 to the phy_conf0.pddq bit field register.
  - Write 1'b1 to the phy_conf0.txpwron bit field register.

9.3.4.3 Step C: Read Sink E-EDID

The E-EDID is a memory present on the receiver side. It contains the video and audio capabilities of the receiver. The E-EDID is read using the E-DDC channel present on the HDMI cable.

- To perform a “normal” read operation
  - Set I2C slave address, write i2cm_slave.slaveaddr[6:0] bit field register.
  - Set I2C register address, write i2cm_address.address[7:0] bit field register.
  - Activate Read operation, write “1” in the i2cm_operation.rd bit field register.
  - Wait for interruption, wait for the ii2cmasterdone interrupt in the ih_i2cm_stat0 register.
  - Read data result, read data in the i2cm_datai.datai[7:0] bit field.
- To perform an E-DDC extended read operation
  - Set I2C slave address, write i2cm_slave.slaveaddr[6:0] bit field register.
  - Set I2C segment address, write the i2cm_segaddr.seg_addr bit field register.
  - Set I2C segment pointer, write i2cm_segptr.segptr bit field register.
  - Activate Read operation, write “1”in the i2cm_operation.rd_ext bit field register.
  - Wait for interruption, wait for the ii2cmasterdone interrupt in the ih_i2cm_stat0 register.

- Read data result, read data in the i2cm_datai.datai[7:0] bit field register.

It is also possible to issue sequential I2C accesses to lower read latency and number of interrupt events.

- To perform a “normal” sequential read operation
  - Set I2C slave address, write i2cm_slave.slaveaddr[6:0] bit field register.
  - Set I2C register address, write i2cm_address.address[7:0] bit field register.
  - Activate Sequential read operation, Write “1” in the i2cm_operation.rd8 bit field register.
  - Wait for interruption, wait for ii2cmasterdone interrupt in the ih_i2cm_stat0 register
  - Read data result, read data of registers i2cm_read_buff0[7:0] to i2cm_read_buff7[7:0].
- To perform an E-DDC extended sequential read operation
  - Set I2C slave address, write i2cm_slave.slaveaddr[6:0] bit field register.
  - Set I2C segment address, write the i2cm_segaddr.seg_addr bit field register.
  - Set I2C segment pointer, write i2cm_segptr.segptr bit field register.
  - Activate Read operation, write “1”in the i2cm_operation.rd8_ext bit field register.
  - Wait for interruption, wait for ii2cmasterdone interrupt in the ih_i2cm_stat0 register
  - Read data result, read data of registers i2cm_read_buff0[7:0] to i2cm_read_buff7[7:0].

9.3.4.4 Step D: Configure Video Mode

The video mode selected has to match what is supported by the source of the video (Transmitter side) and the sink of the video (receiver side). The video capabilities of the receiver are extracted from the E-EDID information (see step C).

To configure any video mode, a similar procedure in step B has to be followed.

Before any change is made on the video modes, is it recommended you power-down the HDMI TX PHY to avoid any unexpected behavior on the receiver side.

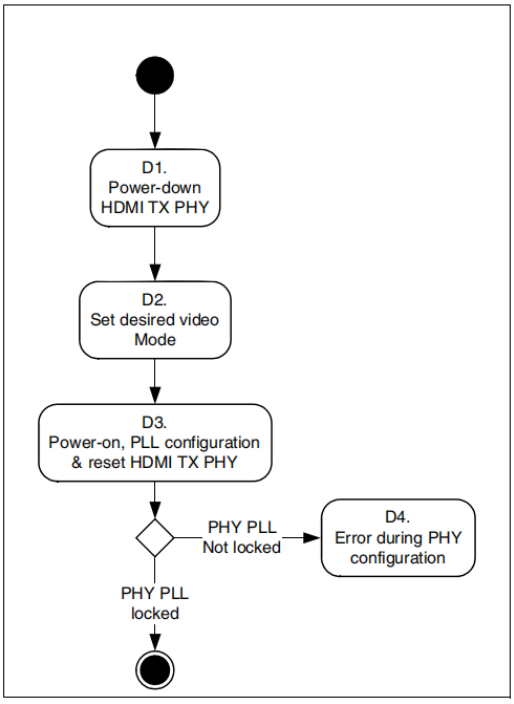


Figure 9-36 Video Mode Configuration

This is the only difference between this setp and Step B:

Select DVI or HDMI mode, user need config this register according to sink device:

Write "0" for DVI in the fc_invidconf.DVI_modez bit field register.

Write "1" for HDMI in the fc_invidconf.DVI_modez bit field register.

9.3.4.5 Step E: Configure Audio Mode

This step is only relevant when the controller is configured in HDMI mode. The audio mode selected has to match what is supported by the source of the audio (Transmitter side) and the sink of the audio (receiver side). The audio capabilities of the receiver are extracted from the E-EDID information.

- Disable the transmission of the ACR packet by writing 1'b0 to the fc_packet_tx_en.acr_tx_en bit field register.
- Disable the audio clock by writing 1'b1 to the mc_clkdis.audclk_disable bit field register.
- Configure I2S input audio interface:
  - Select I2S interface, write "1" in the aud_conf0.i2s_select bit field register.
  - Enable I2S inputs, write "1" in the aud_conf0.i2s_in_en[3:0] bit field register.
  - Set I2S data width [16 bits up to 24 bits], write the aud_conf1.i2s_width[4:0] bit field.
  - Define audio samples transport method (transmission through HBR, AUDS, or MAS packets), write the aud_conf2.HBR bit field register.
  - Define audio samples type (NL-PCM Vs L-PCM), write the aud_conf2.NLPCM bit field register.
  - Define the insertion of PCUV, write the aud_conf2.INSERTPCUV bit field register.
- Configure audio parameters:
  - Set Audio input frequency clock FS ratio factor [64 Fs], write the aud_inputclkfs.lfsfactor bit field register.
  - Set Audio fixed N factor for Audio Clock Regeneration. This factor depends on the audio sampling rate and video mode. For more information, see the Audio Chapter "HDMI 1.4b Specification", write the aud_n1.audN, aud_n2.audN, and aud_n3.audN bit field registers.
  - Set Audio CTS factor for Audio Clock Regeneration. This factor can be generated automatically or manually. For more information, for more information, see the Audio Chapter "HDMI 1.4b Specification".
    - For Automatic CTS generation, write "0" on the bit field "CTS_manual", Register 0x3205: AUD_CTS3
    - For Manual CTS setting, write "1" in the aud_cts3.CTS_manual register bit field, write the aud_cts1.audCTS, aud_cts2.audCTS, aud_cts3.audCTS bit field registers.

Automatic CTS mode is recommended.

9.3.4.6 Step F: Configure HDMI Packets

This step is only relevant when the controller is configured in HDMI mode (see "Step D: Configure Video Mode"). In DVI mode, no Infoframes are transmitted. The configuration of the Infoframes is essential to correctly inform the HDMI Receiver about the content of the video and audio format been transmitted. All the detailed information is provided in the HDMI 1.4b Specification.About all of packet configuration, please see the tables below for details.

Table 9-10 AVI info packet

AVI InfoFrame	Register to Configure
Y2, Y1, Y0 - RGB or YCbCr indicator[1]	Register 0x1019 bits [7],[1],[0]
A0 - Active information present	Register 0x1019 bits [6]
B1,B0 - Bar Info data valid	Register 0x1019 bits [3:2]
S1,S0 - Scan information	Register 0x1019 bits [5:4]
C1,C0 - Colorimetry	Register 0x101A bits [7:6]
M1,M0 - Picture Aspect Ratio	Register 0x101A bits [5:4]
R3…R0 - Active Format Aspect Ratio	Register 0x101A bits [3:0]
ITC - IT Content	Register 0x101B bits [7]
EC2…EC0 - Extended Colorimetry	Register 0x101B bits [6:4]
Q1,Q0 - RGB Quantization Range	Register 0x101B bits [3:2]
SC1,SC0 - Non Uniform Picture Scaling	Register 0x101B bits [1:0]
VIC7…VIC0 - Video Format Identification Code1	Register 0x101C bits [7:0]
YQ1,YQ0 - YCC Quantization Range	Register 0x1017 bits [3:2]
CN1,CN0 - IT Content Type	Register 0x1017 bits [1:0]
PR3…PR0 - (Outgoing) Pixel Repetition	Register 0x10E0 bits [3:0]

Table 9-11 Audio information frame

Audio Infoframe	Register to Configure
CC2...CC0 - Channel Count	Register 0x1025 bits [6:4]
CT3…CT0 - Coding Type	Register 0x1025 bits [3:0]
SS1,SS0 - Sample Size	Register 0x1026 bits [4:3]
SF2…SF0 - Sampling Frequency	Register 0x1026 bits [2:0]
CA7…CA0 - Channel Allocation	Register 0x1027 bits [7:0]
LSV3…LSV0 - Level Shift Value	Register 0x1028 bits [3:0]
DM_INH - Downmix Inhibit	Register 0x1028 bits [4]
LFEPBL1,LFEPBL0 – LFE Playback level information	Register 0x1028 bits [6:5]

HDMI Vendor-Specific Infoframe	Register to Configure
24bits IEEE ID	Register 0x1029-0x1031 bits [7:0]
In case a HDMI 4Kx2K format is desired to be transmitted, the following registers need to be configured:	
HDMI Video Mode	
HDMI_VIDEO_FORMAT – Structure of Extended Video	Register 0x1032 bits [7:5] = 0x01
HDMI_VIC – HDMI Video Format	Register 0x1033 bits [7:0]
In case a HDMI 3D Mode format is desired to be transmitted, the following registers need to be configured:	
3D Video Mode	
HDMI_VIDEO_FORMAT – Structure of Extended Video	Register 0x1032 bits [7:5] = 0x02
3D_Structure – Format of 3D Video data	Register 0x1033 bits [7:4]
3D_Meta_present	Register 0x1033 bits [3]
3D_EXT_DATA – Additional Video Format Information	Register 0x1034 bits [7:4]
3D_Metadata_type	Register 0x1035 bits [7:5]
3D_Metadata_Length	Register 0x1035 bits [4:0]
3D_Metadata	Register 0x1036-0x103D bits [7:0]

Figure 9-37 Vendor Specific Packet config

9.3.4.7 HDCP1.4 configration

- Select DVI or HDMI mode, write “0” in the a_hdcpcfg0.hdmidvi bit field register for DVI mode, write “1” in the a_hdcpcfg0.hdmidvi bit field register for HDMI mode.
- Start the HDCP keepout window by setting fc_invidconf.HDCP_keepout = 1.
- Set the Data enable, Hsync, and VSync polarity, write the dataenpol, vsyncpol, and hsyncpol bit fields of the a_vidpolcfg register.
- Set encryption:
  - Write “64 (0x40)” for “OesswindowSize” in the a_oesswcfg register.
  - Write “0” for “no HDCP bypass” a_hdcpcfg0.BypassEncryption bit field register.
- Write “0” for “HDCP enable” in the a_hdcpcfg1.encryptiondisable bit field register.
- Reset the HDCP 1.4 engine, by writing “0” in the a_hdcpcfg1.swreset bit field register.
- Configure Device Private Keys
- Enable encryption, by writing “1” in the a_hdcpcfg0.rxdetect bit field register.

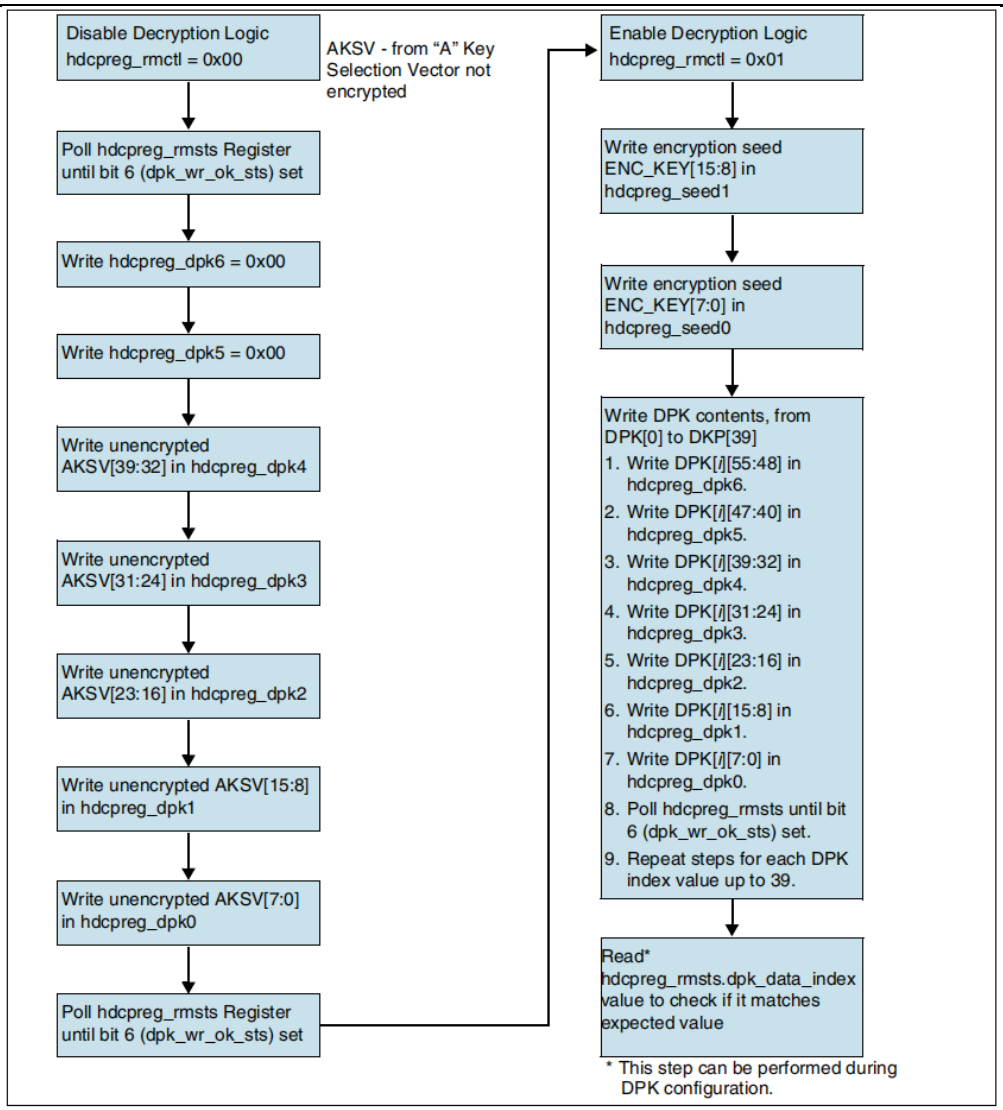


Figure 9-38 HDCP1.4 config steps

9.3.4.8 PHY PLL configration

The configuration parameters of the HDMI PHY PLL are related to the pixel clock frequency, color depth and pixel repetion, PLL are configured through the I2C interface. The following table describes the configuration parameters:

Table 9-12 HDMI PHY 27MHz

Pixel clock(MHz)	Pixel Repetion	Color Depth	Opmode	Divider setting				Op mode		Divider Setting				MPLL Charge Pump Setting												Divider Setting												TMDSCLSSRC
				0x06										0x10												0x11												
				prep_div		mpll_cko_div		opmode		ref_cntrl		mpll_n_cntrl		gmp_cntrl		prop_cntrl				int_cntrl						vco_cntrl				mpll_multiplier								
				Bit13	Bit10	Bit9	Bit8	Bit7	Bit6	Bit4	Bit3	Bit1	Bit0	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Bit10	Bit9	Bit8	Bit7	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	
27	no	8	1.4	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	0	0	27
27	no	10	1.4	0	1	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	1	1	0	0	1	1	0	0	1	0	33.75
27	no	12	1.4	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	0	0	0	1	0	0	0	0	1	0	0	1	1	1	1	0	0	40.5
27	no	16	1.4	1	1	0	0	0	0	0	0	1	0	0	1	0	0	1	0	1	0	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	0	0	54
27	1	8	1.4	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	1	0	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	0	0	54
27	1	10	1.4	0	1	0	0	0	0	0	0	1	0	0	1	0	0	1	0	1	0	0	0	0	0	0	1	0	0	1	1	0	0	1	1	0	0	1	0	67.5
27	1	12	1.4	1	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	1	1	1	0	0	81
27	1	16	1.4	1	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	0	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	0	0	108
27	3	8	1.4	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	1	0	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	0	0	108
27	3	10	1.4	0	1	0	0	0	0	0	0	0	1	1	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	1	1	0	0	1	1	0	0	1	0	135
27	3	12	1.4	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	1	1	0	0	0	0	1	0	0	0	0	1	0	0	1	1	1	1	0	0	162
27	3	16	1.4	1	1	0	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	0	0	216
27	7	8	1.4	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	0	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	0	0	216
27	7	10	1.4	0	1	0	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	0	0	0	0	0	1	0	0	1	1	0	0	1	1	0	0	1	0	270
27	7	12	1.4	1	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	0	0	0	0	1	0	0	0	0	1	0	0	1	1	1	1	0	0	324
27	7	16	1.4	1	1	1	1	0	1	0	0	0	0	1	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	432

Table 9-13 HDMI PHY40~148.5MHz

Pixel clock(MHz)	Pixel Repetion	Color Depth	Opmode	Divider setting				Op mode		Divider Setting				MPLL Charge Pump Setting												Divider Setting										TMDSCLSSRC
				0x06										0x10												0x11										
				prep_div		mpll_cko_div		opmode		ref_cntrl		mpll_n_cntrl		gmp_cntrl		prop_cntrl						int_cntrl						vco_cntrl				mpll_multiplier						
				Bit13	Bit12	Bit10	Bit9	Bit7	Bit6	Bit4	Bit3	Bit1	Bit0	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Bit12	Bit11	Bit10	Bit9	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	
40	no	8	1.4	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0	1	0	0	1	0	1	0	0	0	40
65	no	8	1.4	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	0	0	0	0	1	1	0	0	1	1	0	0	0	1	0	1	0	0	65
74.25	1	8	1.4	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0	74.25
74.25	1	10	1.4	0	1	0	0	0	0	0	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0	1	1	0	0	1	92.8125
74.25	1	12	1.4	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	1	1	0	0	0	0	1	1	1	1	111.375
74.25	1	16	1.4	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0	148.5
148.5	no	8	1.4	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	148.5
148.5	no	10	1.4	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	185.625
148.5	no	12	1.4	1	0	0	0	0	0	0	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0	1	1	0	0	1	222.75
148.5	no	16	1.4	1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	297
148.5	1	8	1.4	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	297
148.5	1	10	2.0	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	1	0	1	0	371.25
148.5	1	12	2.0	1	0	1	1	0	1	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1	445.5
148.5	1	16	2.0	1	1	1	1	0	1	0	0	0	0	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	594
148.5	3	8	2.0	0	0	1	1	0	1	0	0	0	0	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	594

Table 9-14 HDMI PHY 297MHz AND594MHz

Pixel clock(MHz)	Pixel Repetion	Color Depth	Opmode	Divider setting				Op mode		Divider Setting				MPLL Charge Pump Setting												Divider Setting								TMDSCLSSRC
				0x06										0x10												0x11								
				prep_div		mpll_cko_div		opmode		ref_cntrl		mpll_n_cntrl		gmp_cntrl		prop_cntrl						int_cntrl						vco_cntrl				mpll_multiplier								
				Bit13	Bit12	Bit10	Bit9	Bit7	Bit6	Bit4	Bit3	Bit1	Bit0	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Bit12	Bit11	Bit10	Bit9	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	
297	no	8	1.4	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	1	297
297	no	10	2.0	0	1	1	1	0	1	1	1	0	0	1	1	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	1	0	0	1	371.25
297	no	12	2.0	1	0	1	1	0	1	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1	445.5
297	no	16	2.0	1	1	1	1	0	1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	594
297	1	8	2.0	0	0	1	1	0	1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	594
594	no	8	2.0	0	0	1	1	0	1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	594

9.3.4.9 HDCP2.2 Initialization

- The ESM reset is de-asserted and the ESM enters a reset state. However, all values must be valid before booting the ESM using the host library software and cannot change once the ESM is booted.
- The host allocates contiguous memory for the firmware and R/W areas. The buffers are 4K aligned.
- The host configures the ESM's HPI interface, defining the physical memory pointer for the ESM firmware.
- The host copies the ESM firmware from persistent storage to the firmware memory location.
- The host enables the ESM controller, putting it in a running state. At this point the ESM initiates transactions on the AXI bus to fetch its firmware.
- ESM asserts a GPIO to indicate it is booted to indicate it can accept commands. The status can be monitored from the HPI interface.

9.3.5 Interface timing

The display data comes from the OSD, and the interface timing is shown in the DP interface of the OSD.

10 High-Speed Interface

10.1 PCI-Express

10.1.1 Overview

PCI-Express is a high performance I/O bus used to interconnect peripheral device such as computing and communication platforms, and has following key features.

- PCIe Root-Complex(RC) and End-Point(EP) dual mode.
- Support PCIe Gen3 x4, compatible with PCIe Specification Revision 3.0
- EP mode supports one physiacal function.
- EP mode supports 4 BARs, each BAR size is independent configurable.
- EP mode supports INTX, MSI, MSI-X interrupt and PVM function.
- RC mode supports receiving INTX, MSI interrupt.
- Embedded 8 DMA read channals and 8 DMA write channels, and support DMA link-list mode.
- Support iATU(address mapping),4 inbound regions and 32 outbound regions, each outbound size is configurable which up to 8GB.
- Support L0s/L1/L1sub and D1/D2/D3hot low power mode , support PM and ASPM mechanism
- Support PERST#, Hot Reset, FLR reset .
- RC mode supports ATS.
- Max Payload Size is 512Byte, Non-Posted outstanding up to 32.

10.1.2 Block diagram

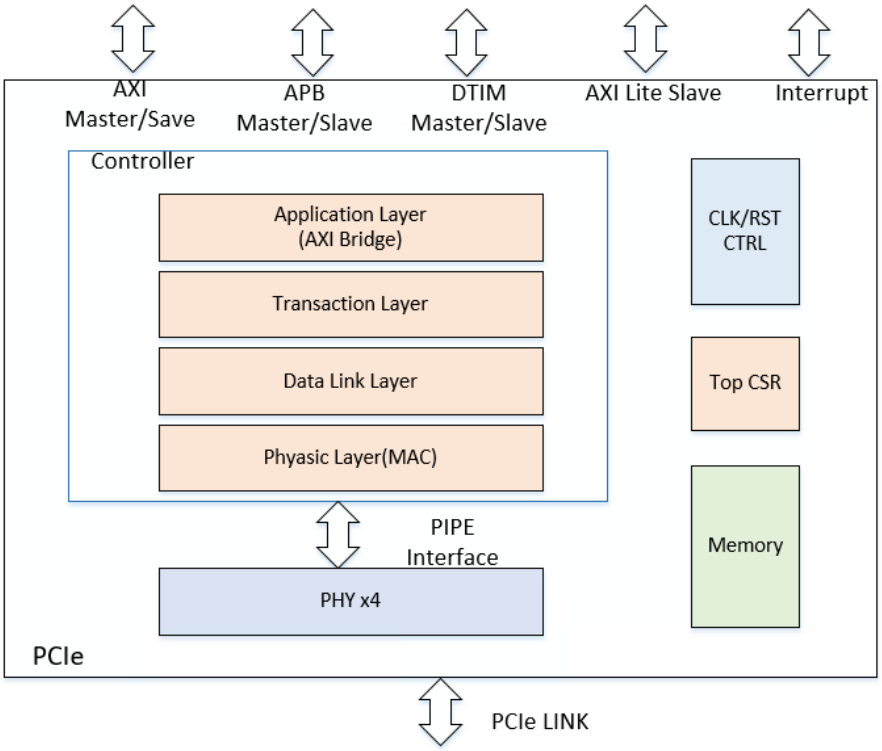


Figure 10-1 PCI-Express subsystem block diagram

10.1.3 Function description

10.1.3.1 Local address map

PCIe local address map as bellow table, there are three regions according to different interface.

Table 10-1 PCIe local address mapping

Interface	Address region	description
APB Slave	0x000_5000_0000 ~ 0x000_5007_ffff	PCIe Top Csr register space
	0x000_5008_0000 ~ 0x000_500f_ffff	PCIe Phy Register Space
AXI-lite Slave	0x000_5400_0000 ~ 0x000_5400_0fff	PCIe 4KB Configure Register Space
	0x000_5410_1000 ~ 0x000_5410_ffff	PCIe Controller Shadow and CS2 Register Space
	0x000_5430_0000 ~ 0x000_5437_ffff	iATU Register Space
	0x000_5438_0000 ~ 0x000_543f_ffff	DMA Register Space
	0x000_54B0_0000 ~ 0x000_54B0_7fff	MSI-X Table Address Space
	0x000_54B0_8000 ~ 0x000_54B0_ffff	MSI-X PBA Address Space
AXI slave	0x000_4000_0000 ~ 0x000_4fff_ffff	Access remote EP Configure and BAR0 Register Space in RC mode
	0x080_0000_0000 ~ 0x09f_ffff_ffff	Outbound Access remote RC Memory in EP mode or Memory Map Access remote EP in RC mode

10.1.3.2 BARs space

As an endpoint, PCIe has five address spaces, such as 4KB configure register space, BAR0, BAR1 BAR2 and BAR4. BAR0 and BAR1 are memory type spaces and non-prefetchable with 32bits address width, BAR2 and BAR4 are also memory type spaces and prefetchable with 64bits address width. BAR0 is responsible for configuring local chip register via remote host, BAR1 corresponds to register address of iATU/DMA/MSI-X table/MSI-X PBA in PCIe subsystem, BAR2 and BAR4 are corresponds to memory map address space for accessing local chip memory as bellow table.

Table 10-2 Remote host access address map

Access Type	Address Region	Access Address
Cfg Request	Configure Space	BDF + Register offset
MRd/MWr	BAR0	BAR0 + Register offset
MRd/MWr	BAR1	BAR1 + DMA Register offset (register of DMA)
		BAR1 + 0x2000 + iATU Register offset (register of iATU)
		BAR1 + 0x8000 + MSI-X Table Register offset(MSI-X Table)
		BAR1 + 0x8400 + MSI-X PBA Register offset(MSI-X PBA)
MRd/MWr	BAR2	BAR2 + Memory Address offset
MRd/MWr	BAR4	BAR4 + Memory Address offset

10.1.3.3 Interrupt

- MSI

As an endpoint, MSI supports 32 vectors, vector[0] is used by PCIe dma interrupt and vector[31:1] is for local mcpu, the 511 interrupts of mcpu are combined into 31 groups, each group corresponds to a vector, vector[1] for mcpu_interrupt[15:0], vector[2] for mcpu_interrupt[31:16]..., and the last vector[31] for mcpu_interrupt[511:480].

As an RC, when the address of received MWr matches MSI_CTRL_ADDR_OFF and MSI_CTRL_UPPER_ADDR_OFF register, a MSI interrupt of USP is detected, RC assert a local interrupt signal (msi_ctrl_int) to mcpu, then the mcpu read MSI_CTRL_INT_0_STATUS_OFF register or MSI_CTRL_INT_0_EN_OFF register (if MSI is masked) and identify the MSI vector.

- MSI-X

RC mode doesn't support MSI-X, when in EP mode, PCIe supports 512 interrupt sources. there are two methods to trigger MSI-X reqeust, one is doorbell through configure the VECTOR and TC fields of MSIX_DOORBELL_OFF register; second is that issues a MWr at AXI slave interface when an address equal to MSIX_ADDRESS_MATCH_LOW_OFF and MSIX_ADDRESS_MATCH_HIGH_OFF, the PCIe controller extracts the vector and TC information from the MWr payload and create the MSI-X request.

- Legacy interrupt

Software configures the SYS_INT field of PCIE_TOP_CSR register to asserts the level-sensitive sys_int and notify the controller that it should send an interrupt message in EP mode, the controller generates two message TLPs, Assert_INTA and Deassert_INTA, in response to the assertion and the de-assertion of this signal. The controller decodes received Assert_INTx/Deassert_INTx messages and pulses the radm_intx_asserted and radm_intx_deasserted output to local mcpu in RC mode, where x=A,B,C,D.

Notes : recommend using MSIX interrupt in EP mode.

10.1.3.4 iATU

The PCIe uses the iATU(internal address translation unit) to implement a local address translation scheme that replaces the TLP address and TLP header fields in the current TLP request header as bellow diagram.

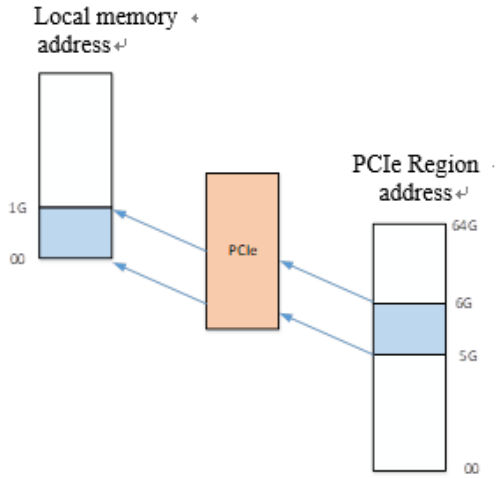


Figure 10-2 One Inbound Region address map

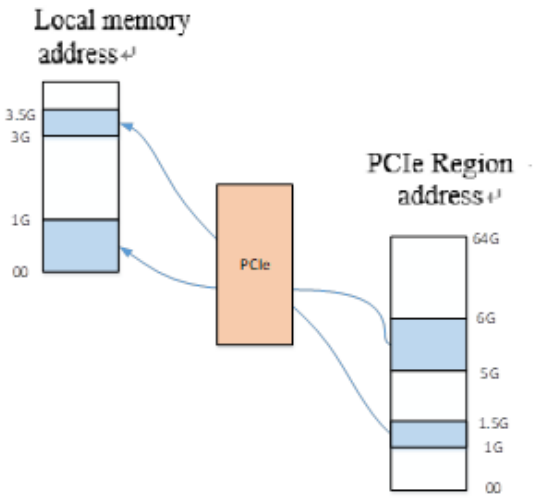


Figure 10-3 Two Inbound Regions address map

- Inbound

There are two inbound regions and iATU processes inbound requests as two translation methods which are address match mode and BAR match mode. TLPs destined for BAR0 or BAR1 in an upstream port are not translated, TLPs that are not error-free(ECRC,malformed,and so on) are not translated

When inbound is configured as address match mode,the field of each request TLP is checked to see if falls into any of the enabled address regions. When an address match if found then the TLP address field is modified as follows:

Address = Address – Base_Address + Target_Address

When the TLP address field matches more than one of inbound address regions,then the first enabled region to be matched is used.

When inbound is configured as BAR match mode,BAR matching mechanism check if the address field of any MEM and I/O request TLP falls into any address region defined by the enbled BAR addresses and masks, when a matched BAR is found,the the iATU compares the BAR ID to the BAR number field in the IATU_REGION_CTRL_2_OFF_INBOUND_i register for all enabled regsions

Inbound programming as following table and provides details of programing examples,you can access the iATU registers through the local cpu or through BAR matched MEM/IO requests.

Table 10-3 iATU Inbound register map

Offset address	Label(i=Region Num)	Description
0x100	IATU_REGION_CTRL_1_OFF_INBOUND_i	Region Control 1
0x104	IATU_REGION_CTRL_2_OFF_INBOUND_i	Region Control 2
0x108	IATU_LWR_BASE_ADDR_OFF_INBOUND_i	Lower Base Address
0x10C	IATU_UPPER_BASE_ADDR_OFF_INBOUND_i	Upper Base Address
0x110	IATU_LIMIT_ADDR_OFF_INBOUND_i	Limit Address
0x114	IATU_LWR_TARGET_ADDR_OFF_INBOUND_i	Lower Target Address
0x118	IATU_UPPER_TARGET_ADDR_OFF_INBOUND_i	Upper Target Address
0x11c	IATU_REGION_CTRL_3_OFF_INBOUND_i	Region Control 3
0x120	IATU_UPPER_LIMIT_ADDR_OFF_INBOUND_i	Upper Limit Address

- Outbound

There are 32 outbound regions,each region size can be configured to 4KB~8GB.Address translation is used for mapping different address ranges to different memory spaces, a typical example maps local memory space to PCI memory space,the iATU also supports type translation.

The iATU supports TYPE translation of MEM and I/O TLPs to Msg/MsgD TLPs.When there is a successful address match on an outbound MEM TLP,and the translated TLP type field is MSG,then the message code field of the TLP is set to the value in the Message Code field of the IATU_REGION_CTRL_2_OFF_OUTBOUND_i register.A MWr with an effective length of 0 is converted to Msg and all other MWr TLPs are converted to MsgD.

The iATU supports translation of I/O and MEM TLPs to CFG TLPs. The 16-bit BDF is located at bit[31:16] of the translated address where:

Translated_Address = Original_Address – Base_Address + Target_Address

As an example

{13'h0，function_num[2:0]}=Original_Address[31:16]

Base_Address[31:16]=16'h0

Target_Address[31:16]={bus_num[7:0],device_num[4:0],3h0}

Then

Translated_Address[31:16]= BDF={ bus_num[7:0],device_num[4:0], bus_num[7:0], device_num[4:0]

}.

For CFG transactions created directly by software,you must ensure that the BDF field does not match

any programmed iATU address region or else unintentional type translation could occur.

Outbound programming as bellow table which provides details of programming examples, you Can access the iATU register through local cpu or through BAR matched MEM/IO requests.

Table 10-4 iATU Outbound register map

Offset address	Label(i=Region Num)	Description
0x000	IATU_REGION_CTRL_1_OFF_OUTBOUND_i	Region Control 1
0x004	IATU_REGION_CTRL_2_OFF_OUTBOUND_i	Region Control 2
0x008	IATU_LWR_BASE_ADDR_OFF_OUTBOUND_i	Lower Base Address
0x00C	IATU_UPPER_BASE_ADDR_OFF_OUTBOUND_i	Upper Base Address
0x010	IATU_LIMIT_ADDR_OFF_OUTBOUND_i	Limit Address
0x014	IATU_LWR_TARGET_ADDR_OFF_OUTBOUND_i	Lower Target Address
0x018	IATU_UPPER_TARGET_ADDR_OFF_OUTBOUND_i	Upper Target Address
0x01c	IATU_REGION_CTRL_3_OFF_OUTBOUND_i	Region Control 3
0x020	IATU_UPPER_LIMIT_ADDR_OFF_OUTBOUND_i	Upper Limit Address

10.1.3.5 DMA

You can configure the DMA(HDMA) to have one to read channels and one to write channels,it can simultaneously perform the following types of memory translations.DMA write,transfer of a block of data from local memory to remote memory;DMA read,transfer of a block of data from remote memory to local memory.Therefore,the DMA supports full duplex operation,processing read and write transfers at the same time, and in parallel with normal(non-DMA)tranffic.upon completion of a DMA transfer or an error,the DMA optionally interrupts the local CPU or sends an interrupt IMWr to the remote CPU.the DMA is highly configurable,and you can program it using the local CPU or over the PCIe wire through BAR1 adress space.

The DMA can operate in non-linked list mode or linked list mode.In linked list mode,the DMA fetches the transfer control information for each transfer from a list of DMA elements that you have constructed in local memory.

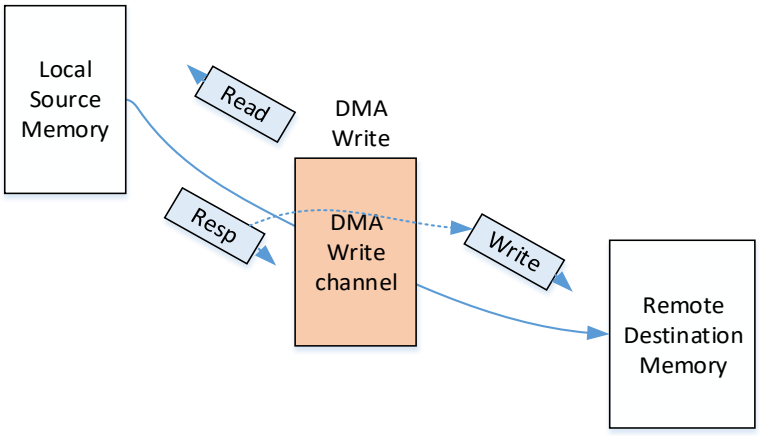


Figure 10-4 Data transfer of DMA write channel

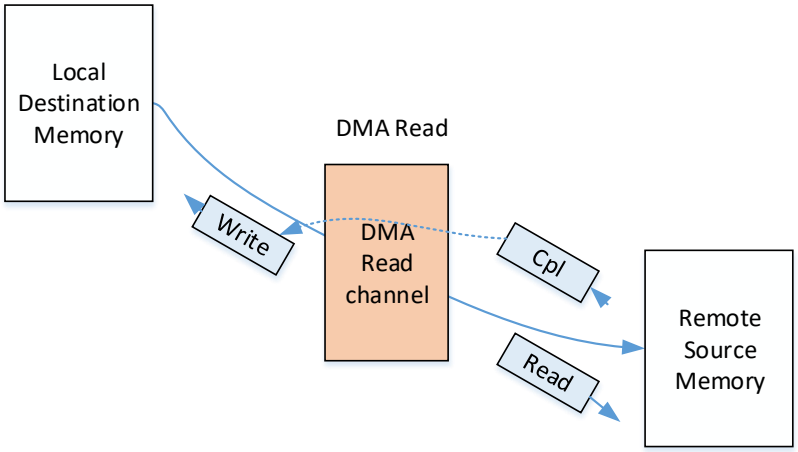


Figure 10-5 Data transfer of DMA read channel

The DMA generates the following interrupts per channel:

- Done: the DMA successfully completes the transfer
- Watermark: Interrupt generated at the end of each watermark event(end of linked list element). Generated in linked list mode only.
- Abort: The DMA fails to complete the transfer,or an error occuers during the transfer.

The interrupts are signaled to the software on local CPU,using one or both of the following mechanisms:

- Locally through the edma_int bus
- Remeotely using a posted memory write(IMWr),which can be interpreted as an MSI or MSI-X when directed toward the remote RC.
- There are three programmable IMWr address per channel,one each for the done, watermark,and abort interrupts.For MSI,you must program all IMWr address registers with the same MSI address,as PCIe only supports a signle MSI address per function.
- A single IMWr data register is used for all the interrupts, so you must read HDMA_INT_ STATUS_OFF_[WR|RD]_i to identify the interrupt type.

Following is the programing example for enable third channel of read ,and enable remote interrupt.

Table 10-5 Third channel of read register map

Local access address	Remote access address	description	Value
0x54380000+0x2C	BAR1+0x2C	DMA read engine enable	0x1
0x54380000+0xCC/D0	BAR1+0xCC/D0	DMA read done IMWR addr	
0x54380000+0xD4/D8	BAR1+0xD4/D8	DMA read abort IMWR addr	
0x54380000+0x0E4	BAR1+0x0E4	DMA read channel 3 IMWr Data	
0x54380000+0x800	BAR1+0x800	DMA channel control1 register Local interrupt Enable (LIE)=0 Remote interrupt Enable (RIE)=1	0x4000008
0x54380000+0x808	BAR1+0x808	DMA transfer size	
0x54380000+0x80C	BAR1+0x80C	DMA SAR low	
0x54380000+0x810	BAR1+0x810	DMA SAR high	
0x54380000+0x814	BAR1+0x814	DMA DAR low	
0x54380000+0x818	BAR1+0x818	DMA DAR high	

Local access address	Remote access address	description	Value
0x54380000+0x010	BAR1+0x010	DMA write doorbell	0x3

The DMA provides a linked list (LL) mode to efficiently move data from source to destination with minimal intervention from the local CPU. This mode provides an alternative to programming the DMA multiple times to transfer multiple blocks of data. The programming information (address, size, and so on) for each block of memory is pre-programmed by your software into a LL element (also known as a descriptor) in local memory. Each element (called a data element) in the LL structure (called a transfer list) can transfer up to 4 GB of data.

You enable LL operation for a channel, by setting the LLEN field of the HDMA_CONTROL1_OFF_[WR|RD]CH_i register to 1. You can enable LL mode independently for each channel. When you enable LL for more than one channel, then you must have a separate LL structure in local memory for each channel. Your must produce the LL element structure in local memory as shown in Figure xx. Normally, all of the elements are contiguous (one after the other) in memory, and each element has six DWORDs containing the information about the block of data to be transferred, RIE is remote interrupt enable,LIE is local interrupt enable,LLP is next linked list pointer,TCB indicate whether repeat to operate this linked list and CB indicate this element is the last one.You program the channel context registers (HDMA_LLP_LOW_OFF_[WR|RD]CH_i and HDMA_LLP_HIGH_OFF_[WR|RD]CH_i) with the location of where you have placed the LL element structure in local memory. following table is the example programming for LL mode.

Table 10-6 Register configuration of Start LL mode

Local access address	Host access address	description	Value
0x54380000+0x30	BAR1+0x30	DMA write enable	0x1
0x54380000+0x200	BAR1+0x200	DMA channel control1 register Linked list enable (LLE) =1	0x4000300
0x54380000+0x21C	BAR1+0x21C	DMA Linked list pointer low	
0x54380000+0x220	BAR1+0x220	DMA Linked list pointer high	
0x54380000+0x10	BAR1+0x10	DMA Write doorbell	0x0

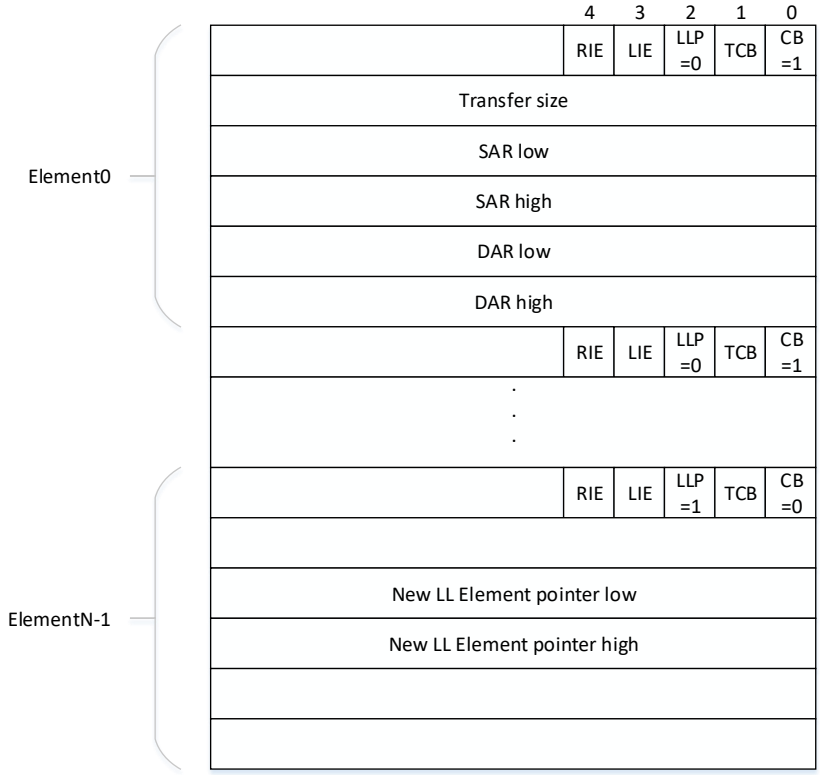


Figure 10-6 Element structure of DMA Linked list

10.1.3.6 Low power

PCIe supports two categories of PM operations to control the device state(D-state) and link state as following table, which are PMSCR and ASPM.

Table 10-7 Low power state

D-State	Link State	State Trigger	PHY State	Active Clock supplies	Pipe_clk Core_clk
D0unitialized	L0	ASPM	P0, P0s	Refclk/Auxclk	On
D0active	L0			Refclk/Auxclk	On
	L0s			Refclk/Auxclk	On
	L1/L1.sub		P1/P1.1/P1.2	Auxclk	Off
D1,D2,D3hot	L1	PCI-PM	P1	Refclk/Auxclk	On
D3cold	L2		P2	Auxclk	Off
	L3		P1.2PG	None	Off

- L0s is a low power state enabled by ASPM：
- configure LINK_CONTROL_STATUS_REG register bit[1:0] to2'b01，enable L0s.
- L0s entry after PCIe being in an idle state for a L0s_ENTRANCE_LATENCY of time if no higher stage of power-down requested.
- L0s exit when any of these condition can be met, such as A DLLP and TLP is pending to be sent or PCIe link partner requests to enter into link recovery.
- L1 is enable by ASPM：

- confgure LINK_CONTROL_STATUS_REG register bit[1:0] to 2'b10，enable L1.
- L1s entry after PCIe being in an idle state for a L0s_ENTRANCE_LATENCY of time if no higher stage of power-down requested.
- L0s exit when any of these condition can be met, such as A DLLP and TLP is pending to be sent or PCIe link partner requests to enter into link recovery and neet to negotiate with partner.
- L1 is enable by ASPM when idle timeout in L0：
- configure LINK_CONTROL_STATUS_REG register bit[1:0] to 2b11，enable L0s and L1.
- L0s exit when any of these condition can be met, such as A DLLP and TLP is pending to be sent or PCIe link partner requests to enter into link recovery and neet to negotiate with partner.
- L1.sub enable and exit by ASPM
- configure LINK_CONTROL_STATUS_REG register to bit[1:0] 为 2'b1x，enable L1.
- configure L1SUB_CONTROL_REG register to enable L1.1 or L1.2.
- enter L1.sub after L1 state，PCIe controls CLKREQ# signal to turn off refclk and then L1.sub.
- L1.sub exit conidition is the same as L1, PCIe controls CLKREQ# signal to turn on refclk and negotiate to exit L1 with partner.
- Enter and exit L1 by PCI-PM
- remote host configure PMCSR of USP enable D1、D2 or D3hot.
- USP initiate a link state transition to L1, Both Link partners must negotiate to go this state, the L1-PM negotiation handshake uses PM_ENTER_L1 and PM_Request_Ack DLLPs.
- A DLLP and TLP is pending to be sent or PCIe link partner requests or configure PMCSR to exit L1

10.1.3.7 ATS

PCIe supports ATS function in RC mode(does not support ATS in EP mode), as following diagram, circumscribed SMMU implements address translation，in response to the translation request and ATS invalidation.

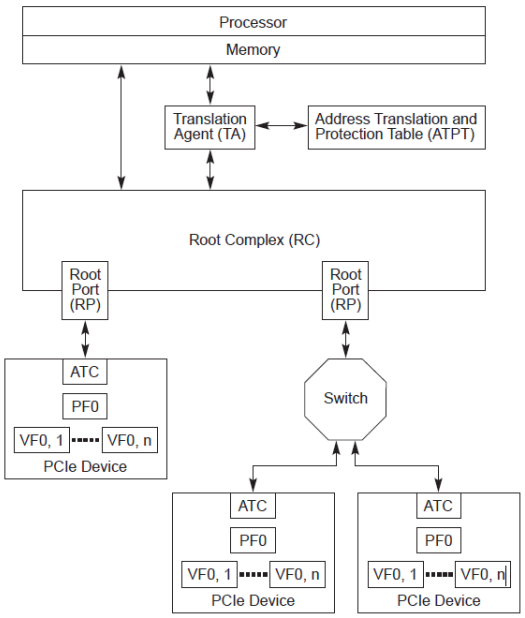


Figure 10-7 A platform of ATS

The AT field of MWr/MRd TLPs from USP illustrates the information of ATS. If AT=2'b10, the address does't need to be translated and pass through SMMU; if AT=2'b00, the address should be translated by SMMU;if AT=2'b01,a translation request is detected,the PCIe will translate this TLP to DTIM interface and inform SMMU to respond to this request. If the address page table of system has changed, SMMU will send invalidation request to USP through DTIM interface.

10.1.3.8 Reset

PCIe of EIC7700X supports four reset mechanism,such as Power Up Reset、Perst#、Hot reset and FLR(function reset). These resets are come from POR of EIC7700X,PCIe slot,Link reset request of DSP and FLR register configuration. Hot reset can be triggerd by Secondary_Bus_Bit register configuration of DSP, then the DSP sends some TS1 TLPs to inform USP to Hot reset , after hot reset the PCIe link will repeat to training.DSP also can configure DEVICE_CONTROL_DEVICE_STATUS register bit[15] to trigger a FLR. If PCIe of EIC7700X complete a hot reset or a FLR,it will generate a interrupt to local cpu.

10.1.4 Initialization sequence

- The initialization sequence in RC mode as following dragram

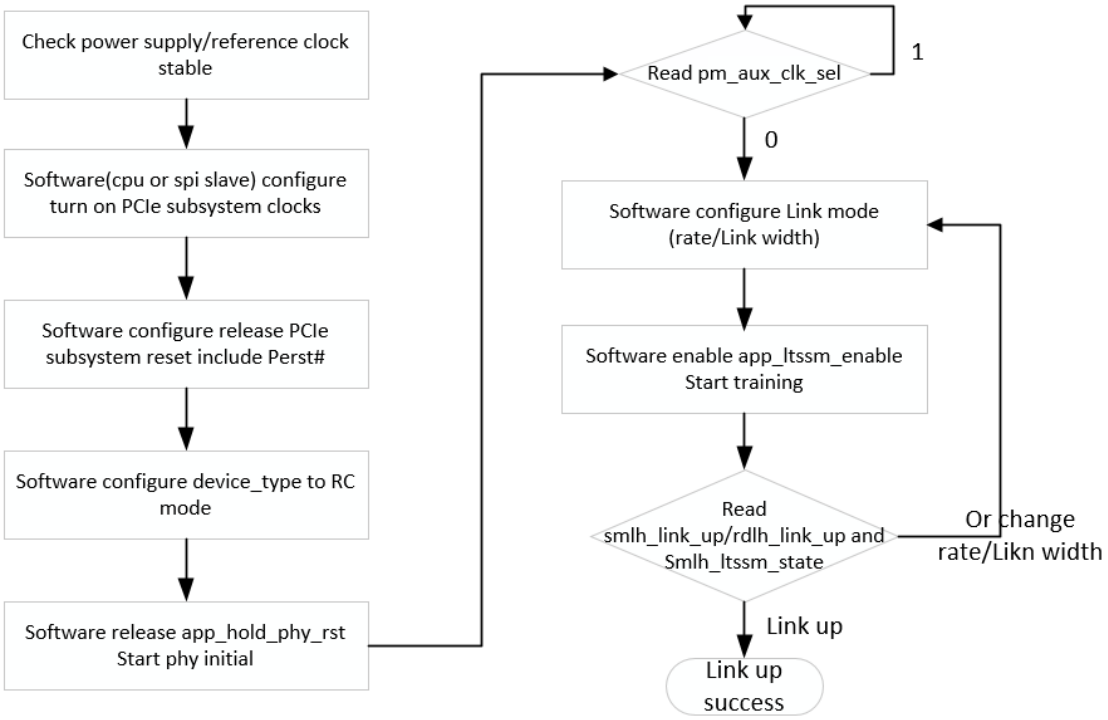


Figure 10-8 Initialization sequence in RC mode

- Above initialization sequence in RC mode
- The initialization sequence in EP mode as following dragram

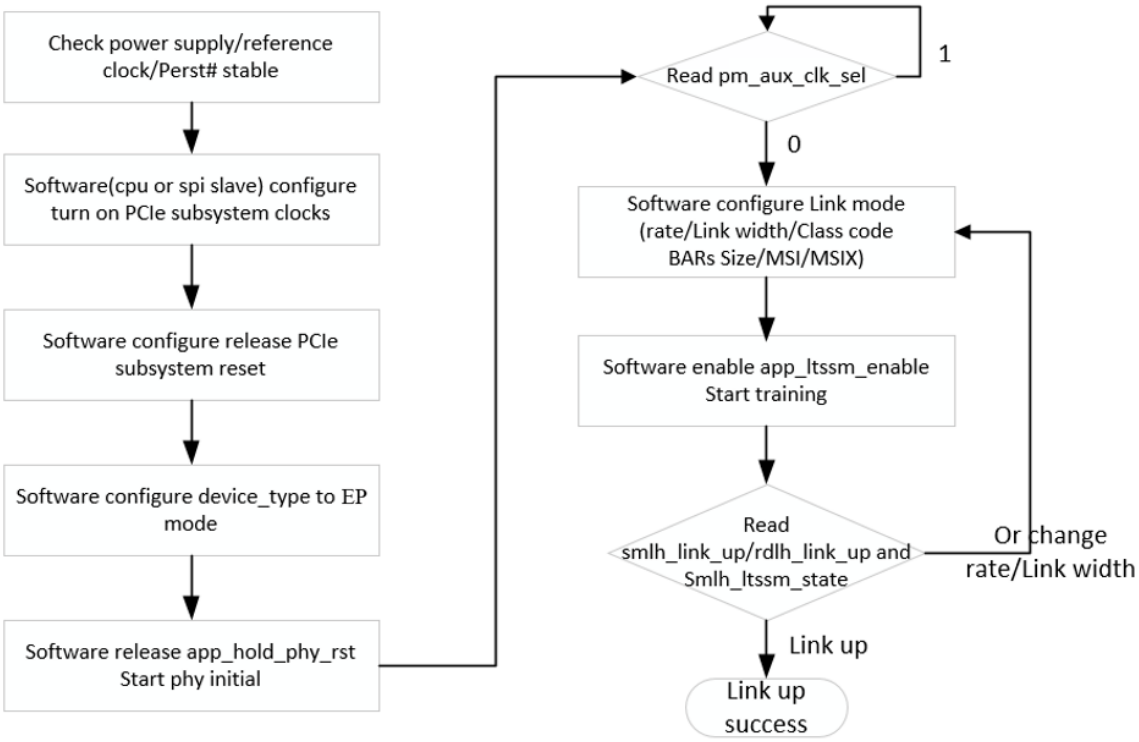


Figure 10-9 Initialization sequence in EP mode

10.2 GMAC

10.2.1 Overview

EIC7700X has two lanes of Ethernet module, which interact with the internet through high-speed IO and can directly connect to a switch or router. and has following key features:

- The two Ethernet modules can share the task of data transmission and implement Ethernet bonding in the protocol stack to achieve load balancing while jointly bearing the data transmission load.
- Implementing Gigabit Ethernet and downward compatibility with 100M and 10M Ethernet.
- Supports full- and half-duplex modes, and flow control under corresponding conditions.
- Support RGMII, RMII interface
- Support IP v4 and IP v6
- Support SA insertion and replacement, and VLAN insertion, replacement and delete
- Supports controlling and managing external PHY chips through MDIO
- Supports up to 8 MAC filtering addresses, supports DA hash filtering, supports VLAN hash filtering, and supports layer 3/4 packet filtering.
- Supports TX/RX queues with a maximum length of 4.
- Supports TCP checksum calculation and insertion.

10.2.2 Block Diagram

The block diagram of a single Ethernet module group is shown above, with each group connected to external PHY interfaces. Inside the module, there are TX and RX sides. The MAC handles Ethernet protocol-related tasks, while the MTL handles data buffering and distribution queue work at the transaction layer. The DMA is responsible for data exchange with the system bus, and CSRs at different levels are responsible for configuring registers in the corresponding modules.

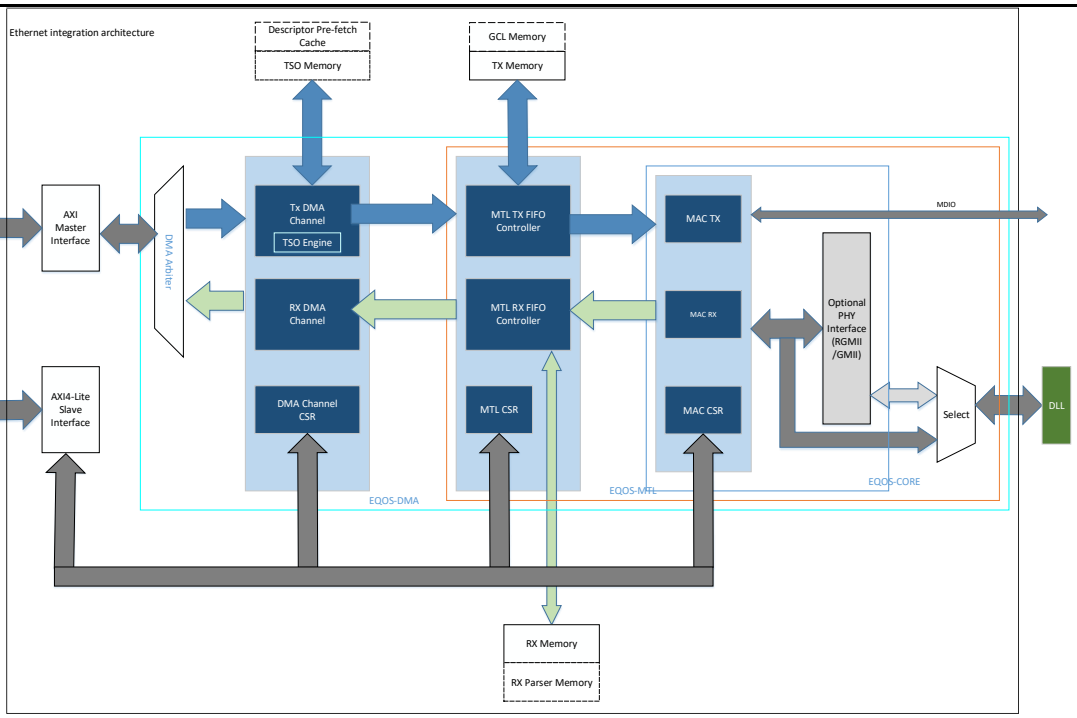


Figure 10-10 GMAC subsystem block diagram

10.2.3 Initialization Sequence

10.2.3.1 DMA Initialization

1. Write register DMA_Mode SWR，reset DMA
2. Poll DMA_Mode SWR=0，reset done
3. Write register DMA_SysBus_Mode
4. Create TX or RX descriptor
5. Configure Tx or Rx RING descriptor length, write register DMA_CH(#i)_TxDesc_Ring_Length
6. Initialize Tx or Rx descriptor address, write register DMA_CH(#i)_TxDesc_List_Address
7. Configure packet length, write register DMA_CH(#i)_Control
8. Enable interrupt, write register DMA_CH(#i)_Interrupt_Enable
9. Start Tx or Rx process, write register DMA_CH(#i)_TX_Control，DMA_CH(#i)_RX_Control

10.2.3.2 MTL Initialization

1. Set Tx or Rx arbitration strategy, write MTL_Operation_Mode
2. Configure mapping from queue to channel, write register MTL_RxQ_DMA_Map0 and MTL_RxQ_DMA_Map1
3. Initialize Tx buffering setting, write register MTL_TxQ0_Operation_Mode
4. Initialize Rx buffering setting, write register MTL_RxQ0_Operation_Mode

10.2.3.3 MAC Initialization

1. Configure mac address, write register MAC_Address0_Low and MAC_Address0_High

2. Configure MAC filtering, write register MAC_Packet_Filter
3. Configure MAC flow control, write reigster MAC_Q(#i)_Tx_Flow_Ctrl
4. Enable interrupt, write register MAC_Interrupt_Enable
5. write register MAC_Configuration.TE, MAC_Configuration.RE, start MAC transmit or receive process

10.2.4 Timing Diagrams

RGMII Interface Timing

- TskewT represents the timing offset between the clock edge and data edge on the transmitting side, with an effective range of 0-0.5ns.
- TskewR represents the timing offset between the clock edge and data edge on the receiving side, with an effective range of 1-2.6ns

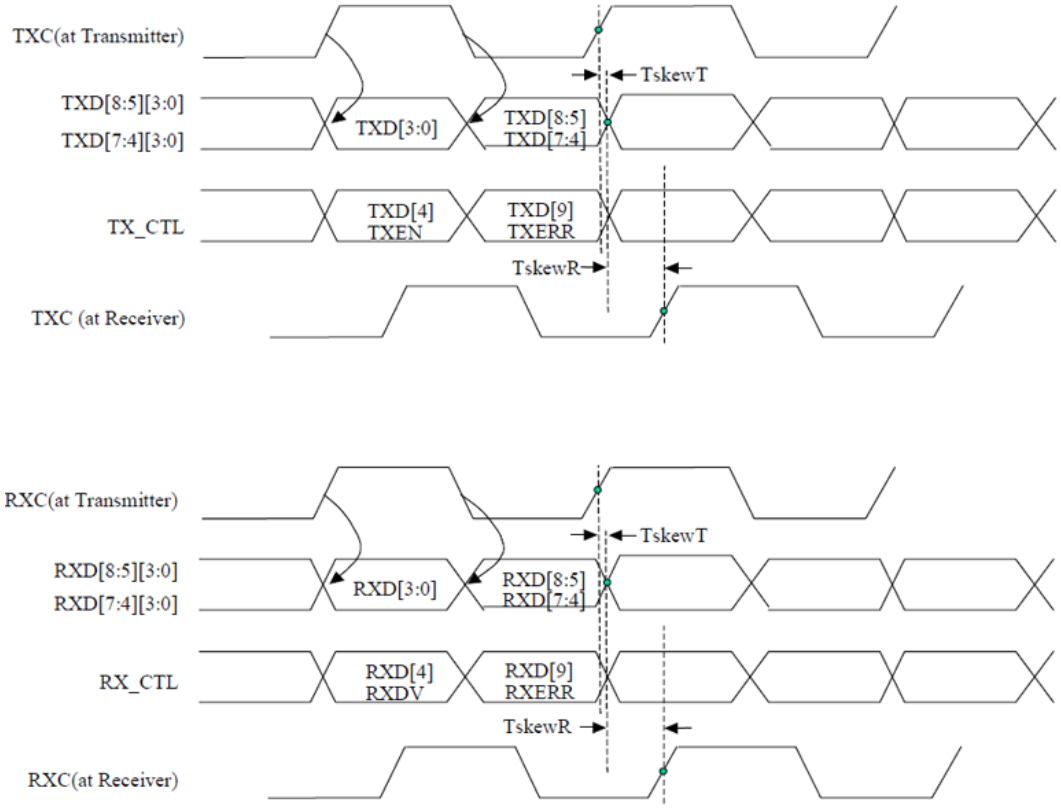


Figure 10-11 RGMII Timing diagram

RMII Interface Timing

- The shadow area in the figure below represents the setup/hold slew

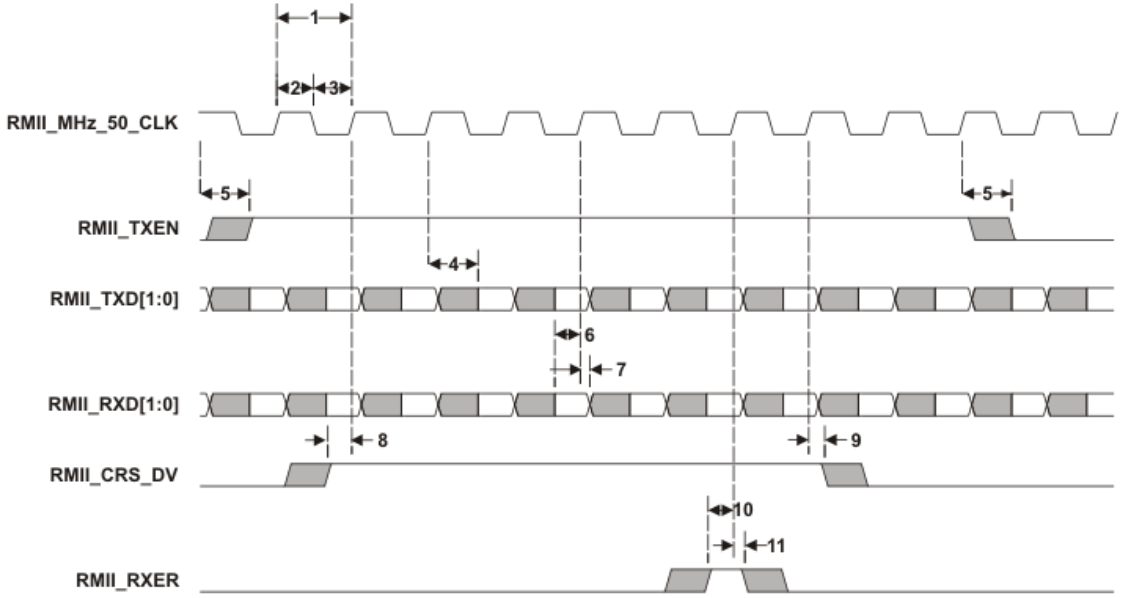


Figure 10-12 RMII Timing diagram

10.3 USB

10.3.1 overview

The chip contains two identical usb,supports Super-speed(5Gb/s)、High-speed(480Mb/s)、Full-speed(12Mb/s) and Low-speed(1.5Mb/s) data transfer.USB controller supports standard USB2.0/3.0 protocol and DRD mode(Dual Role Device), can be configured as Host or Devcie during initialization. The main features are as followed.

10.3.1.1 General Feature

- Universal Serial Bus 3.0 Specification, Revision 1.0
- Universal Serial Bus Specification, Revision 2.0
- LPM protocol in USB2.0 and U0/U1/U2/U3 states for USB3.0
- Dynamic FIFO memory allocation fore endpoints
- Keep-Alive feature in LS mode and (micro-)SOFs in HS/FS modes
- Software controlled standard USB commands
- Interrupt moderation

10.3.1.2 USB3.0 Device Feature

- Up to 16 single directional endpoints;max of 8 active in endpoints
- Flexible endpoint configuration
- Simultaneous IN and OUT transfer support for multiple applications
- Stream-based bulk endpoints with controller automatically initiating data movement
- Isochronous endpoints with isochronous data in data buffers or external hardware FIFOs

10.3.1.3 USB Class-Specific Device Feature

- Stream support for UASP application
- Gathering of scattered packet to support Ethernet over USB
- Scheduling od multiple Ethernet packets without interrupt
- Variable FIFO buffer allocation
- For isochronous applications, scheduling of variable-length payloads for each microframe
- Microframe precise scheduling for isochronous applications
- Configurable endpoint type selection and dynamic FIFO allocation to facilitate multi-function/composite device implementation

10.3.1.4 xHCI Host Feature

- Support 64 devices
- Support 1 interrupter
- Support 1 USB2.0 port and 1 SuperSpeed port
- xHCI1.1 compatible
- Standard or open-source xHCI and class drivers
- Concurrent IN and OUT transfers to get the full 8 Gbps duplex throughput

10.3.2 Block Diagram

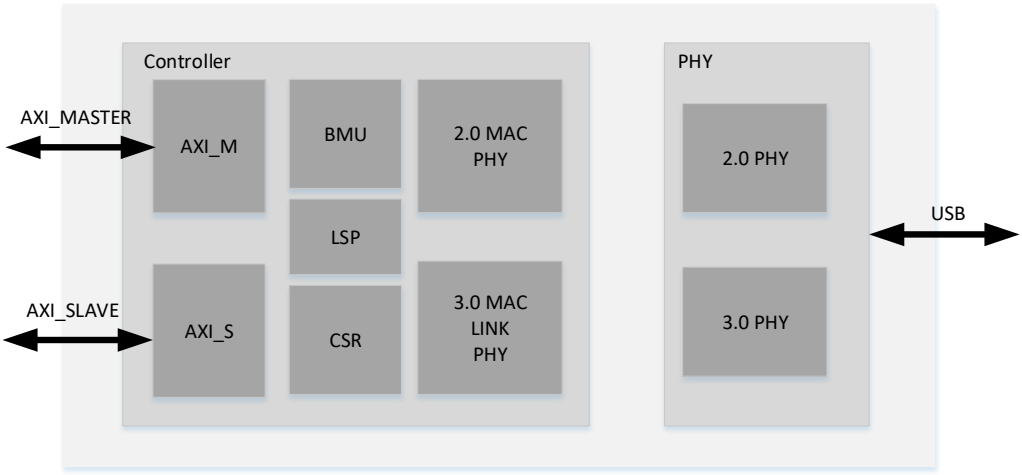


Figure 10-13 USB Diagram

The USB controller interacts with the system through the AXI bus, and mainly contains the following modules:

- BMU handles the buffering requirements between the system bus and USB bus
- LSP manages interaction between the USB bus, the system bus, and the driver
- MAC/LINK/PHY implements the standard USB protocol

10.3.3 Initialization

- Clock gating

  USB0 bus clock can be disabled by setting USB0_CTRL [28] to 0. USB1 corresponding register is USB1_CTRL.
- Reset control

After power up, controller and phy is in reset mode by default, set USB0_CTRL [24] to 1 to release controller reset, set USB0_CTRL [25] to 0 to release USB0 PHY reset.USB1 corresponding register is USB1_CTRL.

- Device mode initialization related registers

Register	Description
DCTL	Set the CSftRst field to '1' and wait for a read to return '0'. This resets the device controller.
GSBUSCFG0/1	Leave the default values if the correct power-on values were selected during configuration.
GTXTHRCFG/ GRXTHRCFG	This is required only if you are planning to enable thresholding. Leave the default values if the correct power-on values were selected during configuration.
GUID	Optionally, the software can program the User ID GUID register.
GUSB2PHYCFG	Program the following PHY configuration fields: USBTrdTim, FSIntf, PHYIf, TOUTCal, or leave the default values if the correct power-on values were selected during configuration. Note: The PHY must not be enabled for auto-resume in device mode. Therefore, the field GUSB2PHYCFG[15] (ULPIAutoRes) must be written with '0' during the power-on initialization

Register	Description
	in case the reset value is '1'.
GUSB3PIPECTL	Program the following PHY configuration fields: DatWidth, PrtOpDir, or leave the default values if the correct power-on values were selected during configuration.
GTXFIFOSIZn	Write these registers to allocate prefetch buffers for each Tx endpoint. Unless the packet sizes of the endpoints are application-specific, it is recommended to use the default value.
GRXFIFOSIZ0	Write this register to allocate the receive buffer for all endpoints. Unless the packet sizes of the endpoints are application-specific, it is recommended to use the default value.
GEVNTADRn/ GEVNTSIZn/ GEVNTCOUNTn	Depending on the number of interrupts allocated, program the Event Buffer Address and Size registers to point to the Event Buffer locations in system memory, the sizes of the buffers, and unmask the interrupt. Note:USB operation stops if the Event Buffer memory is insufficient, because the controller stops receiving/transmitting packets.
GCTL	Program this register to override scaledown, RAM clock select, and clock gating parameters
DCFG	Program device speed and periodic frame interval
DEVTEN	At a minimum, enable USB Reset, Connection Done, and USB/Link State Change events.
DEPCMD0	Issue a DEPSTARTCFG command with DEPCMD0.XferRscIdx set to 0 and CmdIOC set to 0 to initialize the transfer resource allocation. Poll CmdAct for completion.
DEPCMD0/ DEPCMD1	Issue a DEPCFG command for physical endpoints 0 & 1 with the following characteristics, and poll CmdAct for completions: ■ USB Endpoint Number = 0 or 1 (for physical endpoint 0 or 1) ■ FIFONum= 0 ■ XferNRdyEn and XferCmplEn = 1 ■ Maximum Packet Size = 512 ■ Burst Size = 0 ■ EPType = 2'b00 (Control) Note: The command has to be issued for EP0 first, followed by EP1.
DEPCMD0/ DEPCMD1	Issue a DEPXFERCFG command for physical endpoints 0 & 1 with DEPCMDPAR0_0/1 set to 1, and poll CmdAct for completions Note:The command has to be issued for EP0 first, followed by EP1.
DEPCMD0	Prepare a buffer for a setup packet, initialize a setup TRB, and issue a DEPSTRTXFER command for physical endpoint 0, pointing to the setup TRB. Poll CmdAct for completion. Note: The controller attempts to fetch the setup TRB via the master interface after this command completes.
DALEPENA	Enable physical endpoints 0 & 1 by writing 0x3 to this register.
DCTL	Set DCTL.RunStop to '1' to allow the device to attach to the host. At this point, the device is ready to receive SOF packets, respond to control transfers on control endpoint 0, and generate events.

- To initialize the controller as host, the application must perform the steps described in the xHCI specification. If intermediate firmware needs to modify default settings, it can write to the following specific registers at power-on.

Register	Description
GSBUSCFG0/1	Leave the default values if the correct power-on values were selected during configuration.
GTXTHRCFG/ GRXTHRCFG	This is required only if you are planning to enable thresholding. Leave the default values if the correct power-on values were selected during configuration.

Register	Description
GUID	Optionally, the software can program the User ID GUID register.
GUSB2PHYCFG	Program the following PHY configuration fields: USBTrdTim, FSIntf, PHYIf, TOUTCal, or leave the default values if the correct power-on values were selected during configuration.
GUSB3PIPECTL	Program the following PHY configuration fields: DatWidth, PrtOpDir, or leave the default values if the correct power-on values were selected during configuration.
GTXFIFOSIZn/ GRXFIFOSIZn	Program these registers based on the speed used for the FIFO. Unless the packet sizes of the endpoints are application-specific, it is recommended that you use the default value.
GCTL	Program this register to override scaledown, RAM clock select, and clock gating parameters. It is recommended that you set this to 32'h2001004.
GUCTL	If you want to improve the interoperability with different devices, program this register to override the behavior of the controller in Host mode.
DCTL	If you want to improve interoperability as a Debug Capability Target, program the InitU1/U2Ena and AcceptU1/U2Ena fields to '0' to disable U1/U2 entry.

11 system security

11.1 Overview

EIC7700X supports hardware-based secure/non-secure access and supports the following features:

- Secure boot from Secure CPU(SCPU);
- Built-in OTP(128Kb) used to configure the working mode of the chip, which contain security Keys for security boot;
- hardware crypto engine for symmetric encryption and decryption, asymmetric encryption and decryption, digital signature;
- Support DDR memory access protection (MEM-PMP), SCPU handle the master's access rights in the system. The security regions supports up to 16 regions, and each region can be divided into 8 sub-regions.
- If the master CPU(P550) want to access the security subsystem (encryption and decryption module and SCPU TIM), it should be authorized by SCPU;
- Configure OTP can Enable/disable security mode ;
- Configure OTP can Disable JTAG of SCPU;

11.2 secure boot

The security subsystem is implemented by a 32-bit security CPU (call it SCPU). For production applications with security requirements, system boot must from the internal ROM by SCPU. Production applications without security requirements can boot from ROM or nor flash either use SCPU or P550. Whether to use secure boot can be configured through the security bit of OTP.

secure boot mode, only SCPU can access ROM, OTP, sec csr, otp csr, pmp csr and crypto engine directly. The above slaves are referred to as sec island. When other masters want to access the above slaves, All of them need to be authenticated by the security server running on the SCPU.

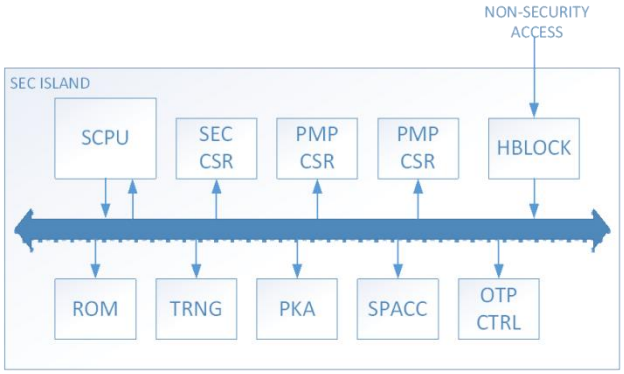


Figure 11-1 Secure block digram

11.3 MEM-PMP

EIC7700X has built-in MEM-PMP module to protect the DDR memory space. By configuring pmp csr, the ddr memory space can be partitioned into security or non-security regions. According to the attributes of regions, pmp will do corresponding processing (block or bypass) for accessing different regions. It is used to protect sensitive data in memory from illegal copying.

11.3.1 PMP feature

- Up to 16 regions are supported, with region0 covering the entire memory space by default. base_addr,size, and subregion_disable are not configurable.
- The priority of the regions is fixed, in descending order according to the region number. If there are overlapping regions, the configuration with the highest priority will prevail.

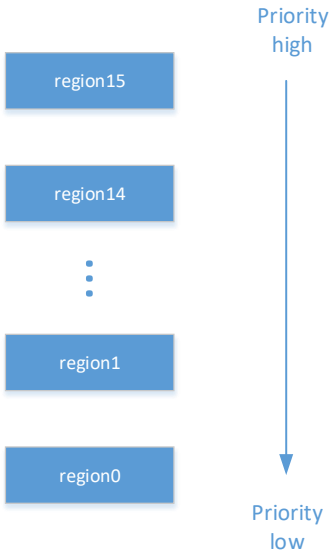


Figure 11-2 PMP region priority

- Each region is equally divided into 8 subregions, and each subregion can be individually configured to disable. The region permission rule of multiple regions overlap is shown in Figure 6.
- Security mode: security mode include two modes: standard and exclusion. In Standard mode, the security master can access to non-security regions. In Exclusion mode, the security master can only access the security region and cannot access the non security region. In any mode, the non-security master can only access non-security regions. Each region supports a separate security region for read and write operations.
- Minimum address range per region is 32KB.

- support speculative accesses.

In the Speculative mode, the PMP directly pass-through the transmission of the addr channel without waiting for the completion of the permission check. When the data arrives from the data path, the corresponding processing is done according to the check result.

The handling of illegal access is as follows:

- Illegal Read: When permission check is an illegal read, return 0 to the upstream master, receive and discard the data returned by the downstream slave.
- Illegal write: When the permission check is an illegal write, the write data issued by the upstream master is discarded and the write resp is returned. At the same time write all zeros to downstream slaves and set strb to all zeros.

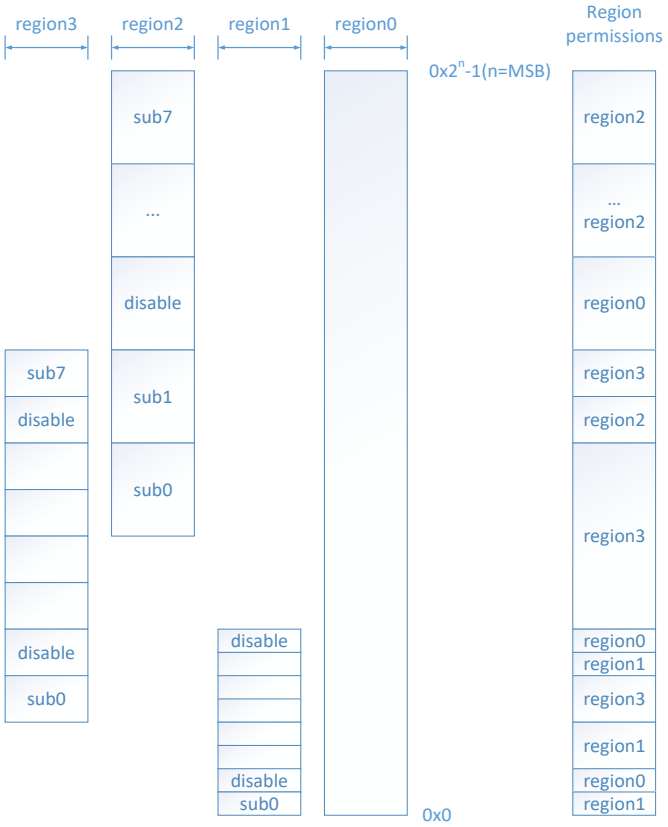


Figure 11-3 Illegal write regions

- PMP supports the secure_lock input, which locks the registers of PMP to prevent misoperation.
- PMP supports report interrupt of illegal access.
- Non-Security cpu allows access to MEMPMP registers in non-secure mode, in non-secure mode, MEMPMP does not check the NS and S bits of the master, and all accesses are filtered by rules.

11.3.2 secure region

The ddr memory space can set to secure regions by configuring pmp csr, and each master can be configured as secure master or non-secure master by configuring sec csr. mem-pmp filters illegal accesses according to the rules configured by the pmp csr.

The master has security bit list as follows, and the register description refers to the Register section

- spacc
- lpcpu
- scpu
- dma1
- dma0

- dsp
- npu
- u84
- gpu
- g2d
- vd/ve/jd/je
- isp/dw200
- eth0/eth1
- sata
- emmc/sd/sdio
- dc8000/hdmi
- pcie
- usb0/usb1

11.4 crypto engine

EIC7700X has three types of hardware-accelerated encryption and decryption engines , which can accelerate encryption and decryption algorithms by hardware. The supported features are as follows.

1. PKA(Public Key Accelerator)

- RSA supported。
- Base ECC supported。
- ECC extension support:ECC-shamir,ECC-512,ECC-512 shamir,ECC-c25519/Ed25519,ECC-Ed25519 shamir
- max RSA operand size:4096 bit。
- max ECC operand size:1024 bit。
- ALU width:128 bit。
- side channel logic:Timing Attack(TA) harden and Differential Power Attack(DPA) harden not supported.

2. SPACC(Security Protocol Accelerator)

- algorithm supported:AES,SM4,DES,HMAC(SHA1,SHA2,SM3),CRC32,Dual core cipher
- algorithm not supported:ChaCHa20,Poly1305,KASUMI,SNOW 3g,ZUC,SHA-3,RC4,MULTI2,Michael-MIC
- max message length:1MB
- max offset:64KB
- contexts number:16
- AES datapath:128bit
- SM4 round/cycle:2
- AES key size:128,192,256
- AES/SM4 block mode:ECB,CBC,OFB,CFB,CTR,XCBC,CCM,GCM,CMAC
- DES datapath:raw DES 16bit/cycle throughput.
- HMAC supported:MD5,SHA1,SHA224,SHA256,SM3,SHA512/224,SHA512/256,SHA384
- HMAC max length:1024

3. TRNG(True Random Number Generator)

- core type:NIST/AIS
- VTRNG number:1
- background noise collection:not support

11.5 otp access

11.5.1 OTP memory organization

Byte Address	Bytes (valid bits)	Field name	Functions	Endianess	U84 accessible	4bit XOR
[0:255]	256(2048)	RSA public key	Secret key for signature validity checking	Little-endian	N	N
[256:511]	256(2048)	Reserved for RSA4096	Reserved for RSA4096, Not used	Little endian	N	N
[512:515]	4(32)	exponent	17 or 65537	Little-endian	N	N
[516:527]	12(96)	Reserved	Reserved	Little-endian	N	N
[528:543]	16(128)	Memory to memory secret key	Commonly used by AES128, SM4, TDES	Little-endian	N	N
[544:559]	16(128)	Reserved for AES256	Reserved for AES256	Little-endian	N	N
[560:575]	16(128)	Boot image decryption key	Commonly used by AES128, SM4 and TDES	Little-endian	N	N
[576:591]	16(128)	Reserved for AES256	Reserved for AES256	Little-endian	N	N
[592:607]	16(128)	Whole memory encryption key (Unused & reserved)	Unused	Little endian	N	N
[608:617]	10(80)	Reserved	Reserved	Little-endian	N	N
[618:873]	256(2048)	Secret Key Array (0~15) (Commonly used for AES/SM4/TDES)		Little-endian	N	N
[874:905]	32(256)	ECDSA	This MUST be the x-	Little-	N	N

Byte Address	Bytes (valid bits)	Field name	Functions	Endianess	U84 accessible	4bit XOR
		256bit key X	coordinate of the ECDSA public key. It MUST be encoded in little-endian format.	endian		
[906:937]	32(256)	ECDSA 256bit key Y	This MUST be the y-coordinate of the ECDSA public key. It MUST be encoded in Little-endian format.	Little-endian	N	N
[938:969]	32(256)	ECDSA 256bit key Z	This MUST be the ECDSA private key. It MUST be encoded in Little-endian format.	Little-endian	N	N
[970:1001]	32(256)	Boot image decryption IV	commonly used by AES128,SM4 and TDES	Little-endian	N	N
[1002:1831]	830(6640)	Reserved	Reserved	Big-endian	N	N
[1832:1847]	16(128)	Password for debugging	Password for enabling of debugging facilities (Unused & Reserved)	Little-endian	N	N
[1848:1975]	128(1024)	Reserved for other keys			Y	N
[1976:2047]	72(576)	Reserved	Reserved for user security data			
[2048:2048]	1(8)		Bit 0: Security Enabled 0H: Security is permanently disabled 1H: Security is permanently enabled [7:4] reserved.1 byte occupied, and only low 4bit used, not mixed with other information.		Y	Y
[2049:2049]	1(8)	Whole otp programming disable	[3:0] OTP programing enable/disable 0 programming is enable 1 programming is disable [7:4] reserved.		Y	Y
[2050:2050]	1(8)	OTP Pad Access disable	[3:0] OTP pad access enable/disable PAD access enabled PAD access disabled		Y	Y

Byte Address	Bytes (valid bits)	Field name	Functions	Endianess	U84 accessible	4bit XOR
		SCPU JTAG disable	[7:4] SCPU JTAG enable/disable SCPU JTAG enabled SCPU JTAG disabled		Y	Y
[2051:2051]	1(8)	rd_mode	set chip mode to rd mode rd mode user mode		Y	Y
[2052:2059]	8(64)	reserved			Y	Y
[2060:2091]	8(64)/32(256)	Market ID			Y (read out by calling of security service)	Y
[2092:2123]	8(64)/32(256)	Device ID			Y (read out by calling of security service)	Y
[2124:2139]	4(32)/16(128)	Functional Feature list	Bit1: HEVC encoding enabled 0H: disabled 1H: enabled		Y (read out by calling security service)	Y
			Bit2: HEVC decoding enabled 0H: disabled 1H: enabled			
			Bit3: AVC encoding enabled 0H: disabled 1H: enabled			
			Bit4: AVC decoding enabled 0H: disabled 1H: enabled			
			Bit5~Bit31 Reserved			
[2140:2155]	4(32)/16(128)	Security Feature	Bit0: Size of RSA key 0:RSA2048	Little Endian	Y	Y

Byte Address	Bytes (valid bits)	Field name	Functions	Endianess	U84 accessible	4bit XOR
		list	1:RSA4096			
			Bit1: Size of Memory-to-Memory key 0:128bit key is used 1:256bit key is used			
			Bit2: Boot image decryption key 0:128bit key is used 1:256bit key is used			
			Bit3: Whole memory encryption key 0: Not present 1: Presented			
			Bit4: ECDSA key 0: Not present 1: Presented			
			Bit5: HDCP key 0: Not present 1: Presented			
			Bit[6]: Password for debugging (Unused) 0: Not present 1: Password to debug			
			Bit 7: security processor firmware decryption key presentation 0: Not present 1: Presented			
			BIT8~BIT31 Reserved			
[2156:3071]	916(7328)	Reserved for other data			Y (read out by calling of security service)	Y
[3072:3087]	16(128)	Unique chip ID			Y (read out by calling of security	N

Byte Address	Bytes (valid bits)	Field name	Functions	Endianess	U84 accessible	4bit XOR
					servic e)	
[3088]	1(8)	Number of DIE	Number of DIE, indicate single Die system or multi-die system 0/1: signle die system others: multi-die system	N/A	Y	N
[3089]	1(8)	DIE ordinal	DIE ordinal	N/A	Y	N
		Reserved	Reserved		Y	N
[3199:3090]		Reserved				
[3239:3200]	40(320)	DFT information	used for DFT information.		Y	N
Reserved						

11.5.2 OTP operation

11.5.2.1 OTP timing

The timing control of OTP access is completed by configuing the registers in otp_csr. The timing unit in csr is clock cycle number of sys_cfg_clk, and the time parameter of otp needs to be calculated according to the clock cycle. The default clock cycle of sys_cfg_clk is 5ns.

OTP initialization timing

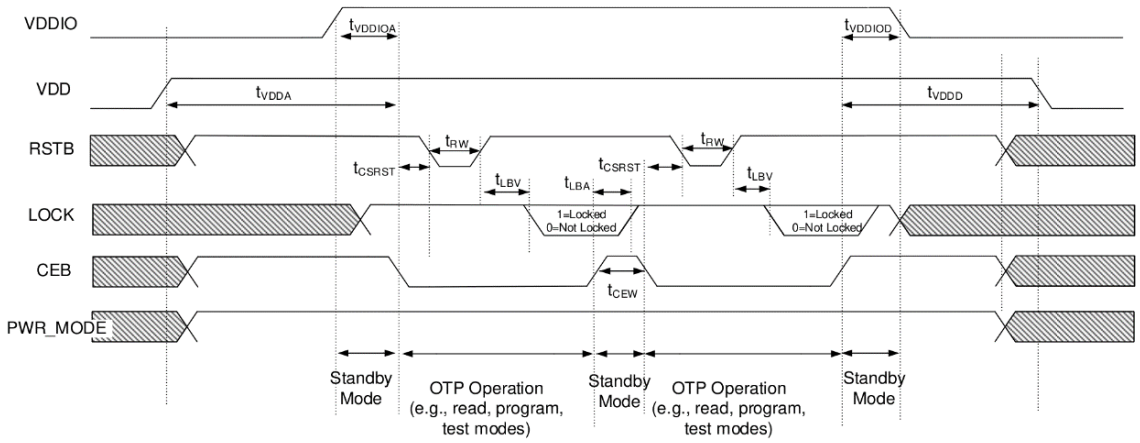


Figure 11-4 OTP initialization timing

Table 11-1 Explanation of OTP initialization timing

Timing	Description	Min.	Typ.	Max.	Unit
Initialization Operation					
t_{RW}	RSTB Pulse Width	20	-	-	ns
t_{CSRST}	CEB Setup Time before RSTB	1000	-	-	ns
t_{CEW}	Minimum CEB High Pulse Width	20	-	-	ns
t_{LBV}	RSTB de-assertion to LOCK output valid Delay	-	-	500	ns
t_{LBA}	CEB de-assertion to LOCK output valid Delay	-	-	10	ns
t_{VDDA}	VDD assertion (up in Spec.) to CEB assertion Delay	100	-	-	ns
t_{VDDIOA}	VDDIO assertion (up in Spec.) to CEB assertion Delay	100	-	-	ns
t_{VDDD}	CEB de-assertion to VDD de-assertion Delay	100	-	-	ns
t_{VDDIOD}	CEB de-assertion to VDDIO de-assertion Delay	100	-	-	ns

OTP read

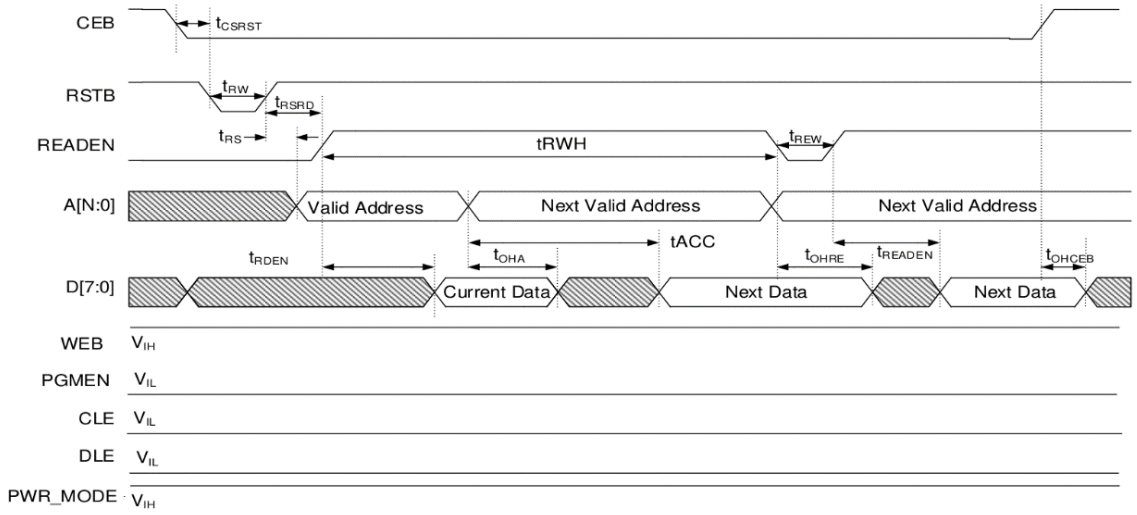


Figure 11-5 OTP read timing

Table 11-2 Explanation of OTP read timing

Timing	Description	Min.	Typ.	Max.	Unit
Read General Timing					
t_{RW}	RSTB Pulse Width	20	-	-	ns
t_{RSRD}	RSTB de-assertion to READEN assertion Delay	1000	-	-	ns
t_{RS}	RSTB de-assertion to Address change Delay	1000	-	-	ns
t_{OHA}	Data Output Hold Time after Address change	0	-	-	ns
t_{OHRE}	Data Output Hold Time after READEN de-asserts	0	-	-	ns
t_{CEW}	Minimum CEB High Pulse Width	20	-	-	ns
t_{OHCEB}	Data Output Hold Time after CEB de-asserts	0	-	-	ns
t_{CSRST}	CEB assertion to RSTB assertion Delay	1000	-	-	ns
t_{REW}	Minimum READEN Low Pulse Width	20	-	-	ns
t_{RWH}	Minimum READEN High Pulse Width	45	-	-	ns
t_{ACC}	Address change to Data Output Delay	-	-	t_{PEH} *	ns
t_{READEN}	READEN assertion to Data Output Delay	-	-	t_{PEH} *	ns

OTP write

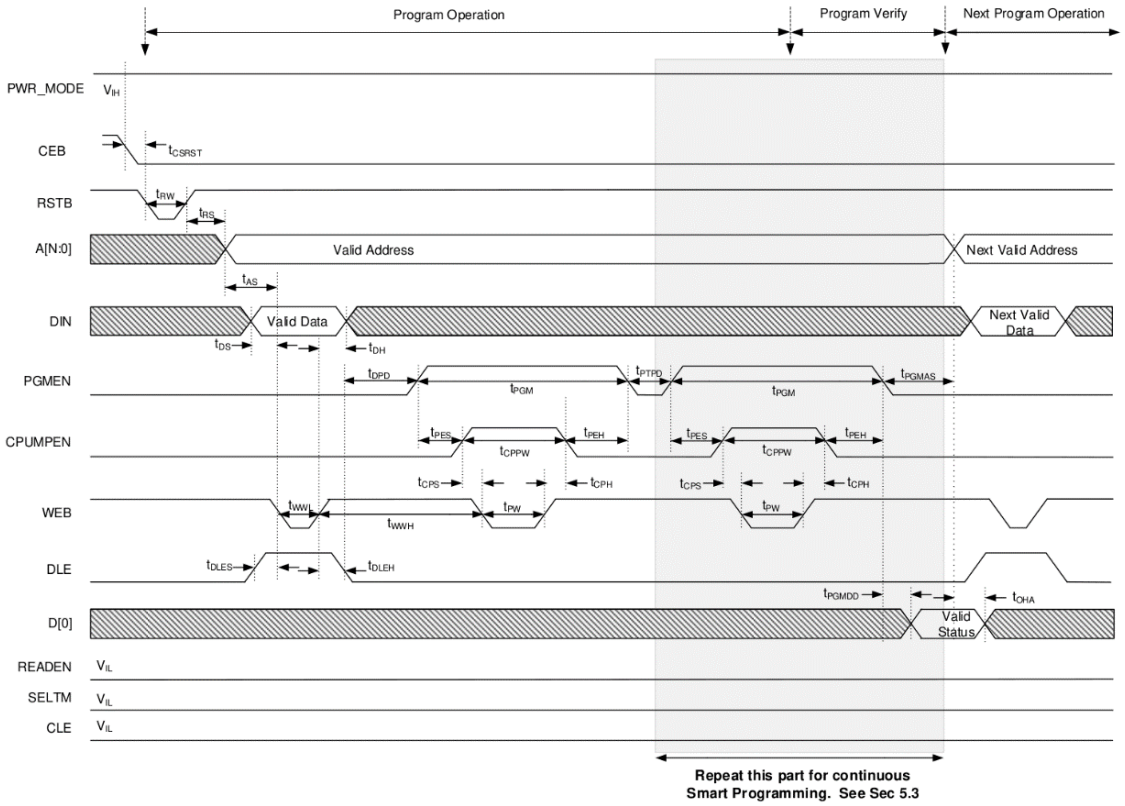


Figure 11-6 OTP write timing

Table 11-3 Explanation of OTP write timing

Timing	Description	Min.	Typ.	Max.	Unit
Programming Operation					
t_{RW}	RSTB Pulse Width	20	-	-	ns
t_{RS}	RSTB Setup Time before Address	1000	-	-	ns
t_{CSRST}	CEB Setup Time before RSTB	1000	-	-	ns
t_{AS}	Address Setup Time before WEB	25	-	-	ns
t_{DS}	Data Setup Time before WEB	20	-	-	ns
t_{DLES}	DLE Setup Time before WEB	20	-	-	ns
t_{WWL}	Write Enable (WEB) Pulse Width Low	20	-	-	ns
t_{WWH}	Write Enable (WEB) Pulse Width High	50	-	-	ns
t_{DLEH}	DLE Hold Time after WEB	20	-	-	ns
t_{DH}	Data Hold Time after WEB	20	-	-	ns
t_{PES}	PGMEN Setup Time before CPUMPEN	1	-	-	μs
t_{CPS}	CPUMPEN Setup Time before WEB	1	-	-	μs
t_{PW}	Program Pulse Width for WEB	4.5	5	5.5	μs
t_{CPH}	CPUMPEN Hold Time after WEB	1	-	-	μs
t_{CPPW}	Charge Pump Program Pulse Width (min. of $t_{CPS} + t_{PW} + t_{CPH}$)	6.5	-	-	μs
t_{PGM}	Program Time (min. of $t_{PES} + t_{CPPW} + t_{PEH}$)	7.545	-	-	μs
t_{PEH} *	CPUMPEN de-assert to PGMEN de-assert Delay	45	-	200	ns
t_{PGMAS}	PGMEN de-assert to Address change Delay	100	-	-	ns
t_{PGMRD}	PGMEN de-assert to READEN assert Delay	100	-	-	ns
t_{DPD}	DLE de-assertion to PGMEN assertion Delay	20	-	-	ns
t_{PGMDD}	PGMEN de-assert to Data Output Delay	-	-	0	ns
t_{OHA}	Output Hold Time after Address	0	-	-	ns
t_{PTPD}	PGMEN de-assert to PGMEN assert Delay	100	-	-	ns

11.5.2.2 otp operation

software read operation as follows

- verify that OTP has been initialized;
- Software query busy status, confirm the otp controller is IDLE;
- Software configure the start address, read size, the maximum data length is 128Byte of a single write/read operation;
- Configure start word: 0X6accacc6, the otp controller turn to busy state, then hardware automatically generate read timing, read back data into buffer;
- When the data read back, it set the valid data flag to 1. When the data in the buffer are read out, the controller status go back to IDLE.

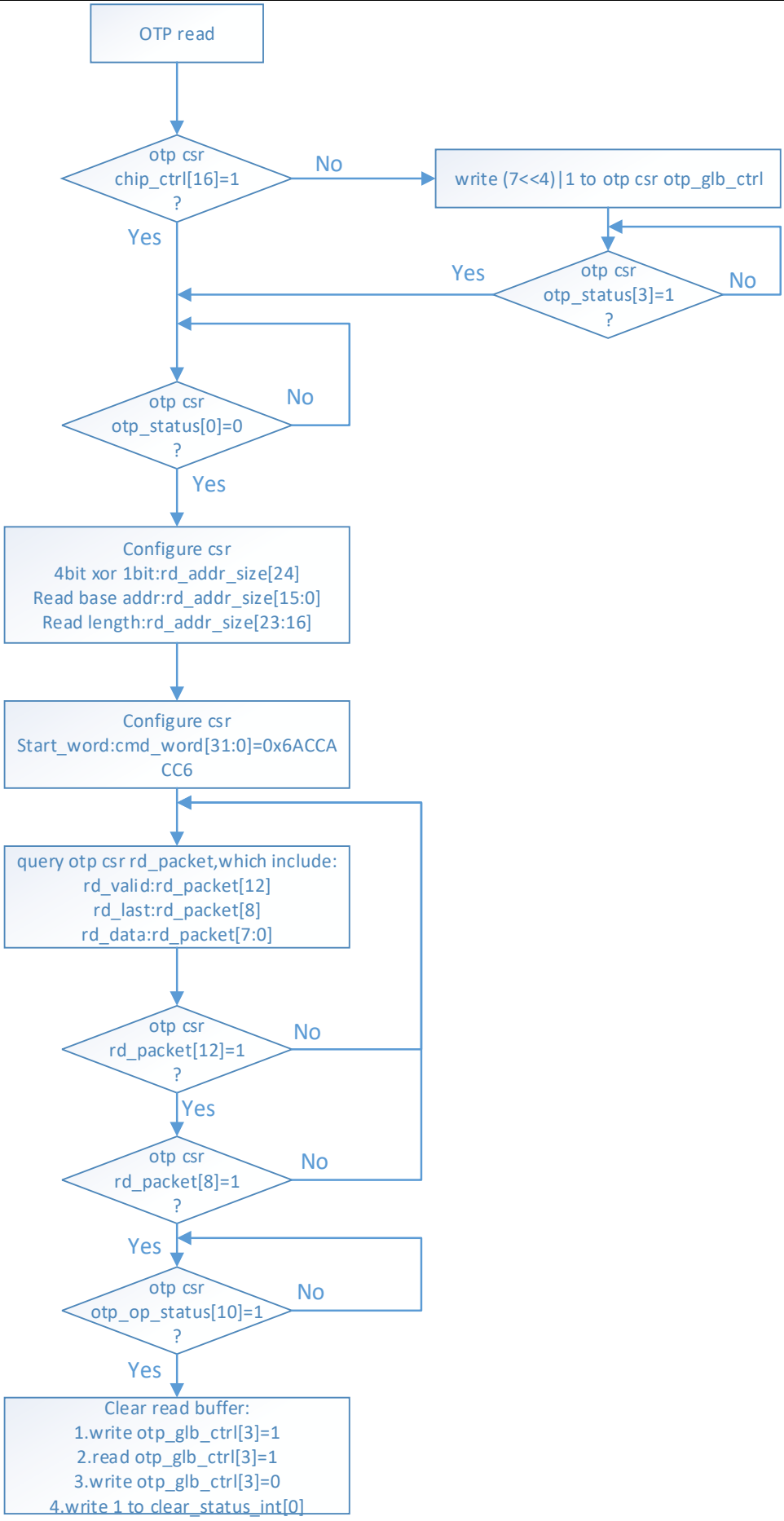


Figure 11-7 otp read

write opteration，see follows:

- verify that OTP has been initialized;
- Software query busy status and lock flag, confirm the controller is IDLE and in the non-lock state;
- Software configure re-write pulse number(if write fail), write address, write data and bitmask;
- Software configure start command 0X9accacc9, controller turn to busy state, hardware generate writing timing;
- When the write is complete, controller provide the write status information, and a FAIL message.

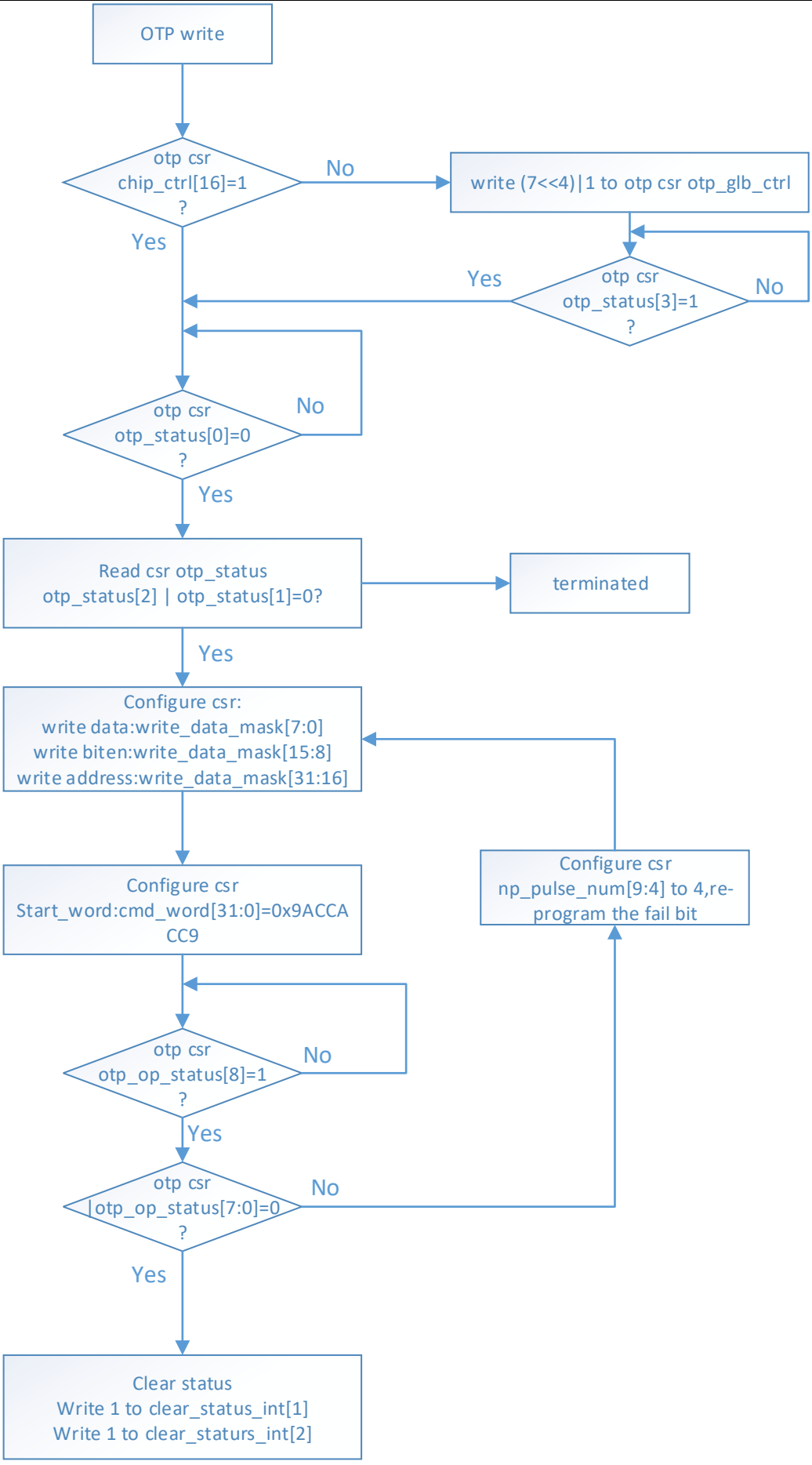


Figure 11-8 otp write

PGMLOCK operation，see follows:

- Confirm that OTP has been initialized
- Software query busy status and lock information, confirm controller is IDLE and in the non-lock state;
- Configure the PGMLOCK command 0x10C610C6 to start the PGMLOCK process;
- Check the status of pgmlock_done csr and confirm whether otp_pgm_lock was successful.

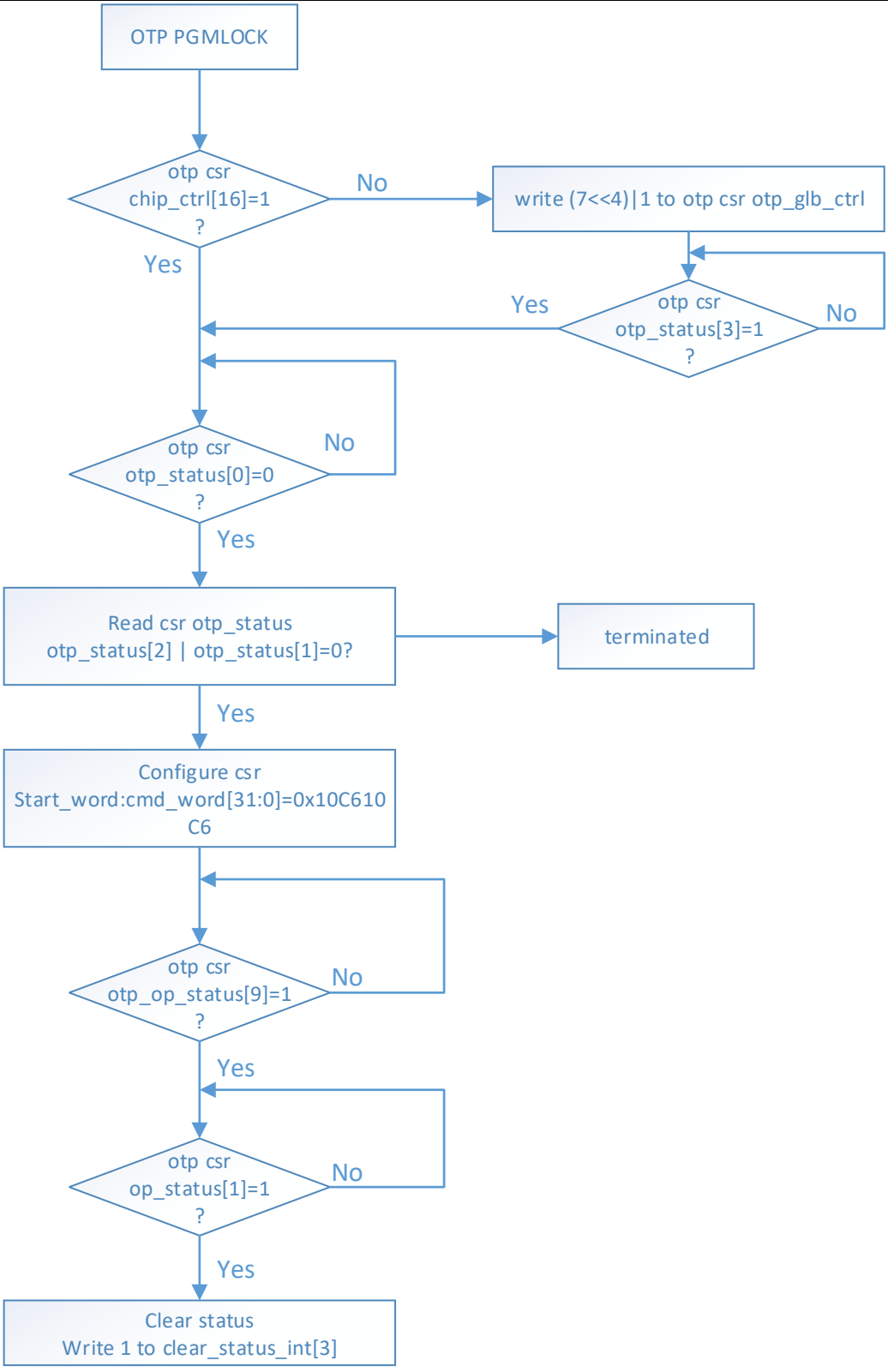


Figure 11-9 otp lock

12 Peripheral Devices

EIC7700X includes the following interfaces:

- 12×I2C, Supports speeds up to 1000 kbit/s
- 5×UART, uart1 supports automatic flow control

- 2×SPI, supports up to 50 MHz and only supports master mode
- 1×PWM, Three programmable timers are internally instantiated, supporting 0%~100% duty cycle
- 4×TIMER,Each TIMER internally instantiates 8 programmable timers, supporting 0%~100% duty cycle
- 112×GPIO, The 0~31 GPIO supports interrupt mode
- 3×JTAG, Supports up to 20 MHz
- 3×I2S, Supports up to 192 kbps
- 1 fan control interface (FAN_CTRL) to receive fan speed information

I2C/UART/SPI/PWM/FAN_CTRL all have soft reset and gating. Before enable the peripheral, you need to release the reset and disable clock gating. For some peripheral, you need to configure the corresponding IO pad. The reset and clock of the GPIO module are the same as those of the CLMM.

The base address of the low-speed peripheral is as follows:

Table 12-1 Peripheral address space.

Name	Address space	Size	Remarks
wdt0	0x5080_0000~0x5080_3FFF	16k	LSP APB2
wdt1	0x5080_4000~0x5080_7FFF	16k	LSP APB2
wdt2	0x5080_8000~0x5080_BFFF	16k	LSP APB2
wdt3	0x5080_C000~0x5080_FFFF	16k	LSP APB2
spi0	0x5081_0000~0x5081_3FFF	16k	LSP APB2
spi1	0x5081_4000~0x5081_7FFF	16k	LSP APB2
pwm	0x5081_8000~0x5081_BFFF	16k	LSP APB2
uart0	0x5090_0000~0x5090_FFFF	64k	LSP APB3
uart1	0x5091_0000~0x5091_FFFF	64k	LSP APB3
uart2	0x5092_0000~0x5092_FFFF	64k	LSP APB3
uart3	0x5093_0000~0x5093_FFFF	64k	LSP APB3
uart4	0x5094_0000~0x5094_FFFF	64k	LSP APB3
i2c0	0x5095_0000~0x5095_FFFF	64k	LSP APB3
i2c1	0x5096_0000~0x5096_FFFF	64k	LSP APB3
i2c2	0x5097_0000~0x5097_FFFF	64k	LSP APB3
i2c3	0x5098_0000~0x5098_FFFF	64k	LSP APB3
i2c4	0x5099_0000~0x5099_FFFF	64k	LSP APB3
i2c5	0x509A_0000~0x509A_FFFF	64k	LSP APB3
i2c6	0x509B_0000~0x509B_FFFF	64k	LSP APB3
i2c7	0x509C_0000~0x509C_FFFF	64k	LSP APB3
i2c8	0x509D_0000~0x509D_FFFF	64k	LSP APB3
i2c9	0x509E_0000~0x509E_FFFF	64k	LSP APB3
GPIO	0x5160_0000~0x5160_007F	128B	
CLMM control	0x5160_0000~0x517F_FFFF	2M-128B	

Name	Address space	Size	Remarks
i2c10	0x5183_0000~0x5183_7FFF	32k	
i2c11	0x5183_8000~0x5183_FFFF	32k	
TIMER0	0x5184_0000~0x5184_7FFF	32k	
TIMER1	0x5184_8000~0x5184_FFFF	32k	
TIMER2	0x5185_0000~0x5185_7FFF	32k	
TIMER3	0x5185_8000~0x5185_FFFF	32k	

12.1 IO Control（CLMM）

12.1.1 Overview

CLMM（Chip-Level Mode Mux）provides a pad reuse mechanism for many functional modules that require IO ports for internal and external data/control interaction.

Supports the following functions:

- Supports apb.2.0. The data bit width is 32bit.
- Support IO input enable control
- Support IO driving strength setting
- Control IO multiplexing through function sel
- Support Schmidt trigger
- Support RGMII pad, support 1.8V, 3.3V voltage
- Support pull-up/pull-down settings
- Clock input via OSC Pad is supported
- Only need to set the function selection, pull up and down, etc., it is not recommended to set the driving capacity

12.2 I2C

12.2.1 Overview

I2C is a configurable, programmable control bus that supports the communication between devices. It is a simple two-wire bus protocols for system control for temperature sensors and voltage level translators, general-purpose I/O, A/D and D/A converters, codecs and many types of microprocessors.

The following features are supported:

- Support APB4.0, bus data bit width 32bit
- Master OR slave I2C operation
- Three speeds:
  - Standard mode: 0~100kb/s
  - Fast mode: ⩽400kb/s
  - fast mode plus: ⩽1000kb/s
- 7-bit addressing7bit
- Bulk transmit mode
- Interrupt or polled-mode operation
- Support DMA transmission
- Receive/send data fifo depth 32
- Handles Bit and Byte waiting at all bus speeds
- Programmable SDA hold time (tHD;DAT)
- SMBus/PMBus support
- Bus clear feature

12.2.2 Block Diagram

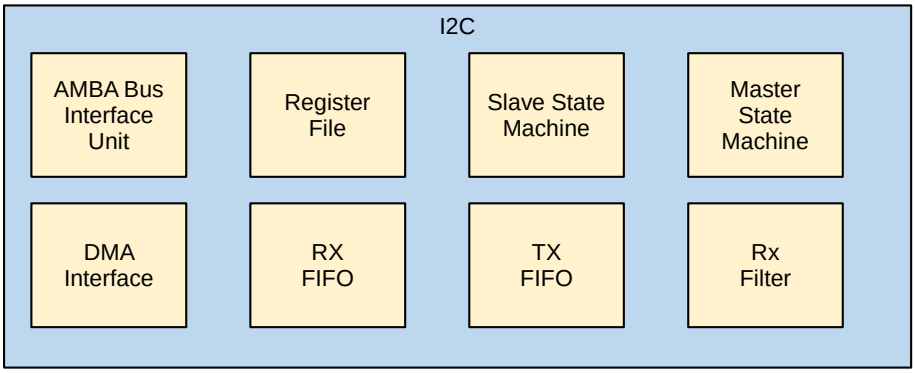


Figure 12-1 I2C block diagram

12.2.3 Function Description

12.2.3.1 Initialize

In addition to configuring the I2C itself, you also need to configure:

- Release the I2C apb bus reset, that is, the value of sw_i2c_rst_n register is 1
- Configure the CLMM func_sel to enable IE for pads
- Configure the APB clock

12.2.3.2 TXFIFO Data Format

DATA –Read/Write field; data retrieved from slave is read from this field; data to be sent to slave is written to this field.
CMD –Write-only field; this bit determines whether transfer to be carried out is Read (CMD=1) or Write (CMD=0)

Figure 12-2 TX fifo Data format

12.2.3.3 Spike Suppression

This logic is based on counters that monitor the input signals (SCL and SDA), checking if they remain stable for a predetermined amount of ic_clk cycles before they are sampled internally. There is one separate counter for each signal (SCL and SDA). The number of ic_clk cycles can be programmed and should be calculated taking into account the frequency of ic_clk and the relevant spike length specification.

Each counter is started whenever its input signal changes its value. Depending on the behavior of the input signal, one of the following scenarios occurs:

- The input signal remains unchanged until the counter reaches its count limit value. When this happens, the internal version of the signal is updated with the input value, and the counter is reset and stopped. The counter is not restarted until a new change on the input signal is detected.
- The input signal changes again before the counter reaches its count limit value. When this happens, the counter is reset and stopped, but the internal version of the signal is not updated. The counter remains stopped until a new change on the input signal is detected.

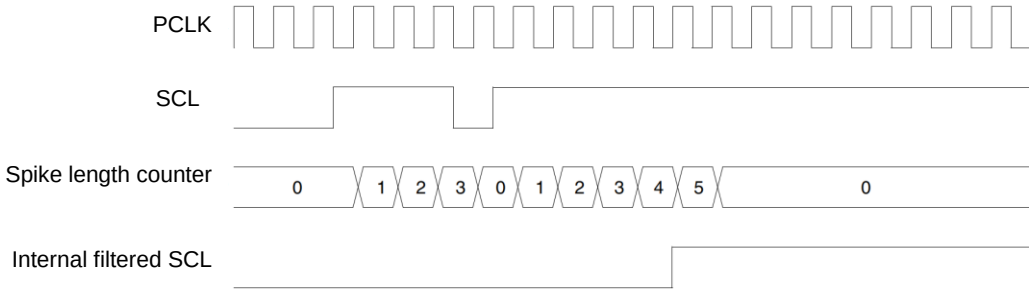


Figure 12-3 Spike Suppression Example

The spike suppression length in SS and FS modes is controlled by IC_FS_SPKLEN registers. The registers are 8 bits wide and can be read and written through the APB interface, however, they can only be written when i2c is disabled. The minimum value that can be programmed to these registers is 1; values less than 1 are treated as 1.

12.2.3.4 SCL Rate Calculation

When i2c is running as an I2C master, in send and receive transmissions:

- IC_SS_SCL_LCNT/IC_FS_SCL_LCNT should bigger than IC_FS_SPKLEN + 7
- IC_SS_SCL_HCNT/IC_FS_SCL_HCNT should bigger than IC_FS_SPKLEN + 5

Regardless of SCL_Rise_time and SCL_Fall_time, the following is an example:

- The data rate of fast mode is 400kb/s; Indicates that the SCL period is 1/400khz = 2.5us
- Minimum SCL high and low time:
  - MIN_SCL_LOWtime_FS = 1300ns
  - MIN_SCL_HIGHtime_FS = 600ns
- The APB clock takes 200MHz as an example, that is, the period is 5ns

SPKLEN does not directly affect the SCL frequency, but the master will check whether the SCL of the bus is pulled low by the slave when the master outputs SCL high. That is to say, the SCL input will be sent to the internal after SPKLEN, and then the master will decide whether to continue transmission according to the input SCL, so that SPLKEN will reduce the SCL frequency. In practice, it is also affected by the rise time and fall time of SCL. Theoretical calculated value after connecting the slave:

$$scl\_low\_time\_fs = (IC\_FS\_SCL\_LCNT + 1) * apb\_clk\_period$$

$$scl\_high\_time\_fs = (IC\_FS\_SCL\_HCNT + IC\_FS\_SPKLEN + 7) * apb\_clk\_period$$

if low time =1600ns(bigger than 1300ns)，high time = 900 ns(bigger than 600ns)，IC_FS_SPKLEN = 10:

$$1600 = (IC\_FS\_SCL\_LCNT + 1) * 5$$

$$900 = (IC\_FS\_SCL\_HCNT + 10 + 7) * 5$$

Then:

$$IC\_FS\_SCL\_LCNT = 0x1F3$$

$$IC\_FS\_SCL\_HCNT = 0xA3$$

12.2.3.5 Master Mode

1. Disable the I2C by writing 0 to bit 0 of the IC_ENABLE register.
2. Write to the IC_CON register to set the maximum speed mode supported for slave operation (bits 2:1) and to specify whether the I2C starts its transfers in 7 bit addressing mode when the device is a slave (bit 3)
3. Write to the IC_TAR register the address of the I2C device to be addressed. It also indicates

whether a General Call or a START BYTE command is going to be performed by I2C

4. Enable the I2C by writing a 1 to bit 0 of the IC_ENABLE register

5. Now write the transfer direction and data to be sent to the IC_DATA_CMD register

12.2.3.6 Master Transmit And Master Receive

The i2c supports switching back and forth between reading and writing dynamically. To transmit data, write the data to be written to the lower byte of the I2C Rx/Tx Data Buffer and Command Register (IC_DATA_CMD). The CMD bit [8] should be written to 0 for I2C write operations. Subsequently, a read command may be issued by writing "don't cares" to the lower byte of the IC_DATA_CMD register, and a 1 should be written to the CMD bit. The i2c master continues to initiate transfers as long as there are commands present in the transmit FIFO. If the transmit FIFO becomes empty it checks to see if IC_DATA_CMD[9] is set to 1.

- If set to 1, it issues a STOP condition after completing the current transfer.
- If set to 0, it holds SCL low until next command is written to the transmit FIFO

12.2.3.7 Slave Mode

To use the i2c as a slave, perform the following steps:

1. Disable the i2c by writing a '0' to bit 0 of the IC_ENABLE register.

2. Write to the IC_SAR register (bits 9:0) to set the slave address. This is the address to which the i2c responds.

3. Put the i2c in slave-only mode by writing '0' to the 6th bit (IC_SLAVE_DISABLE) of the IC_CON and '0' to the 0th bit (MASTER mode)..

4. Enable the i2c by writing a '1' in bit 0 of the IC_ENABLE register.

12.2.3.8 SDA Hold Time

The I2C protocol specification requires 300ns of hold time on the SDA signal (tHD;DAT) in standard mode and fast mode, and a hold time long enough to bridge the undefined part between logic 1 and logic 0 of the falling edge of SCL in high speed mode and fast mode plus.

The i2c contains a software programmable register (IC_SDA_HOLD) to enable dynamic adjustment of the SDA hold-time.

The bits [15:0] are used to control the hold time of SDA during transmit in both slave and master mode (after SCL goes from HIGH to LOW).

The bits [23:16] are used to extend the SDA transition (if any) whenever SCL is HIGH in the receiver (in either master or slave mode).

IC_SDA_HOLD register:

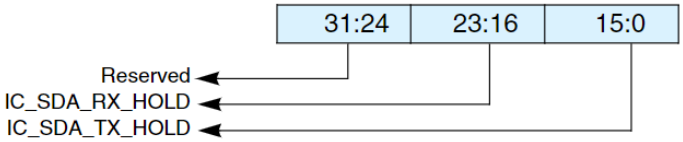


Figure 12-4 IC_SDA_HOLD Register

- RX hold

When i2c acts as a receiver, according to the I2C protocol, the device should internally hold the SDA line to bridge undefined gap between logic 1 and logic 0 of SCL. IC_SDA_RX_HOLD can be used to change the internal hold time that I2C applies to an inbound SDA line. Each value in the IC_SDA_RX_HOLD register represents a unit of ic_clk period. The minimum value of the IC_SDA_RX_HOLD is 0. This hold time only applies when the SCL is HIGH. After changing to LOW inside the SCL, the receiver does not

extend the SDA.

When the IC_SDA_RX_HOLD is equal to or greater than 3, the waveform is shown below

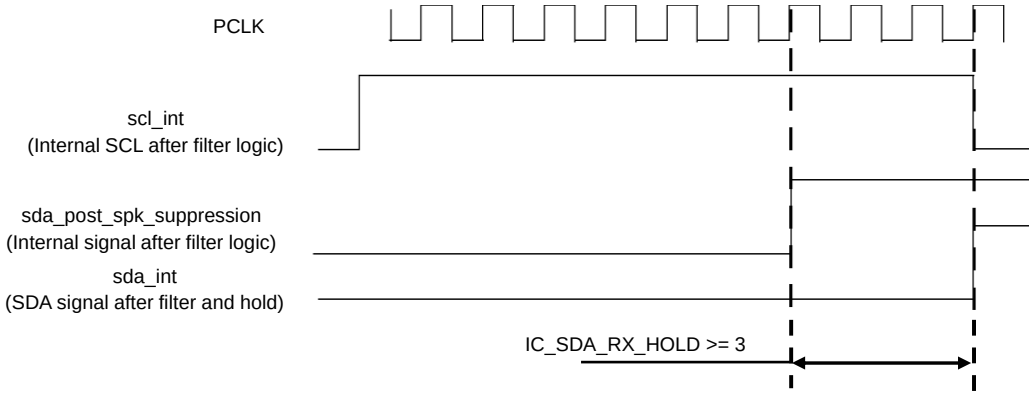


Figure 12-5 RX Hold time diagram

- TX hold

Note: The hold time of tx cannot exceed scl_lcnt - 2

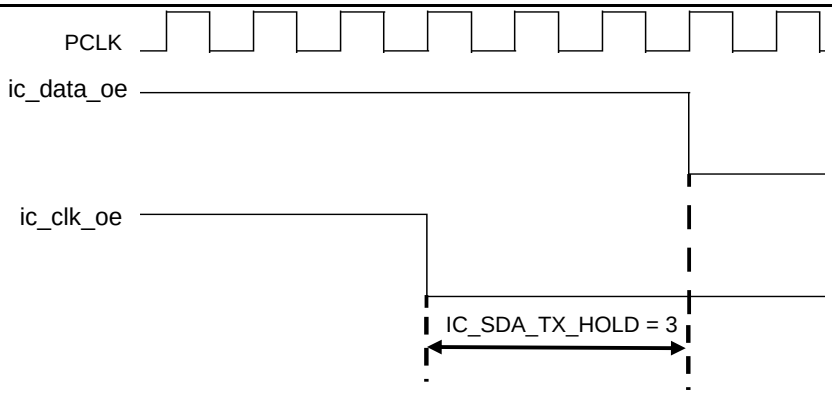


Figure 12-6 TX Hold time diagram.

12.2.4 Programming Example

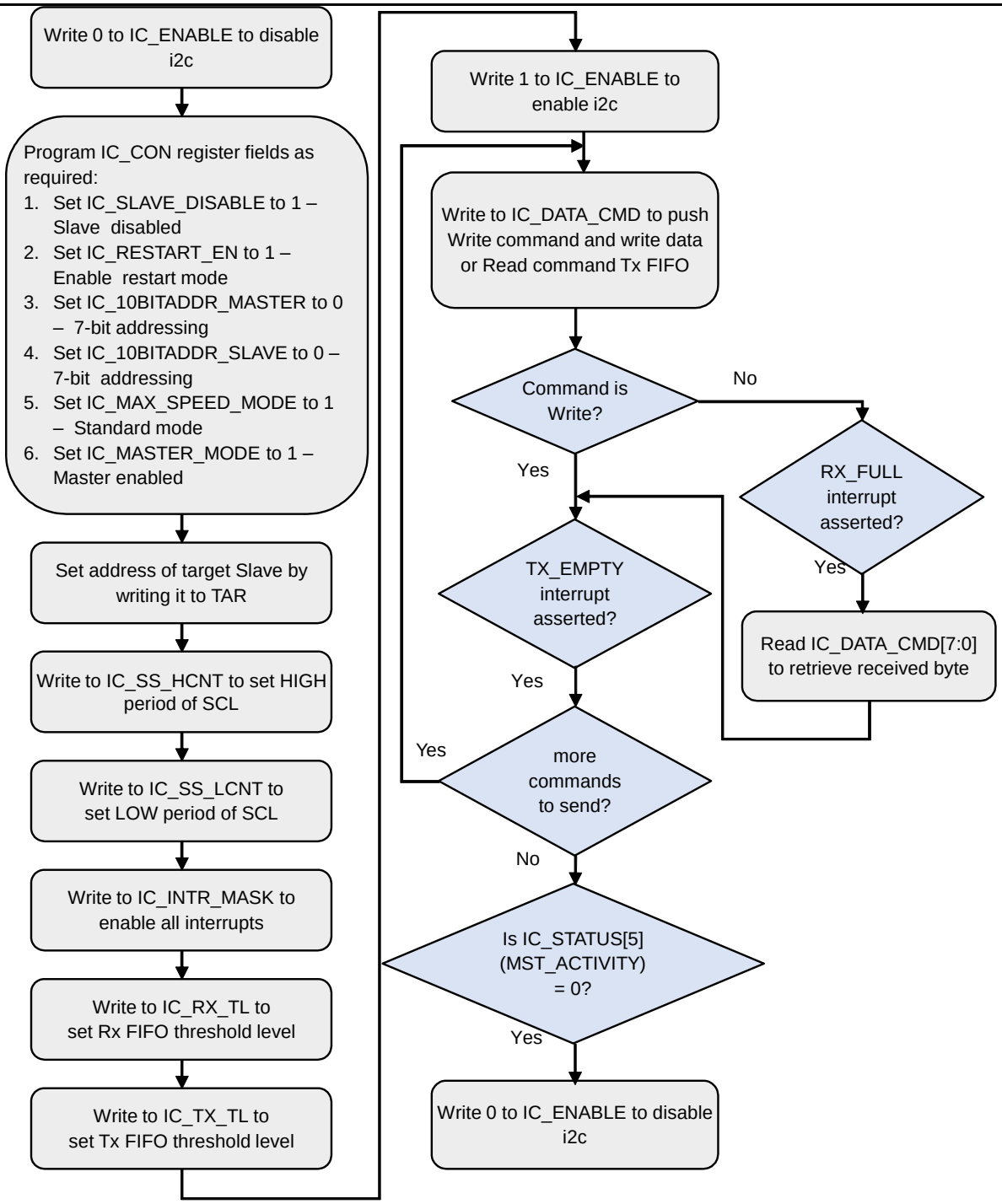


Figure 12-7 I2C MASTER programming example

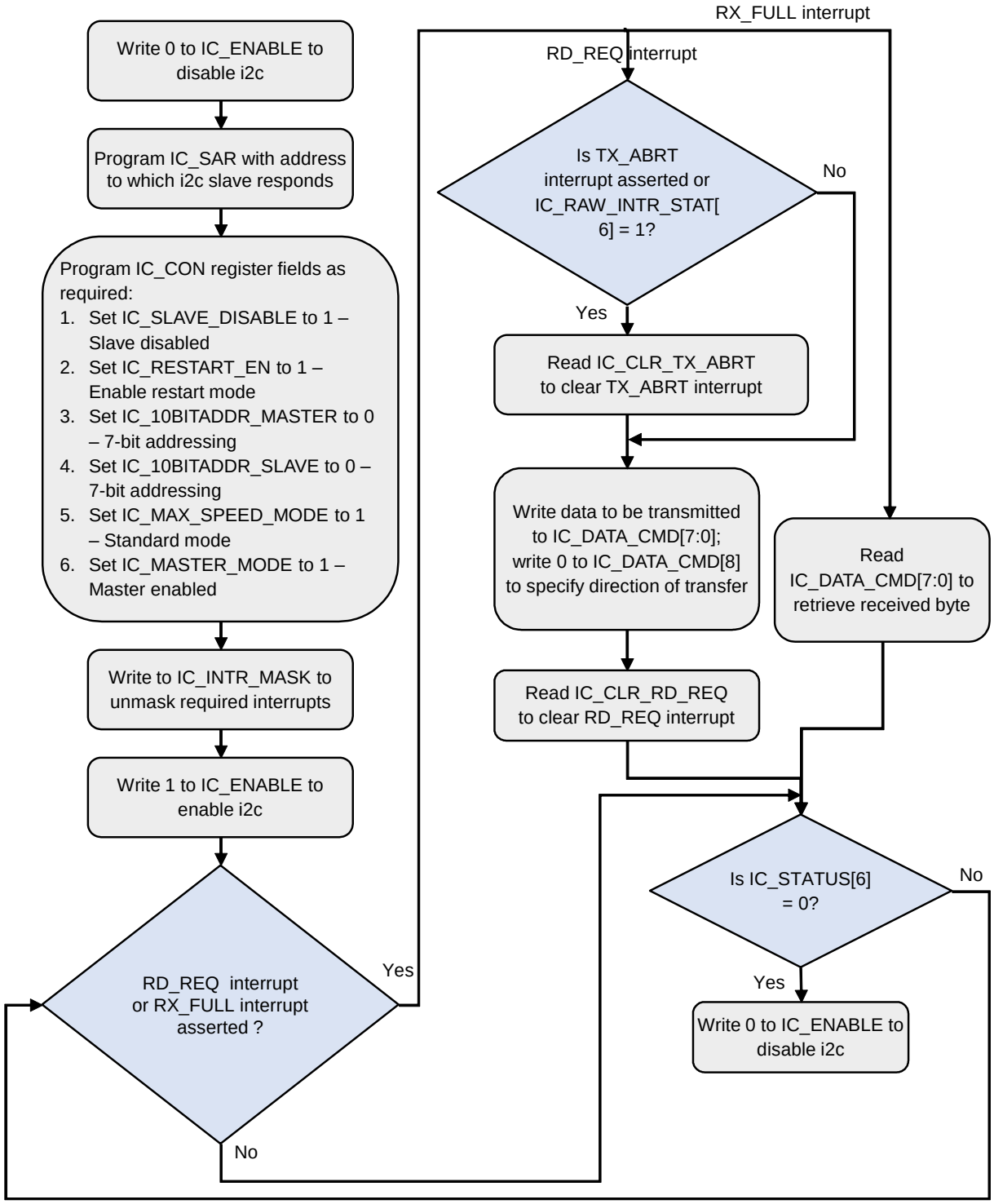


Figure 12-8 I2C slave programming example

12.2.5 Timing Sequence

Name	Description	Max(ns)	Min(ns)
t1	Start condition		
t2	Stop conditiion		
t3	Output SDA setup		
T4	Output SDA hold		

Name	Description	Max(ns)	Min(ns)
T5	SCL period		
T6	SCL pulse width high		
T7	SCL pulse width low		
T8	Input SDA setup		
T9	Input SDA hold		
T10	Start condition stup		

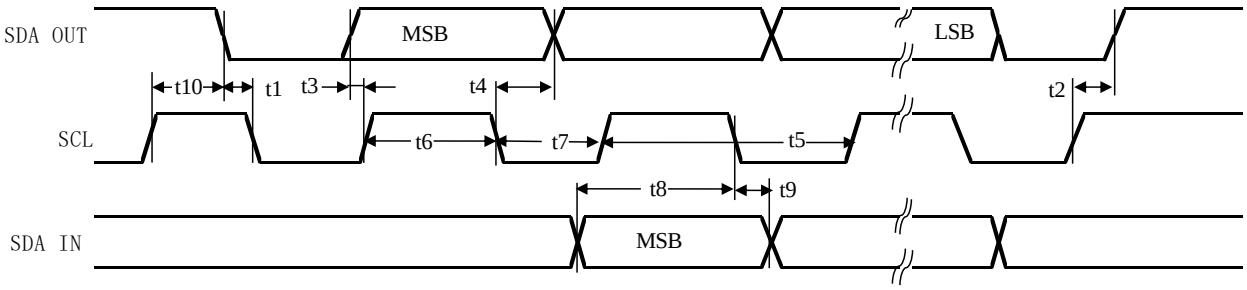


Figure 12-9 I2C timing sequence

12.3 I2S

12.3.1 Overview

The I2S bus (Inter-IC sound bus) is a serial link used for the transmission of digital audio data between devices in a system. Commonly used I2S bus devices include ADC, DAC, DSP, CPU, etc. With the I2S interface, we can connect audio devices and embedded SoC platforms together to provide an audio interface solution for the system.

The following features are supported:

- APB2.0 bus, 32-bit data bit width
- I2S master mode
- Supports I2S transmitter and reveiver
- Dual-channel audio transmission is supported
- Audio data resolution 12, 14, 16, 20, 24, 32 bits can be matched
- Support for SCLK gating
- Audio data transmission FIFO depth 16
- DMA transmission is supported

12.3.2 Block Diagram

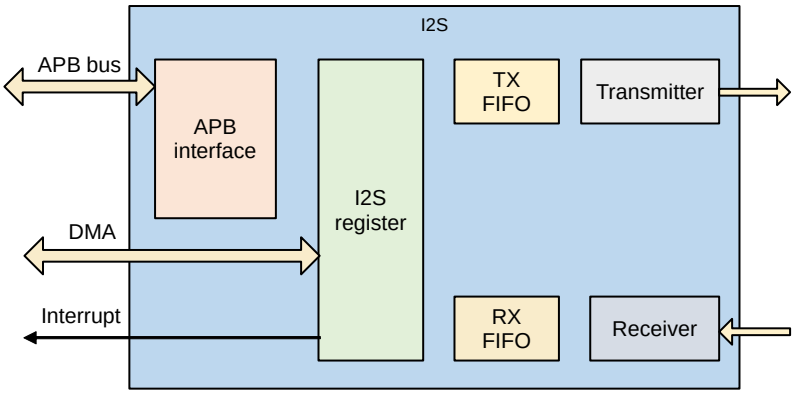


Figure 12-10 I2S block diagram

12.3.3 Function Description

12.3.3.1 Enable I2S

Before receiving or sending any data to the fifo, the IEN bit of the I2S Enable Register must be set to 1 - Enable I2s. Set "IER[0]" to "0" to disable "i2s". When disabled, the following events occur:

- TX and RX FIFOs are cleared, and read/write pointers are reset;
- Any data in the process of being transmitted or received is lost;
- All other programmable enables (such as transmitter/receiver block enables and individual TX/RX
- channel enables) in the component are overridden;
- Generation of master mode clock signals sclk_en, ws_out and sclk_gate are disabled (for instance, they are held low).

When I2S is enabled and configured as the master device, the device always starts at the left stereo data cycle (WS = 0) and then one SCLK cycle transitions to the right stereo data cycle (WS = 1).

12.3.3.2 Write Send Channel

The stereo data pairs to be transmitted by a TX channel are written to the TX FIFOs through the Left Transmit Holding Register (LTHRx, where x is the channel number) and the Right Transmit Holding Register (RTHRx, where x is the channel number). All stereo data pairs must be written using the following two-stage process:

1. Write left stereo data to LTHRx
2. Write right stereo data to RTHRx.

You must write stereo data to the device in this order, otherwise, the interrupt and status lines values are invalid, and the left/right stereo pairs might be transmitted out of sync

When TX DMA (I2S_TX_DMA = 1) is enabled, data for the TX channel is written to the TX fifo through the TXDMA registers instead of through LTHRx and RTHRx.

12.3.3.3 Reading From A Receive Channel

The stereo data pairs received by a RX channel are written to the left and right RX FIFOs. These FIFOs can be read through the Left Receive Buffer Register (LRBRx, where x is the channel number) and the Right Receive Buffer Register (RRBRx, where x is the channel number). All stereo data pairs must be read using the following two-stage process:

1. Read the left stereo data from LRBRx.
2. Read the right stereo data from RRBRx.

When RX DMA (I2S_RX_DMA=1) is enabled, data can be read from the RX fifo via the RXDMA registers instead of through LRBRx and RRBRx.

12.3.3.4 Clock Generation

The Clock Generation Enable (CLKEN) bit of the Clock Enable Register (CER) enables and disables the master mode clock signals: ws_out, sclk_en, and sclk_gate. To enable these signals, set CER[0] to 1; to disable them, set this bit to 0, in which case ws_out is held low (ws_out = 0).

When the CLKEN bit is disabled, any incoming or outgoing data is lost. However, data already in the RX and TX FIFOs are preserved. After this bit is enabled, transmission recommences at the start of the next stereo frame.

12.3.4 Programming Example

12.3.4.1 I2S As Transmitter（Master Mode）

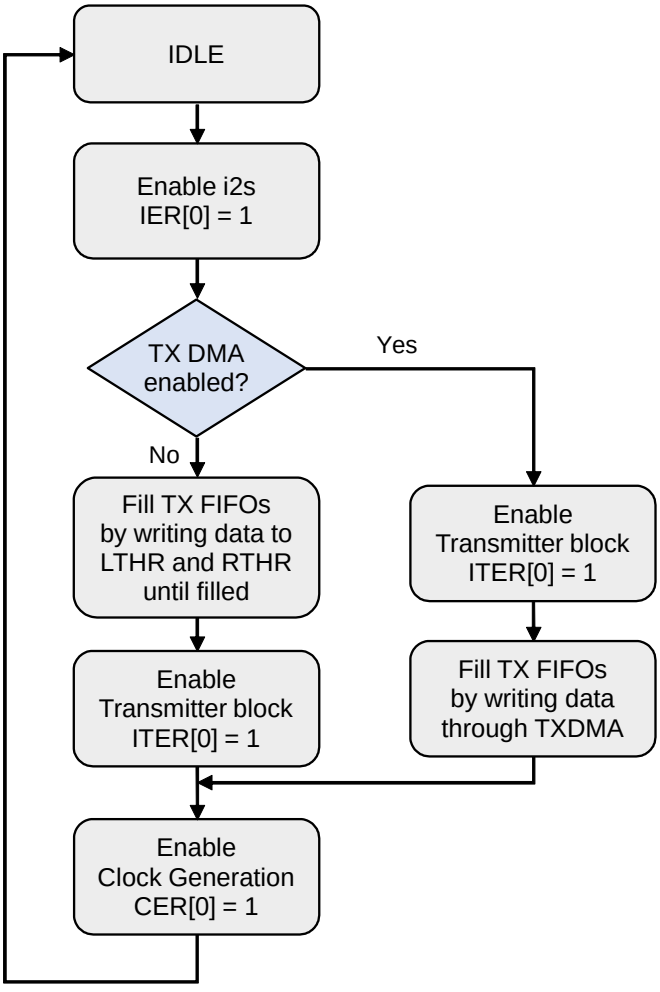


Figure 12-11 master transmiting process

12.3.4.2 I2S As Receiver（Master Mode）

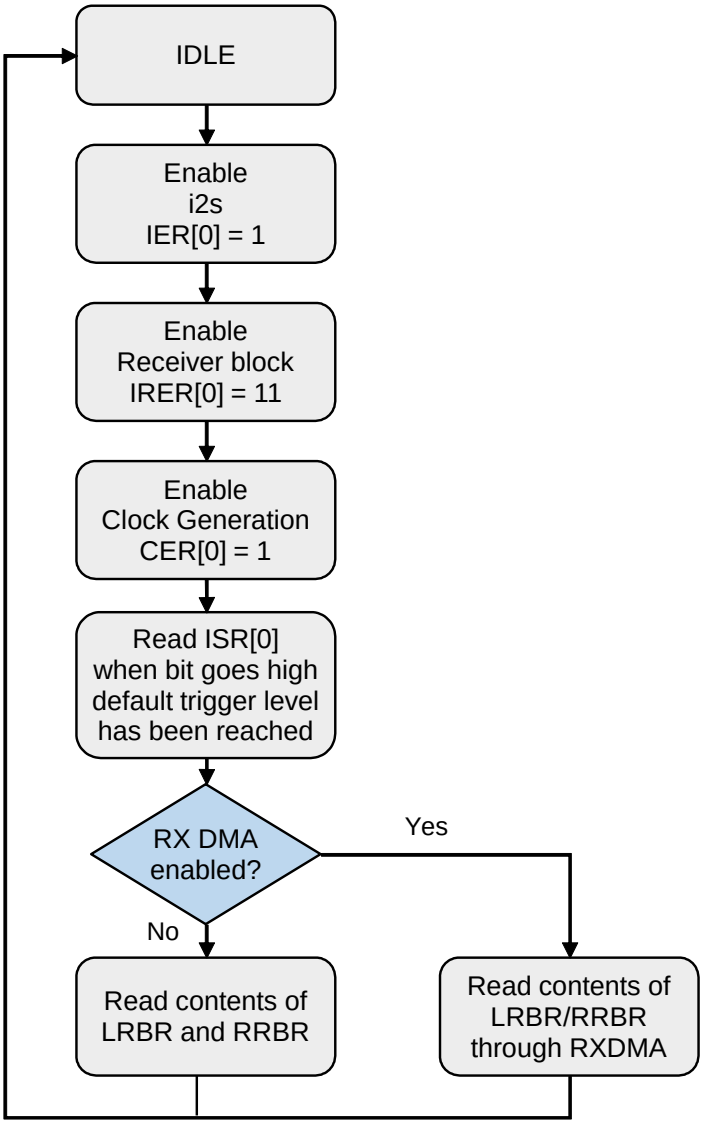


Figure 12-12 I2S MASTER receiving flow chart

12.3.5 Timing Sequence

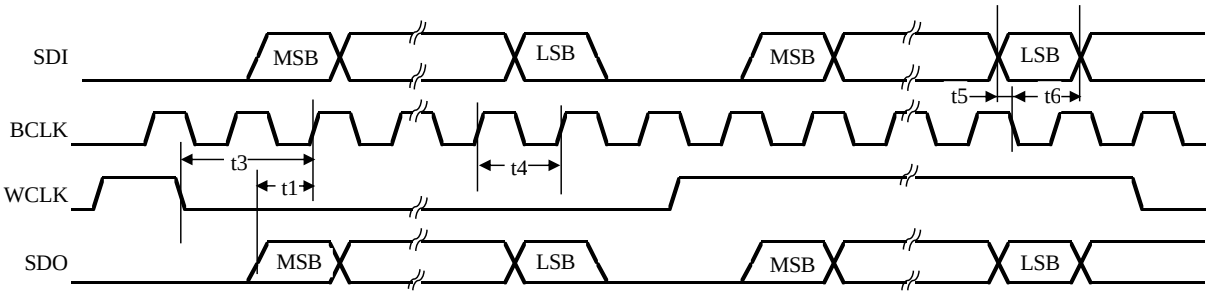


Figure 12-13 I2S timing sequence

12.4 UART

12.4.1 Overview

Data can be written from the host (CPU) to the UART via the APB bus, and then converted to serial form and transmitted to the target device. Serial data is also received by the UART and stored for the host (CPU) to read back.

Supported features:

- Support APB4.0, bus data bit width 32 bits
- Support DMA mode, automatic flow control mode
- Transmit and receive FIFO depth 32
- Functionality based on the 16550 industry standard
  - Number of data bits per character (5-8)
  - Optional parity bit (with odd, even select or Stick Parity)
  - Number of stop bits (1, 1.5 or 2)
  - Line break generation and detection
  - DMA signaling with two programmable modes
  - Prioritized interrupt identification
- Programmable decimal baud rate is supported
- Separate system resets for each clock domain to prevent metastability

12.4.2 Block Diagram

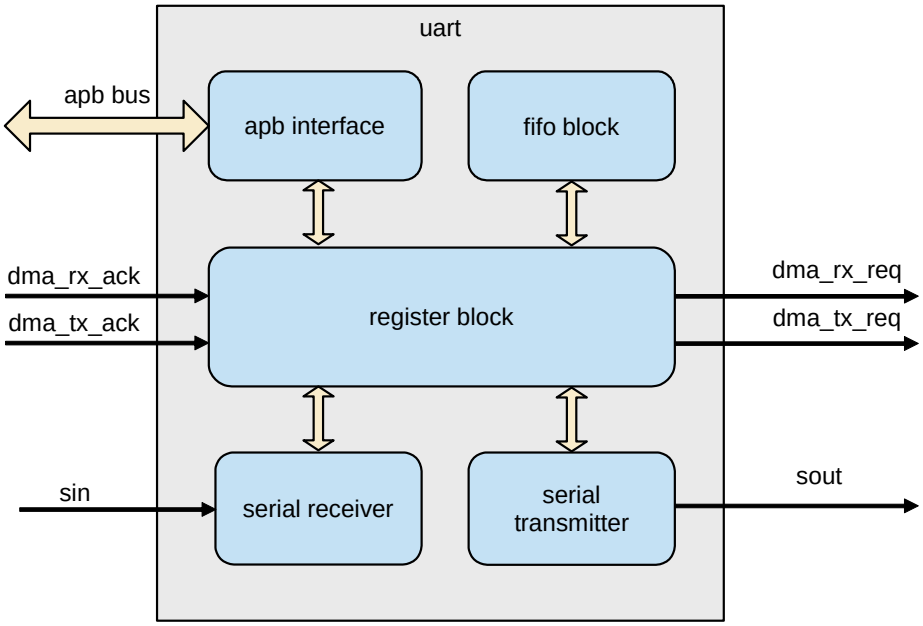


Figure 12-14 uart block diagram

12.4.3 Function Description

12.4.3.1 UART(RS232)

Serial communication between UART devices is asynchronous, with the start and stop bits indicating the start and end in the serial data. These bits are used to synchronize the two devices.

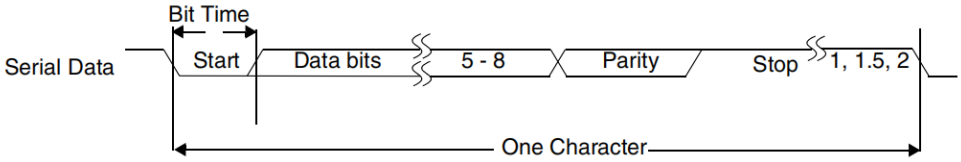


Figure 12-15 Serial Data Format

12.4.3.2 Baudrate

UART supports fractional Baud Rate:

$$\text{Baud Rate Divisor} = \frac{\text{Serial Clock Frequency}}{16 \times \text{Required Baud Rate}} = \text{BRDI} + \text{BRDF}$$

BRD_I - Integer part of the divisor.

BRD_F - Fractional part of the divisor, DLF/ 2^{DLF_SEZE} .

Serial Clock Frequency – apb clk frequency

Consider the following parameters:

Target baud rate(RBR) = 38400

APB clk frequency(PCLK)= 200 MHz

DLF_SIZE = 4

Then:

$BRD_I + BRD_F$ = 200 MHz / (16 x 38400) = 325.520833

BRD_I = 325，BRD_F = 0.520833

Therefore, the baud rate divisor fractional value (DLF) is:

DLF = 0.520833 × 2^{DLF_SIZE} = 8.3 = 8(Rounded)

The actual baud rate (GDR) is:

$$\text{GBR} = \frac{200MHz}{16 \times (\frac{8}{2^{DLF\_SIZE}} + BRD_I)} = 38{,}402.457$$

Percentage ERROR:

$$\text{Error} = \frac{GBR - RBR}{RBR} = \frac{38{,}402.457 - 38400}{38400} = 0.006\%$$

12.4.3.3 Auto Flow Control

The uart can be configured to have a 16750-compatible Auto RTS and Auto CTS serial data flow control mode available; if FIFOs are not implemented, this mode cannot be selected. When Auto Flow Control is not selected, none of the corresponding logic is implemented and the mode cannot be enabled, reducing overall gate counts. When Auto Flow Control mode is selected, it can be enabled with the Modem Control Register (MCR[5]).

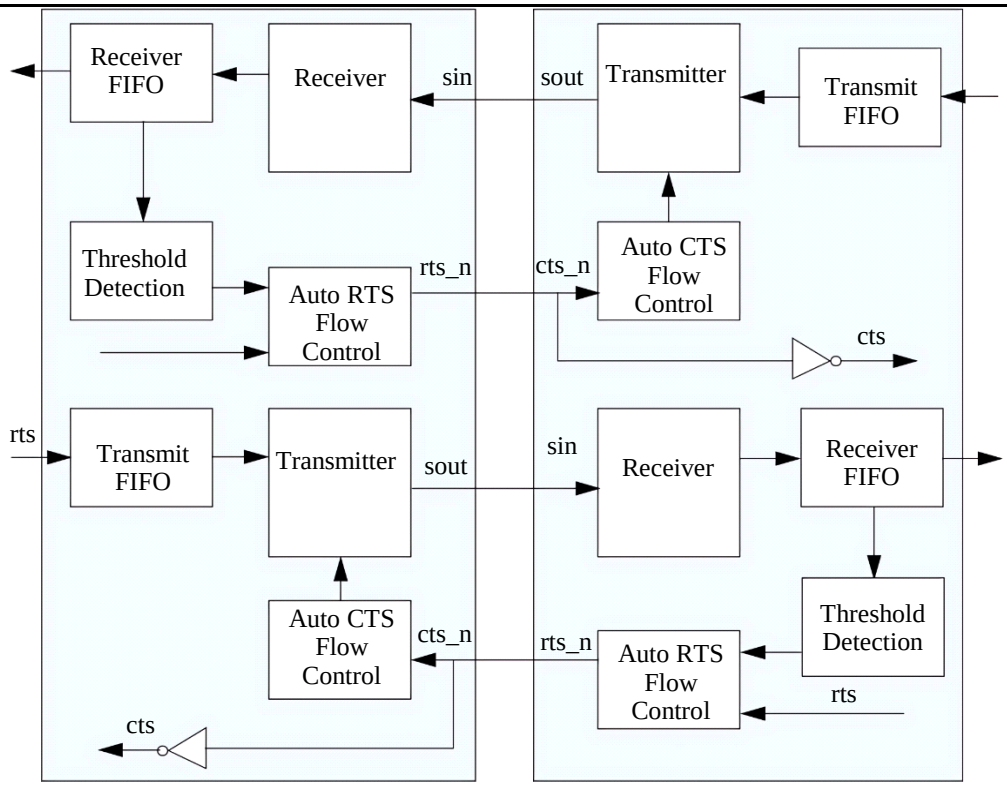


Figure 12-16 uart Auto Flow Control

Auto RTS – Becomes active when the following occurs:

- Auto Flow Control is selected during configuration
- FIFOs are implemented
- RTS (MCR[1] bit and MCR[5]bit are both set)
- FIFOs are enabled (FCR[0]) bit is set)
- SIR mode is disabled (MCR[6] bit is not set)

When Auto RTS is enabled, the rts_n output is forced inactive (high) when the receiver FIFO level reaches the threshold set by FCR[7:6], but only if the RTC flow-control trigger is disabled. Otherwise, the rts_n output is forced inactive (high) when the FIFO is almost full, where “almost full" refers to two available slots in the FIFO. When rts_n is connected to the cts_n input of another UART device, the other UART stops sending serial data until the receiver FIFO has available space; that is, until it is completely empty.

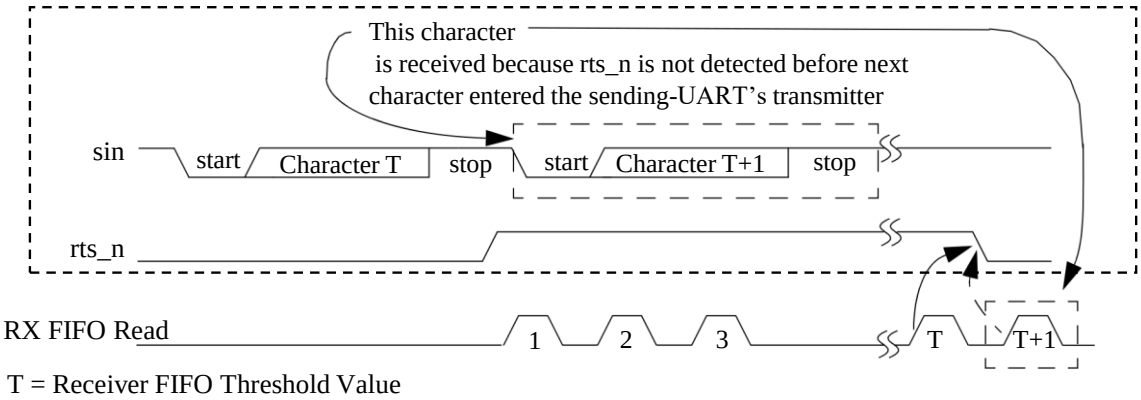


Figure 12-17 Auto CTS timing

Auto CTS – becomes active when the following occurs:

- Auto Flow Control is selected during configuration
- FIFOs are implemented

- AFCE (MCR[5] bit = 1)
- FIFOs are enabled through FIFO Control Register FCR[0] bit
- SIR mode is disabled (MCR[6] bit = 0)

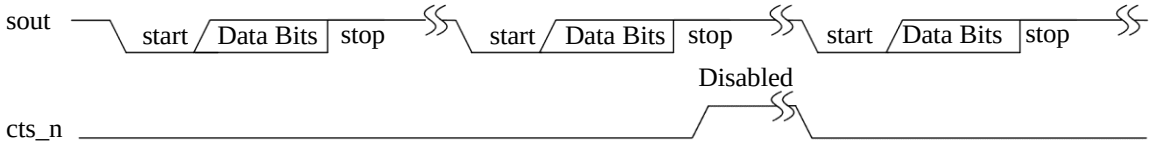


Figure 12-18 Auto CTS Timing

When Auto CTS is enabled (active), the UART transmitter is disabled whenever the cts_n input becomes inactive (high); this prevents overflowing the FIFO of the receiving UART.

When using the "FIFO full" status, software can poll this before each write to the Transmitter FIFO; for details, see "Programmable THRE Interrupt". When the cts_n input becomes active (low) again, transmission resumes.

12.4.3.4 Programmable Thre Interrupt

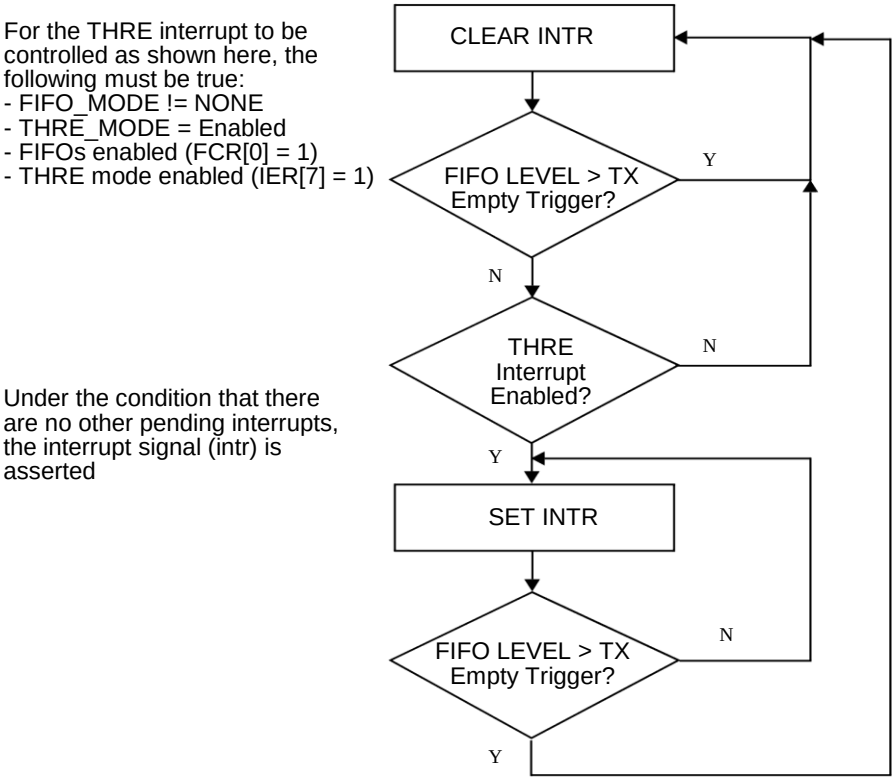


Figure 12-19 Flowchart of Interrupt Generation for Programmable THRE Interrupt Mode

12.4.3.5 Interrupt

The UART interrupt output signal (INTR) is a combined interrupt that is pulled up when one or more of the following are in effect. Specific interrupt information can be read in the IIR registers:

- Receiver Error
- Receiver Data Available
- Character Timeout (in FIFO mode only)

- ransmitter Holding Register Empty at/below threshold (in Programmable THRE interrupt mode)
- Modem Status
- Busy Detect Indication

Table 12-2 Interrupt Control Functions

Interrupt ID				interrupt set and reset functions			
Bit 3	Bit 2	Bit 1	Bit 0	Priority Level	Interrupt type	Interrupt Source	Interrupt Reset Control
0	0	0	1	–	None	None	–
0	1	1	0	Highest	Receiver line status	Overrun/parity/ framing errors, break interrupt, or address received interrupt	For Overrun/parity/framing/break interrupt reset control, the behavior is as follows: ■ If LSR_STATUS_CLEAR=0 (RBR Read or LSR Read), then the status is cleared on: - Reading the line status register Or - In addition to an LSR read, the Receiver line status is also cleared when RX_FIFO is read. ■ If LSR_STATUS_CLEAR=1 (LSR Read), the status is cleared only on: - Reading the line status register. ■ For address received interrupt, the status is cleared on: - Reading the line status register
0	1	0	0	Second	Received data available	Receiver data available (non-FIFO mode or FIFOs disabled) or RCVR FIFO trigger level reached (FIFOmode and FIFOs enabled)	Reading the receiver buffer register (non-FIFO mode or FIFOs disabled) or the FIFO drops below the trigger level (FIFO mode and FIFOs enabled)
1	1	0	0	Second	Character timeout indication	No characters in or out of the RCVR FIFO during the last 4 character times and there is at least 1 character in it during this time	Reading the receiver buffer register

Interrupt ID				interrupt set and reset functions			
Bit 3	Bit 2	Bit 1	Bit 0	Priority Level	Interrupt type	Interrupt Source	Interrupt Reset Control
0	0	1	0	Third	Transmit holding register empty	Transmitter holding register empty (Prog. THRE Mode disabled) or XMIT FIFO at or below threshold (Prog. THRE Mode enabled)	Reading the IIR register (if source of interrupt); or, writing into THR (FIFOs or THRE Mode not selected or disabled) or XMIT FIFO above threshold (FIFOs and THRE Mode selected and enabled).
0	0	0	0	Fourth	Modem status	Clear to send or data set ready or ring indicator or data carrier detect. Note that if auto flow control mode is enabled, a change in CTS (that is, DCTS set) does not cause an interrupt.	Reading the Modem status register
0	1	1	1	Fifth	Busy detect indication	UART_16550_COMPATIBLE = NO and master has tried to write to the Line Control Register while the uart is busy (USR[0] is set to 1)	Reading the UART status register

12.4.4 Programming Example

12.4.4.1 Transmission Flow

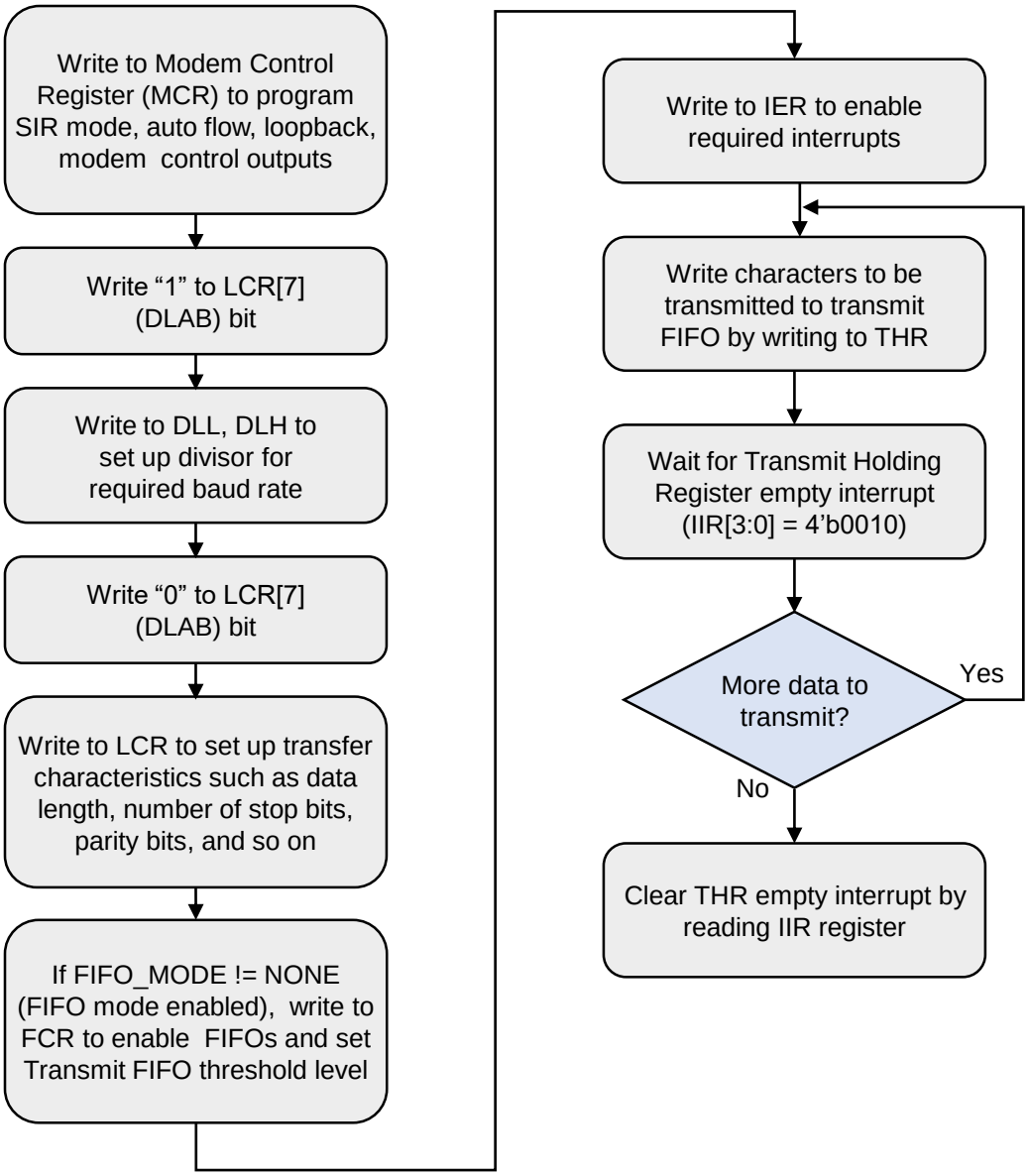


Figure 12-20 Flowchart for uart Transmit Programming Example

12.4.4.2 Receive Flow

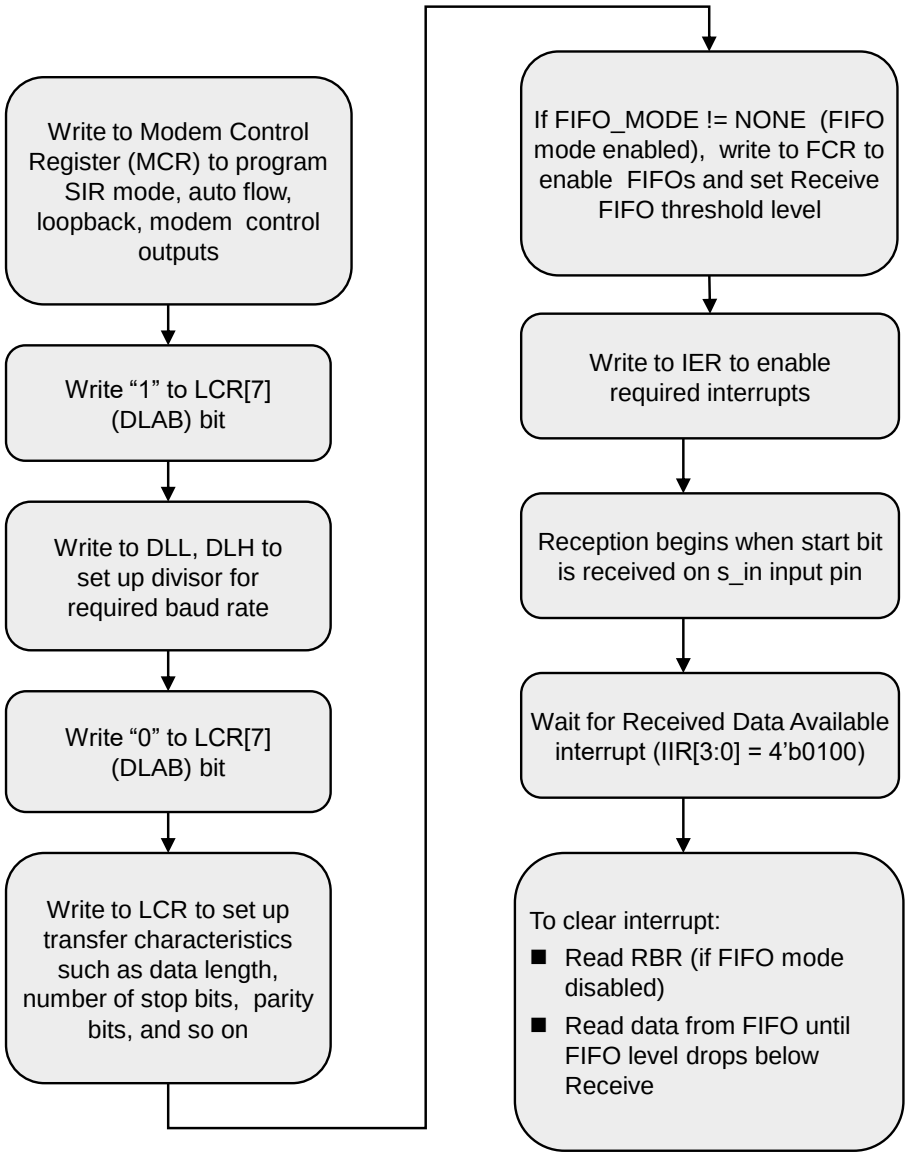


Figure 12-21 Flowchart for uart Receive Programming Example

12.5 SPI

12.5.1 Overview

The following features are supported:

- Support APB4.0, bus data bit width 32
- Configurable and programmable Dual/Quad SPI support in Master Mode
- Support standard SPI transmission protocol
- Support DMA mode
- An high validity interrupt,
- Configurable features:
  - Serial interface operation – Choice of Motorola SPI or Texas Instruments Synchronous Serial Protocol.
  - Clock bit-rate – Dynamic control of the serial bit rate of the data transfer; used in only serial-master mode of operation.

- Data Item size (4 to 32 bits) – Item size of each data transfer under the control of the programmer.

■ FIFO depth 32

12.5.2 Block Diagram

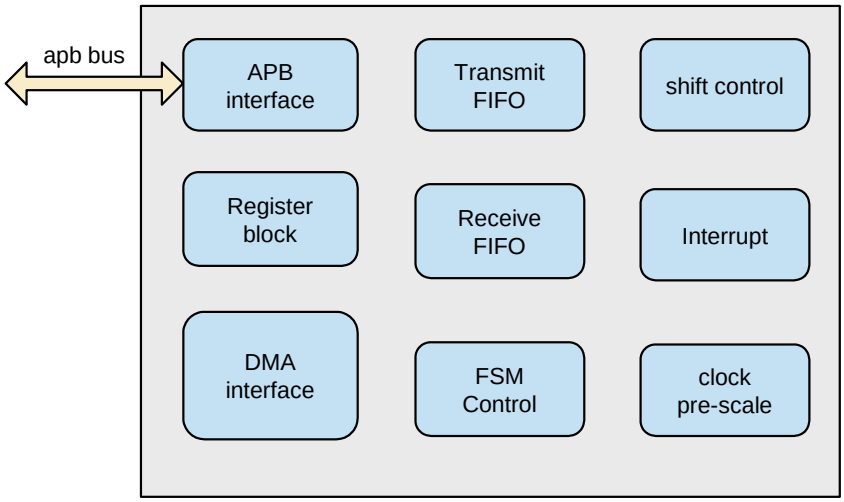


Figure 12-22 SPI block diagram

12.5.3 Function Description

12.5.3.1 Transfer Modes

Transmit and Receive

When TMOD = 2'b00, both transmit and receive logic are valid. The data transfer occurs as normal according to the selected frame format (serial protocol). Transmit data are popped from the transmit FIFO and sent through the txd line to the target device, which replies with data on the rxd line. The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame.

Transmit Only

When TMOD = 2'b01, the receive data are invalid and should not be stored in the receive FIFO. The data transfer occurs as normal, according to the selected frame format (serial protocol). Transmit data are popped from the transmit FIFO and sent through the txd line to the target device, which replies with data on the rxd line. At the end of the data frame, the receive shift register does not load its newly received data into the receive FIFO. The data in the receive shift register is overwritten by the next transfer. You should mask interrupts originating from the receive logic when this mode is entered.

Receive Only

When TMOD = 2'b10, the transmit data are invalid. When configured as a slave, the transmit FIFO is never popped in Receive Only mode. The txd output remains at a constant logic level during the transmission. The data transfer occurs as normal according to the selected frame format (serial protocol). The receive data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame. You should mask interrupts originating from the transmit logic when this mode is entered.

EEPROM Read

When TMOD = 2'b11, the transmit data is used to transmit an opcode and/or an address to the EEPROM device.. Typically this requires three data frames (8-bit opcode followed by an 8-bit upper address and 8-bit lower address). During the transmission of opcodes and addresses, the receive logic does not capture any data (as long as the SPI host transmits data on its txd line, the data on the rxd line is ignored). The SPI master continues to transmit data until the sending FIFO is empty. Therefore, you should only

have enough data frames in the transmit FIFO to provide opcodes and addresses to the EEPROM. If there are more data frames in the sending FIFO than needed, the read data is lost.

When the transmit FIFO becomes empty (all control information has been sent), the data on the receive line (rxd) is valid and stored in the receive FIFO, and the TXD output remains at a constant logic level. the txd output is held at a constant logic level. The serial transfer continues until the number of data frames received by the SPI master matches the value of the NDF field in the CTRLR1 register + 1.

12.5.3.2 Compatible Interfaces

12.5.3.2.1 Motorola Serial Peripheral Interface (SPI)

With the SPI, the clock polarity (SCPOL) configuration parameter determines whether the inactive state of the serial clock is high or low. To transmit data, both SPI peripherals must have identical serial clock phase (SCPH) and clock polarity (SCPOL) values. The data frame can be 4 to 32-bits in length.

When the configuration parameter SCPH = 0, data transmission begins on the falling edge of the slave select signal.。

(1) Serial Format Continuous Transfers (SCPH = 0) when CTRLR0. SSTE = 1

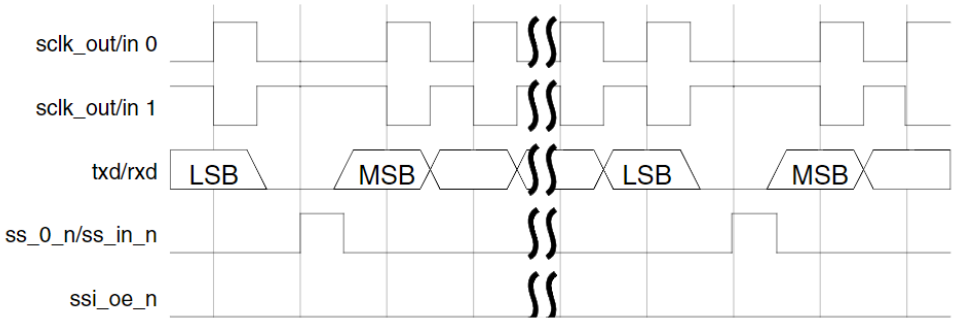


Figure 12-23 Motorola Serial Peripheral Interface(SCPH = 0, SSTE = 1)

(2) Serial Format Continuous Transfers (SCPH=0) when CTRLR0. SSTE = 0

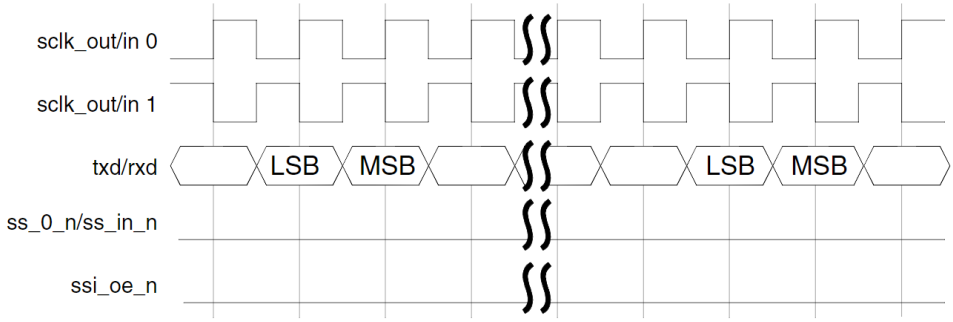


Figure 12-24 Motorola Serial Peripheral Interface(SCPH = 0, SSTE = 0)

When the configuration parameter SCPH = 1, both master and slave peripherals begin transmitting data on the first serial clock edge after the slave select line is activated. The first data bit is captured on the second (trailing) serial clock edge. Data are propagated by the master and slave peripherals on the leading edge of the serial clock. During continuous data frame transfers, the slave select line may be held active-low until the last bit of the last frame has been captured.

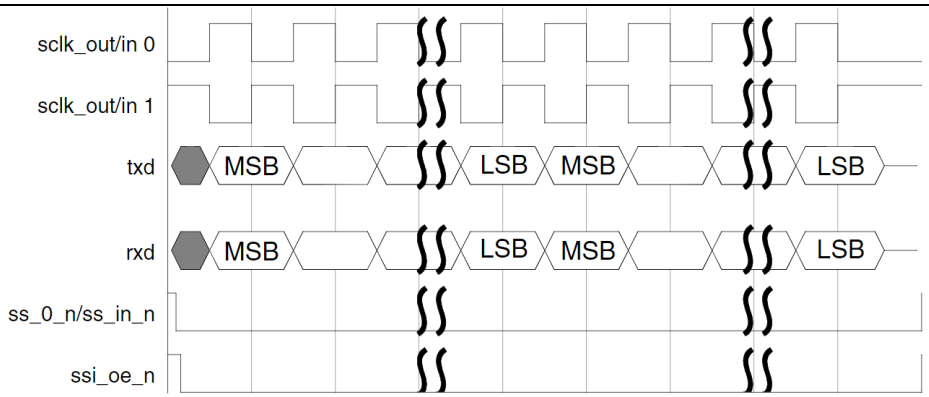


Figure 12-25 SPI Serial Format Continuous Transfer (SCPH = 1)

12.5.3.2.2 National Semiconductor Microwire

When the SPI is configured as a serial master, data transmission begins with the falling edge of the slave-select signal (ss_0_n). One-half serial clock (sclk_out) period later, the first bit of the control is sent out on the txd line. The length of the control word can be in the range 1 to 16 bits and is set by writing bit field CFS (bits 15:12) in CTRLR0. The remainder of the control word is transmitted (propagated on the falling edge of sclk_out) by the SPI serial master. During this transmission, no data are present (high impedance) on the serial master's rxd line.

The direction of the data word is controlled by the MDD bit field (bit 1) in the Microwire Control Register (MWCR). When MDD=0, this indicates that the SPI serial master receives data from the external serial slave. One clock cycle after the LSB of the control word is transmitted, the slave peripheral responds with a dummy 0 bit, followed by the data frame, which can be 4 to 32-bits in length. Data are propagated on the falling edge of the serial clock and captured on the rising edge.

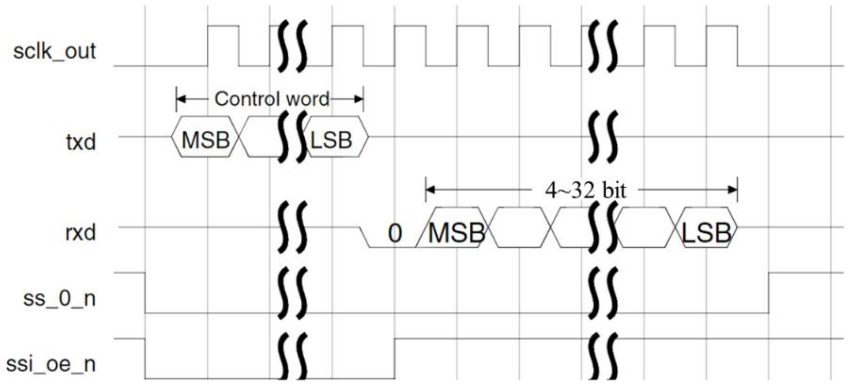


Figure 12-26 Single Master Microwire Serial Transfer (MDD=0)

12.5.3.2.3 Enhanced SPI Modes

You can choose the dual/quadl modes of SPI using the SSI_SPI_MODE configuration parameter. The possible values for this parameter are Standard, Dual SPI, Quad SPI modes. When dual, quad mode is selected for this parameter, the width of txd, rxd and ssi_oe_n signals change to 2, 4, or 8, respectively. Hence, the data is shifted out/in on more than one line, increasing the overall throughput. All four combinations of the serial clock's polarity and phase are valid in this mode and it works same as in normal SPI mode. The mode of operation (write/read) can be selected using the CTRLR0.TMOD field.

The following register fields are used for a write operation:

- CTRLR0.SPI_FRF - Specifies the format in which the transmission happens for the frame.
- SPI_CTRLR0 (Control Register 0 register) – Specifies length of instruction, address, and data.
- SPI_CTRLR0.INST_L – Specifies length of an instruction (possible values for an instruction are 0, 4, 8,
- or 16 bits.)

- SPI_CTRLR0.ADDR_L – Specifies address length (See Table 2-3 for decode values)
- CTRLR0.DFS or CTRLR0.DFS_32 – Specifies data length.

The instruction, address and data can be programmed to send in dual/quad mode, which can be selected from the SPI_CTRLR0.TRANS_TYPE and CTRLR0.SPI_FRF fields. As shown below, Instruction transmitted in standard and address transmitted in Enhanced SPI format.

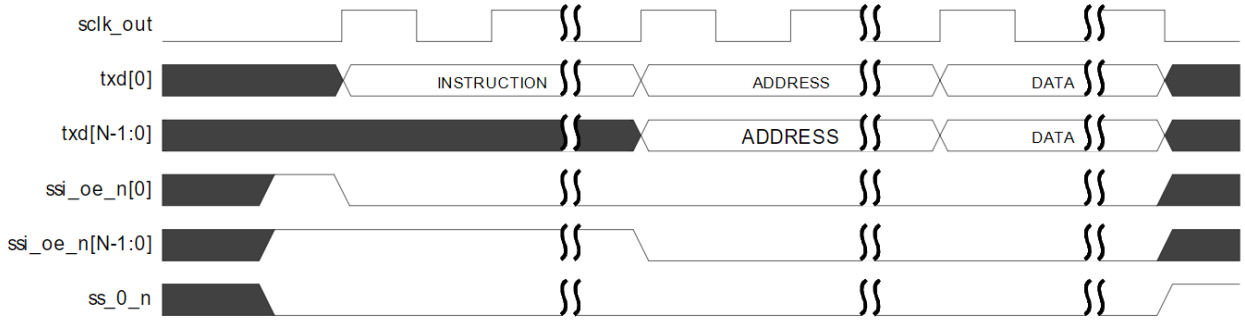


Figure 12-27 Instruction Transmitted in Standard and Address Transmitted in Enhanced SPI Format

For Reading Operation, wait Cycles can be programmed using SPI_CTRLR0.WAIT_CYCLES field. The value programmed into SPI_CTRLR0.WAIT_CYCLES is mapped directly to sclk_out times. For example, WAIT_CYCLES=0 indicates no Wait, WAIT_CYCLES=1, indicates 1 wait cycle and so on. The wait cycles are introduced for target slave to change their mode from input to output and the wait cycles can vary for different devices.

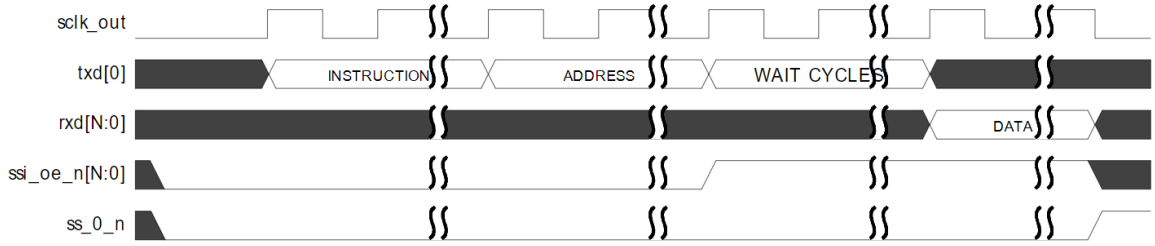


Figure 12-28 Typical Read Operation in Enhanced SPI Mode

12.5.3.2.4 RXD Sample Delay

Due to the data delay, the data from slave may not reach the host within one clock cycle, and the sampling delay needs to be set. Specifically, set the RX_SAMPLE_DLY to increment from 0 until the data is received normally, and record the value n set at this time, and continue to increase the RX_SAMPLE_DLY until the data cannot be received, and record the delay value m set at this time. Then the RX_SAMPLE_DLY is round((m+n-1)/2) which is the optimal delay value that can receive data stably.

12.5.3.3 Interrupts

SPI supports combined interrupt requests, and each interrupt request can be masked. The merged interrupt request is the result of all other SPI interrupts after masking. All SPI interrupts are active-high level interrupts.

The SPI interrupts are described as follows:

- Transmit FIFO Empty Interrupt (ssi_txe_intr) – Set when the transmit FIFO is equal to or below its threshold value and requires service to prevent an under-run. The threshold value, set through a software-programmable register, determines the level of transmit FIFO entries at which an interrupt is generated. This interrupt is cleared by hardware when data are written into the transmit FIFO buffer, bringing it over the threshold level.
- Transmit FIFO Overflow Interrupt (ssi_txo_intr) – Set when an APB access attempts to write into the transmit FIFO after it has been completely filled. When set, data written from the APB is

discarded. This interrupt remains set until you read the transmit FIFO overflow interrupt clear register (TXOICR).

- Receive FIFO Full Interrupt (ssi_rxf_intr) – Set when the receive FIFO is equal to or above its threshold value plus 1 and requires service to prevent an overflow. The threshold value, set through a software-programmable register, determines the level of receive FIFO entries at which an interrupt is generated. This interrupt is cleared by hardware when data are read from the receive FIFO buffer, bringing it below the threshold level.
- Receive FIFO Overflow Interrupt (ssi_rxo_intr) – Set when the receive logic attempts to place data into the receive FIFO after it has been completely filled. When set, newly received data are discarded. This interrupt remains set until you read the receive FIFO overflow interrupt clear register (RXOICR).
- Receive FIFO Underflow Interrupt (ssi_rxu_intr) – Set when an APB access attempts to read from the
- receive FIFO when it is empty. When set, zeros are read back from the receive FIFO. This interrupt remains set until you read the receive FIFO underflow interrupt clear register (RXUICR).

12.5.3.4 SCLK Clock Ratios

Output frequency:

$$F_{sclk\_out} = \frac{F_{ssi\_clk}}{SCKDV}$$

ssi_clk is the apb clock

sclk_out is the output clock, with a maximum support of 50MHz

12.5.4 Programming example

12.5.4.1 Master SPI Transfer Flow

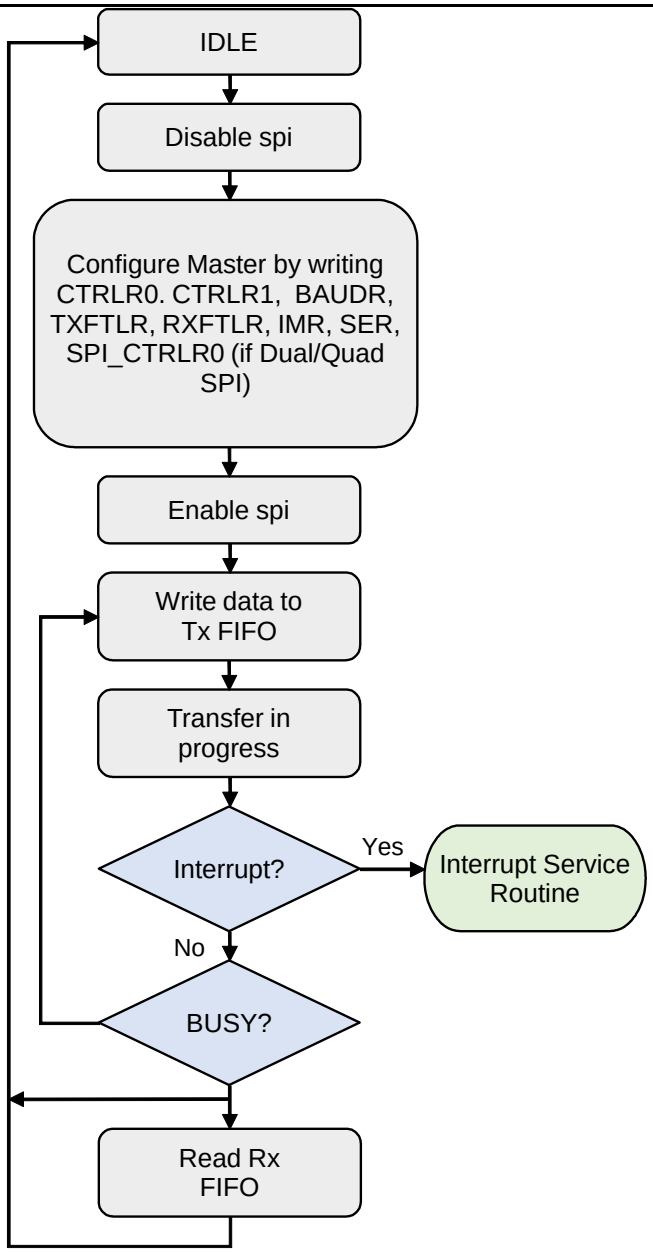


Figure 12-29 Master SPI Transfer Flow

12.5.4.2 Master Microwire Transfer Flow

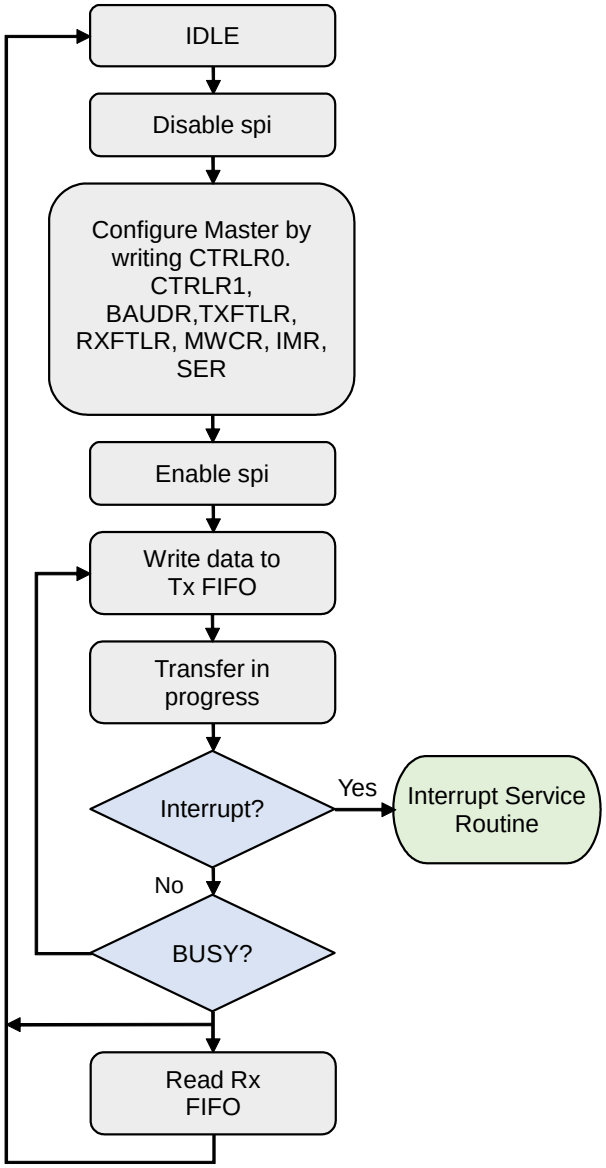


Figure 12-30 Master Microwire Transfer Flow

12.5.5 Timing Sequence

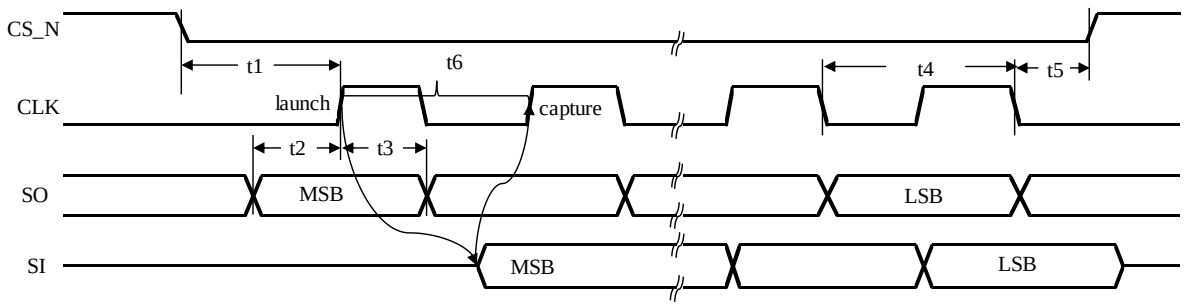


Figure 12-31 SPI timing sequence

12.6 GPIO

12.6.1 Overview

GPIO is a programmable, general-purpose I/O peripheral. This component is an APB slave. The GPIO controls the output data and direction of the external I/O. It can also use memory-mapped registers to read back external data.

The following features are supported:

- 112 independently configurable signals
- A/B/C/D port. This 4 ports map to 112 GPIO
- Each signal has its own data register and data direction register
- Configurable interrupt mode for port A
- Debounce logic can be configured to debounce interrupt the signal
- Configurable synchronization of interrupt signals

12.6.2 Block Diagram

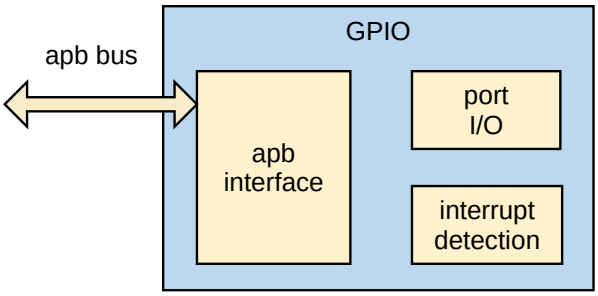


Figure 12-32 GPIO block diagram

12.6.3 Function Description

12.6.3.1 Software Control

The data and direction control of the signal come from the data register (gpio_swportx_dr) and the output enable register (gpio_swportx_ddr), respectively, where x is A, B, C, or D. When used as input, the IE corresponding to the IO must be enable, and the external input value is read out by the register gpio_ext_portx.

12.6.3.2 Interrupt Function(PortA)

GPIO 0~31:

- Active-high and level
- Active-low and level
- Rising edge
- Falling edge

For edge-detected interrupts, you can clear the interrupt by writing a 1 to the gpio_porta_eoi register for the corresponding bit to disable the interrupt.

12.6.3.3 Debounce

When the porta is working in interrupt mode, it can be configured whether or not to debounce to remove any glitch that is less than one cycle of the debounce clock.

12.6.4 Programming example

GPIO Features:

1. Select the corresponding pad
2. Disable PortA interrupt function

3. Assert Output Enable(gpio_swportx_ddr)
4. Configure the level of the output signal(gpio_swportx_dr)

12.7 JTAG

12.7.1 JTAG Chain

There are three groups of JTAG interfaces:

- JTAG0 chain：MCPU LPCPU NPU reserved, no JTAG reset
- JTAG1 chain：SCPU, no JTAG reset
- JTAG2 chain：DSP

JTAG bypass control is controlled by register jtag_chain_ctrl, and 0~4bit controls LPCPU, MCPU, NPU, reserved and DSP respectively。

12.7.2 Timing Sequence

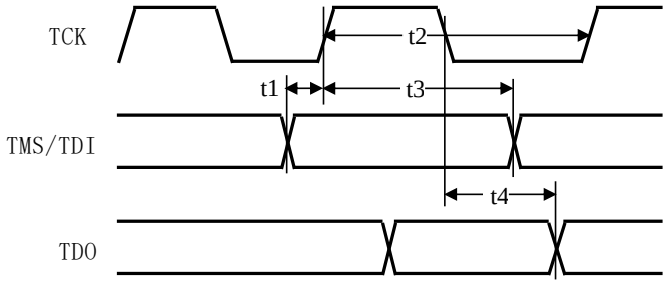


Figure 12-33 JTAG timing sequence

12.8 PWM/TIMER

12.8.1 Overview

TITIME and PWM have the same characteristics. Unless otherwise specified, the feature is both.

The following features are supported:

- APB4.0 bus interface, 32-bit data bit width
- Internally instantiate N programmable timers
  - 3 timers are instantiated inside the PWM module
  - 8 timers are instantiated inside the TIMER module
- Support two modes of operation:
  - Free-running
  - User-defined
- Each timer generates a separate high active interrupt signal
- The duty cycle of the output signal is 0%~100% adjustable
- The TIMER module supports the pause function, but the PWM module does not support the pause function

12.8.2 Block Diagram

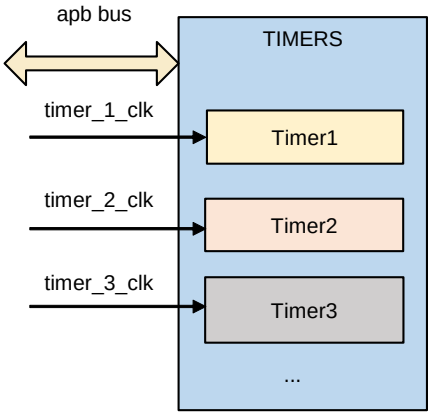


Figure 12-34 PWM/TIMER block diagram

12.8.3 Function Description

12.8.3.1 The Working Mode

When the timer is enabled after being reset or disable, the count value is loaded from the TimerNLoadCount register; this happens in both Free-running and User-defined count modes.

When a timer counts down to 0, it loads one of two values, depending on the timer operating mode:

- User-defined count mode – Timer loads the current value of the TimerNLoadCount register. Use this mode if you want a fixed, timed interrupt. Designate this mode by writing a "1" to bit 1 of TimerNControlReg.
- Free-running mode – Timer loads the maximum value, which is dependent on the timer width; that is, the TimerNLoadCount register is comprised of $2^{TIMER\_WIDTH\_N - 1}$ bits, all of which are loaded with 1s. The timer counter wrapping to its maximum value allows time to reprogram or disable the timer before another interrupt occurs. Use this mode if you want a single timed interrupt. Designate this mode by writing a "0" to bit 1 of TimerNControlReg.

12.8.3.2 Duty Cycle Calculation

Each timer can be be pulse-width modulated, when register bit TimerNControlReg[4] (TIMER_PWM bit) is set to 1, the HIGH and LOW periods of the toggle outputs can be controlled separately by programming the TimerNLoadCount2 and TimerNLoadCount registers.

The pulse widths of the toggle outputs are controlled as follows:

- Width of timer_N_toggle HIGH period = (TimerNLoadCount2 + 1) * timer_N_clk clock period
- Width of timer_N_toggle LOW period = (TimerNLoadCount + 1) * timer_N_clk clock period

12.8.3.3 0%~100% Duty Cycle

PWM/TIMER supports the programming for 0% and 100% duty cycle pulse width modulation of toggle outputs (timer_N_toggle) through the TimerNLoadCount and TimerNLoadCount2 registers, when 0% and 100% duty cycle mode is enable. You can enable the duty cycle mode either by setting the TimerNControlReg [4] register.

- 0% duty cycle
  - TimerNLoadCount = Do not care
  - TimerNLoadCount2 = 0
- 100% duty cycle
  - TimerNLoadCount = 0
  - TimerNLoadCount2 = Do not care
- Other duty cycle – When 0% and 100% duty cycle mode is enabled (with timer PWM mode and the user-defined count mode is enabled), the definition of the toggle high and low period changes as follows for a duty cycle other than 0% or 100%:

- Width of timer_N_toggle HIGH period = TimerNLoadCount2 * timer_N_clk clock period
- Width of timer_N_toggle LOW period = TimerNLoadCount * timer_N_clk clock period

12.8.3.4 Interrupt

When the counter reaches 0, an interrupt is generated. Interrupts can be cleared and masked by register.

12.8.3.5 Pause Function (Supported By TIMER Only)

The TIMER pause function is controlled by the system control register timerN_pause, 0~7bit controls the 1-8 internal timer of TIMERN respectively.

12.8.4 Input Clock

Timer8 of TIMER3 is controlled by a timer3_clk8 clock； The timerN_clk of TIMER0, TIMER1, TIMER2, and TIMER3 (except timer8) are gated by timer_clk[N], and the timerN_clk of PWM is the same as timer_pclk.

12.8.5 Programming Example

You can use the following programming flow to enable the 0% and 100% duty cycle mode:

1. Disable the timer enable bit in the TimerNControlReg register.
2. Program the TimerNLoadCount and TimerNLoadCount2 registers with appropriate values.
3. Enable the 0% and 100% duty cycle mode bit, the Pulse width modulation bit and set the Timer mode to user-defined count mode in the TimerNControlReg register.
4. Set the timer enable bit in the TimerNControlReg register such that the toggle output is 100% (high) or 0% (low).

When the 0% and 100% duty cycle mode is enabled, internal timer is disabled. The internal timers can be enabled again by switching to Normal toggle output mode or to Pulse width modulation toggle output mode.

12.9 FAN_CTRL

12.9.1 Overview

FAN_CTRL for fan speed monitoring, supporting:

- APB3.0 bus, data bit width 32bit
- Monitor external fan speed

12.9.2 Programming Example

The FAN_CTRL operation process is as follows:

1. Set the REG_FAN_INT [0] value to 32'h1 to enable speed detection
2. Wait for the REG_FAN_INT[1] interrupt
3. Read the REG_FAN_RPM, configurate REG_FAN_INT [2] to 1, and clear the interrupt
4. Repeat the process for the next read