FLC Memory Expander Hardware uArch Specifications

Revision 1.60
7/29/2024

FLC TECHNOLOGY GROUP | 4655 Old Ironsides Dr Suite 490, Santa Clara, CA 95054

Table of Contents

Chapter 1: Feature list

1.1 FLC, Memory Controller and In-Package Memory (IPM)

- Fully Associative Look-up Engine for Cache
- 2 (or More) Levels

1.2 FLC (Final Level Cache)

- Gigantic Entries: (ex) Millions
- Cache Line: (ex) 4KB
- # of Entries: 32K per 1Gb
- Very High (~99%) Hit Rate

1.3 Cache & System Memory

1. 1.3.1 In-Package Memory (IPM)
    - 1Gb density
    - 2CH x 64b @2133Mbps BW 32GB/s
    - Access time ~ 33ns
    - Access power ~2pJ/bit
    - Support parallel ECC 128 + 8 SECDED
- 1.3.2 DDR4
    - 1CH x 64b @3200Mbs BW 25.6GB/s
    - Support parallel ECC 64 + 8 SECDED

1.4 High-end Memory Controller

RAS (Reliability, Availability, Serviceability):

- End-to-end protection
    - Accepts error sideband signal form upstream masters and report error for master
    - Parity protected SRAMs
- Supports ECC for all supported DDR bus widths
    - SECDED
    - Error logging for correctable and non-correctable errors
    - Error reporting interrupt
- Row Hammer Protection
- Memory scrubbing
    - Automatically scans system memory to correct errors and report uncorrectable errors
    - Programmable auto timer between scrubbing or software-triggered scrubber with programmable start and end system address

Performance:

- Sophisticated DRAM scheduling
- Large write buffer with built-in heuristics
- Agile E2E Quality of Service arbitration with improved latency and bandwidth for critical transactions
- Bandwidth regulators among multiple masters
- Bandwidth monitors
- Read data forwarding

- Write coalescing

Debug:

- Error poisoning
  - 1-bit or 2-bit ECC errors injection
  - CRC data error injection
- DRAM status monitoring
- Memory controller status monitoring

DDR Repair:

- Supports DDR4 post-package repair (hPPR and sPPR)
- Shadow memory repair

Performance and Activity Monitoring:

- Event-based performance monitors
  - AXI transactions
  - Empty/full status for all internal FIFO/buffer/pools
  - DDR commands counter
- Fully user-defined start and end conditions-based monitoring
- Critical visibility to AXI ports

Power Saving:

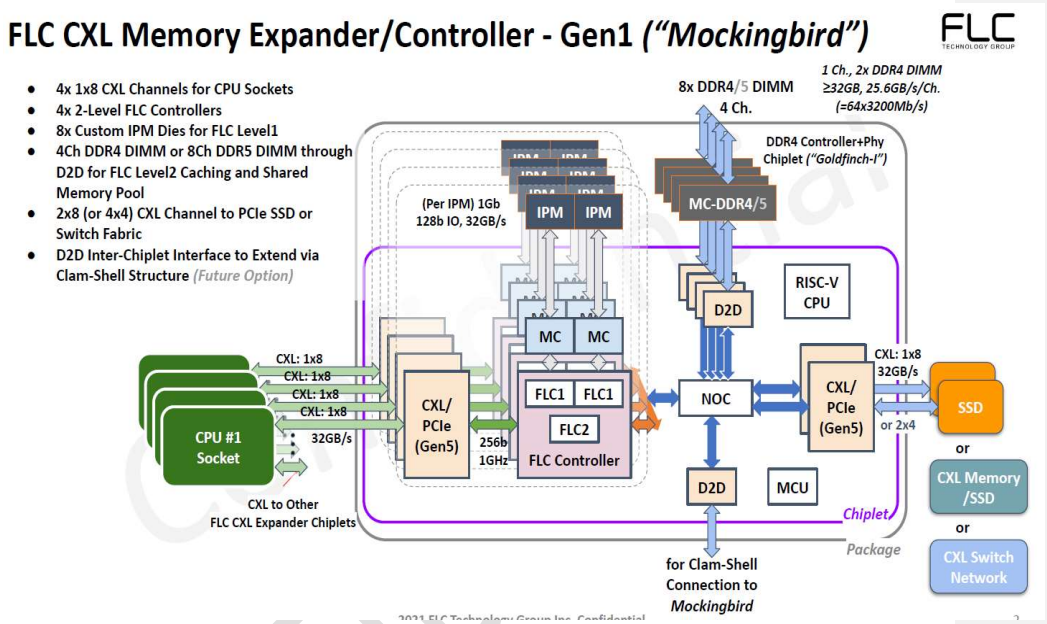
- Auto power saving control to place DRAM into self-refresh or when no activity detected for extended timeframe
- Aggressive clock gating signals PMU to partially or fully gate off MC clock
- Supports firmware or software based Dynamic Frequency Change (DFC) to adjust speed of the memory controller and DRAM in order to balance between performance and power consumption

uC Subsystem Components:

- Processor: SiFive E21
- Processor speed: 200MHz
- Operating mode: Machine mode (privilege mode), User mode
- Physical memory protection: 4 regions (configurable up to 16). PMP is equivalent to ARM's PMU
- Debug interface: JTAG
- Peripheral bus: AHB and APB for embedded peripherals/memory/CSR. Interface to NOC/FLC/CXL
- Interrupt: CLIC (63 interrupt sources).
- Timer: Local Timer (x2) and processor embedded Timer. Watchdog Timer
- SRAM (TCM): 256KB, 2 banks (2x128KB)
- Mask ROM: 32KB
- XIP NOR serial flash: x4 data bus with SRAM cache
- OTP
- UART (x1): console terminal and firmware download
- I2C (x1)
- SPI (x2): master and device modes

- GPIO: 20 pins.
- DMA: 8 channels.
- Security Engine: AES and SHA.
- Run-time temperature and performance monitoring.

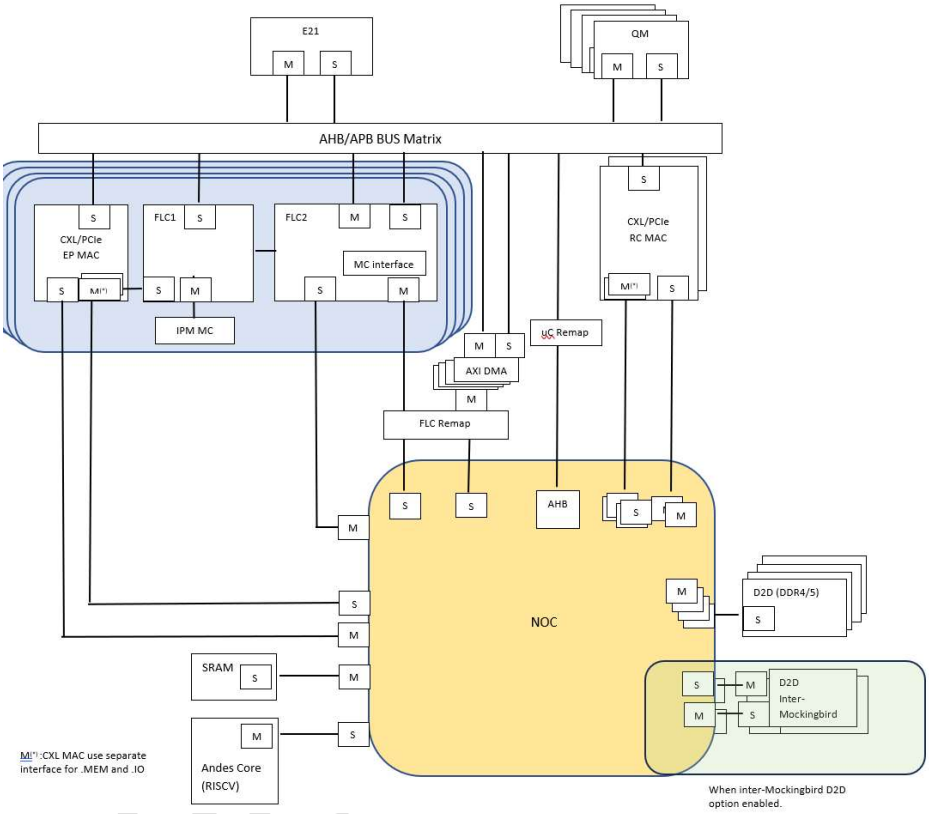
Chapter 2: Block Diagram



MKB 1x16 D2D

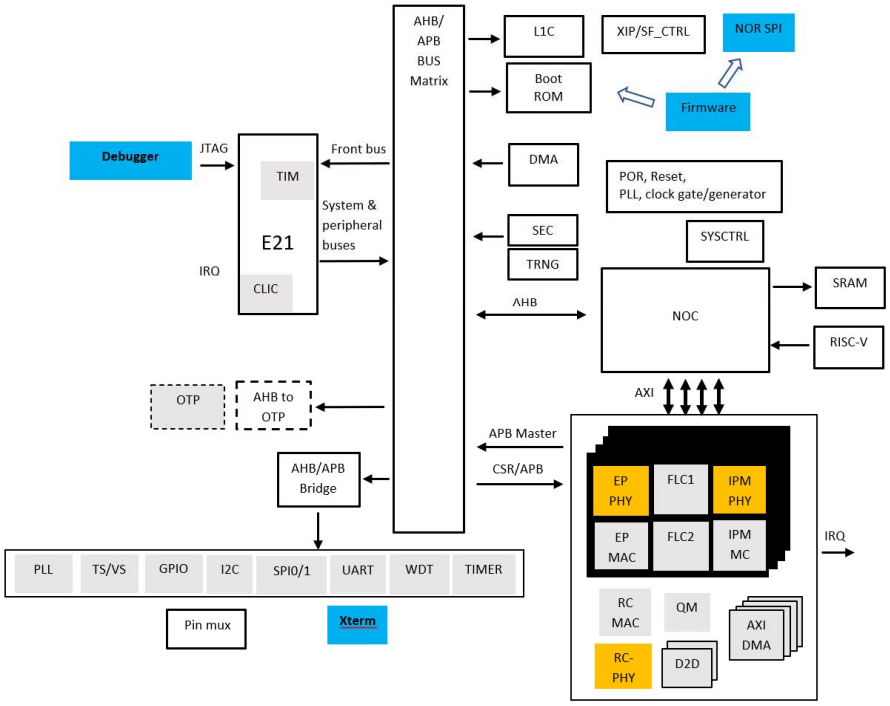
	D2D	Orientation	NOC interface: AXI target	NOC interface: AXI initiator	Peak BW
GFH	4x(1x16)		4	0	4x32GB/s
Inter-Chiplet	2x(1x16)		2	2	2x32GB/s
Total	6(1x16)		6	2	192GB/s

2.3 NOC and AHB/APB bus Matrix

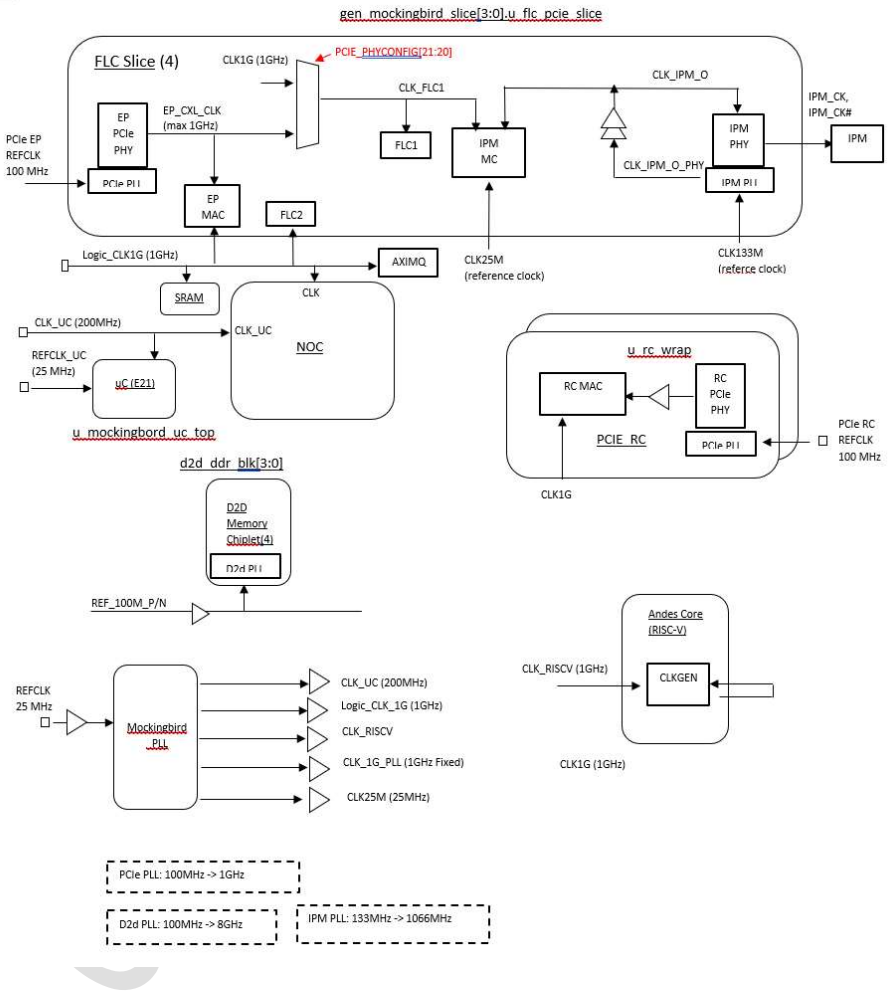


2.4 uC Subsystem

uC buses and modules



2.5 Clock Domain



FLC slice clocking table

uC CSR: 0x2600003c. PCIE_phyconfig[21:20]:

FLC1_axi_slv_config	00	RW	00 = CXL.mem -> flc1 01 = CXL.io -> flc1 1x = Reserved.

When FLC1_axi_slv_config == 2'b00

Clock source selection			
lpm_mc_axi_p0_sync	flc1_axi_slave_sync	flc1_axi_slave	flc1
x	1	ep_cxl_clk	ep_cxl_clk

ep_cxl_clk: PCIe PHY generated clock. Frequency varies based on mode negotiation.

When FLC1_axi_slv_config != 2'b00

Clock source selection			
lpm_mc_axi_p0_sync	flc1_axi_slave_sync	flc1_axi_slave	flc1
x	1	clk_nic (clk1g)	clk_nic (clk1g)

SOC clock table

Name	Working Frequency	Source	Comment
clk1g (clk1g_slice[3:0])	1GHz	Mockingbird PLL	Main clock for each slice.
clk1g_noc, clk1g_noc_lp	1GHz	Mockingbird PLL	NOC clock
clk_riscv	1Ghz	Mockingbird PLL	Andes CPU
clk1g_axi4tg	1Ghz	Mockingbird PLL	AXI4 traffic gen
clk_uc	200MHz	Mockingbird PLL	uC subsystem
clk_ipm	1066MHz	IPM PHY	
clk_flc1_slave	Up to 1Ghz	Mockingbird PLL or PCIe EP PHY	FLC1 slave port
clk_flc1	1Ghz	Mockingbird PLL	FLC1 core clock
clk_flc2	1Ghz	Mockingbird PLL	FLC2 core clock
clk_rc_x8[1:0]	Up to 1Ghz	RC x8 PHY	
clk_rc_x4[1:0]	Up to 1Ghz	RC x4 PHY	
ref_100m_p[1:0], ref_100m_n[1:0]	100MHz	External	100MHz differential clock for D2D PLL
ref_ipm[3:0]	133.33MHz	External	IPM PLL reference clock
ep_cxl_ref_p[3:0], ep_cxl_ref_n[3:0]	100MHz	External	PCIe/CXL endpoint PHY reference clock
rc_cxl_ref_p[1:0], rc_cxl_ref_n[1:0]	100MHz	External	PCIe/CXL RC PHY reference clock
clk25m	25MHz	External	

External clock sources:

Frequency	Part Number	comment
25MHz	ASD3-25.000MHZ-LR-T	±25ppm
133.33MHz	SIT8009BI-13-18E-133.333333	±50ppm
100MHz	Si52204-A01BGMR	0.085ps rms, PCIe Gen 6.

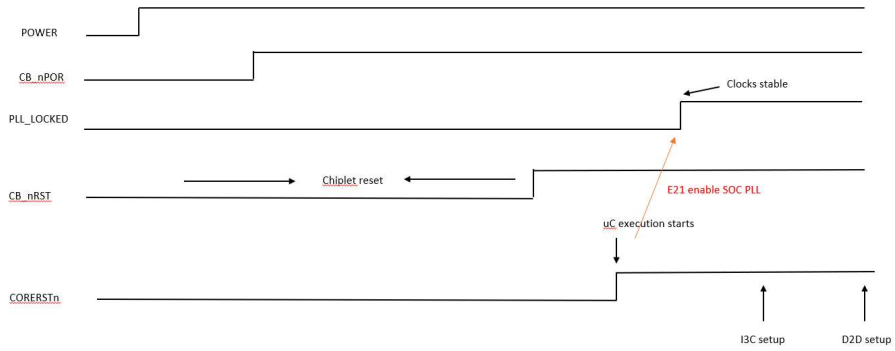
Reset and power up

- Power applied. system clock stable.
- E21 boot.
- PHY (IPM, PCIe, DDR, D2D) FW download.
- PHY (IPM, PCIe, DDR, D2D) init.
- PHY output clock stable (IPM, D2D, PCIe clock).
- I3C, D2D LL sync
- Functional unit init.
    - PCIe MAC
    - IPM, DDR MC
    - FLC

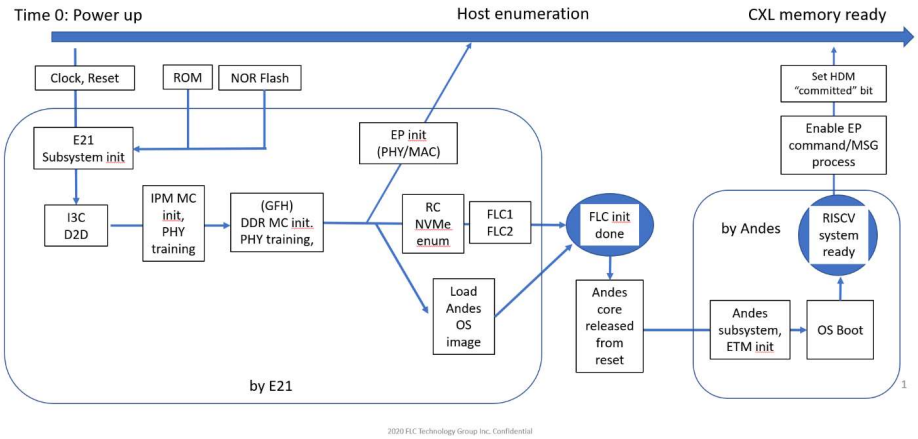
- RISC-V system bring up
  - Code download
  - ETM

Chip Reset and clock ports

- CB_nPOR: Active low, power on reset. Reset PLL and clock circuitry.
- CB_nRST: Active low, global chip reset.
- CORERSTn: Active low, reset embedded uC.
- External reset chip required to detect stable voltage level and reset timing.
- OSC: 25MHz single ended reference clock for SOC PLL
- Ref_100m_p/n[4:0] 100MHz reference (CML/differential) for D2D high speed PLL
- Ref_ipm[4:0] 133.33Mhz reference single end clock for IPM PHY



MKB start up timeline (CXL.mem mode)



Chapter 3: Chiplet Modules

3.1 FLC

FLC handles hierarchy memory with FLC controller between each tier is illustrated below. In the chiplet, conventional memory is expanding into a 3-tier hierarchy with IPM serves as the Level 1 cache, DDR serves as the Level 2 cache and main memory resides in NVMe (NAND)

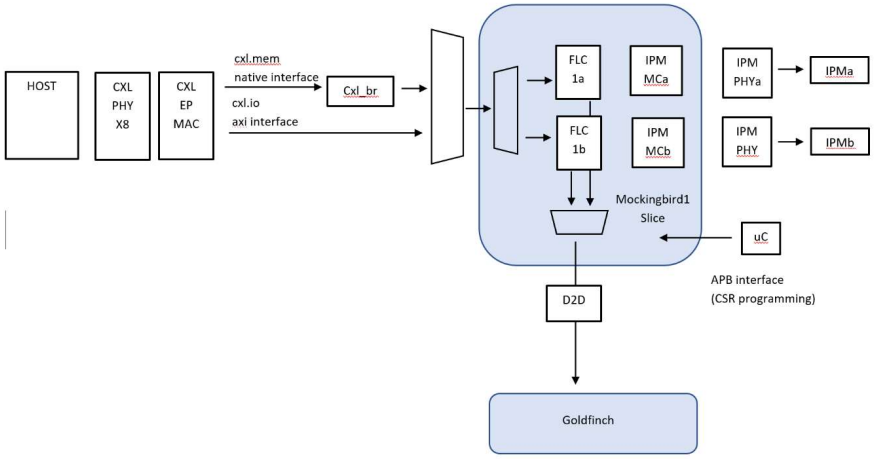
FLC1 controller manages data between IPM and DDR. FLC2 controller manages data between DDR, NAND, or remote memories (CXL, PCIe, NVMe).

uC programs module CSRs within FLC during initialization.

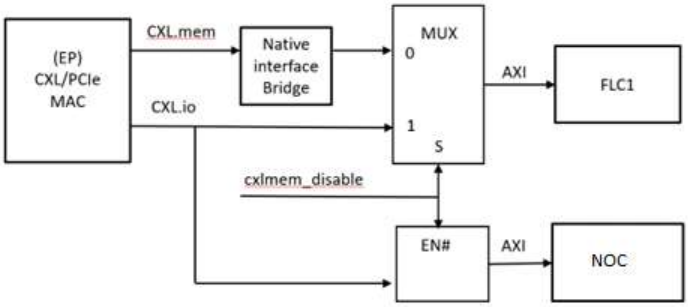
FLC2 issues AXI cycle to access DDR, NAND or remote through NOC.

FLC1 controller, FLC2 controller, IPM memory controller and LP4X memory controller are on the same Mockingbird slice. IPM will be on a different die but same package. Both DDR and NAND are off-chip.

There are total of 4 Mockingbird slices per generation 1 chiplet.



End Point: CXL.mem CXL.io access mux to FLC1
CSR bit "cxlmem_disable=1" enables CXL.io traffic to FLC1



3.3 uC subsystem

uC subsystem design is based on SiFive E21 RISCV core and Bouffalo Lab peripheral IPs. Firmware is stored in on-chip mask ROM and off-chip XIP NOR flash. An AHB/APB bus matrix is used to connect CPU, memory and IO. uC subsystem also interface with chiplet's NIC via AHB to access chiplet resources (SRAM, DDR).

After reset, uC performs system initialization through AHB/APB bridge (refer to section 4.2 "uC address map"). There are 2 local timers, watchdog timer and core embedded timer for system timing control. Interrupt via E21's CLIC module with total of 63 maskable interrupt sources and 1 non-maskable NMI. (Refer to chapter 5 "uC IRQ map")

Two on-core tightly couple memories (TIM), 128KB each are included for fast execution.

3.3.1 Embedded IO summary

Name	Mode	FIFO	DMA	Interrupt	Comment
UART	Slave	32	Yes	Yes	Interface for debug and firmware download.
I2C	Master	2	Yes	Yes	
SPI0	Master/Device	4	Yes	Yes	
SPI1	Master/Device	4	Yes	Yes	
TIMER	Slave			Yes	2 programmable timers, 1 watchdog timer
GPIO	Slave	NA		Yes	20 general purpose IO pins with configurable driving mode and pull up/down. Refer to section 6.1 "uC pin multiples" and section 7.2 "GPIO Control Register".
DMA				Yes	
SEC	AHB master			Yes	Security engine supports hardware-based AES and SHA,
OTP	Slave	NA		Yes	Only in MKB chiplet

3.3.2 AHB/APB bus matrix

The uC bus matrix supports 10 AHB masters and 4 AHB slaves with hardcoded address range. There are also subsection decoders for individual module's CSR. In order to program multiple FLC slices IPM and PCIE PHY, "phy_sel" register in system control (0x26000038) needs to be set with target unit ID. Refer to section 7.2 for details.

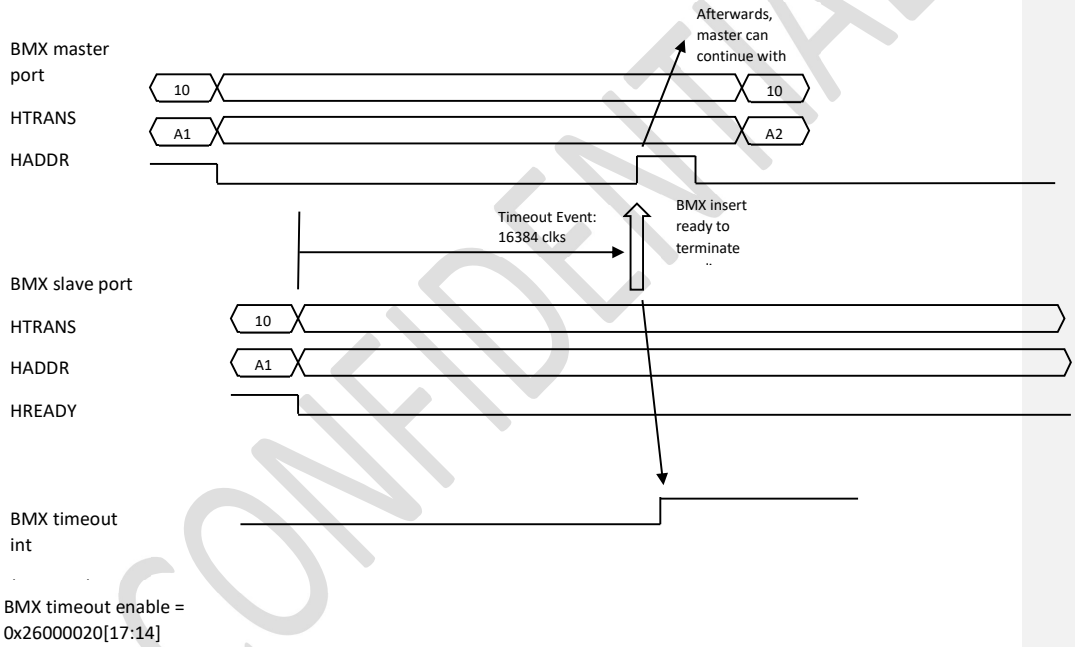
Matrix masters

	Name	Comment
0	uC system bus	uC (E21) system port for code fetch
1	FLC APB master2	APB master from (EP) PCIe ELBI
2	DMA	Connect to DMA engine's AHB master port
3	EMAC	EMAC AHB master
4	CONFIG_SPI	(RTL simulation only) SPI BFM port
5	uC peripheral bus 1	uC (E21) peripheral port 1
6	uC peripheral bus 2	uC (E21) peripheral port 2
7	AES	Security engine (AES) master port
8	SHA	Security engine (SHA) master port
9	FLC APB master	FLC APB master for hardware based NVMe queue management

Matrix Slaves

	Name	Comment
0	Mem	Memory region, mask ROM and XIP NOR.
1	Peripheral	Peripheral region, includes uC embedded IO and FLC, PCIe controller CSRs.
2	NOC	NOC AHB slave port for DDR, NOC-SRAM access.
3	TIM	uC embedded memory (SRAM).

Bus timeout



BMX timeout enable = 0x26000020[17:14]

"PHY_sel" register: Select PHY to configure.

	Name	Comment
3:0	Pcie_PHY_sel	0=RCx8, 1=host cluster 1, 2=host cluster 2, 3=host cluster 3, 4=host cluster 4. *5=RCx4*, 6=RC1x8, 7=RC1x4, F=Broadcast. Note: in Mockingbird, value of 0,1,2,3,4,5,f is valid.
7:4	IPM_PHY_sel	{0,1}=1st FLC slice IPM, {2,3}=2nd FLC slice IPM, {4,5}=3rd FLC slice IPM, {6,7}=4th FLC slice IPM, 0xF=Broadcast. Select the corresponding PHY for firmware download and MCU control. Note: in Mockingbird, value of 0,1,2,3,4,5,6,7,f is valid.
11:8	D2d_PHY_sel	4=MKB D2D0, 5=MKB_D2D1. 0..3=DDR chiplet (1st 1x16). F=Broadcast. Note: in Mockingbird, value of 0,1,2,3,4,5 f is valid.

Steps to access PCIe/D2D PHY CSR:

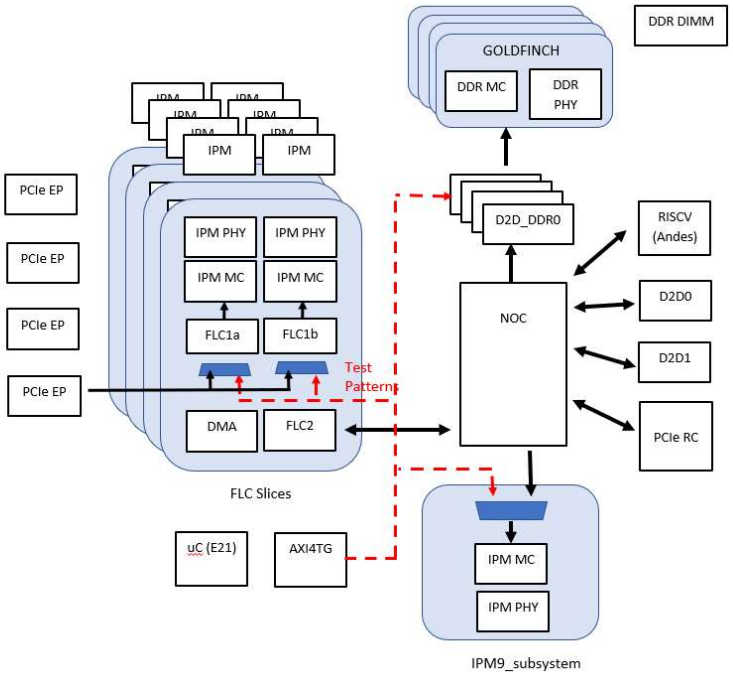
1. Set PHY_sel register (0x26000038) to enable one of the PCIe or D2D PHY unit.
2. Access unit CSR based on uC address map.

IPM AXI4TG

Build in AXI4 traffic generator to test IPM, bypass FLC1.

APB sequence:

1. E21 init IPM PHY, MC
2. E21 programs AXI4TG module.
3. E21 selects one of the IPM to test via CSR. There are total of 9 IPMs in Mockingbird.
4. E21 sets FLC1_MC_TOP's AXI MUX to select path from AXI4TG AXI.
5. E21 set CSR bit to start generating traffic at IPM MC's AXI interface.
6. After test done, E21 access AXI4TG to check result.

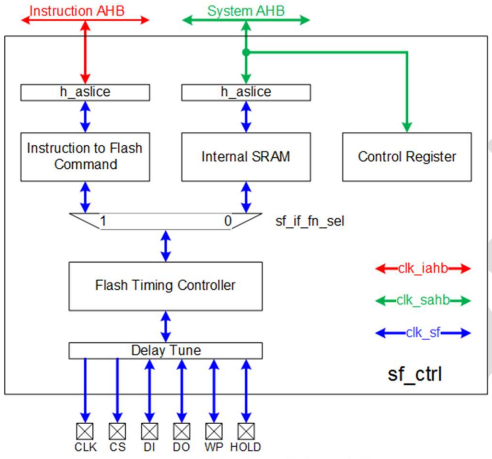


SF CTRL

- Execute-In-Place (XIP) for MCU application
- Configurable flash controller supports all brands of QSPI Flash
- Support Max 256MB Flash capacity
- Support Max 133MHz Flash speed (or faster according to timing constraint)
- The Flash Controller is a QSPI timing controller. It converts data to QSPI protocol in the following two modes:

    1. Indirect mode: read / erase / program flash from system bus. The Flash controller execute the sram data to / from QSPI interface.

    2. XIP mode: read flash data from instruction bus. Once the Flash controller received the target address from instruction bus, it execute read command immediately and read back the data from QSPI flash.



SPI

Overview

Serial Peripheral Interface (SPI) Bus is a synchronous serial communication interface specification for short-range communication. Devices communicate in the full duplex mode, which is a master-slave mode where one master controls one or more slaves.

SPI completes full-duplex communication with 4 signal lines, namely CS (chip select), SCLK (clock), MOSI (master output slave input), and MISO (master input slave output).

Features

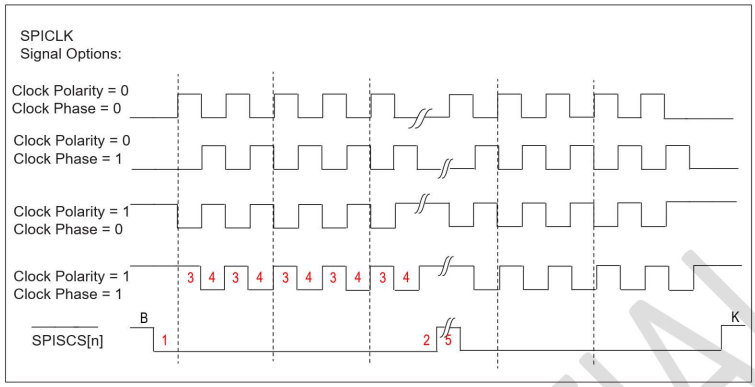
- Can be used as SPI master or slave
- Both master and slave support 4 operating modes (CPOL, CPHA)
- Both master and slave support 1/2/3/4-byte transfer mode
- The sending and receiving channels each have a FIFO with a depth of 32 bytes
- The adaptive FIFO depth variation characteristic suits high-performance applications
    - When the Frame is 32Bits, the depth of FIFO is 8
    - When the Frame is 24Bits, the depth of FIFO is 8
    - When the Frame is 16Bits, the depth of FIFO is 16
    - When the Frame is 8Bits, the depth of FIFO is 32
- Adjustable byte transfer sequence
- Flexible clock configuration
- Configurable MSB/LSB transfer priority
- Receive ignore function: You can set to ignore the reception of data from the specified location
- Supports timeout mechanism in the slave mode
- Supports DMA transfer mode

Functional Description

Clock Control

Due to different clock phases and polarity settings, the SPI clock has four modes, which can be set by cr_spi_sclk_pol (CPOL) and cr_spi_sclk_ph (CPHA) in the register spi_config. CPOL determines the level of SCK clock signal when it is idle. If CPOL=0(cr_spi_sclk_pol=0), the idle level is low, and if CPOL=1(cr_spi_sclk_pol=1), the idle level is high. CPHA determines the sampling time. If CPHA=0(cr_spi_sclk_ph=1), sampling is made at the first lock edge of each cycle, and if CPHA=1(cr_spi_sclk_ph=0), that is made at the second clock edge of each cycle.

By setting the registers spi_prd_0 and spi_prd_1, you can also adjust the duration of the start and end levels of the clock, time of phase 0/1, and interval between frames of data. The settings of the four modes are shown as follows:



SPI Timing

The meanings of numbers are as follows:

- "1" denotes the length of the START condition, which is configured by the cr_spi_prd_s in the register spi_prd_0.
- "2" denotes the length of the STOP condition, which is configured by the cr_spi_prd_p in the register spi_prd_0.
- "3" denotes the length of phase 0, which is configured by the cr_spi_prd_d_ph_0 in the register spi_prd_0.
- "4" denotes the length of phase 1, which is configured by the cr_spi_prd_d_ph_1 in the register spi_prd_0.
- "5" denotes the interval between frames of data, which is configured by the cr_spi_prd_i in the register spi_prd_1.

Master Continuous Transfer Mode

After this mode is enabled, the CS signal will not be released when the current data is sent and there are still available data in FIFO.

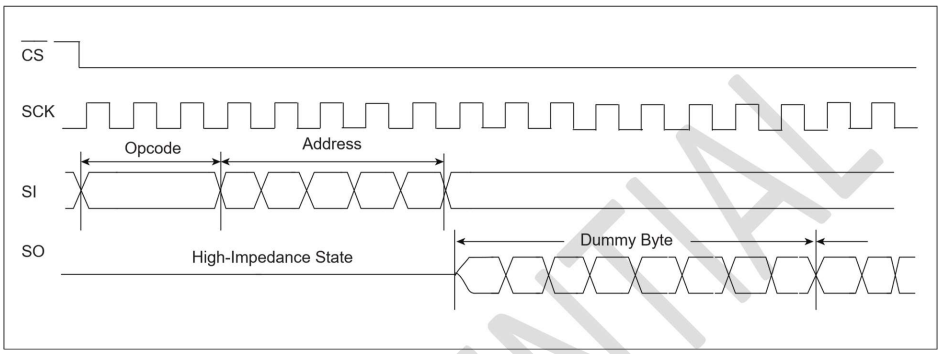
Master-Slave: Transfer and Receive Data

The same framesize shall be set for the master and slave for transferring and receiving data by configuring the cr_spi_frame_size in the register spi_config. When the master and slave agree to communicate at a 32 bits framesize, if the clk of the master does not meet 32 bits due to an exception in a frame of data, the following symptoms occur.

- The data sent by the master cannot be transferred to the RX FIFO of the slave. The slave cannot receive data from the master.
- When the slave sends data, it will skip this frame of data and continue to send the next frame of data when the master's clk is normal again.

Receive Ignore Function

When the start and end bits to be filtered out are set, SPI will discard the corresponding data segments in the received data, as shown below:



SPI Ignore Waveform

You can enable this function by configuring the cr_spi_rxd_ignr_en in the register spi_config. The start bit of this function is set by configuring the cr_spi_rxd_ignr_s in the register spi_rxd_ignr. The end bit of this function is set by configuring the cr_spi_rxd_ignr_p in the register spi_rxd_ignr.

In the above figure, the start bit to be filtered is set to 0, and if the end bit is set to 7, Dummy Byte will be received; if the end bit is set to 15, Dummy Byte will be discarded.

Filtering Function

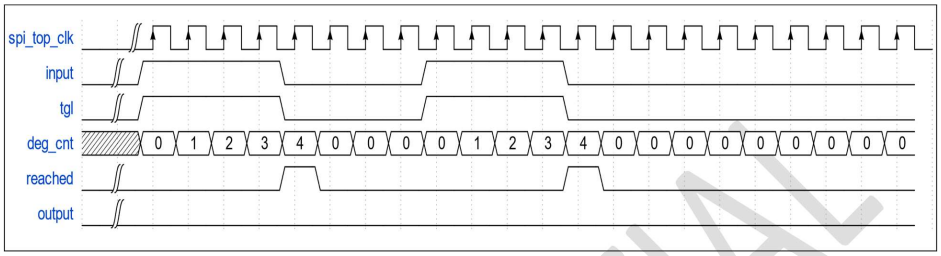
When this function is enabled and a threshold is set, SPI will filter the data less than or equal to the width threshold.Assuming that the SPI top clock is 160MHz and the threshold is set to 4, any data with a width below (4/160MHz = 25ns) will be filtered out.

When this function is enabled by setting the cr_spi_deg_en in the register spi_config and the threshold is set by configuring the cr_spi_deg_cnt, SPI will filter out the data that cannot reach the width threshold. As shown in the figure below, the data width is 4, "Input" is the initial data and "output" is the filtered data.

Filtering logic process:

- "Tgl" is the exclusive OR result of input and output.
- "Deg_cnt" counts from 0, and the counting condition is that "tgl" is at the high level and "reached" is at the low level.
- "Reached" means whether the current deg_cnt count reaches the set cr_spi_deg_cnt, and it is at a high level once reached.
- When "reached" is at a high level, "input" is output to "output".
- Note: user-defined condition for deg_cnt: "tgl" is at a high level and "reached" is at a low level. In other cases, "deg_cnt" will be cleared to 0.

SPI Filter Waveform



Configurable MSB/LSB Transfer

The configurable MSB/LSB transfer mode is limited to the priority transfer sequence of 8 bits in one byte, and the transfer sequence of bits in one byte is set by configuring the cr_spi_bit_inv bit in the register spi_config. 0 indicates MSB and 1 indicates LSB.

For example, for data transfer where the frame size is 24 bits, the data format is Data[23:0]=0×123456.

When MSB transfer is set, the transfer sequence is: 01010110 (binary, 1st byte: 0×56); 00110100 (binary, 2nd byte: 0×34); 00010010 (binary, 3rd byte: 0×12). When LSB transfer is set, the transfer sequence is: 01101010 (binary, 1st byte: 0×56); 00101100 (binary, 2nd byte: 0×34); 01001000 (binary, 3rd byte: 0×12).

Adjustable Byte Transfer Sequence

The adjustable byte transfer sequence is limited to the priority transfer sequence between different bytes in FIFO. The transfer sequence of bytes in FIFO is set by configuring the cr_spi_byte_inv bit in the register spi_config. 0 means sending LSB first, and 1 means sending MSB first.

For example, for data transfer where the frame size is 24 bits, the data format is Data[23:0]=0×123456.

When LSB transfer priority is set, the transfer sequence is 0×56 (1st byte: LSB); 0×34 (2nd byte: intermediate byte); 0×12 (3rd byte: MSB). When MSB transfer priority is set, the transfer sequence is 0×12 (3rd byte: MSB); 0×34 (2nd byte: intermediate byte); 0×56 (1st byte: LSB).

Adjustable byte transfer can be used in conjunction with configurable MSB/LSB transfer.

Slave Mode Timeout Mechanism

When a timeout threshold is set, an interrupt will be triggered when SPI in the slave mode receives no clock signal after the threshold exceeds.

I/O Transfer Mode

The chip communication processor can perform FIFO padding and clearing operations in response to the interrupt from the FIFO. Each FIFO has a programmable FIFO trigger threshold to trigger an interrupt. When rx_fifo_cnt in the register

spi_fifo_config_1 is greater than the trigger threshold of rx_fifo_th, an interrupt will be generated to send a signal to the chip communication processor to clear the RX FIFO. When rx_fifo_cnt in the register spi_fifo_config_1 is greater than rx_fifo_th, an interrupt will be generated to send a signal to the chip communication processor to re-pad the TX FIFO.

You can query the SPI status register to determine the sampled value in the FIFO and the FIFO status. The software must provide correct trigger thresholds for RX FIFO and TX FIFO, and prevent the overflow of RX FIFO and the underflow of TX FIFO.

DMA Transfer Mode

SPI supports the DMA transfer mode. To enable this mode, you must set the thresholds of TX FIFO and RX FIFO respectively. Setting spi_dma_tx_en in the register spi_fifo_config_0 to 1 can enable the DMA sending mode. Setting spi_dma_rx_en in the register spi_fifo_config_0 to 1 can enable the DMA receiving mode. When this mode is enabled, UART will check the TX/RX FIFO. Once the tx_fifo_cnt/rx_fifo_cnt in the register spi_fifo_config_1 is greater than tx_fifo_th/rx_fifo_th, a DMA request will be initiated, and DMA will transfer data into TX FIFO or remove data from RX FIFO as configured.

SPI Interrupt

SPI supports the following interrupt control modes:

- SPI end of transfer interrupt
  - In the master mode, the SPI end of transfer interrupt will be triggered when the transfer of each frame of data ends.
  - In the slave mode, that interrupt is triggered when the CS signal is released.
- TX FIFO request interrupt
  - The TX FIFO request interrupt will be triggered when the FIFO available count value is greater than the preset threshold, and the interrupt flag will be cleared automatically when the condition is unmet.
- RX FIFO request interrupt
  - The RX FIFO request interrupt will be triggered when the FIFO available count value is greater than the preset threshold, and the interrupt flag will be cleared automatically when the condition is unmet.
- Slave mode transfer timeout interrupt
  - The slave mode transfer timeout interrupt will be triggered when no clock signal is received in the slave mode after the threshold exceeds. If the TX/RX FIFO overflows or underflows, it will trigger the TX/RX FIFO overflow interrupt.
- Slave mode TX overload interrupt
  - Slave mode TX overload interrupt will trigger when the TX is not ready to transmit in slave mode.
- TX/RX FIFO overflow interrupt
  - TX/RX FIFO overflow interrupt is triggered if an overflow or underflow occurs in the TX/RX FIFO. When the tx_fifo_clr/rx_fifo_clr bit in the FIFO clear register spi_fifo_config_0 is set to 1, the corresponding FIFO will be cleared and the overflow interrupt flag will be cleared automatically.

You can query the interrupt status through the register SPI_INT_STS and write 1 to the corresponding bit to clear the interrupt.

I2C

Overview

Inter-Integrated Circuit (I2C) is a serial communication bus, which uses a multi-slave and multi-master architecture and is connected to low-speed peripherals. Each device has a unique address identifier and can be used as a transmitter or receiver. The address of each device connected to the bus can be set by software through a unique address and the existing master or slave relation. The master can work as a master transmitter or a master receiver. If two or more masters are initialized at the same time, data can be prevented from being damaged through conflict detection and arbitration during transmission.

To facilitate communication with slaves. With the FIFO of 2-word depth and interrupt function, it can be used with DMA to improve efficiency and supports flexible adjustment of clock frequency.

Features

- Master mode
- Multi-master mode and arbitration function
- Flexible control of the level duration of the start, end and data transmission phases in segments
- Supports 7-bit address mode and 10-bit address mode
- Supports DMA transfer mode
- Supports multiple interrupt mechanisms

Functional Description

I2C pin list

Name	Type	Description
I2Cx_SCL	Input/output	I2C serial clock signal
I2Cx_SDA	Input/output	I2C serial data signal

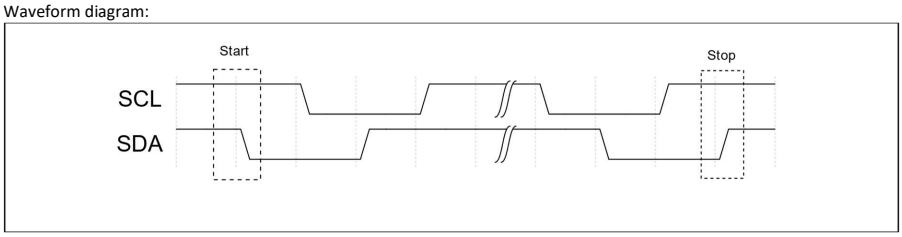
Start and stop conditions

All transmissions start with a START condition and end with a STOP condition. The START and STOP conditions are generally generated by the master. The bus is considered to be busy after the START condition and to be idle for a certain period of time after the STOP condition.

START condition: SDA produces a high-to-low level transition when SCL is high;

STOP condition: SDA produces a low-to-high level transition when SCL is high.

Waveform diagram:

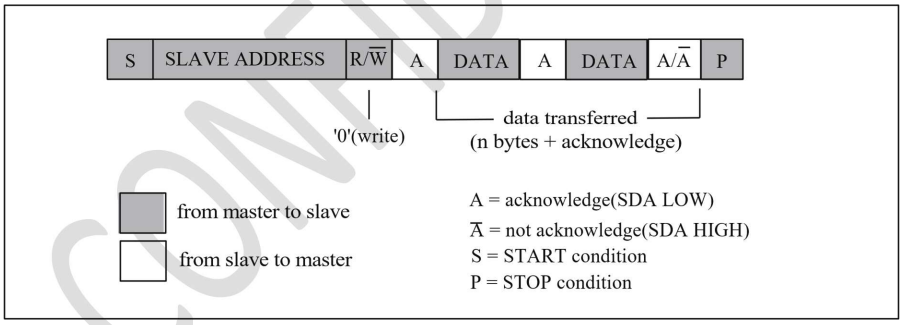


Start and stop conditions of I2C

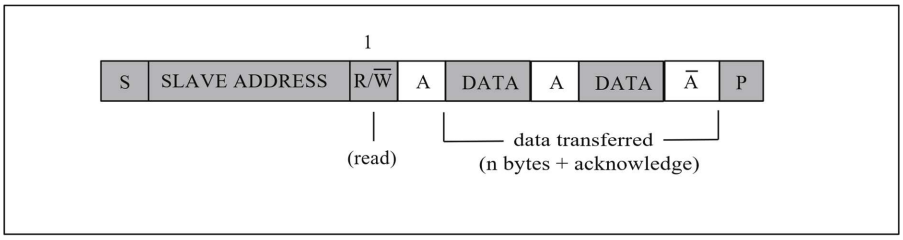
Data Transfer Format

1. 7-bit address mode:

The first 8 bits transferred are addressing bytes, including a 7-bit slave address and a 1-bit direction bit. Sending or receiving data by the master is controlled by the 8th bit in the first byte sent by the master. If it is 0, it means that the data is sent by the master, while "1" indicates that data is received by the master. After the direction bit is the answer bit (ACK), which is sent by the slave to answer (pull the signal low) and the host starts transmitting the specified length of data after receiving the answer. Upon data transfer completed, the master sends out a STOP signal, with waveform shown below:

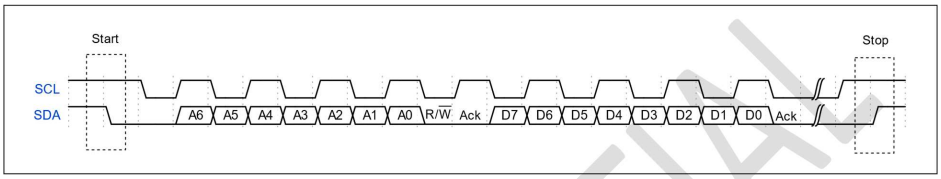


Master transmit and slave receive data formats



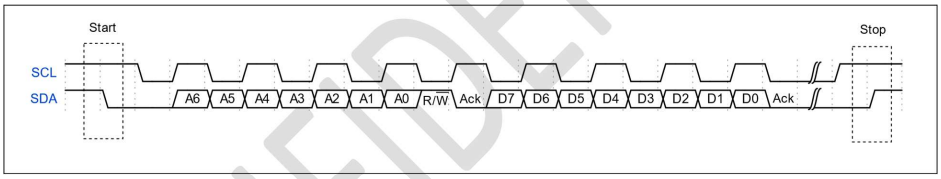
Master receive and slave transmit data formats

Master Transmit and Slave Receive Timing



Timing of master transmitter and slave receiver

Master Receive and Slave Transmit Timing

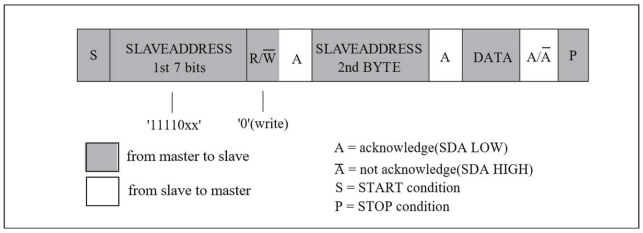


Timing of master receiver and slave transmitter

2. 10-bit address mode

The cr_i2c_10b_addr_en in the register i2c_config must be set to 1 before use.

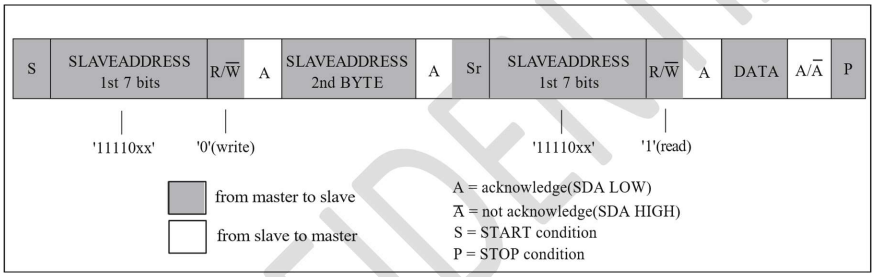
The 10-bit slave address consists of the two bytes after the START condition (S) or the repeated START condition (Sr). The first 7 bits of the first byte are 1111 0XX, where XX are the first two bits of MSB of the 10-bit address. The 8th bit of the first byte is the read/write bit that determines the transfer direction. The second byte is the remaining low 8 bits of the 10 bit address. The data transfer format is as follows.



Master transmit and slave receive data format (10bit slave address)

When receiving the 10-bit address following the START condition, the slave compares the first byte (1111 0XX) of the slave address with its own address, and checks whether the eighth bit (read/write bit) is 0. If the value of XX in the first byte is the same as the top two bits of the slave's 10 bit address, the first byte match passes and the slave will give answer A. If there are multiple slave devices connected to the bus, more than one device may match and generate answer A. Next, all slaves start to match the second byte (XXXX XXXX), where only one slave will have the exact same lower eight bits of the 10 bit address as the second byte, and that slave will give answer A. The slave that is addressed by the master will remain addressed until it receives a termination condition or a repeat start condition.

Master receive and slave transmit data format (10bit slave address)



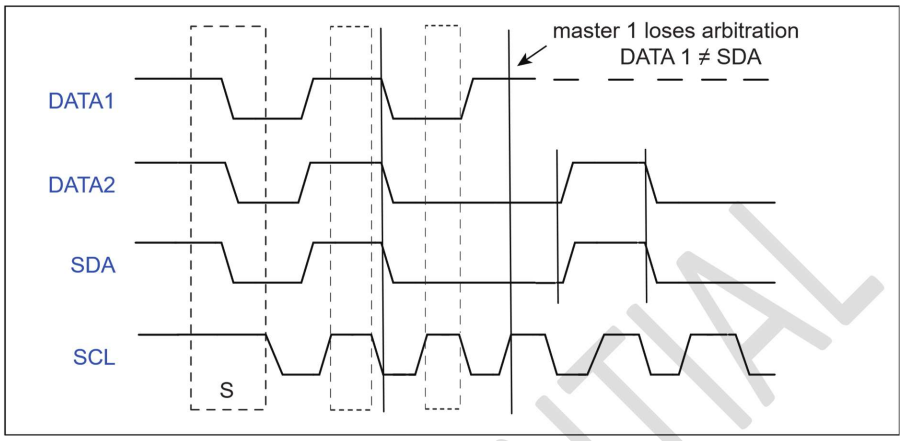
Before the second acknowledgement A, the process is the same as that of the master-transmitter addressing the slave-receiver. After the repeated START condition (Sr), the matched slave will remain in the addressed state. This slave will check whether the first 7 bits of the first byte after Sr are 1111 0XX, and then test whether the 8th bit is 1 (read). If this also matches, the slave considers that it is addressed as a transmitter and generates an acknowledgement (A). The slave-transmitter will remain in the addressed state until it receives the STOP condition (P) or the repeated START condition (Sr) followed by a different slave address. Then, under Sr, all the slaves will compare their addresses with 11110XX and test the eighth bit (read/write bit). However, they will not be addressed, because for 10-bit devices, the read/write bit is 1, or for 7-bit devices, the slave addresses of 1111 0XX do not match.

Arbitration

When there are multiple masters on I2C bus, it may happen that multiple masters start data transfer at the same time. At this time, the arbitration mechanism will decide which master has the right to transfer data, while other masters have to give up the control of the bus and wait until the bus is idle before transferring data again.

During data transfer, all masters must check whether the SDA is consistent with the data they want to send when SCL stays high. When the SDA level is different from the expected one, it means that other masters are transferring data at the same time. The masters with different SDA levels will lose the arbitration and other masters will complete the data transfer.

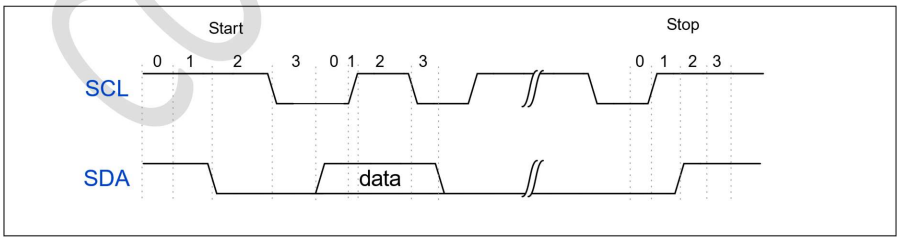
The waveform of two masters transferring data and initiating the arbitration mechanism at the same time is as follows:



Waveform of simultaneous data transfer

I2C Clock Setting

I2C clock can be derived from bclk (bus clock) and xclk, and frequency division can be done on this basis. The duration of the start condition, each bit of data and the end condition are set by registers i2c_prd_start, i2c_prd_data and i2c_prd_stop respectively. Each of these durations can be subdivided into 4 phases, and the number of samples in each phase is controlled by a separate byte in the register (the actual value is the register value plus 1). The 4 phase settings in the data section together determine the frequency division factor of the i2c clock. As shown in the figure below, suppose the I2C clock source is selected as 80M bclk and the register i2c_prd_data is set to 0x09070b09, then the second 0 in the figure is 0x09+1=0x0a, the second 1 is 0x07+1=0x08, the second 2 is 0x0b+1=0x0c, and the second 3 is 0x09+1=0x0a. Then the clock frequency of I2C is 80MHz/(0x0a+0x08+0x0c+0x0a) = 2MHz. Similarly, the first 0, 1, 2 and 3 are set by register i2c_prd_start, which determines the duration of the start condition, and the third 0, 1, 2 and 3 are set by register i2c_prd_stop, which determines the duration of the end condition.



I2C clock setting

I2C Configuration Flow

Configuration Items

- Read/write flag bit
- Slave address
- Slave register address
- Slave register address length
- Data (TX: configure the sent data; RX: store the received data)
- Data length
- Enable signal

Read/Write Flag Bit

I2C supports TX and RX working statuses. The cr_i2c_pkt_dir in the register i2c_config represents the TX/RX status, "0" for TX status and "1" for RX status.

Slave Address

Each slave connected to I2C will have a unique device address, which is usually 7 bits long. This address will be written into the cr_i2c_slv_addr in the register i2c_config. I2C will automatically shift to the left by 1 bit before sending the address, and the TX/RX direction bit will be added to the LSB.

Slave Register Address

The slave register address represents the register address where I2C needs to read and write a slave register. The slave register address is written to the register i2c_sub_addr, and the cr_i2c_sub_addr_en in the register i2c_config must be set to 1. If cr_i2c_sub_addr_en in the register i2c_config is set to 0, the I2C master will skip the slave register address field when sending.

Slave Register Address Length

The slave register address length is subtracted by 1 and then written to cr_i2c_sub_addr_bc in the register i2c_config.

Data

It refers to the data that needs to be sent to or received from the slave. When sending data, I2C must write the data (in word) into the register i2c_fifo_wdata. When receiving data, I2C must read out the data (in word) from the register i2c_fifo_rdata.

Data Length

The cr_i2c_pkt_len in the i2c_config register sets the send data length (the value written to the register + 1 is the send data length), and the maximum send length is 256 bytes.

Enable Signal

After the above items are configured, when cr_i2c_m_en in the enable signal register i2c_config is set to 1, the I2C sending process will be started automatically.

When the read/write flag bit is configured as 0, I2C sends data. Take sending 2 bytes as an example, the master's transmission flow is as follows:

1. Start bit

2. (The slave address shifts to the left by 1 bit + 0) + ACK
3. Slave register address + ACK
4. 1-byte data + ACK
5. 1-byte data + ACK
6. Stop bit

When the read/write flag bit is configured as 1, I2C receives data. Take receiving 2 bytes as an example, the master's transmission flow is as follows:

1. Start bit
2. (The slave address shifts to the left by 1 bit + 0) + ACK
3. Slave register address + ACK
4. Start bit
5. (The slave address shifts to the left by 1 bit + 1) + ACK
6. 1-byte data + ACK
7. 1-byte data + ACK
8. Stop bit

FIFO Management

I2C FIFO has a 2-word depth, and I2C includes RX FIFO and TX FIFO. The rx_fifo_cnt in the register i2c_fifo_config_1 represents how much data (in word) in RX FIFO needs to be read. The tx_fifo_cnt in the register i2c_fifo_config_1 represents how much free space (in word) in TX FIFO for writing.

I2C FIFO status:

- RX FIFO underflow: When the data in RX FIFO is completely read out or empty, if I2C continues to read data from RX FIFO, the rx_fifo_underflow in the register i2c_fifo_config_0 will be set to 1;
- RX FIFO overflow: When I2C receives data until the two words of RX FIFO are filled, without reading RX FIFO, if I2C receives data again, the rx_fifo_overflow in the register i2c_fifo_config_0 will be set to 1;
- TX FIFO underflow: When the data size filled into TX FIFO does not meet the configured I2C data length: cr_i2c_pkt_len in i2c_config, and no new data is filled into TX FIFO, the tx_fifo_underflow in the register i2c_fifo_config_0 will be set to 1;
- TX FIFO overflow: After the two words of TX FIFO are filled, before the data in TX FIFO is sent out, if data is filled into TX FIFO again, the tx_fifo_overflow in the register i2c_fifo_config_0 will be set to 1.

Use with DMA

I2C can send and receive data through DMA. Setting i2c_dma_tx_en in the register i2c_fifo_config_0 to 1 will enable the DMA TX mode. After the channel for I2C is allocated, DMA will transfer data from memory to the i2c_fifo_wdata register. Setting i2c_dma_rx_en in the register i2c_fifo_config_0 to 1 will enable the DMA RX mode. After the channel for I2C is allocated, DMA will transfer the data in the register i2c_fifo_rdata to memory. When I2C is used with DMA, DMA will automatically transfer data, so it is unnecessary for CPU to write data into I2C TX FIFO or read data from I2C RX FIFO.

DMA Sending Flow

1. Set read/write flag bit to 0
2. Set slave address
3. Set slave register address
4. Set slave register address length
5. Data Length
6. Set enable signal register to 1

7. Configure DMA transfer size
8. Configure the transfer width of DMA source address
9. Configure the transfer width of DMA destination address (when I2C is used with DMA, the transfer width of destination address must be set to 32 bits, which is word-aligned)
10. Configure the DMA source address as the memory address for storing sent data
11. Configure the DMA destination address to I2C TX FIFO address, i2c_fifo_wdata
12. Enable DMA

DMA Receiving Flow

1. Set read/write flag bit to 1
2. Set slave address
3. Set slave register address
4. Set slave register address length
5. Data Length
6. Set enable signal register to 1
7. Configure DMA transfer size
8. Configure the transfer width of DMA source address (when I2C is used with DMA, the transfer width of source address must be set to 32 bits, which is word-aligned)
9. Configure the transfer width of DMA destination address
10. Configure the DMA source address to I2C RX FIFO address, i2c_fifo_rdata
11. Configure the DMA destination address as the memory address for storing received data
12. Enable DMA

Interrupt

I2C includes the following interrupts:

- I2C_TRANS_END_INT * I2C transfer end interrupt, which is generated when I2C completes a transfer
- I2C_TX_FIFO_READY_INT * When tx_fifo_cnt in i2c_fifo_config_1 is greater than tx_fifo_th, a TX FIFO request interrupt will be generated, and the interrupt flag will be automatically cleared when the condition is not satisfied
- I2C_RX_FIFO_READY_INT * When rx_fifo_cnt in i2c_fifo_config_1 is greater than rx_fifo_th, an RX FIFO request interrupt will be generated, and the interrupt flag will be automatically cleared when the condition is not satisfied
- I2C_NACK_RECV_INT * When the I2C module detects a NACK state, a NACK interrupt is generated
- I2C_ARB_LOST_INT * I2C arbitration lost interrupt
- I2C_FIFO_ERR_INT * FIFO ERROR interrupt is generated when TX/RX FIFO overflows or underflows

UART

Overview

The Universal Asynchronous Receiver/Transmitter (UART) provides a flexible way to exchange full-duplex data with external devices. uC subsystem is provided with 1 UARTs, which can be used together with DMA to achieve efficient data communication.

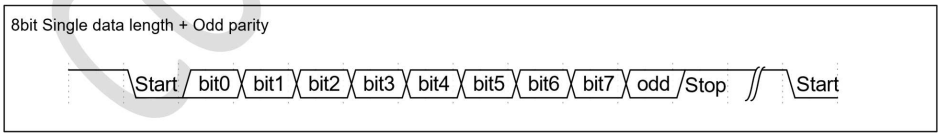
Features

- Full-duplex asynchronous communication
- Optional data bit length: 5/6/7/8-bit
- Optional stop bit length: 0.5/1/1.5/2-bit
- Supports odd/even/none check bit
- Error-detectable start bit
- Abundant interrupt control modes
- Hardware flow control (RTS/CTS)
- Convenient baud rate programming
- Configurable MSB/LSB transfer priority
- Automatic baud rate detection of ordinary/fixed characters
- 32-byte TX/RX FIFO
- Supports DMA transfer mode
- Supports baud rate of 10 Mbps and below
- Supports the LIN bus protocol
- Supports the RS485 mode
- Optional clock sources: 160M/BCLK/XCLk
- Support filter function

Functional Description

Data Formats

The normal UART communication data consists of start bits, data bits, parity check bits, and stop bits. The UART of xx supports configurable data bits, parity check bits, and stop bits, which are set in registers utx_config and urx_config. The waveform of a frame of data is shown as follows:



UART Data Format

The start bit of the data frame occupies 1 bit, and the stop bit can be 0.5/1/1.5/2-bit wide by configuring the cr_utx_bit_cnt_p bit in the register utx_config. The start bit is at a low level and the stop bit is at a high level. The data bit width can be set to 5/6/7/8-bit by the cr_utx_bit_cnt_d bit in the register utx_config.

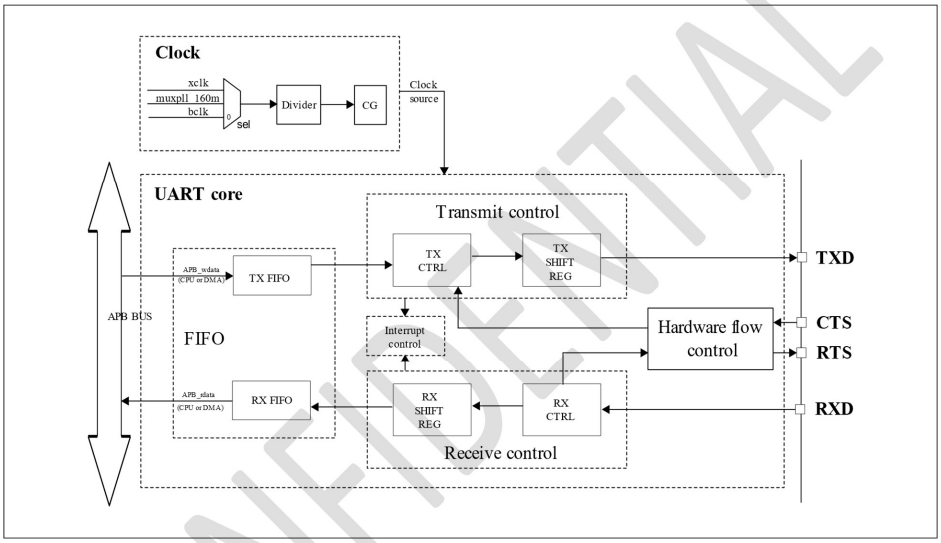
When the cr_utx_prt_en bit in the register utx_config and the cr_urx_prt_en bit in the register urx_config are set, the data frame adds a parity check bit after the data. The cr_utx_prt_sel bit in the register utx_config and the cr_urx_prt_sel bit in the

register urx_config are used to select odd or even parity check. When the receiver detects the check bit error of the input data, it will generate the check error interrupt. However, the received data will still be stored into the FIFO.

Calculation method of odd parity check: If there is an odd number of "1" in the current data bit, the odd parity check bit is set to 0. Otherwise, it is set to 1.

Calculation method of even parity check: If there is an odd number of "1" in the current data bit, the even parity check bit is set to 1. Otherwise, it is set to 0.

Basic Architecture



UART Architecture

The UART has 3 clock sources: XCLK, 200MHz CLK and BCLK. The frequency divider in the clock is used to divide the frequency of the clock source and then generate the clock signal to drive the UART module.

The UART controller is divided into two functional blocks: transmitter and receiver.

Transmitter

The transmitter contains a 32-byte TX FIFO to store the data to be sent. When the transmission enable bit is set, the data stored in FIFO will be output from the TX pin. Software can transfer data into TX FIFO through DMA or APB bus. Software can check the status of transmitter by querying the remaining free space count value of TX FIFO through tx_fifo_cnt in the register uart_fifo_config_1.

FreeRun mode of transmitter:

- If the FreeRun mode is disabled, transmission will be terminated and an interrupt will be generated when the sent bytes reach the specified length. Before next transmission, you need to re-disable and enable the TxE bit.
- If the FreeRun mode is enabled, the transmitter will send when there is data in the TX FIFO, and will not stop working because the sent bytes reach the specified length.

Receiver

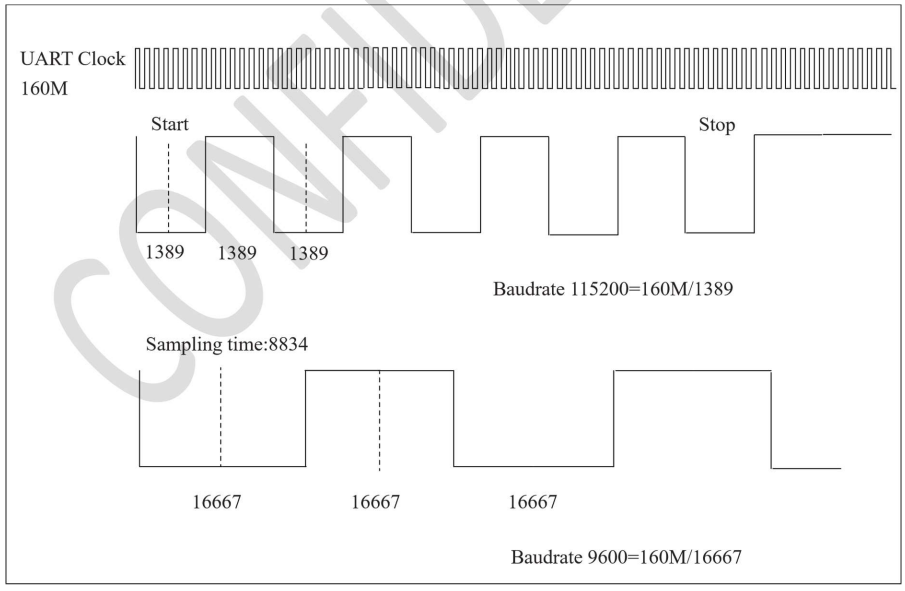
The receiver contains a 32-byte RX FIFO to store the received data. Software can check the status of receiver by querying the available data count value of RX FIFO through rx_fifo_cnt in the register uart_fifo_config_1. The low 8 bits of the register URX_RTO_TIMER are used to set a receiving timeout threshold, which will trigger an interrupt when the receiver fails to receive data beyond the threshold. The cr_urx_deg_en and cr_urx_deg_cnt in the register urx_config are used to enable the deburring function and set the threshold, which control the filtering part before sampling by UART. UART will filter out the burrs whose width is lower than the threshold in the waveform and then send it to sampling.

Baud Rate Setting

Baudrate = \frac{UART_clk}{uart_prd + 1}

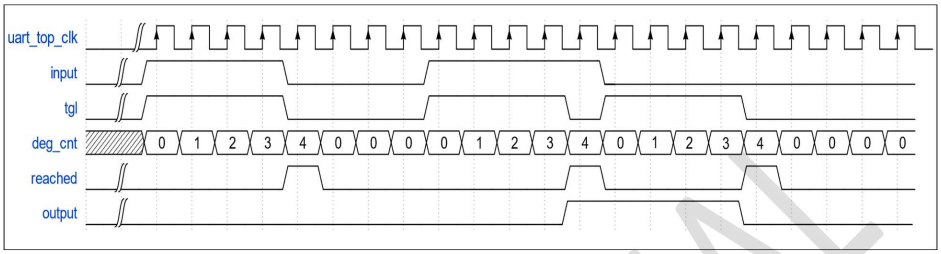
The user can set the baud rate of RX and TX separately. Take TX as an example: the value of uart_prd is the value of the lower 16 bits cr_utx_bit_prd of the register UART_BIT_PRD. Since the maximum value of the 16-bit bit width coefficient is 65535, the minimum baud rate supported by UART is : UART_clk/65536.

Before sampling the data, UART will filter the data to remove the burrs in the waveform. Then, the data will be sampled at the intermediate value of the 16-bit width factor, so that the sampling time can be adjusted based on baud rates, to ensure that the intermediate value is always sampled, providing much higher flexibility and accuracy. The sampling process is shown as follows:



UART Sampling Waveform

Filtering



UART Filter Waveform

When this function is enabled by configuring cr_urx_deg_en and the threshold is set by configuring cr_urx_deg_cnt in the register urx_config, UART will filter out the data that cannot meet the width threshold. As shown in the figure below, when the data width is 4, setting cr_urx_deg_cnt to 4 can meet this condition. "Input" is the initial data and "output" is the filtered data.

Filtering logic process:

- "Tgl" is the exclusive OR result of input and output.
- "Deg_cnt" counts from 0, and the counting condition is that "tgl" is at the high level and "reached" is at the low level.
- If the count value of deg_cnt reaches the value set by cr_urx_deg_cnt, reached is high level.
- When "reached" is at a high level, "input" is output to "output".
- Note: user-defined condition for deg_cnt: "tgl" is at a high level and "reached" is at a low level. In other cases, "deg_cnt" will be cleared to 0.

Automatic Baud Rate Detection

The UART module supports automatic baud rate detection, which is divided into two modes, a generic mode and a fixed character (square wave) mode. The cr_urx_abr_en in the urx_config register enables auto baud rate detection, and when it is turned on, both detection modes are enabled.

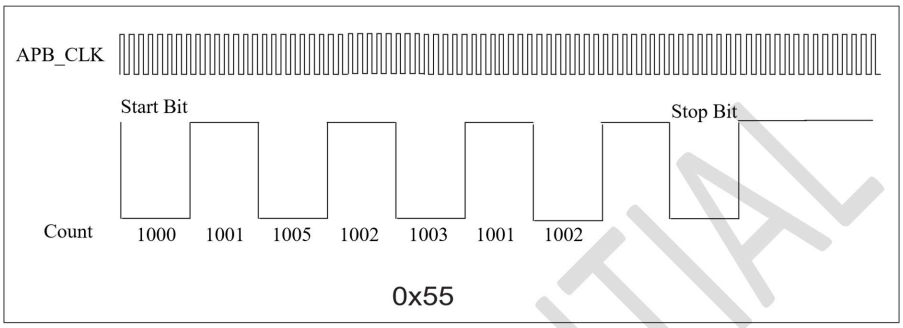
Generic mode

For any character data received, UART will count the number of clocks in the start bit width, which will then be written into the low 16 bits sts_urx_abr_prd_start in the register STS_URX_ABR_PRD and used to calculate the baud rate. So the correct baud rate can be obtained when the first received data bit is 1, such as '0x01' under LSBFIRST.

Fixed character (square wave) mode

In this mode, after the UART module counts the number of clocks in the bit width, it will continue to count the number of clocks in the subsequent data bits and compare it with the start bit. The check is passed, otherwise the count value is discarded. The allowable error can be set by setting the cr_urx_abr_pw_tol bit in the register urx_abr_pw_tol, and the unit is the clock source of the UART.

Therefore, only when the fixed character '0x55'/'0xD5' under LSB-FIRST or '0xAA'/'0xAB' under MSB-FIRST is received, the UART module will write the clock count value in the starting bit width into the high 16-bit sts_urx_abr_prd_0x55 of the register STS_URX_ABR_PRD. As shown below:



Waveform of UART in fixed character mode

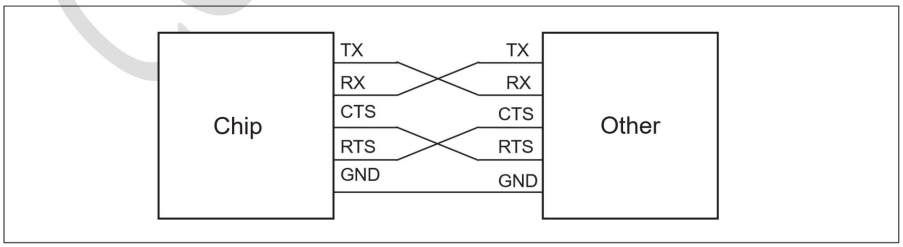
As shown above, assuming the maximum allowable error set is 4, for a received data with unknown baud rate, the UART uses UART_CLK to count the bit width of the starting bit as 1000, the bit width of the second bit as 1001, which is not more than 4 UART_CLK up or down from the previous bit width, then the UART will continue to count the third bit. The third bit is 1005, the difference with the starting bit is more than 4, the detection is not passed and the data is discarded. UART compares the first 6 bit widths of the data bits with the starting bit in turn.

Formula for calculating the detected baud rate:

Baudrate = \frac{UART_clk}{Count + 1}

Hardware Flow Control

UART supports hardware flow control in CTS/RTS mode to prevent data in FIFO from being lost due to too late processing. Hardware flow control connection is shown as follows:



UART hardware flow control

Require To Send (RTS) is an output signal, which indicates whether the chip is ready to receive data from the other side. This is valid at a low level denoting that the chip can receive data.

Clear To Send (CTS) is an input signal, which determines whether the chip can send data to the other side. This is valid at a low level denoting that the chip can send data to the other side.

When the hardware flow control function is enabled, the low level of chip's RTS indicates requesting the other side to send data, and the high level of that indicates informing the other side to stop sending data. When the chip detects that CTS goes high, TX will stop sending data, and continue sending until CTS goes low. If CTS goes high or low at any time during communication, it does not affect the continuity of data sent by TX, and the other side also can receive continuous data.

Two ways for hardware flow control of the transmitter:

- Hardware control (the cr_urx_rts_sw_mode in the register uart_sw_mode is 0): RTS goes high when cr_urx_en in the register urx_config is not turned on or the RX FIFO is almost full (one byte left).
- Software control(the cr_urx_rts_sw_mode in the register uart_sw_mode is 1): The level of RTS can be changed by configuring cr_urx_rts_sw_val in the register uart_sw_mode.

DMA Transfer

UART supports DMA transfer. Using DMA transfer, the TX and RX FIFO thresholds need to be set respectively by tx_fifo_th and rx_fifo_th in register uart_fifo_config_1. When this mode is enabled, if tx_fifo_cnt in uart_fifo_config_1 is greater than tx_fifo_th, a DMA TX request will be triggered. After the DMA is configured, when the DMA receives the request, it will move the data from the memory to the TX FIFO according to the settings. If the rx_fifo_cnt in uart_fifo_config_1 is greater than rx_fifo_th, the DMA RX request will be triggered. After the DMA is configured, when the DMA receives the request, it will transfer the data of the RX FIFO to the memory according to the settings.

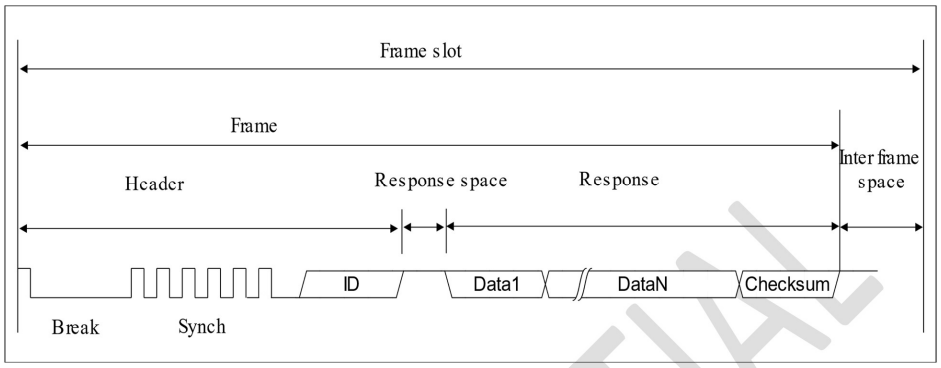
In order to ensure the correctness of the data transferred by the chip DMA TX Channel, the following conditions need to be met in the Channel configuration: (transferWidth * burstSize) ≤ (tx_fifo_th + 1).

In order to ensure the integrity of the data transferred by the chip DMA RX Channel, the following conditions need to be met in the Channel configuration: (transferWidth * burstSize) = (rx_fifo_th + 1).

Support for LIN Bus

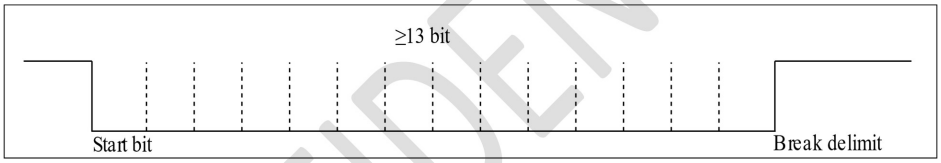
The protocol for the Local Interconnect Network (LIN) is based on the Volcano-Lite technology developed by the Volvo spin-out company—Volcano Communications Technology (VCT). LIN is a complementary protocol to CAN and SAE J1850, suitable for applications that have low requirement for time or require no precise fault tolerance (as LIN is not as reliable as CAN). LIN aims to be easy to use as a low-cost alternative to CAN. The vehicle parts where LIN can be used include window regulator, rearview mirror, wiper, and rain sensor.

UART supports the LIN bus mode.The LIN bus is under the master-slave mode, and data is always initiated by the master node. The frame (header) sent by the master node contains synchronization interval field, synchronization byte field, and identifier field. A typical LIN data transfer is shown as follows.



A typical LIN frame

1. LIN Break Field



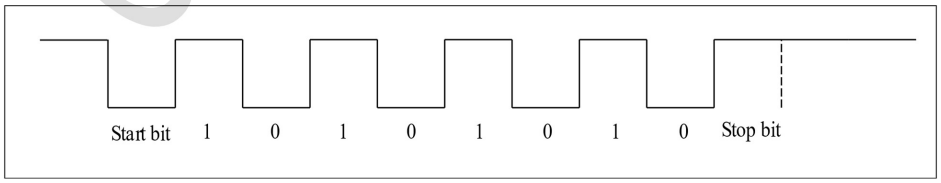
Break Field of LIN

The synchronization interval field indicates the start of the message, with at least 13 dominant bits (including the start bit). The synchronization interval ends with an "interval separator", which contains at least one recessive bit.

The length of the Break in the LIN frame can be set by cr_utx_bit_cnt_b in utx_config.

2. LIN Sync Field

A synchronization byte field is sent to determine the time between two falling edges, to determine the transmission rate used by the master node. The bit pattern is 0x55 (01010101, maximum number of falling edges).



Sync Field of LIN

3. LIN ID Field

The identifier field contains a 6-bit identifier and two parity check bits. The 6-bit identifier contains information about the sender and receiver, and the number of bytes required in the response. The parity check bit is calculated as follows: The check bit P0 is the result of logical OR operation among ID0, ID1, ID2, and ID4. The check bit P1 is the result of inversion after logical OR operation among ID1, ID3, ID4, and ID5.

ID Field of LIN

The slave node waits for the synchronization interval field, and then starts to synchronize between master and slave nodes through the synchronization byte field. Depending on the identifier sent by the master node, the slave node will receive, send or do not respond. The slave node that should send data sends the number of bytes requested by the master node, and then ends the transmission with a checksum field.

UART supports the LIN transfer mode. To enable this mode, you need to configure the cr_utx_bit_cnt_b by setting cr_utx_lin_en in the register utx_config so that the synchronization interval field consists of at least a 13-bit dominant level.

RS485 mode

UART supports the RS485 mode. After the cr_utx_rs485_en in the register UTX_RS485_CFG is set, UART can work in the RS485 mode. Then, UART can be connected to the RS485 bus through an external RS485 transceiver. In this mode, the RTS pin in the module performs the Dir function of transceiver. When UART has data to send, it will automatically control the RTS pin at a high level, so that the transceiver can send data to the bus. Contrarily, when UART has no data to send, it will automatically control the RTS at a low level, to keep the transceiver in the RX state.

UART supports the RS485 transfer mode. To enable this mode, you need to set cr_utx_rs485_pol and cr_utx_rs485_en in the register UTX_RS485_CFG.

UART Interrupt

UART supports the following interrupt control modes:

- TX end of transfer interrupt
- RX end interrupt
- TX FIFO request interrupt
- RX FIFO request interrupt
- RX timeout interrupt
- RX parity check error interrupt
- TX FIFO overflow interrupt
- RX FIFO overflow interrupt
- RX BCR interrupt
- LIN synchronization error interrupt
- Auto baud rate detection (universal mode) interrupt
- Auto baud rate detection (fixed characters mode) interrupt

TX/RX end of transfer interrupt

You can set a transfer length for TX and RX respectively by configuring the high 16 bits of the registers utx_config and urx_config. When the number of transferred bytes reaches this value, the corresponding TX/RX end of transfer interrupt will be triggered. While this interrupt is generated, TX stops working. To continue to use TX, you must re-initialize this module. Then, RX resumes to work. If the preset transfer length of TX is less than the data volume actually sent by TX, the other side can only receive the data equal to the transfer length, and the remaining data will be stored in TX FIFO. After this module is re-initialized, the data in TX FIFO can be sent out.

For TX, if the data continuously filled into the TX FIFO is greater than the set transmission length value, only the data of the set length value will be transmitted on the TX pin, and the excess data will be kept in the TX FIFO, and the TX will be re-enabled. After the function, the remaining data in the TX FIFO will be sent out.

For example: set the TX transmission length value to 64, enable the TX function, first fill 63 bytes into the TX FIFO, these 63 bytes will be transmitted on the pins, but no TX transmission completion interrupt is generated, and then the TX FIFO is sent to the TX FIFO. Fill in 1 byte, at this time, the transmission length reaches the transmission length value set by TX, a transmission completion interrupt will be generated, and the TX function will stop. Continue to fill 1 byte into the TX FIFO, you will find that there is no data transmission on the pin, the byte is still retained in the TX FIFO, and the TX function is turned off and re-enabled, and the byte is sent out on the pin.

For RX, if the data length sent by the other party exceeds the set transmission length, RX can continue to receive data after the RX transmission completion interrupt is generated.

For example: set the RX transmission length value to 16, the other party sends 32 bytes of data, RX will generate an RX transmission completion interrupt when it receives 16 bytes of data, and continue to receive the remaining 16 bytes of data, all saved in the RX FIFO.

TX/RX FIFO request interrupt

A TX FIFO request interrupt will be generated when tx_fifo_cnt in uart_fifo_config_1 is greater than tx_fifo_th. When the condition is not met, the interrupt flag will be cleared automatically.

A RX FIFO request interrupt will be generated when rx_fifo_cnt in uart_fifo_config_1 is greater than rx_fifo_th. When the condition is not met, the interrupt flag will be cleared automatically.

TX/RX supports multiple rounds of transmission/receiving, instead of reaching the value set by tx_fifo_th/rx_fifo_th at a time.

E.g:

1. Set tx_fifo_th/rx_fifo_th in register uart_fifo_config_1 to 16.
2. Set cr_utx_frm_en in register utx_config to enable free run mode.
3. Set cr_utx_frdy_mask/cr_urx_frdy_mask in register uart_int_mask to 0, and enable FIFO interrupt of TX/RX.
4. Set cr_utx_en/cr_urx_en in register utx_config/urx_config to enable TX/RX.
5. TX FIFO interrupt: TX will always enter the FIFO interrupt, when the chip sends 128 bytes, set cr_utx_frdy_mask to 1 to shield the interrupt. If you want to enter the TX FIFO interrupt again, set cr_utx_frdy_mask to 0.
6. RX FIFO interrupt: the other party first sends 15 bytes, no interrupt is generated, at this time, the value of rx_fifo_cnt is 15, and an interrupt is generated when 1 byte is sent again to reach the value set by rx_fifo_th. After the transmission is interrupted, the other party sends the data again, and the chip can receive the data.

RX timeout interrupt

The RX timeout interrupt generation condition is: after receiving data last time, the receiver will start timing, and the interrupt will be triggered when the timing value exceeds the timeout threshold and the next data has not been received. The time-out threshold value is in the unit of communication bit.

When the other party sends data to the chip, a timeout interrupt will be generated after the set timeout period is reached.

RX parity check error interrupt

The RX parity check error interrupt will be generated when a parity check error occurs. But it does not affect the RX, which still can correctly receive and analyze the data sent by the other side. When receiving data, RX takes the first 8 bits as data bits and ignores parity check bits, ensuring data consistency.

For example, you can enable parity check by setting cr_utx_prt_en/cr_urx_prt_en in the register utx_config/urx_config, and select the parity check type by setting cr_utx_prt_sel/cr_urx_prt_sel in the register utx_config/urx_config. When the other side sends data to the chip through odd/even parity check, the parity check of RX is disabled, but RX can receive correct data.

TX/RX FIFO overflow interrupt

If the TX/RX FIFO overflows or underflows, it will trigger the corresponding overflow interrupt. When the tx_fifo_clr and rx_fifo_clr bits in the FIFO clear bit register UART_FIFO_CONFIG_0 are set to 1, the corresponding FIFO will be cleared and the overflow interrupt flag will be cleared automatically. You can query the interrupt status through the register UART_INT_STS, and clear the interrupt by writing 1 to the corresponding bit in the register UART_INT_CLEAR.

RX BCR interrupt

A BCR interrupt will be generated when the data received by RX reaches the value set by cr_urx_bcr_value in the register urx_bcr_int_cfg.

The difference from RX END interrupt is that END interrupt is suitable for receiving data of known length, while BCR interrupt can be used to receive interrupts of unknown length. The trigger position of the END interrupt is controlled by cr_urx_len, and the counter will be cleared to 0 when an interrupt is triggered. The trigger position of BCR interrupt is controlled by cr_urx_bcr_value. When the interrupt is triggered, the counter will accumulate instead of being cleared to 0, but it can be cleared by software (cr_urx_bcr_clr). When the BCR interrupt is used together with the chained DMA, if you do not know how much data has been transferred by DMA, you can check it through "count".

LIN synchronization error interrupt

When cr_utx_lin_en in utx_config is enabled, the LIN mode is enabled. Then, if the synchronization field of the LIN bus is not detected when data is received in this mode, the LIN synchronization error interrupt will be generated.

Auto baud rate detection (universal/fixed characters mode) interrupt

In the auto baud rate detection mode, when a baud rate is detected, the auto baud rate detection (universal/fixed characters mode) interrupt will be generated as configured.

DMA

With the management of DMA controller, peripherals and memory directly exchange data without CPU intervention. After data transfer is complete, the device send a DMA end signal to CPU, returning bus control.

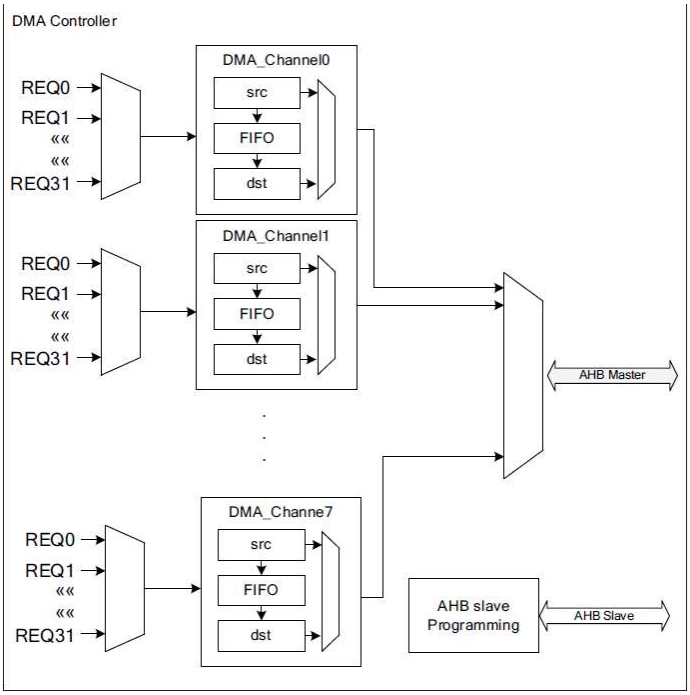
The DMA controller includes a set of AHB master interfaces and a set of AHB slave interfaces. The AHB master interface actively accesses memory or peripheral through the system bus according to current configuration requirements, as a port of data movement. The AHB slave interface is used to configure DMA interface and only supports 32-bit access.

DMA supports 8 channels in total, each channel does not interfere with each other and can run at same time. The following us configuration process of DMA channel x.

1. Set 32-bit source address in DMA_C0SrcAddr.
2. Set 32 bit target address in DMA_C0DstAddr.
3. Configure SI (source) and DI (destination) in DMA_C0Control register to set whether to enable automatic address accumulation.
4. Set the transfer data width STW (source) and DTW (destination) in DMA_C0Control. The options are single byte, double and four byte.
5. Set burst type, SBS (source) and DBS (destination). The options are Single, INCR4, INCR8, INCR16.
6. A single burst cannot exceed 16 bytes.
7. Set data transmission length range: 0-4095

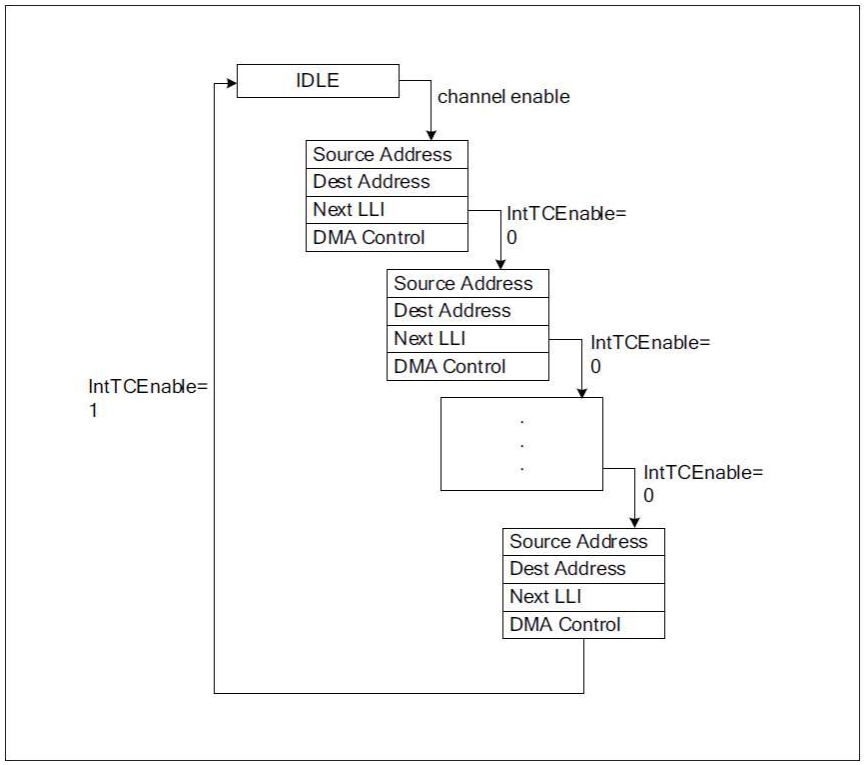
Peripheral port

0	1	2	3	4	5	6	7
Uart.tx	Uart.rx	Spi0.tx	Spi0.rx	Spi1.tx	Spi1.rx	I2c.tx	I2c.rx



DMA architecture

DMA support linked list operation mode. When performing DMA read or write operation, you can fill data in next linked list. After completing the data transfer of current list, read the DMA_C0LLI register to obtain the start address of next linked list, and directly transfer data.



: LLI architecture

when source peripheral requests a trigger, the source configuration burst will be stored in the buffer and will stop the set number of moves is reached. On the other hand, when internal cache is enough for target burst number, DMA will automatically move the cached content. DMA_C0SrcAddr and DMA_C0SrcDstAddr specifies source and target address respectively.

Transmission modes:

Memory to memory

Memory to peripheral

Peripheral to memory:

Peripheral to peripheral:

SEC ENG

Overview

SEC ENG has a variety of built-in computing modules, including AES, SHA. ~~CRC, GMAC, PKA, and TRNG~~.

Features

- AES
  - Supports 128-bit, 192-bit and 256-bit key lengths
  - Support encryption and decryption of multiple link modes (ECB/CBC/CTR/XTS)
  - Exclusive AES LINK function
- SHA
  - Support SHA1/SHA224/SHA256/SHA384/SHA512
  - Exclusive SHA LINK function
- ~~Support MD5, CRC-16, CRC-32~~

Functional Description

AES Accelerator

key

The key length required for encryption mode and decryption mode can be selected by configuring se_aes_0_mode and se_aes_0_dec_en in the register se_aes_0_ctrl.

amode	adec en	运算
0	0	AES-128 加密
1	0	AES-256 加密
2	0	AES-192 加密
0	1	AES-128 解密
1	1	AES-256 解密
2	1	AES-192 解密

AES operation mode diagram

Select whether to enable the hardware key through aes_0_hw_key_en in the register se_aes_0_ctrl. If you use a software key, you also need to configure the registers se_aes_0_key_0~se_aes_0_key_7 to store the key, and each register stores a 4-byte key.

link mode

Different link modes can be selected through se_aes_0_block_mode in the register se_aes_0_ctrl. Currently, ECB, CBC, CTR, and XTS modes are supported.

Plaintext, ciphertext

- Plaintext or ciphertext needs to be a multiple of 16
- The register se_aes_0_msa stores the plaintext address entered during encryption or the ciphertext address entered during decryption
- The register se_aes_0_mda stores the ciphertext address output during encryption or the plaintext address output during decryption
- se_aes_0_msg_len in the register se_aes_0_ctrl is used to set the length of ciphertext or plaintext (in units of 16 bytes)

initialization vector

The registers se_aes_0_iv_0~se_aes_0_iv_3 store the initialization vector IV. You can choose whether to use a new iv by configuring aes_0_iv_sel in se_aes_0_ctrl. You need to clear 0 when configuring iv for the first time, and you need to set 1 if you continue to use this iv or automatically update iv.

Encryption and decryption configuration process

- Enable AES with se_aes_0_en in the configuration register se_aes_0_ctrl
- Configuration register se_aes_0_endian, including se_aes_0_dout_endian, se_aes_0_din_endian, se_aes_0_key_endian, se_aes_0_iv_endian, se_aes_0_twk_endian, if the value is 0, it means little-endian, if the value is 1, it means big-endian
- se_aes_0_block_mode in the configuration register se_aes_0_ctrl selects the link mode
- se_aes_0_mode in the configuration register se_aes_0_ctrl to select the key length
- To use software key, configure registers se_aes_0_key_0~se_aes_0_key_7 to store the key. To use a hardware key, set aes_0_hw_key_en in register se_aes_0_ctrl
- Configure registers se_aes_0_iv_0~se_aes_0_iv_3 to set IV, the filling order for MSB is se_aes_0_iv_0~se_aes_0_iv_3, and the filling order for LSB is se_aes_0_iv_3~se_aes_0_iv_0
- se_aes_0_dec_en in the configuration register se_aes_0_ctrl selects the encryption or decryption mode
- The configuration register se_aes_0_msa sets the source address of the data to be processed
- The configuration register se_aes_0_mda sets the destination address where the processing result is stored
- se_aes_0_msg_len in the configuration register se_aes_0_ctrl sets the length of the data to be processed, in units of 16 bytes
- se_aes_0_trig_1t in the configuration register se_aes_0_ctrl triggers AES to run

The result is output to the destination address specified by se_aes_0_mda.

SHA Accelerator

SHA mode

The se_sha_0_mode in the register se_sha_0_ctrl: 0:SHA-256 1:SHA-224 2:SHA-1 3:SHA-1 4:SHA-512 5:SHA-384 6:SHA-512/224 7:SHA-512/256

The se_sha_0_mode_ext in the register se_sha_0_ctrl: hash mode extention; 0:SHA 1:MD5 2:CRC-16 3:CRC-32

Plaintext and ciphertext

- The register se_sha_0_msa stores the plaintext address.
- The register se_sha_0_hash_l_0~se_sha_0_hash_l_7 stores the ciphertext.

Operation flow

- Configure se_sha_0_mode in the register se_sha_0_ctrl to set the specific mode of SHA
- Enable SHA by configuring se_sha_0_en in the register se_sha_0_ctrl
- Configure se_sha_0_hash_sel in the register se_sha_0_ctrl; 0 means starting a new HASH calculation, and 1 means using the last result for HASH calculation
- Configure the register se_sha_0_msa to set the source address of the data to be processed
- Configure se_sha_0_msg_len in the register se_sha_0_ctrl to set the length of the data to be processed (512 bits for SHA1, SHA224 and SHA256, while 1024 bits for SHA512, SHA384, SHA512/224, and SHA512/256)
- Configure se_sha_0_trig_1t in the register se_sha_0_ctrl to trigger SHA
- The output result is stored in se_sha_0_hash_l_0~se_sha_0_hash_l_7, MSB:se_sha_0_hash_l_0~se_sha_0_hash_l_7, LSB:se_sha_0_hash_l_7~se_sha_0_hash_l_0

~~**Random Number Generator (RNG)**~~

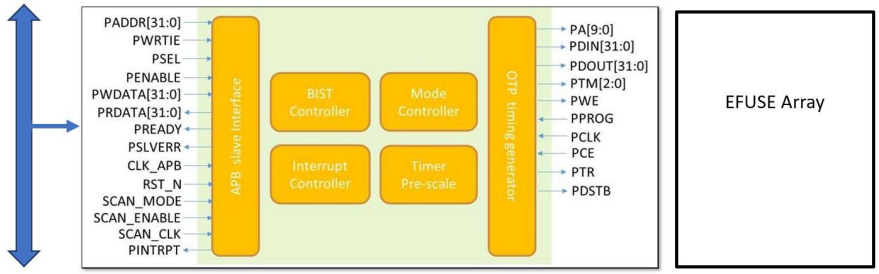
~~The random numbers generated by the built-in true RNG can be used as the basis for encryption and other operations.~~

- ~~True random numbers: They are generated through physical phenomena, such as coin tossup, dicing, wheel rotation, noise from using electronic components, and nuclear fission. Such RNGs are called physical RNGs, and their weaknesses are high technical requirements.~~
- ~~Pseudo-random numbers: Truly random numbers (or random events) are randomly generated in a generation process according to the distribution probability shown in the experimental process, and the result is unpredictable and invisible. The random function in the computer is simulated according to a specified algorithm, and its result is deterministic and visible. We may consider that the probability of this foreseeable result is 100%. Hence the "random number" generated by computer random function is not truly random, but pseudo-random.~~

~~**Usage process**~~

- ~~Enable TRNG by configuring se_trng_0_en in the register se_trng_0_ctrl_0~~
- ~~Configure se_trng_0_trig_1t in the register se_trng_0_ctrl_0 to trigger TRNG~~
- ~~The output result is stored in se_trng_0_dout_0~se_trng_0_dout_7~~

OTP



System APB bus

OTP is integrated into Mockingbird's uC subsystem (base address: 0x2a080000, IRQ21).

- System bus APB is connected to uC.
- "PINTRPT" is connected to E21 IRQ21.
- EFUSE=4KB (1Kx32)
- System APB bus runs at uC APB speed: 200MHz,

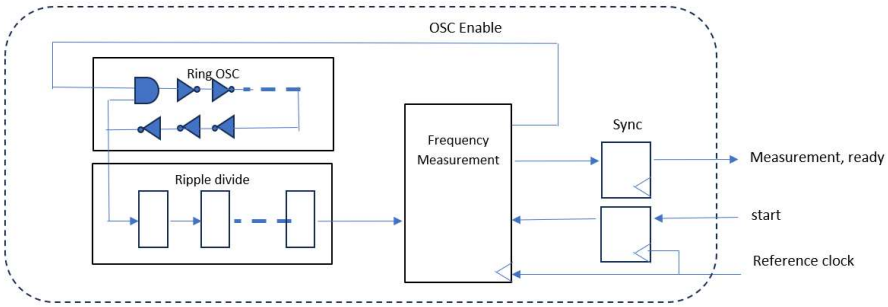
OTP controller address

Offset	Size	Usage	Comment
0x8000~0x8ffc	32K	OTP array	1KB eFuse (0x8000~0x8fff)
0x0800~0x081c		PTR	Test row address pa[0]~pa[7]
0x3400~0x3508		CFG	CSR in register section.
Others		CMD	

OTP array assignment

Offset	Size	Usage	Comment
0~0x3f	512b	AES key	
0x40~0x7f	512b	SHA key	
0x80~0x8f	128b	BIOS Straps	
0x90~0xfff		Process data and misc use.	

DRO for process monitor



Reference clock: 200MHz (uc_clk)

rosc chain: 83 stages, typical case, period= ~938ps

- Multiple Vt and/or combined cells. Harden at PAR stage.
- Ring stopped automatically after measurement complete (ready=1).

Multiple instances at different locations.

ripple: 8 stage, period = 938ps*256 = 240128ps

period count by 200Mhz clock: measurement = 240128/5000 =~ 48

CSR bit

- Start (DRO_START): bit 0 (RW)
- Measurement Ready (DRO_READY): bit 1 (RO)
- Measurement result (DRO_MEA): bit[15:8] (RO)

EMAC

Emac introduction

The EMAC module is a 10/100Mbps Ethernet MAC (Ethernet Media Access Controller) compatible with IEEE 802.3. It includes status and control register group, transceiver module, transceiver buffer descriptor group, host interface, MDIO, physical layer chip (PHY) interface.

The status and control register group contains the status bits and control bits of the EMAC, which is the interface with the user program, and is responsible for controlling data receiving and sending and reporting the status.

The transceiver module is responsible for obtaining the data frame from the designated memory location according to the control word in the transceiver descriptor, adding the preamble, CRC, and expanding the short frame before sending it out through the PHY; Or receive data from the PHY, and put the data into the designated memory according to the transmit and receive buffer descriptor. Configure related event flags after sending and receiving. If the event interrupt is enabled, the interrupt request will be sent to the host for processing.

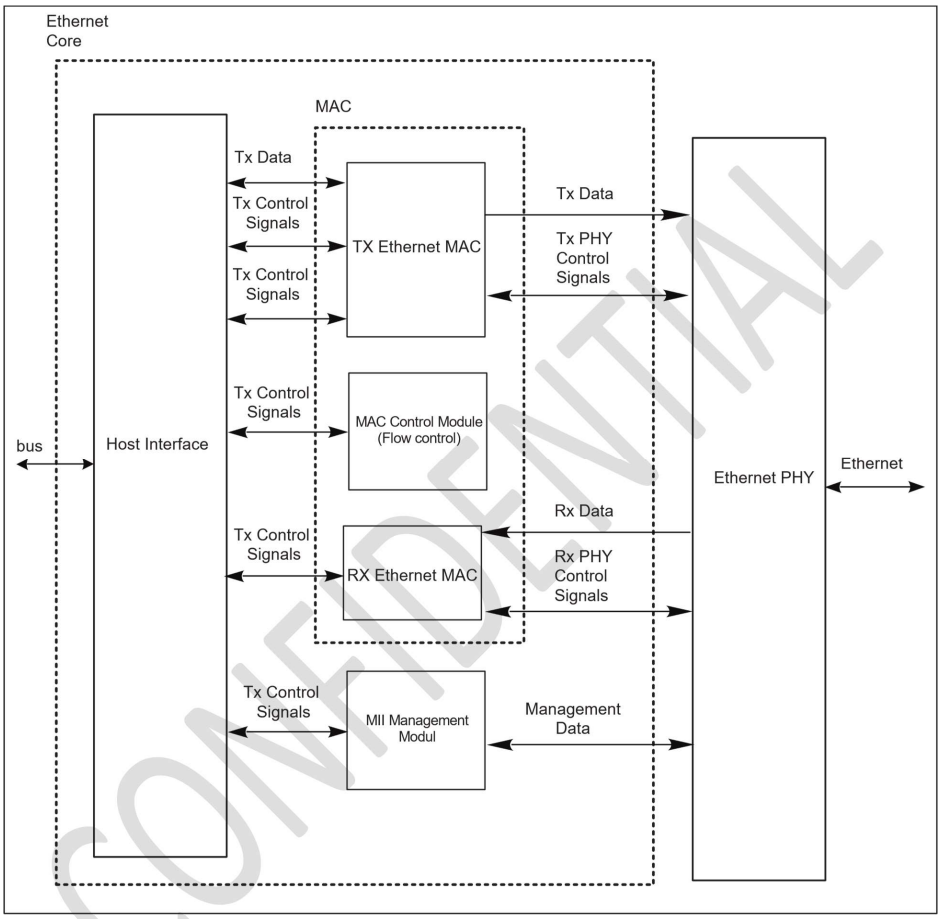
The MDIO and MII/RMII interfaces are responsible for communicating with the PHY, including reading and writing PHY registers and sending and receiving data packets.

Emac main features

- Compatible with the MAC layer functions defined by IEEE 802.3
- Support MII and RMII interface PHY defined by IEEE 802.3
- Interaction with PHY through MDIO
- Support 10Mbps and 100Mbps Ethernet
- Support half duplex and full duplex
- In full duplex mode, support automatic flow control and control frame generation
- Support collision detection and retransmission in half-duplex mode
- Support CRC generation and verification
- Data frame preamble generation and removal
- When sending, automatically expand short data frames
- Detect too long or too short data frame (length limit)
- Can transmit long data frames (> standard Ethernet frame length)
- Automatically discard packets that exceed the number of retransmissions or the frame gap is too small
- Broadcast packet filtering
- Internal RAM for storing up to 128 BD (Buffer Descriptor)
- Various event flags sent/received
- Generate a corresponding interrupt when an event occurs

Emac function description

The composition of the EMAC module is shown in the figure below



EMAC architecture

The module's control register can read and write the PHY register through MDIO to realize configuration, select mode (half/full duplex), initiate negotiation and other operations.

The receiving module filters and checks the received data frame: whether there is a legal preamble, FCS, length, etc. And according to the descriptor, the data frame is stored in the designated buffer address.

The sending module obtains data from the memory according to the data buffer descriptor, adds preamble, FCS, pad, etc., and then sends the data according to the CSMA/CD protocol.

If CRS is detected, the retry will be delayed.

The send and receive buffer descriptor group is connected to the external RAM, which is used to store the Ethernet data frames sent and received. Each descriptor contains the corresponding control status word and the corresponding buffer memory address. There are 128 groups of descriptors, which can be flexibly allocated for sending or receiving.

Emac clock

The EMAC module needs a clock for synchronous transmission and reception (at 100Mbps, 25MHz (MII) or 50MHz (RMII); at 10Mbps, 2.5MHz). This clock must be synchronized between EMAC and PHY.

Send and receive buffer descriptor (BD, Buffer Descriptor)

The transceiver buffer descriptor is used to provide the association between the EAMC and the data frame buffer address information, to control the transceiver data frame, and to provide the transceiver status prompt.

Each descriptor is composed of two consecutive words (32 bits). The low address word provides the length, control and status bits of the data frame contained in the buffer; the high address word is a memory pointer.

Specific BD description can refer to the register description chapter.

Note that: For BD, you need to write in word.

The EMAC module supports 128 BDs, which are shared by the sending/receiving logic and can be freely combined. But the sending BD always occupies the previous continuous area (the number is specified by the TXBDNUM field in the MAC_TX_BD_NUM register).

EMAC processes the sending/receiving BD in the order of BD, until it encounters the BD marked as WR, it wraps around to send/receive the respective first BD.

PHY interaction

The PHY interaction register group provides the commands and data communication methods needed for PHY interaction. EMAC controls the working mode of PHY through MDIO and ensures that the two match (rate, full/half duplex).

The data packet interacts between EMAC and PHY through the MII/RMII interface, and can be selected by RMII_EN in the EMAC mode register (EMAC_MODE). When this bit is 1, select RMII mode, otherwise it is MII mode.

Both MII and RMII modes support the 10Mbps and 100Mbps transmission rates specified in the IEEE 802.3u standard.

The transmission signal description of MII and RMII is shown in the table below.

Transmission signal

Name	MII	RMII
EXTCK_EREFCK	ETXCK: send clock signal	EREFCK: reference clock
ECRS	ECRS: carrier detection	-
ECOL	ECOL: collision detection	-
ERXDV	ERXDV: data valid	ECRSDV: Carrier detect/data valid
ERX0-ERX3	ERX0-ERX3: 4-bit receive data	ERX0-ERX1: 2-bit receive data
ERXER	ERXER: Receive error indication	ERXER: Receive error indication
ERXCK	ERXCK: Receive clock signal	-
ETXEN	ETXEN: transmit enable	ETXEN: transmit enable
ETX0-ETX3	ETX0-ETX3: 4-bit transmit data	ETX0-ETX1: 2-bit transmit data
ETXER	ETXER: Send error indication	-
EMDC	MDIO Clock	MDIO Clock
EMDIO	MDIO Data Input Output	MDIO Data Input Output

The RMII interface has fewer pins, and a 2-bit data line is used for receiving and sending. At a rate of 100Mbps, a 50MHz reference clock is required.

Programming process

PHY initialization

- According to the PHY type, set the RMII_EN bit in the EMAC_MODE register to select the appropriate connection method
- Set the MAC address of EMAC to EMAC_MAC_ADDR0 and EMAC_MAC_ADDR1
- Set the appropriate clock for the MDIO part by programming the field CLKDIV in the EMAC_MIIMODE register
- Set the corresponding PHY address to the FIAD field of the register EMAC_MIIADDRESS
- According to the PHY manual, send commands through the EMAC_MIICOMMAND and EMAC_MIITX_DATA registers
- The data read from the PHY will be stored in the EMAC_MIIRX_DATA register

- The status of interaction with PHY commands can be queried through the EMAC_MIISTATUS register

After the basic interaction is completed, the PHY should enter the auto-negotiation state. After the negotiation is completed, program the mode to the FULLD bit in the EMAC_MODE register according to the negotiation result.

Send data frame

- Configure bit fields such as data frame format and interval in the EMAC_MODE register
- By configuring the TXBDNUM field in the EMAC_TX_BD_NUM register to specify the number of BDs used for transmission, the remaining BDs are RX BDs
- Prepare the data frame to be sent in the memory
- Fill in the address of the data frame into the data pointer field (word 1) corresponding to the sent BD
- Clear the status flag in the control and status field (wrod 0) corresponding to the sent BD, and set the control field (CRC enable, PAD enable, interrupt enable, etc.)
- Write the length of the data frame and set the RD field to inform EMAC that this BD data needs to be sent; if necessary, set the IRQ bit to enable interrupts
- In particular, if it is the last BD sent, the WR bit needs to be set, and EMAC will "wrap around" to the first sent BD for processing after processing this BD
- If there are multiple BDs to be sent, repeat the steps of setting BD to fill all sending BDs
- If you need to enable the transmit interrupt, you also need to configure the TX related bits in the EMAC_INT_MASK register
- Configure the TXEN bit in the EMAC_MODE register to enable transmission
- If the interrupt is enabled, in the interrupt sent, the current BD can be obtained through the TXBDNUM field in the EMAC_TX_BD_NUM register
- Perform corresponding processing according to the current BD status word
- For the BD whose data has been sent, the RD bit in the control field will be cleared by hardware and will not be sent again; after filling in new data, set RD, and this BD can be used for sending again

Receive data frame

- Configure bit fields such as data frame format and interval in the EMAC_MODE register
- By configuring the TXBDNUM field in the EMAC_TX_BD_NUM register to specify the number of BDs used for transmission, the remaining BDs are RX BDs
- The area in the memory that is ready to receive data
- Fill in the address of the data frame into the data pointer field (word 1) corresponding to the received BD
- Clear the status flag in the control and status field (wrod 0) corresponding to the sending BD, and set the control field (interrupt enable, etc.)
- Write the receivable data frame length and set the E bit field to inform EMAC that the BD is idle and can be used for data reception; if necessary, set the IRQ bit to enable interrupts

- In particular, if it is the last valid receiving BD, the WR bit needs to be set, and EMAC will "wrap around" to the first receiving BD for processing after processing this BD
- If there are multiple BDs available to receive data, repeat the steps of setting BD to fill all BDs
- If you need to enable the receive interrupt, you also need to configure the RX related bits in the EMAC_INT_MASK register
- Configure the RXEN bit in the EMAC_MODE register to enable reception
- If the interrupt is enabled, in the received interrupt, the current BD can be obtained through the RXBDNUM field in the EMAC_TX_BD_NUM register
- Perform corresponding processing according to the current BD status word
- For the received BD, the E bit in the control field will be cleared by hardware and will not be used for receiving again; the data needs to be taken away, and E is set, this BD can be used for receiving again

3.4 CXL, PCIe

3.4.1 PCIe PHY

Alphawave PCIe PHY CSR access:

PHY selected by “pcie_sel”: 0=RCx8, 1=host1, 2=host2, 3=host3, 4=host4, 5=RCx4

RC interface is 4/8 lane, CXL/PCIe 5 root complex, connects to external switch or NVMe SSD.

Host interface is 8 lane, CXL endpoint, connects to hosts

PMA CSR: 0x2800_0000~0x283f_ffff.

Address [23:12]	Address[11:0]	Comment
0x000	CSR offset	Common Register
0x010	CSR offset	RX register
0x011	CSR offset	TX register
0x012	CSR offset	ETH register
0x013	CSR offset	DFX register

PMA SRAM: 0x2840_0000~0x287f_ffff. SRAM configuration = 4kx32 (per PHY).

PCS CSR: 0x2881_0000~0x28af_ffff. Address bit 21 = broadcast. Address bit[20:16]= lane ID.

Address [23:12]	Address[11:0]	Comment
0x810	CSR offset	Lane 1
0x820	CSR offset	Lane 2
0x830	CSR offset	Lane 3
0x840	CSR offset	Lane 4
0x850	CSR offset	Lane 5 (host phy only)
0x860	CSR offset	Lane 6 (host phy only)
0x870	CSR offset	Lane 7 (host phy only)
0x880	CSR offset	Lane 8 (host phy only)
0x8a0	CSR offset	Broadcast

Note: No PCS SRAM

PCIe MAC (EP0..EP3, RC) AXI slave port: MAC CSR and PCIe/CXL access (e.g., NVMe door bell, RC CXL.mem, RC CXL.io)

Synopsys PCIE/CXL controller DBI

PCIe MAC access:
Base address = 0x29000000
AXI DBI address A[23:0] = A[23:0]
AXI DBI address A[29:24] = 0
AXI DBI address A[30] = 0
AXI DBI address A[31] = 0

Note: PCIe/CXL interfaces

interface	MAC's role	protocol	clock	Data bus width	comment
EP cxl.mem	master	Native	synchronous	512b	cxl-axi bridge to flc1
EP cxl.io	master	AXI	asynchronous	256b	AXI ID=8b
EP cxl.mem	slave	Native	synchronous	512b	Not used
EP cxl.io	slave	AXI	asynchronous	256b	AXI ID = 0b
RC cxl.mem	master	Native	synchronous	512b	Not used
RC cxl.io	master	AXI	asynchronous	128b	AXI ID=8b
RC cxl.mem	slave	Native	synchronous	512b	Axi-cxl bridge to nic/noc
RC cxl.io	slave	AXI	asynchronous	128b	AXI ID=8b

Note 1: "synchronous" mode: I.e., PCIe/CXL MAC does not include CDC layer.
Question: AXI ID width okay for master (8b)/ slave (16b) interface?

ELBI access

For type 3 CXL device below registers are to be implemented outside of the controller and through ELBI interface.

1. HDM Capability is to be implemented. - BAR0
2. CXL Device Registers (mailbox) -BAR0
3. DOE Capability Registers – CFG space.
4. Intel DVSEC – CFG Space
5. PCIE DVSEC for Test Capability - CFG Space, (Address Registers are in mem space but, we are not reporting/implementing any test capability (so all 0s)).

ELBI access is required for CXL2.0 .mem to implement CXL HDM decoder. MAC glue logic will route ELBI access to SRAM. Address is determined by: {ELBI_SRAM_BASE[2:0],slice#[1:0], SRAM_OFFSET[10:0]}. See the following table:

Registers	SRAM Address range	ELBI address
Memory mapped (BAR0) CXL Capability Registers	0-15	0x1010-0x101f
Memory mapped (BAR0) HDM decoder Capability and other Capability can be inserted here.	16-511	0x1200- (1200+496 bytes)
CFG mapped, any DVSEC can be inserted here.	512- 512+256 bytes	BAR=3'b111, address starts from **0xd00** of config address, (Intel NDR, Test DVSEC)
BAR1 mapped, for application registers (Here also BAR is configurable	768 ~ 768+256	BAR is configurable, size is configurable (16Bytes) granularity. Default BAR1 is decoded here, size is 256bytes.
BAR is configurable	1KB~2KB	1. 64bytes of CXL Device Capabilities Registers and 2. CXL Status Registers (size 8bytes) 3. Mailbox (512+32bytes) 4. CXL Memory Device Capabilities (8bytes) Total = 616bytes

Total SRAM area occupied = 2KB/EP, total 4 endpoints = 8KB.

ELBI_BASE is defined in uC CSR "SYS_CTRL" 0x26000008, bit[26:24] with 8KB allocation.

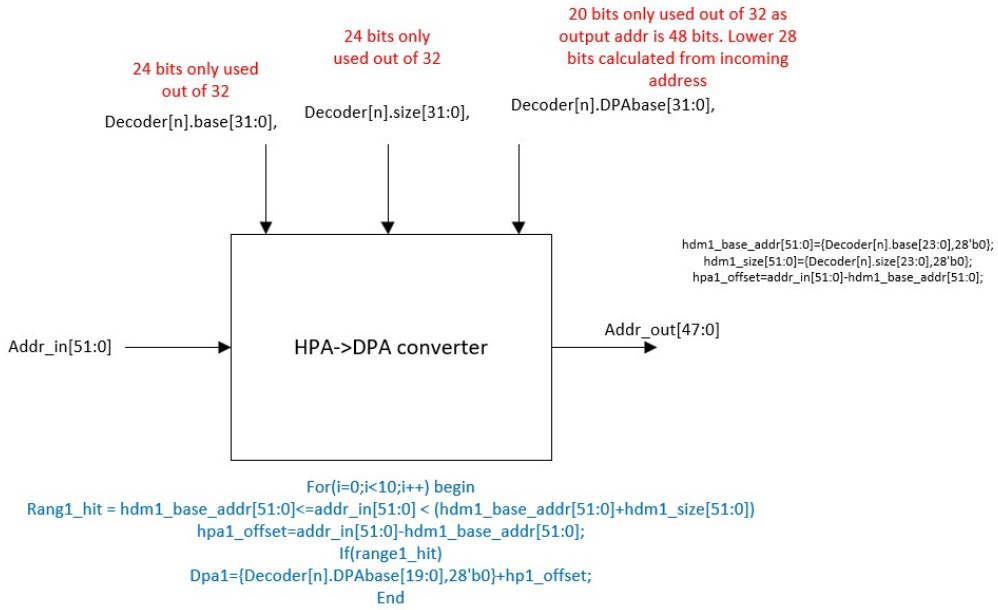
26:24	ELBI_SRAM_BASE	0xf	RW	ELBI access base address in SRAM with 8KB allocation
27	Reserved			
31:28	ELBI_ACC_FLAG[3:0]	0	RWc	Set to "1" when ELBI write access. Write 1 to clear.

Address to SRAM:

A[31:16]	A[15:13]	A[12:11]	A[10:0]
0x4000	ELBI_SRAM_BASE	Slice#	offset

When there is an ELBI access, ELBI_ACC_FLAG[EP] is set to “1” and uC can be notified via interrupt. uC can write “1” to corresponding ELBI_ACC_FLAG bit to clear the latched flag.

New HDM decoder format (06-01-2023)



Reference from MAC specification:

Table 3-18 CXL 2.0 Component Specific Register Space Implementation

Range	Size	Destination	Implementation
0000_0000h to 0000_0FFFh	4k	CXL.io registers	External (ELBI)[a]
0000_1000h to 0000_100Fh	4K	CXL.cache and CXL.mem registers	Internal
0000_1010h to 0000_101Fh[b]			External (ELBI)
0000_1020h to 0000_11FFh			Internal
0000_1200h to 0000_1FFFh[c]			External (ELBI)
0000_2000h to 0000_DFFFh	48K	Implementation specific registers	External (ELBI)
0000_E000h to 0000_E3FFh	1K	CXL ARB/MUX registers	Internal
0000_E400h to 0000_FFFFh	7K	Reserved	External (ELBI)

a. The output signal lbc_ext_cxl_mbar0_access indicates accesses to the ELBI mapped regions of the CXL 2.0 Port Specific Component Registers. You can use signal lbc_ext_cxl_mbar0_access along with lbc_ext_addr[15:0] to decode accesses to ELBI mapped CXL 2.0 Port Specific Component Registers region.
b. You must implement this address range on ELBI for CXL_HDM_Capability_Header, CXL_Extended_Security_Capability_Header, CXL_IDE_Capability_Header, and CXL_Snoop_Filter_Capability_Header registers.
c. You must implement this address range on ELBI for CXL_HDM_Capability, CXL_Extended_Security_Capability, CXL_IDE_Capability, and CXL_Snoop_Filter_Capability registers.

Figure 4-26 ELBI Transaction: 32-bit Write Access to External Registers

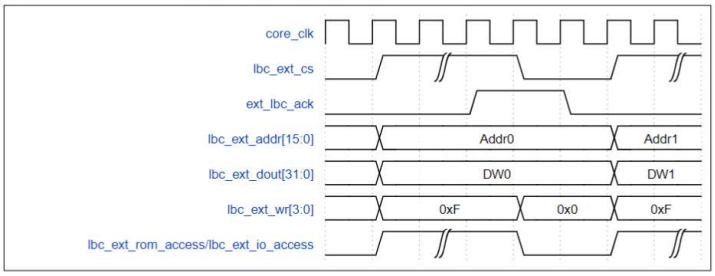
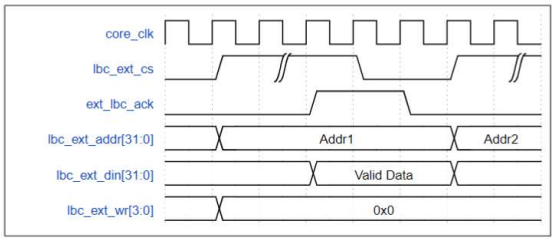


Figure 4-27 ELBI Transaction: 32-bit Read Access to External Registers



QM SRAM access

(JIRA:MOC-7)

QM's submission queue (SQ) and completion queue (CQ) are placed in a SRAM (size: 8KB) within QM module.

This SRAM is mapped into QM's address space and QM's queue handler SM will access this SRAM via APB.

We need to provide a path from PCIe RC to access this SRAM (read/write queue contents), the path is:

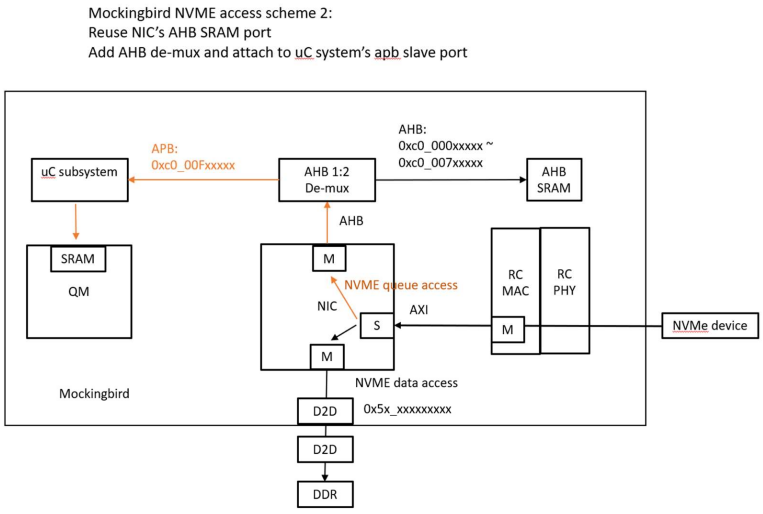
PCIe RC (AXI master) -> NOC -> uC bus matrix -> QM's apb slave port -> QM SRAM

Note the same PCIe RC's AXI master interface also carries data transfer, which is:

PCIe RC (AXI master) -> NOC -> DDR4 MC

(NIC will route traffic based on RC's address to either MC or uC…)

Note: QM SRAM address is 0xc0_21fx_xxxx (1MB). QM0=0xc0_21f0_xxxx (64KB), QM1=0xc0_21f1_xxxx (64KB), QM2=0xc0_21f2_xxxx (64KB), QM3=0xc0_21f3_xxxx (64KB).



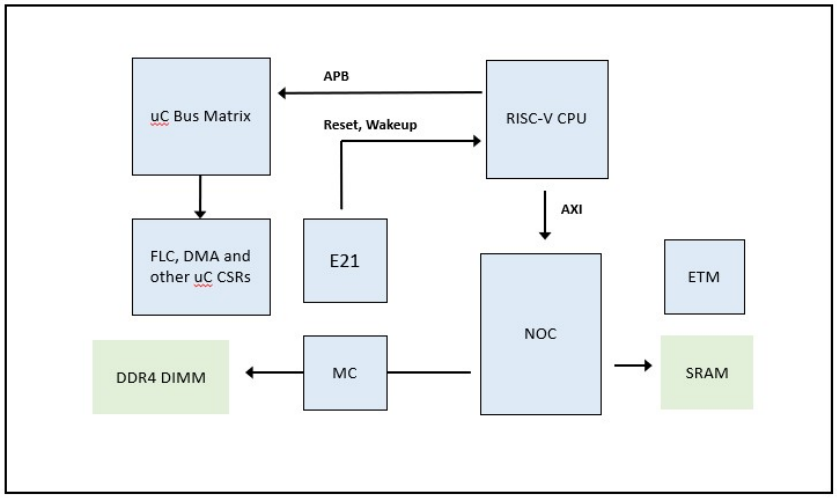
AHB SRAM

AHB SRAM connected to NOC; size16Kx32 (total=64KB), running at NOC clock (=1Ghz). It is shared by various units and served as system storage. E21 FW manages SRAM usage via CSR.

Offset	Size	Usage	CSR	Comment
0xc000~0xffff	16K	Reserved		For Andes boot code
0xa000~0xbfff	8K	Reserved		For RC use
0x8000~9fff	8K	PCIe EP config/MMIO registers.	(SYS_CTRL) 0x26000008 ELB_SRAM_BASE	PCIe MAC AXI interface, 2K per endpoint for CXL.io. Slice# appended to A[12:11]
0x4000~0x7fff	16K	NVMe management queue		For management queue use. Data and response queues are handled by QM
0x80~0x3fff	~16K	Reserved		
0~0x7f	128B	PM message		16B/message per device. 8x16B=128B message buffer for EP (4x) or RC (4x).

RISCV: AE350 platform

RISCV CPU connection Diagram



RISCV OS image in DDR4 DIMM. Bare-metal test image can be in SRAM or DDR4.

Andes address space: 0xF3000000~0xFFFFFFFF mapped into E21's CSR

RISCV	E21	Comment
0xF3xx_xxxx	0x21xx_xxxx	QM
0xF[4..7]00_xxxx	0x2[2..5]00_xxxx	IPM MC0
0xF[4..7]01_xxxx	0x2[2..5]01_xxxx	FLC1_0
0xF[4..7]02_xxxx	0x2[2..5]02_xxxx	FLC2
0xF[4..7]03_xxxx	0x2[2..5]03_xxxx	AXI Master
0xF[4..7]04_xxxx	0x2[2..5]04_xxxx	IPM MC1
0xF[4..7]05_xxxx	0x2[2..5]05_xxxx	FLC1_1
0xF8xx_xxxx	0x26xx_xxxx	uC system peripherals
0xFAxx_xxxx	0x28xx_xxxx	PCIE PHY
0xFBxx_xxxx	0x29xx_xxxx	PCIE_MAC
0xFC00_xxxx	0x2a00_xxxx	I3C master
0xFC01_xxxx	0x2a01_xxxx	I3C master
0xFC02_xxxx	0x2a02_xxxx	AXI4TG
0xFC03_xxxx	0x2a03_xxxx	ETM
0xFDxx_xxxx	0x2bxx_xxxx	EMAC

Table 151: AE350 Memory Map

Address		Description
Begin	End	
0x00000000	0x7FFFFFFF	RAM Bridge
0x80000000	0x87FFFFFF	SPI1 AHB Memory/ Reset Vector
0xA0000000	0xA01FFFFF	Hart 0 Local Memory Slave Port: ILM
0xA0200000	0xA03FFFFF	Hart 0 Local Memory Slave Port: DLM
0xA0400000	0xA05FFFFF	Hart 1 Local Memory Slave Port: ILM
0xA0600000	0xA07FFFFF	Hart 1 Local Memory Slave Port: DLM
0xA0800000	0xA09FFFFF	Hart 2 Local Memory Slave Port: ILM
0xA0A00000	0xA0BFFFFF	Hart 2 Local Memory Slave Port: DLM
0xA0C00000	0xA0DFFFFF	Hart 3 Local Memory Slave Port: ILM
0xA0E00000	0xA0FFFFFF	Hart 3 Local Memory Slave Port: DLM
0xC0000000	0xC00FFFFF	BMC
0xC0200000	0xC02FFFFF	DFS
0xE0000000	0xE00FFFFF	AHB Decoder
0xE0500000	0xE05FFFFF	L2C
0xE4000000	0xE43FFFFF	PLIC
0xE6000000	0xE60FFFFF	Machine Timer
0xE6400000	0xE67FFFFF	PLIC-SWINT
0xE6800000	0xE68FFFFF	Debug Module/ Debug Vector
0xF0000000	0xF00FFFFF	APBBRG
0xF0100000	0xF01FFFFF	SMU
0xF0200000	0xF02FFFFF	UART1
0xF0300000	0xF03FFFFF	UART2
0xF0400000	0xF04FFFFF	PIT
0xF0500000	0xF05FFFFF	WDT
0xF0600000	0xF06FFFFF	RTC
0xF0700000	0xF07FFFFF	GPIO

Continued on next page...

Table 151: (continued)

Address		Description
Begin	End	
0xF0A00000	0xF0AFFFFF	I2C
0xF0B00000	0xF0BFFFFF	SPI1
0xF0C00000	0xF0CFFFFF	DMAC
0xF0F00000	0xF0FFFFFF	SPI2
0xF2000000	0xF20FFFFF	DTROM
0x100000000	0x1FFFFFFFF	Uncacheable alias to region 0x00000000 - 0xFFFFFFFF. The data in cacheable regions will be cached into L1 caches of the processor. This region is an uncacheable alias to the cacheable regions. Accesses to this region will bypass the L1 caches. This is useful when the processor and external bus masters need to share data in the same memory space. This region is present only when BIU_ADDR_WIDTH is 33 bits.

Note

- The RAM bridge space indicates the size of the RAM behind this bridge. It can be different from the size of the address space allocated to the bridge on the bus. The default setting allocates a 2GiB space (0x00000000 – 0x7FFFFFFF) to the bridge on the bus. When the bridge sees a transaction for addresses outside of the addressable RAM, it will return an error response.
- In addition to the bus view described here, ILM/DLM are accessible by the processor through private address spaces visible only to the processor:
  - The address ranges of these private address spaces are controlled by `milmb.IBPA` and `mdlmb.DBPA`.
  - The private address spaces have higher priority than the bus address spaces in the processor. Accesses will be directed to go through the local memory interfaces and bypass the bus address spaces if they hit the private address spaces.
  - If overlapping of address spaces is not desired, the `milmb` and `mdlmb` control registers could be programmed to avoid overlapping.
  - ILM is visible to the processor at 0x00000000 in the default setting.
  - DLM is visible to the processor at 0x00200000 in the default setting.
  - Debug Module region will be mapped to the default slave when macro `PLATFORM_NO_DEBUG_SUPPORT` is defined

0x10000000 = DLM base (256KB)

Andes core interrupts:

	Source	Note
1	RTC_PERIOD	
2	RTC_ALARM	
3	PIT	Programmable Interval Timer
4	SPI1	NA
5	SPI2	NA
6	I2C	NA
7	GPIO	
8	UART1	
9	UART2	NA
10	DMA	
15:11	0	
16	L2C_ERR	

17	SSP	NA
18	SDC	NA
19	MAC	NA
20	LCD	NA
21	FLC1	
22	FLC2	
23	IPM_MC	
24	IPM_PHY	
25	AXI_DMA	
26	PCIE PHY	
27	PCIE MAC	
28	ETM	
29	E21	
30	EMAC	
31	0	

RISC-V CPU boot up procedure.

1. Download uBoot and OS image from host (via PCIe) to DDR or on-die SRAM
2. As an alternative, E21 can download RISC-V code from QSPI SF (lower performance).
3. E21 setup reset vector in CSR: 0x26000014~0x2600001c.
4. E21 release RISC-V reset.

- Sample code: RISC-V core 0 boot from SRAM, address 0x84

W 26000018, 0x000c0000 //reset vec base = 0xc0_00000000 (SRAM)

W 26000014, 0x00085001 //reset vec offset = 0x84, release core 0 reset.

Reset vector calculation

- CSR: sysctrl_riscv = 0x26000014.
- CSR: sysctrl_riscv_rstvec = 0x26000018
- CSR: sysctrl_riscv_rstvec1=0x2600001c
  - When sysctrl_riscv[12]=0, “core0_reset_vector” = 0, else “core0_reset_vector” = {sysctrl_riscv_rstvec[11:0],sysctrl_riscv[31:24],2’b0}
  - When sysctrl_riscv[13]=0, “core1_reset vector” = 0, else “core1_reset_vector” = {sysctrl_riscv_rstvec1[31:2],2’b0}

To match NOC decoder, high bits of reset vector added to both cores before sending transactions to NOC.

Final address to NOC:

- Core0 access = {sysctrl_riscv_rstvec[26:12],32’h0} + “core0_reset_vector”.

- Core1 access = {sysctrl_riscv_rstvec[26:12],32'h0} + "core1_reset_vector".

ETM buffer access: ETM SRAM is mapped at NOC with fixed location: 0xa[3:0]_00003fff.

Reg 0x26000010 bit[31:16] (riscv_etm) defines etm access region from Andes core to ETM.

riscv_etm[31] = ETM region enable.

riscv_etm[30:16] = region's riscv address[30:16].

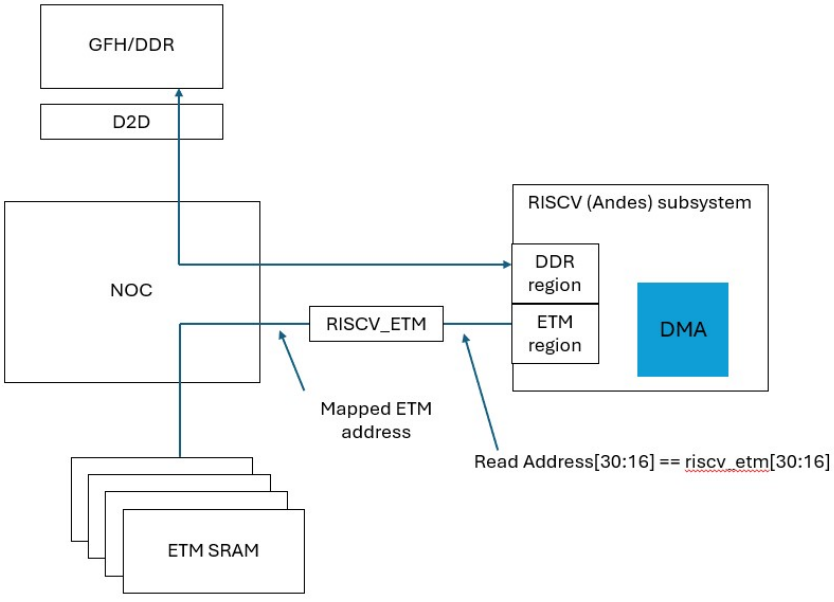
Region is 64K aligned.

ETM region is in lower 2GB address space, i.e. riscv address[31]=0.

ETM's address map in NOC.

ETM0	16K	0xa0_00000000	0xa0_00003fff	
ETM1	16K	0xa1_00000000	0xa1_00003fff	
ETM2	16K	0xa2_00000000	0xa2_00003fff	
ETM3	16K	0xa3_00000000	0xa3_00003fff	

Mapped address to NOC = {4'ha, 2'b0, Andes address[15:14], 18'h0, Andes address[13:0]}



Inter-chiplet notification

Companion chiplets -> Mockingbird, E21 IRQ

40	D2D_0_0	DDR chiplet #1, slice 0
41	D2D_1_0	DDR chiplet #2, slice 0
42	D2D_2_0	DDR chiplet #3, slice 0
43	D2D_3_0	DDR chiplet #4, slice 0
44	MKB_D2D_0	Inter MKB D2D
45	MKB_D2D_1	Inter MKB D2D

E21 CSR: d2d_irq, address 0x2600_0060

9:0	d2d_irq_o	RW	D2D Chiplet IRQ output. Bit 0..3 = Goldfinch [3:0], slice 0 Bit 4 = MKB_D2D_0 Bit 5 = MKB_D2D_1 Bit 6..9 = Reserved
19:10	d2d_irq_i	R	D2D Chiplet IRQ input status. Corresponding to E21 IRQ[47:40, 37:36] Bit 0..3 = Goldfinch [3:0], slice 0 Bit 4 = MKB_D2D_0 Bit 5 = MKB_D2D_1 Bit 6..9 = Reserved
31:20	Reserved		

Virtual Wires: D2D port “NMI_in” and “NMI_out” are used.

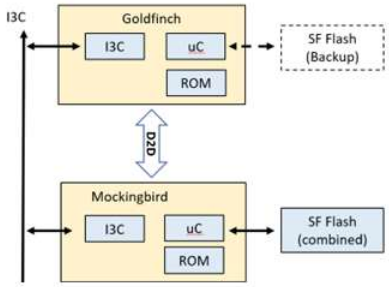
e.g.

Mockingbird -> Goldfinch:

(Mockingbird) E21 CSR (d2d_irq) -> NMI_in (u_d2d_top_wrapper) -> … D2D … -> NMI_out (u_d2d_top_wrapper) -> E21 IRQ/CSR (Goldfinch)

Goldfinch -> Mockingbird:(Goldfinch) E21 CSR -> NMI_in (u_d2d_top_wrapper) -> … D2D … -> NMI_out (u_d2d_top_wrapper) -> E21 IRQ/CSR (Mockingbird)

Chiplet Flash configuration



- Internal ROM for I3C and D2D init
- Combined SF flash for all chiplets' FW.
- Goldfinch uC read chiplet-specific FW through D2D or I3C.
- Keep SF interface for chiplets other than Mockingbird as backup option.

FW init sequence

Step		Source	Comment
1	Power up/reset	ROM and SF Flash	Each chiplet's uC fetch D2D and I3C initialization code from internal ROM.
2	D2D init		uC programs D2D and I3C. I3C is ready after this step.
3	D2D Sync	I3C	uC trains inter-D2D timing with I3C. D2D is ready after this step.
4	Chiplet init	SF Flash	For Mockingbird ~~and Robin~~: uC fetch init code from SF Flash and perform FLC, PCIE, IPM init. For other chiplets: Mockingbird~~/Robin~~ uC executes chiplet system init code through D2D. I3C used for interrupt messages.

QSPI flash table

QSPI connected to Mockingbird and stores PHY FW, FSBL and APP code for mockingbird system. Following sample table based on 32MB QSPI.

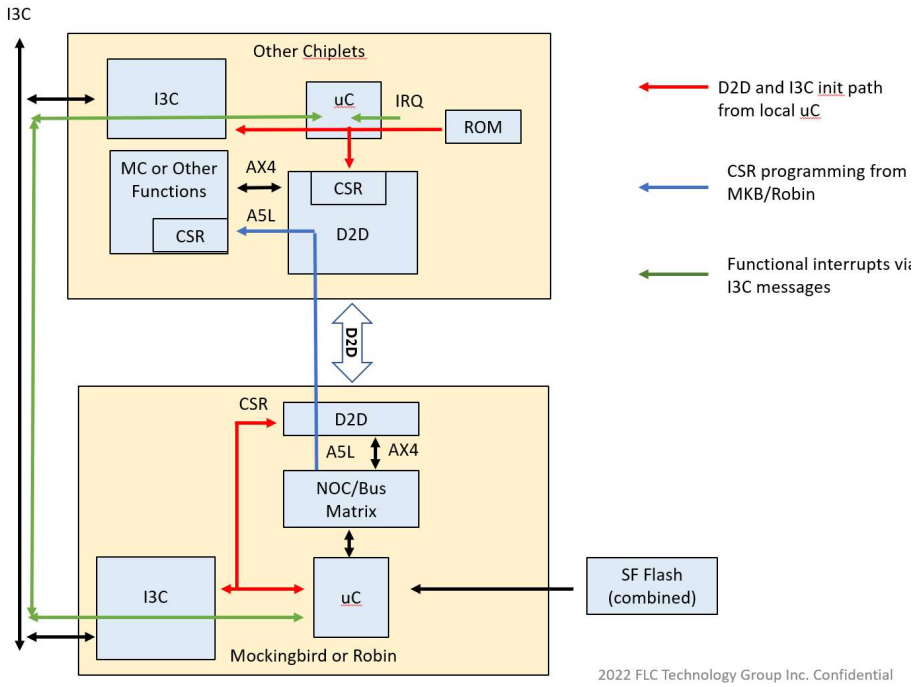
Offset	Size	Usage	Comment
0x0		I3C Driver	
		D2D Driver	
	128K	...	Reserved for other drivers
0x20000	256K	AW PHY FW	
0x60000	256K	IPM PHY FW	
0xa0000	256K	DDR PHY FW	

0xe000	128K		Reserved
0x1_00000	1MB	Free RTOS	E21
0x2_00000	29MB	Linux/Bare metal	Andes
0x1f_00000	1MB	Memory repair	

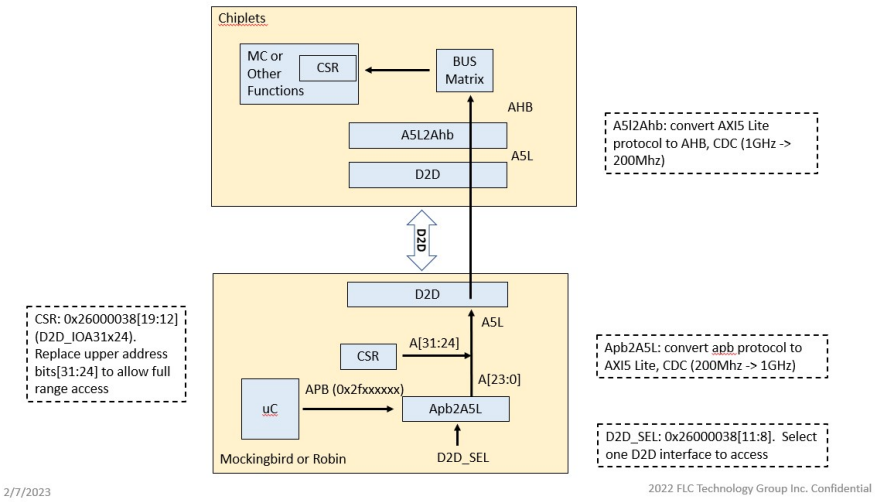
QSPI connected to Mockingbird and stores PHY FW, FSBL and APP code for mockingbird system. Following sample table based on 256MB QSPI.

Offset	Size	Usage	Comment
0x0		I3C Driver	
		D2D Driver	
	128K	...	Reserved for other drivers
0x20000	256K	AW PHY FW	
0x60000	256K	IPM PHY FW	
0xa0000	256K	DDR PHY FW	
0xe000	128K		Reserved
0x1_00000	1MB	Free RTOS	E21
0x2_00000	238MB	Linux/Bare metal	Andes
0xf0_00000	16MB	Memory repair	

Transaction Flow



Detailed Block Diagram



2/7/2023

Chiplet test mode, AXI traffic generator

uC subsystem includes a traffic generator to test IPM and D2D interface without FLC and NOC.

D2D AXI4TG

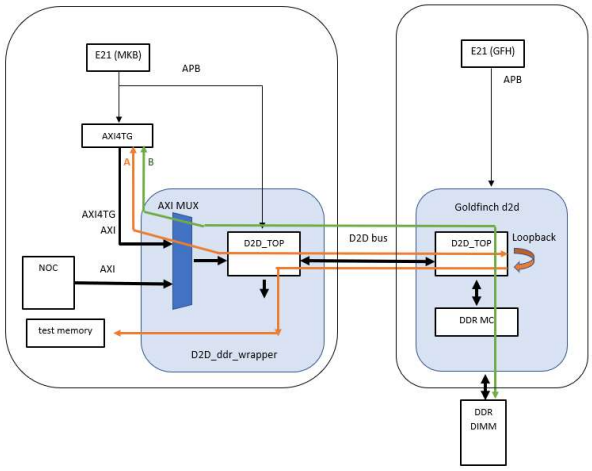
Build in AXI4 traffic generator to test D2D, bypass NOC.

Test flow A with D2D loopback

1. E21 configures D2D.
2. E21 programs pattern in AXI4TG
3. AXI4TG generates AXI4 write traffic through d2d_ddr_wrapper to GFH d2d (loopback), write to test memory
4. AXI4TG generates AXI4 read traffic through d2d_ddr_wrapper to GFH d2d (loopback), read from test memory
5. E21 checks AXI4TG read buffer
6. Test memory: AHB memory at E21 address space: 0x40xx_xxxx

Test flow B without D2D loopback

1. E21 configures D2D, DDR MC
2. E21 programs pattern in AXI4TG
3. AXI4TG generates AXI4 write traffic through d2d_ddr_wrapper to GFH d2d, write to DDR
4. AXI4TG generates AXI4 read traffic through d2d_ddr_wrapper to GFH d2d, read from DDR
5. E21 checks AXI4TG read buffer

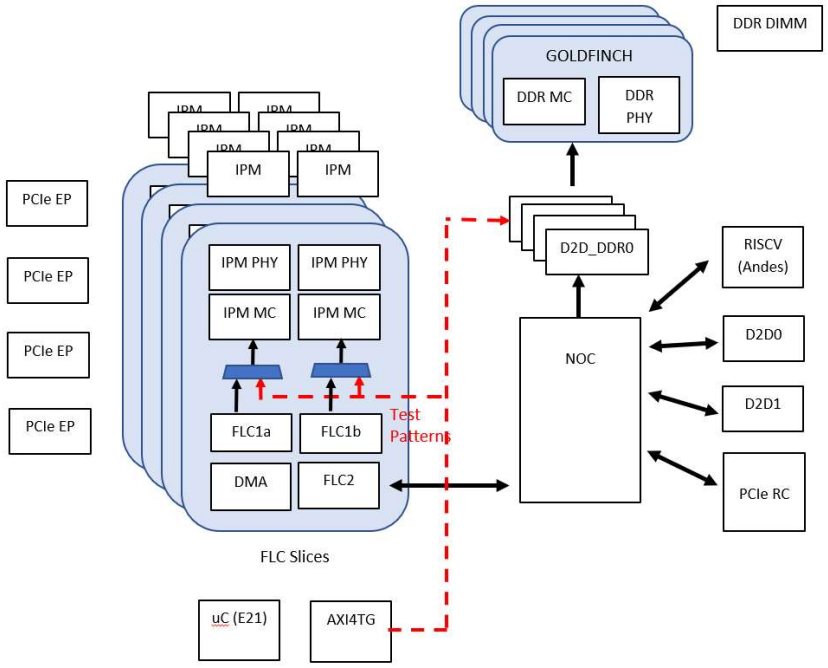


IPM AXI4TG

Build in AXI4 traffic generator to test IPM, bypass FLC1.

APB sequence:

1. E21 init IPM PHY, MC
2. E21 programs AXI4TG module.
3. E21 selects one of the IPM to test via CSR. There are total of 9 IPMs in Mockingbird.
4. E21 sets FLC1_MC_TOP's AXI MUX to select path from AXI4TG AXI.
5. E21 set CSR bit to start generating traffic at IPM MC's AXI interface.
6. After test done, E21 access AXI4TG to check result.



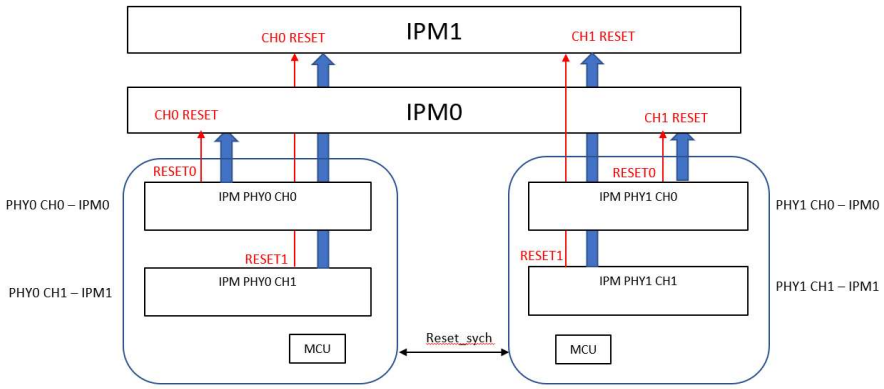
AXI4TG CSR

AXI4TG IP base addres: 0x2a02_0000
E21 CSR: AXI4TG_CTRL (0x2600_0418)

bit		RW	default	Comment
0	Axi4tg_en	RW	0	
1	Axi4tg_start	RW	0	
6:2	Axi4tg_sel	RW	0	DUT selection. 0={slice0, ipm0}, 1={slcie0, ipm1}, 2={slice1, ipm0}, 3={slice1, ipm1}, 4={slice2, ipm0}, 5={slcie2, ipm1}, 6={slice3, ipm0}, 7={slice3, ipm1}, 8={Goldfinch, d2d0, 1st slice}, 9={Goldfinch, d2d1, 1st slice}, 10={Goldfinch, d2d2, 1st slice}, 11={Goldfinch, d2d3, 1st slice}, 12=mkb d2d0, 13=mkb d2d1.
29:5	Reserved		0	
30	Axi4tg_irq	R	0	
31	Axi4tg_err	R	0	

Interleaved IPM

- IPM PHY and memory connections are interleaved for balanced RDL routing.
- When IPM1 is not installed, only ch0 of each PHY is initialized.



IPM init sequence with PHY MCU

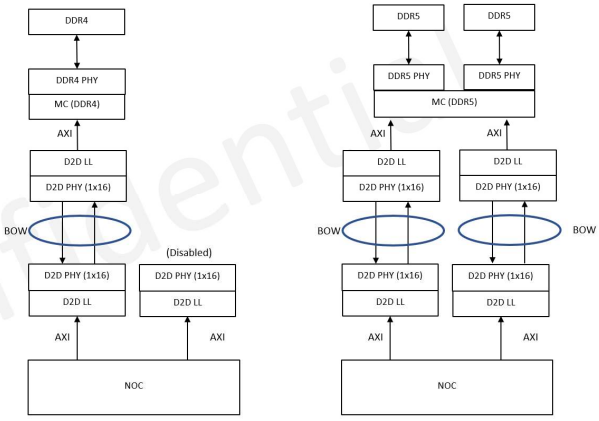
1. VDD=1, PHY.boot_i=1, PHY.mcu_core_rstn=0, MC.clk_resetn=0
2. E21 load FW into PHY (via APB bus).
3. PHY.mcu_core_rstn=1 (E21 programs CSR 0x26000064, bit 0 = 1)
4. wait 0.5ms
5. PHY.boot_i=0, MC.clk_resetn=1(E21 programs CSR 0x26000064, bit 5 = 0)
6. MC csr init (by E21)

Goldfinch D2D interface

~~For future chiplet with 64GB/s, change D2D module from 2x16 D2D to 2x(1x16). Add extra AXI4 interface to NOC. No change in current GFH.~~

Mockingbird D2D interface to Goldfinch

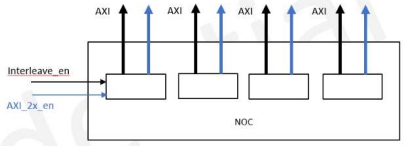
- 256b AXI @1Ghz, peak BW = 256Gb/s
- 1x16 D2D peak BW = 256Gb/s.
- To support 2x16 D2D, 2x AXI bus to be used for NOC connection.
- E21 FW controls enable/disable of 2nd D2D LL/PHY
- DDR5 MC supports 2 AXI interfaces, with arbitration among DDR5 channels.
- Client chiplet (LPDDR5) CMN option:
  - single 256b AXI @ 2GHz



NOC_CFG: E21 CSR: 0x26000008 (sysctrl) bit [23:15] controls noc interleave and other options.

NOC: Mockingbird-Goldfinch D2D interleave

- Interleave_en[1:0]
  - 00 : Interleave disabled.
  - 01: Interleave 2 targets.
  - 10: Interleave 4 targets.
- AXI_2x_en
  - 0: single AXI link
  - 1: dual AXI link



Simulation setup

CONFIG_SPI: External SPI agent to access uC bus matrix. At beginning of simulation, test case in "../spi_cfg/*hex" is loaded automatically to start. It'll not be included in chip tapeout. In source code, definition "REMOVE_CONFIG_SPI" will remove this module.

Command format:
// Store the serial in packet
// 112 bit packet: | Comand[7:0] | Address[39:0]| DataIn[31:0] | DataOut[31:0]
// Comand
// READ: 0x20
// WRITE: 0xA0
// Wait: 0x10 "Address" field is # cycle to wait.

Example:
A000_2001_0010_0000_0010_0000_0000 //0 w 0x20010010 0x00000010, gpio4 oe

32KHz clock speed up: program SYS_CTRL (0x26000008) bit 2 ("Simulation_speed_up") = 1.

I3C previsional ID register

SLV_MIPI_ID_VALUE

- **Name:** Provisional ID Register
- **Description:** I3C MIPI Manufacturer ID Register.
  This register is used in slave mode of operation.
- **Size:** 32 bits
- **Offset:** 0x70

31:16	15:1	0
Rsvd	SLV_MIPI_MFG_ID	SLV_PROV_ID_SEL

Table 20. Fields for Register: SLV_MIPI_ID_VALUE

Bits	Name	Memory Access	Description
31:16			**Reserved Field:** Yes
15:1	SLV_MIPI_MFG_ID	R/W	Specifies the MIPI Manufacturer ID. (PID[47:33]). Reset value of this register field is considered from input port signal 'slv_pid[47:33]'. **Value After Reset:** 0x0
0	SLV_PROV_ID_SEL	R/W	Specifies the Provisional ID Type Selector (PID[32]). Reset value of this register field is considered from input port signal 'slv_pid[32]'. (1'b1: Random Value, 1'b0: Vendor Fixed Value) **Value After Reset:** 0x0

SLV_PID_VALUE

- **Name:** Provisional ID Register
- **Description:** I3C Normal Provisional ID Register.
  This register is used in slave mode of operation.
- **Size:** 32 bits
- **Offset:** 0x74

31:16	15:12	11:0
SLV_PART_ID	SLV_INST_ID	SLV_PID_DCR

Table 21. Fields for Register: SLV_PID_VALUE

Bits	Name	Memory Access	Description
31:16	SLV_PART_ID	R/W	Specifies the Part ID of DWC_mipi_i3c device (PID[31:16]) Reset value of this register field is considered from input port signal 'slv_pid[31:12]'. **Value After Reset:** 0x0
15:12	SLV_INST_ID	R/W	This field is used to program the instance ID of the Slave. Reset value of this register field is considered from input port signal 'slv_pid[15:12]'. **Value After Reset:** 0x0
11:0	SLV_PID_DCR	R/W	Specifies the additional 12-bit ID of DWC_mipi_i3c device (PID[11:0]). Reset value of this register field is considered from input port signal 'slv_pid[11:0]'. **Value After Reset:** 0x0

Offset 0x70 = 0x00000A50 (MID=0x528)

Offset 0x74 = 0x000100c6 (PID=01, SLV_INST_ID=0, SLV_PID_DCR=0xc6)

Chapter 4: Address space

4.1 NIC address map

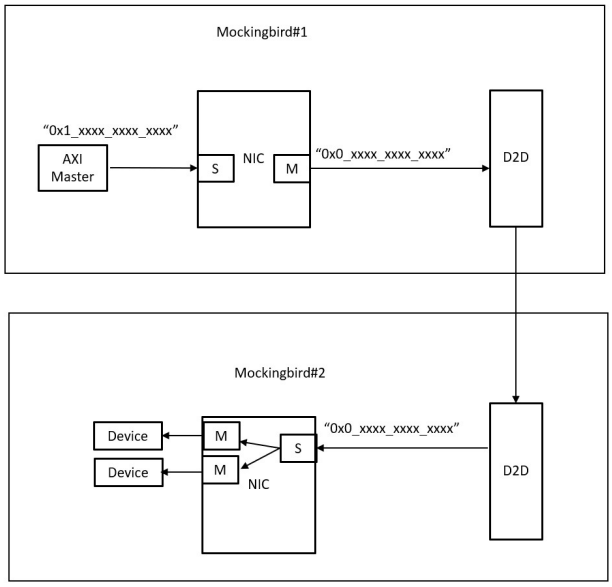
- 52 bit address.
- Fixed map at NOC.
- Programmable system address region via external CSR.

Device	Size	Start	End	Comment
		0	0x9f_ffffffff	Reserved
ETM0	16K	0xa0_00000000	0xa0_00003fff	
ETM1	16K	0xa1_00000000	0xa1_00003fff	
ETM2	16K	0xa2_00000000	0xa2_00003fff	
ETM3	16K	0xa3_00000000	0xa3_00003fff	
	128M	0xb0_00000000	0xb0_07ffffff	Reserved
SRAM	16M	0xc0_00000000	0xc0_00ffffff	Actual SRAM size may be smaller
FLC2	1T	0x0_0100_00000000	0x0_01ff_ffffffff	FLC2 slice 0 slave address
FLC2	1T	0x0_0200_00000000	0x0_02ff_ffffffff	FLC2 slice 1 slave address
FLC2	1T	0x0_0300_00000000	0x0_03ff_ffffffff	FLC2 slice 2 slave address
FLC2	1T	0x0_0400_00000000	0x0_04ff_ffffffff	FLC2 slice 3 slave address
		0x0_0500_00000000	0x0_05ff_ffffffff	Reserved
D2D_0	256T	0x1_0000_00000000	0x1_ffff_ffffffff	Inter-chiplet
D2D_1	256T	0x2_0000_00000000	0x2_ffff_ffffffff	Inter-chiplet
CXL_PCIe_EP0	1T	0x0_4000_00000000	0x0_40ff_ffffffff	Host0 CXL_PCIe AXI slave address
CXL_PCIe_EP1	1T	0x0_4100_00000000	0x0_41ff_ffffffff	Host1 CXL_PCIe AXI slave address
CXL_PCIe_EP2	1T	0x0_4200_00000000	0x0_42ff_ffffffff	Host2 CXL_PCIe AXI slave address
CXL_PCIe_EP3	1T	0x0_4300_00000000	0x0_43ff_ffffffff	Host3 CXL_PCIe AXI slave address
		0x0_4400_00000000	0x0_4fff_ffffffff	Reserved
D2D_DDR0	1T	0x0_5000_00000000	0x0_50ff_ffffffff	Goldfinch0
D2D_DDR1	1T	0x0_5100_00000000	0x0_51ff_ffffffff	Goldfinch1
D2D_DDR2	1T	0x0_5200_00000000	0x0_52ff_ffffffff	Goldfinch2
D2D_DDR3	1T	0x0_5300_00000000	0x0_53ff_ffffffff	Goldfinch3
				Reserved
	256T	0x8_0000_00000000	0x8_ffff_ffffffff	(Reserved)
	256T	0x9_0000_00000000	0x9_ffff_ffffffff	(Reserved)
	256T	0xa_0000_00000000	0xa_ffff_ffffffff	(Reserved)
	256T	0xb_0000_00000000	0xb_ffff_ffffffff	(Reserved)
				Reserved
CXL_PCIe_RC1_x8	8T	0x0_c000_00000000	0x0_c7ff_ffffffff	Downstream CXL.mem (RC1_x8)
CXL_PCIe_RC1_x8	8T	0x0_c800_00000000	0x0_cfff_ffffffff	Downstream CXLio/PCIe (RC1_x8)
CXL_PCIe_RC1_x4	8T	0x0_d000_00000000	0x0_d7ff_ffffffff	Downstream CXL.mem (RC1_x4)

CXL_PCIe_RC1_x4	8T	0x0_d800_00000000	0x0_dffff_ffffffff	Downstream CXLio/PCIe (RC1_x4)
CXL_PCIe_RC0_x8	8T	0x0_e000_00000000	0x0_e7ff_ffffffff	Downstream CXL.mem (RC0_x8)
CXL_PCIe_RC0_x8	8T	0x0_e800_00000000	0x0_effff_ffffffff	Downstream CXLio/PCIe (RC0_x8)
CXL_PCIe_RC0_x4	8T	0x0_f000_00000000	0x0_f7ff_ffffffff	Downstream CXL.mem (RC0_x4)
CXL_PCIe_RC0_x4	8T	0x0_f800_00000000	0x0_fffff_ffffffff	Downstream CXLio/PCIe (RC0_x4)

Inter-chiplet transaction mapping

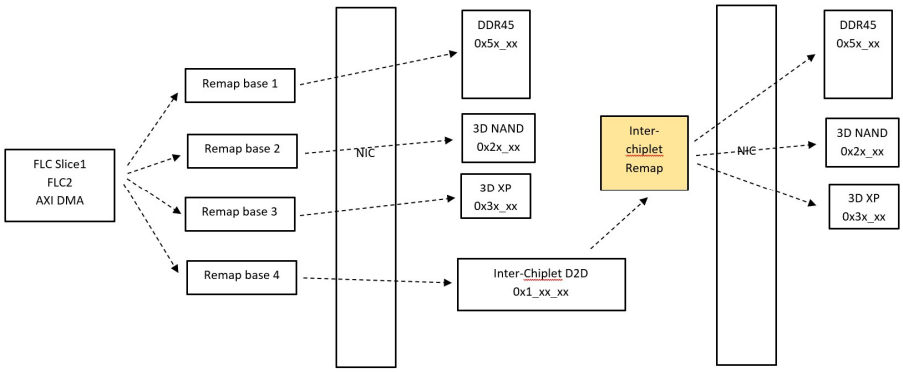
NIC D2D0/D2D1 (inter-chiplet) AXI master interface is assigned address space as 0x1_xxxx_xxxx_xxxx/0x2_xxxx_xxxx_xxxx. When transactions passing through D2D to reach targeting chiplet, it'll be decoded as 0x_0_xxxx_xxxx_xxxx for devices within.



Address remap from AXI/AHB master to NIC slave port

- Modify request address from masters to fit (fixed) NIC map.

- Run time programmable CSR for memory pooling.
- Logical address from host vs chiplet memory partition.
- Expand address bit: uC (32) -> NIC (52).
- Applied on master interfaces: FLC2 (AXI), RC master and uC master (AHB).
- CSR fields
  - Master BAR (base, size): to compare address field with incoming request.
  - Slave BAR (base): new base address for outgoing request.
  - 4 sets of slave BAR per FLC AXI master to access shared devices: DDR45, inter-chiplet, 3DNAND and CXL_PCIe RC.



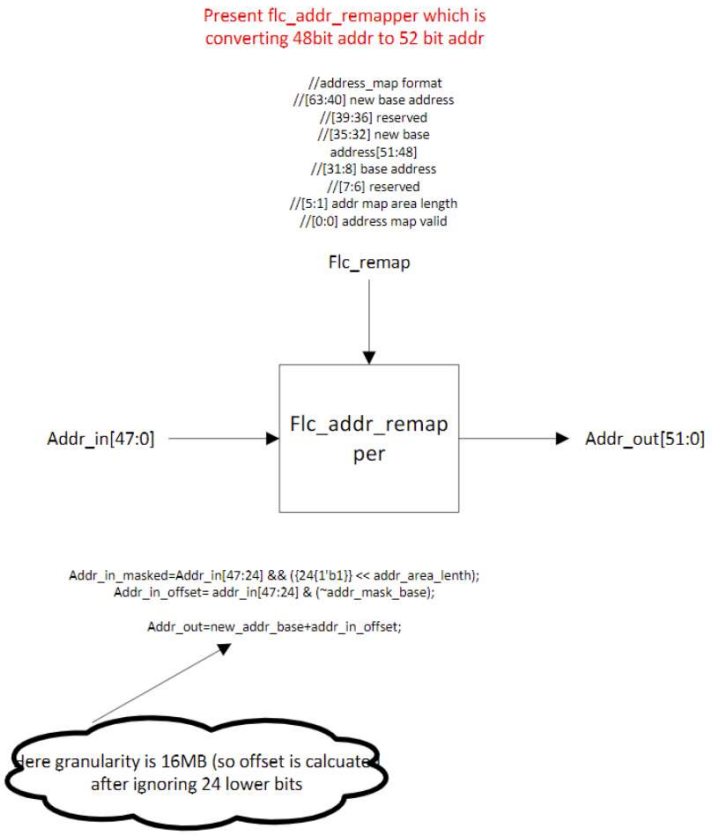
There are 4 sets of mappers for each FLC slice. Defined in uC's system control register (0x26000080~0x260000ff).

Each mapper register is 64b, remaps memory block of 16MB to a new location.

Address offset	Bit	Name	default	Comment
0x0	0	valid	0	1=mapper valid
	5:1	length	0	Length of mapped region (2's power of 16MB). 0=16MB, 1=32MB, 2=64MB..
	7:6	RSVD		Reserved
	31:8	base_address	0	Incoming address (to be mapped) A[47:24]
0x4	3:0	Remapped_address_h	0	Outgoing (remapped) address A[51:48]
	7:4	RSVD		
	31:8	Remapped_address	0	Outgoing (remapped) address A[47:24]

Mapper allocation (sysctrl base = 0x26000000)

uC Address offset	FLC Slice	Mapper	Comment
0x80~0x87	0	1	
0x88~0x8f	0	2	
0x90~0x97	0	3	
0x98~0x9f	0	4	
0xa0~0xa7	1	1	
0xa8~0xaf	1	2	
0xb0~0xb7	1	3	
0xb8~0xbf	1	4	
0xc0~0xc7	2	1	
0xc8~0xcf	2	2	
0xd0~0xd7	2	3	
0xd8~0xdf	2	4	
0xe0~0xe7	3	1	
0xe8~0xef	3	2	
0xf0~0xf7	3	3	
0xf8~0xff	3	4	


Present flc_addr_remapper which is converting 48bit addr to 52 bit addr
//address_map format
//[63:40] new base address
//[39:36] reserved
//[35:32] new base address[51:48]
//[31:8] base address
//[7:6] reserved
//[5:1] addr map area length
//[0:0] address map valid
Flc_remap
Addr_in[47:0]
Flc_addr_remapper
Addr_out[51:0]
Addr_in_masked=Addr_in[47:24] && {{24{1'b1}} << addr_area_lenth);
Addr_in_offset= addr_in[47:24] & (~addr_mask_base);
Addr_out=new_addr_base+addr_in_offset;
Here granularity is 16MB (so offset is calcuated after ignoring 24 lower bits

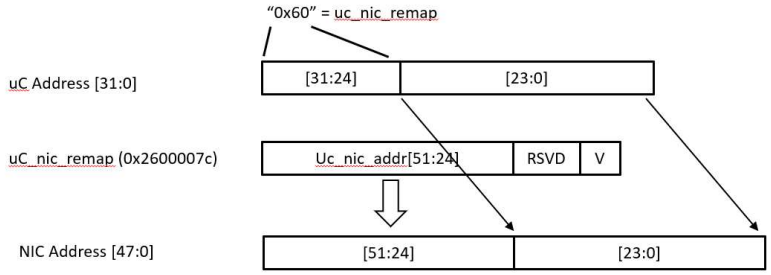
uC to NOC Mapper

uC address space is 4G (32b). All uC access to NIC goes through fixed mapping window.

uC Address window	Address to NOC	Region	Size
0x4000_0000 ~ 0x40ff_ffff	0x00c0_0000_0000 ~ 0x00c0_00ff_ffff	SRAM	16M
0x5000_0000 ~ 0x50ff_ffff	CSR: dbl_rc0_remap	(RCx8) PCIe/NVMe doorbell	Mmio Location within RC's PCIe space
0x5100_0000 ~ 0x51ff_ffff	CSR: uc_rc0_remap	(RCx8) Downstream CXL/PCIe	16T
0x5200_0000 ~ 0x52ff_ffff	CSR: uc_ep0_remap	upstream CXL/PCIe ep 0	1T
0x5300_0000 ~ 0x53ff_ffff	CSR: uc_ep1_remap	upstream CXL/PCIe ep 1	1T
0x5400_0000 ~ 0x54ff_ffff	CSR: uc_ep2_remap	upstream CXL/PCIe ep 2	1T
0x5500_0000 ~ 0x55ff_ffff	CSR: uc_ep3_remap	upstream CXL/PCIe ep 3	1T
0x5600_0000 ~ 0x56ff_ffff	Reserved		
0x5700_0000 ~ 0x57ff_ffff	Reserved		
0x5800_0000 ~ 0x58ff_ffff	CSR: dbl_rc0_x4_remap	(RCx4) PCIe/NVMe doorbell	16T
0x5900_0000 ~ 0x59ff_ffff	CSR: uc_rc0_x4_remap	(RCx4) CXL/PCIe	16T
0x5a00_0000 ~ 0x5aff_ffff	CSR: dbl_rc1_x8_remap	(RC1x8) PCIe/NVMe doorbell	16T
0x5b00_0000 ~ 0x5bff_ffff	CSR: uc_rc1_x8_remap	(RC1x8) CXL/PCIe	16T
0x5c00_0000 ~ 0x5cff_ffff	CSR: dbl_rc1_x4_remap	(RC1x4) PCIe/NVMe doorbell	16T
0x5d00_0000 ~ 0x5dff_ffff	CSR: uc_rc1_x4_remap	(RC1x4) CXL/PCIe	16T
0x6000_0000 ~ 0x6fff_ffff	CSR: uc_nic_remap	UC_mapper (DDR45 D2D)	256T (default to 0x5000_0000_0000)
0x7000_0000 ~ 0x7fff_ffff	Reserved		

"CSR: Uc_nic_remap" is added for software to remap uC to NIC access at run time. By default this register is used to map uC access to DDR45 of 1st D2D interface.

Address	name	bit	field			comment
0x2600007c	uc_nic_remap	0	valid	0	RW	1=remap is valid
		3:1	Reserved	0	RW	
		27:4	uc_nic_addr_47x24	0x500000	RW	A[47:24] of uc to NIC access
		31:28	uc_nic_addr_51x48	0	RW	A[51:48] of uc to NIC access



"CSR: dbl_rc_remap" is added for software to remap FLC2/QM to RC doorbell access at run time.

FLC2/QM access window = 0x50xx_xxxx
System address after remap = 0xf_<dbl_rc_a_43_24>_xx_xxxx

Address	name	bit	field			Comment
0x26000068	dbl_rc_remap	19:0	Dbl_rc_a_43_24	0	RW	
		31:20	Reserved	0		

"CSR: uc_rc_remap" is added for software to remap uC to RC AXI slave port access at run time.

uC access window = 0x51xx_xxxx
System address after remap = 0xf_<uc_rc_a_43_24>_xx_xxxx

Address	name	Bit	field			Comment
0x2600006c	uc_rc_remap	19:0	uc_rc_a_43_24	0	RW	
		31:20	Reserved	0		

"CSR: uc_ep01_remap" is added for software to remap uC to EP0/EP1 AXI slave port access at run time.

uC access window = 0x52xx_xxxx
System address after remap = 0x40_<uc_ep0_a_39_24>_xx_xxxx

uC access window = 0x53xx_xxxx
System address after remap = 0x41_<uc_ep1_a_39_24>_xx_xxxx

Address	name	bit	field			Comment
0x26000070	uc_ep01_remap	15:0	uc_ep0_a_39_24	0	RW	
		31:16	uc_ep1_a_39_24	0	RW	

"CSR: uc_ep23_remap" is added for software to remap uC to EP2/EP3 AXI slave port access at run time.

uC access window = 0x54xx_xxxx
System address after remap = 0x42_<uc_ep2_a_39_24>_xx_xxxx

uC access window = 0x55xx_xxxx
System address after remap = 0x43_<uc_ep3_a_39_24>_xx_xxxx

Address	name	bit	field			Comment
0x26000074	uc_ep23_remap	15:0	uc_ep2_a_39_24	0	RW	
		31:16	uc_ep3_a_39_24	0	RW	

ETM AXI port mapping

0x5e[y]x_xxxx will be mapped to 0xa[y]_000x_xxxx. This is for E21 subsystem to access ETM's AXI slave port through NOC.

e.g. 0x5e00_0000 -> 0xa0_0000_0000 (ETM0 offset 0)

e.g. 0x5e20_0010 -> 0xa2_0000_0020 (ETM2, offset 0x20)

e.g. 0x5e30_3ff0 -> 0xa3_0000_3ff0 (ETM3, offset 0x3ff0)

PHY JTAG connection:

MKB has 3 PHY JTAG chains (IPM, PCIE, D2D).

They shared: TMS, TCK, nTRST.

TDI and TDO are selected via GPIO[8:4]:

GPIO[8:4]	TDI/TDO	GPIO[8:4]	TDI/TDO
0	Slice 0, IPM0.	13	PCIe EP2 PMA
1	Slice 0, IPM1.	14	PCIe EP3 PCS
2	Slice 1, IPM0.	15	PCIe EP3 PMA
3	Slice 1, IPM1.	16	PCIe RC0 PCS
4	Slice 2, IPM0.	17	PCIe RC0 PMA
5	Slice 2, IPM1.	18	PCIe RC1 PCS
6	Slice 3, IPM0.	19	PCIe RC1 PMA
7	Slice 3, IPM1.	20	D2D_DDR0 (GFH0)
8	PCIe EP0 PCS	21	D2D_DDR1 (GFH1)
9	PCIe EP0 PMA	22	D2D_DDR1 (GFH2)
10	PCIe EP1 PCS	23	D2D_DDR2 (GFH3)
11	PCIe EP1 PMA	24	MKB D2D0
12	PCIe EP2 PCS	25	MKB D2D1
		26	JTAG_APB_bridge

D2D PHY JTAG loopback test: Apply GPIO[15:17] to enable D2D LL and wrapper during JTAG test sequence.

Conditions in D2D loopback test mode (CORERSTn==0):

- GPIO[15] -> EXT_PHY_RST_EN
- GPIO[16] -> EXT_TX_PHY_RSTN
- GPIO[17] -> EXT_RX_PHY_RSTN

JTAG APB TDR:

IR: Data reg =1 (write), Status reg (read) =2

DR:
Data and status reg format:

[71]:	reset_n	Reset jtag bridge.
[70]:	start	Set high to trigger apb write/read command, set low before start new apb transaction.
[69:67]:	pprot	Apb used
[66]:	pwrite	Apb used
[65:34]:	paddr	Apb used
[33:02]:	pwdata/prdata	Apb used
[01]:	pslverr	Apb used
[00]:	busy	Status of apb write/read command

Steps:

1. Toggle ntrst. TMS=0. “start” in TDR reset to “0”.
2. Write TDR: {address, pprot, pwrite..}, set “start”=1 to start APB cycle.
3. Read TDR, polling for “busy”=0;
4. “busy”=0 indicates APB cycle completed, {prdata and pslverr} valid.
5. Write TDR with “start”=0.
6. Back to step 2 for more APB transactions.

4.2 E21 address map

Start Address	End Address	Size	Description	Comment
0x0000_0000	0x0000_0fff	4K	Debug controller control	
0x0000_3000	0x0000_3fff	4K	Error-device	
0x0000_4000	0x0000_4fff	4K	Test status	
0x0170_0000	0x0170_0fff	4K	Bus error unit	
0x0200_0000	0x02ff_ffff	16M	Interrupt controller	
0x1000_0000	0x1000_0fff	4K	Trace encoder	Only in trial version
0x2000_0000	0x2000_0fff	4K	SPI0	
0x2000_1000	0x2000_1fff	4k	SPI1	
0x2000_2000	0x2000_2fff	4K	I2C	I2C host
0x2000_3000	0x2000_3fff	4K	LTMR	Local timer, includes WDT
0x2000_4000	0x2000_4fff	4K	UART	
0x2000_5000	0x2000_5fff	4K	GPIO	
0x2000_6000	0x2000_ffff	40K	Reserved	
*moved to Goldfinch				
0x2100_0000	0x210f_ffff	1M	DDR4/5 MC	DDR4/5 MC CSR
0x2110_0000	0x211f_ffff	1M	DDR4/5 PHY	DDR4/5 PHY CSR
0x2120_0000	0x21ff_ffff	14M	QM	Queue Manager CSR QM0=0x2120_xxxx QM1=0x2121_xxxx QM2=0x2122_xxxx QM3=0x2123_xxxx QM0 SRAM = 0x21f0_xxxx QM1 SRAM = 0x21f1_xxxx QM2 SRAM = 0x21f2_xxxx QM3 SRAM = 0x21f3_xxxx
"s" = (FLC slice# +2). E.g., [0x2200_0000:0x2200_ffff] = FLC slice 0, MC.				
0x2s00_0000	0x2s00_ffff	64k	IPM MC0	[Mockingbird slice/IPM MC0]
0x2s01_0000	0x2s01_ffff	64k	FLC1_0	[Mockingbird slice/FLC1_0]
0x2s02_0000	0x2s02_ffff	64k	FLC2	[Mockingbird slice/FLC2]
0x2s03_0000	0x2s03_ffff	64k	AXI_MASTER	[Mockingbird slice/AXIM]
0x2s04_0000	0x2s04_ffff	64k	IPM MC1	[Mockingbird slice/IPM MC1]
0x2s05_0000	0x2s05_ffff	64k	FLC1_1	[Mockingbird slice/FLC1_1]

0x2s06_0000	0x2s0f_ffff	640k	Reserved	
0x2s10_0000	0x2s1f_ffff	1M	IPM_PHY0	[Mockingbird slice/IPM PHY0]
0x2s20_0000	0x2s2f_ffff	1M	IPM_PHY1	[Mockingbird slice/IPM PHY1]
0x2s30_0000	0x2sff_ffff	13M	Reserved	Reserved

Broadcast address for IPM MC, FLC1, FLC2, AXI Master				
0x2208_0000	0x2208_ffff	64k	IPM_MC broadcast	
0x2209_0000	0x2209_ffff	64k	FLC1 broadcast	
0x220a_0000	0x220a_ffff	64k	FLC2 broadcast	
0x220b_0000	0x220b_ffff	64k	AXI Master broadcast	
0x22f0_0000	0x22ff_ffff	1M	IPM PHY broadcast	

0x2600_1000	0x2600_1fff	4k	SF_CTRL	SPI serial flash controller, SRAM buffer offset: 0x300
0x2600_2000	0x2600_2fff	4k	L1C	L1C cache controller for XIP flash
0x2600_3000	0x2600_3fff	4k	Reserved	
0x2600_4000	0x2600_4fff	4k	DMA	DMA controller
0x2600_5000	0x2600_5fff	4k	SEC	Security Module

0x2601_0000	0x2601_0fff	4k	UC_PCIE	uC – PCIe control
0x2608_0000	0x2608_ffff	64K	NEO_EFUSE	(OTP) NEO eFuse cell
0x2700_0000	0x27ff_ffff	16M	D2D_PHY	[D2D PHY]

Start Address	**End Address**	**Size**	**Description**	**Comment**
0x2800_0000	0x287f_ffff	8M	PCIE_PHY (PMA)	PCIE PHY
0x2881_0000	0x28af_ffff	8M	PCIE_PHY (PCS)	PCIE PHY
0x2900_0000	0x29ff_ffff	16M	PCIE MAC	CXL_PCIE_EP, CXL_PCIE_RC config registers (DBI)

0x2a00_0000	0x2a00_ffff	64K	I3C Master	I3C controller
0x2a01_0000	0x2a01_ffff	64K	I3C Slave	I3C controller
0x2a02_0000	0x2a02_ffff	64k	AXI4TG	Axi4 test generator
0x2a03_0000	0x2a03_ffff	64K	ETM	ETM0=0x2a030000~0x2a030fff ETM1=0x2a031000~0x2a031fff ETM2=0x2a032000~0x2a032fff ETM3=0x2a033000~0x2a033fff
0x2a04_0000	0x2a04_ffff	64K	~~IPM9_MC~~	

0x2a10_0000	0x2a1f_ffff	1M	~~IPM9_PHY~~	
0x2a21_0000	0x2aff_ffff		Reserved	APBx
0x2b00_0000	0x2b00_ffff	64K	EMAC	
0x2b01_0000	0x2bff_ffff		Reserved	AHBx
0x2c00_0000	0x2c0f_ffff	1M	D2D_PHY	Alternate window for D2D PHY 0x2c00_xxxx: GFH0 D2D 0x2c01_xxxx: GFH1 D2D 0x2c02_xxxx: GFH2 D2D 0x2c03_xxxx: GFH3 D2D
0x2c10_0000	0x2eff_ffff		Reserved	
0x2f00_0000	0x2fxx_xxxx	16M	D2D_IO	D2D A5L access window for remote chiplet CSR
0x3000_0000	0x300f_ffff	1M	Boot ROM	Boot ROM: 32-256KB
0x3010_0000	0x3fff_ffff	255M	XIP NOR Flash	SPI NOR Flash (XIP access)
0x4000_0000	0x40ff_ffff	16M	SRAM	Chiplet SRAM
0x4100_0000	0x4fff_ffff	240M	Reserved	
0x5000_0000	0x50ff_ffff	24M	RC0_x8_DBL	CXL_PCIe RC0_x8 door bell
0x5100_0000	0x51ff_ffff	24M	RC0_x8_AXIS	CXL_PCIe RC0_x8 AXI slave port
0x5200_0000	0x52ff_ffff	24M	EP0_AXIS	CXL_PCIe slice 0 AXI slave port
0x5300_0000	0x53ff_ffff	24M	EP1_AXIS	CXL_PCIe slice 1 AXI slave port
0x5400_0000	0x54ff_ffff	24M	EP2_AXIS	CXL_PCIe slice 2 AXI slave port
0x5500_0000	0x55ff_ffff	24M	EP3_AXIS	CXL_PCIe slice 3 AXI slave port
0x5600_0000	0x56ff_ffff	24M	Reserved	
0x5700_0000	0x57ff_ffff	24M	Reserved	
0x5800_0000	0x58ff_ffff	24M	RC0_x4_DBL	CXL_PCIe RC0_x4 door bell
0x5900_0000	0x59ff_ffff	24M	RC0_x4_AXIS	CXL_PCIe RC0_x4 AXI slave port
0x5a00_0000	0x5aff_ffff	24M	RC1_x8_DBL	CXL_PCIe RC1_x8 door bell
0x5b00_0000	0x5bff_ffff	24M	RC1_x8_AXIS	CXL_PCIe RC1_x8 AXI slave port
0x5c00_0000	0x5cff_ffff	24M	RC1_x4_DBL	CXL_PCIe RC1_x4 door bell
0x5d00_0000	0x5dff_ffff	24M	RC1_x4_AXIS	CXL_PCIe RC1_x4 AXI slave port
0x5e00_0000	0x5eff_ffff	24M	ETM_AXI	ETM AXI slave port (read only)
0x5f00_0000	0x5fff_ffff	24M	~~DBI~~ Reserved	~~CXL_PCIe DBI interface selected by "pcie_sel".~~
0x6000_0000	0x6fff_ffff	256M	uc_nic_remap	Default access to DDR45, controlled by register "uc_nic_remap" (0x2600007c)
0x7000_0000	0x7fff_ffff	256M	Reserved	

0x8000_0000 ~ 0xffff_ffff: E21 core targets

0x8000_0000	0x8001_ffff	128K	TIM0
0x8002_0000	0x8003_ffff	128K	TIM1

Chapter 5: E21 IRQ map

IRQ	Device	Comment
0	UART	
1	SPI0	
2	SPI1	
3	I2C	I2C controller
4	Reserved0x	Reserved for Local timer 1
5	TMR2	Local timer 2
6	TMR3	Local timer 3
7	GPIO	
8	DMA	
9	I3C Master	I3C Master
10	I3C Slave	I3C Slave
11	ELBI Access	
12	PCIe MAC	
13	RC_msi_ctrl_int	apbx_irq[1]
14	EP0_PERST	apbx_irq[2]
15	EP1_PERST	apbx_irq[3]
16	EP2_PERST	apbx_irq[4]
17	SEC-SHA	
18	SEC-AES	
19	SEC-TRNG	
20	SF_CTRL	XIP/SPI controller
21	OTP	
22	L1C	
23	BMX_Timeout	
24	BMX_Error	
25	DMA_interr	
26	DMA_inttc	
27	EP3_PERST	apbx_irq[5]
28	Reserved	
29	Reserved	
30	ETM IRQ	ETM
31	RISCV_IRQ	From RISCV GPIO0
32	FLC1	FLC1
33	FLC2	FLC2

34	IPM MC	IPM MC
35	IPM PHY	IPM PHY
36	Reserved	
37	Reserved	
38	FLC_AXIM	FLC AXI Master
39	QM	Queue Manager
40	D2D_0_0	DDR chiplet #1, slice#1
41	D2D_1_0	DDR chiplet #2, slice#1
42	D2D_2_0	DDR chiplet #3, slice#1
43	D2D_3_0	DDR chiplet #4, slice#1
44	MKB_D2D0	
45	MKB_D2D1	
46	Reserved	
47	Reserved	
48	Radm_pm_turnoff	
49	~mac_mstr_aw_fifo_empty	
50	~m2sreq_memInv_NT_fifo_empty	
51	~m2sreq_specrd_fifo_empty	
52	m2sreq_memInv_NT_rsp_fifo_full	
53	Hot_reset_req	RCx8, RCx4, EP[3:0]
54	RC_radm_pme_or_err	RCx8 or RCx4
55	EMAC	
56	~Vdm_empty	Ep[1:4] vdm_0 or vdm_1 fifo not empty
57	TS_IRQ	Temperature Sensor
58	VS_IRQ	Voltage Sensor
59	Reserved	
60-55	Reserved	
61	Reserved	
62	Watchdog Timer	Local timer WDT
127-63	Reserved	
NMI	Watchdog Timer	Local timer WDT

Chapter 6: IO pins

6.1 uC pin multiplex

uC's GPIO pins are multiplexed with embedded peripheral functions.

GPIO #	Function Select	Function Pins	Direction	Pull Up	Pull Down	comment
0	1	uart_rts	o	n	n	riscv_txd
1	1	uart_cts	i	n	n	riscv_rxd
2	1	uart_txd	o	n	n	
3	1	uart_rxd	i	n	n	
4	2	spi_mosi	o if master mode	n	n	master mode=0x80.0
5	2	spi_miso	i if master mode	n	n	swap miso/mosi=0x80.1
6	2	spi_ss_o	o if master mode	n	n	
7	2	spi_clk	o if master mode	n	n	
8	3	i2c_scl	bi	Y		
9	3	i2c_sda	bi	y		
10	0	gpio_reg	bi			CHIP_ID/spi1_mosi
11	0	gpio_reg	bi			CHIP_ID/spi1_miso
12	0	gpio_reg	bi			BOOT_LED[0]/spi1_ss_o
13	0	gpio_reg	bi			BOOT_LED[1]/spi1_clk
14	0	gpio_reg	bi			BOOT_LED[2]/config_spiclk
15	0	gpio_reg	bi			I3C_ready (GFH3's GPIO18)/config_spi_di
16	0	gpio_reg	bi			I3C_ready (GFH2's GPIO18)/config_spi_do
17	0	gpio_reg	bi			I3C_ready (GFH1's GPIO18)
18	0	gpio_reg	bi			I3C_ready (GFH0's GPIO18)
19	0	gpio_reg	bi			FW_MODE

6.2 Pin List

Type	Pin Name	Category	comment
power	VDD_IO	PWR	1.8V LVCMOS standard IO power
power	VDD_CORE	PWR	Core power, 0.8V
ground	VSS_IO	GND	
ground	VSS_CORE	GND	
power	PCIE_PA_VDDH	PCIe-PWR	PCIe PHY analog 1.2V power
power	PCIE_PA_VDDL	PCIe-PWR	PCIe PHY analog 0.75V power
ground	PCIE_PA_VSS	PCIe-GND	PCIe PHY analog ground
power	PCIE_PHY_P_VDD	PCIe-PWR	
ground	PCIE_PHY_P_VSS	PCIe-GND	
power	IPM_VDDQ	IPM-PWR	IPM PHY 0.3V IO power
power	IPM_VDD2	IPM-PWR	IPM PHY 1.1V IO/PLL power
ground	IPM_VSS	IPM-GND	
Power	PLL_VDDHV	SOC-PLL-PWR	1.8v
Power	PLL_VDDPOST	SOC-PLL-PWR	0.8v
Power	PLL_VDDREF	SOC-PLL-PWR	0.8v
ground	PLL_VSS	SOC-PLL-GND	
Power	HS_PLL_VDDHV	D2D-PLL-PWR	1.8v
Power	HS_PLL_VDDPOST	D2D-PLL-PWR	0.8v
Power	HS_PLL_VDDREF	D2D-PLL-PWR	0.8v
ground	HS_PLL_VSS	D2D-PLL-GND	
Power	D2D_VDDQ_S	D2D-PWR	D2D PHY
Power	D2D_VDDCLK_S	D2D-PWR	
Power	D2D_VDDA_S	D2D-PWR	
Power	D2D_VDD_S	D2D-PWR	
ground	D2D_VSS_S	D2D-GND	
pwr	Otp_vdd	OTP	0.8v

pwr	Otp_vdd2	OTP	1.8v
ground	Otp_vss	OTP	
Input	CB_nRST	Reset	Functional reset for chiplet
input	CB_nPOR	Reset	Power on reset
input	CORERSTn	Reset	CPU subsystem reset
input	OSCCLK[0]	Clock	SOC Reference clocks: 25MHz.
Input	Ref_100m_n[1:0]		Reference clock for D2D PLL (LVDS)
input	Ref_100m_p[1:0]		Reference clock for D2D PLL (LVDS)
inout	GPIO[19:0]	GPIO	General purpose IO
output	sf_clk	SF-NOR	Serial flash clock
output	sf_cs	SF-NOR	Serial flash chip select
inout	sf_io[3:0]	SF-NOR	Serial flash data
input	ep_cxl_ref_p[3:0], ep_cxl_ref_n[3:0]	CXL_PCIe	Reference clock for host side CXL/PCIe PHY, 100MHz.
input	ep_cxl_rx_n[3:0][7:0], ep_cxl_rx_p[3:0][7:0]	CXL_PCIe	RX lane for host side CXL/PCIe.
output	ep_cxl_tx_n[3:0][7:0], ep_cxl_tx_p[3:0][7:0]	CXL_PCIe	TX lane for host side CXL/PCIe.
input	rc_cxl_ref_p[1:0], rc_cxl_ref_n[1:0]	CXL_PCIe	Reference clock for RC side CXL/PCIe PHY, 100MHz.
input	rc_cxl_rx_n[1:0][7:0], rc_cxl_rx_p[1:0][7:0]	CXL_PCIe	RX lane for RC side CXL/PCIe.
output	rc_cxl_tx_n[1:0][7:0], rc_cxl_tx_p[1:0][7:0]	CXL_PCIe	TX lane for RC side CXL/PCIe.
input	ep_perstn[3:0]	CXL_PCIe	Endpoint [3:0] PCIe bus reset
output	rc_perstn[1:0]	CXL_PCIe	RC port bus reset
	IPM*	IPM	IPM interface for FLC slice [3:0], 2xIPM each slice
	D2D_DDR_0	D2D 1x16	D2D interface for DDR4 chiplet
	D2D_DDR_1	D2D 1x16	D2D interface for DDR4 chiplet
	D2D_DDR_2	D2D 1x16	D2D interface for DDR4 chiplet
	D2D_DDR_3	D2D 1x16	D2D interface for DDR4 chiplet

	D2D_0	D2D 1x16	(When MKB D2D enabled) D2D interface for inter-chiplet.
	D2D_1	D2D 1x16	(When MKB D2D enabled) D2D interface for inter-chiplet.
input	nTRST	JTAG	JTAG reset
input	TMS	JTAG	JTAG mode select
input	TCK	JTAG	JTAG clock
input	TDI	JTAG	JTAG data input: TDI_IPM_PHY, TDI_PCIE_PHY, TDI_D2D
output	TDO	JTAG	JTAG data output: TDO_IPM_PHY, TDO_PCIE_PHY, TDO_D2D
input	nTRST_E21	JTAG	JTAG reset
input	TMS_E21	JTAG	JTAG mode select
input	TCK_E21	JTAG	JTAG clock
input	TDI_E21	JTAG	JTAG data input: TDI_E21
output	TDO_E21	JTAG	JTAG data output: TDO_E21
input	nTRST_RISCV	JTAG	JTAG reset
input	TMS_RISCV	JTAG	JTAG mode select
input	TCK_RISCV	JTAG	JTAG clock
input	TDI_RISCV	JTAG	JTAG data input: TDI_RISCV
output	TDO_RISCV	JTAG	JTAG data output: TDO_RISCV
input	scan_mode	SCAN	Scan mode
input	boot_xip	boot	0=uC boot from internal ROM. 1=uC boot from serial flash
output	Pll_locked	Status	1= pll locked, clock ready
inout	I3c_master_scl	I3C	I3C clock
inout	I3c_master_sda	I3C	I3C data
input	I3c_slave_scl	I3C	I3C clock
inout	I3c_slave_sda	I3C	I3C data

input	EXTCK_EREFCK	EMAC	
input	ECRS	EMAC	
input	ECOL	EMAC	
input	ERXDV	EMAC	
input	ERX[3:0]	EMAC	
output	ERXER	EMAC	
output	ETXEN	EMAC	
output	ETX[3:0]	EMAC	
output	ETXER	EMAC	
inout	EMDC	EMAC	
inout	EMDIO	EMAC	
Power	VDDA_TS		Temperature sensor analog power
Power	VDDA_VS		Voltage sensor analog power
Analog	Ts_anio[1:0]		Temperature sensor analog access bus
Analog	Vs_anio[1:0]		Voltage sensor analog access bus

Chapter 7: Registers

7.1 uC subsystem

CSR registers are uC controlled and selected by uC address map. Refer to section 4.2 for complete address space assignment.

7.1 System Control: base address 0x26000000

Address	regname	bit	bitname	default		comment
0x26000000	GPIO_POS	19:0	GPIO_POS[19:0]	x		Sampled value of GPIO[19:0] at chiplet reset.
		31:20	Reserved			
0x26000004	PMU_CTRL	0	UC_HCLK_HALT_EN	0	RW	1=stop uC hclk when WFI
		1	flc2_slave_p1_sync_mode	1	RW	1=flc2 slave port 1 run in sync mode
		2	flc2_slave_p0_sync_mode	0	RW	
		3	ipm_mc_axi_sync_mode	0	RW	1=ipm_mc's axi interface runs in synchronous mode.
		4	flc1_axi_slave_sync_mode	1	RW	1=Flc1's axi slave interface runs in synchronous mode.
		5	Reserved	0	RW	
		6	flc1_axi_master_sync_mode	1	RW	1=Flc1's axi master interface runs in synchronous mode
		7	Boot_tim1	0	RW	1=E21 boot from tim1 (0x80020000)
		11:8	D2D_DDR[3:0]_RSTN	0	RW	0= Put D2D_DDR (goldfinch D2D interface) in reset state 1= D2D_DDR interface is enabled Bit 8 = goldfinch#1, slice#0 Bit 9 = goldfinch#2, slice#0 Bit 10 = goldfinch#3, slice#0 Bit 11 = goldfinch#4, slice#0
		15:12	SF_IO[3:0]_PULL_UP_EN	0	RW	Bit 12 = SF_IO[0] pull up en Bit 13 = SF_IO[1] pull up en Bit 14 = SF_IO[2] pull up en Bit 15 = SF_IO[3] pull up en
		16	MKB_D2D0_RSTN	0	RW	0=MKB_D2D0 interface in reset state
		17	MKB_D2D1_RSTN	0	RW	0=MKB_D2D1 interface in reset state
		18	Reserved	0	RW	
		19	force_clk_ipm_on	0	RW	1=disable ipm clock gated

		20	force_flc1_clk_on	0	RW	1=disable flc1 clock gate
		21	Force_flc2_clk_on	0	RW	1=disable flc2_clk_gate
		22	I2c_en	0	RW	1=I2C pin mux to GPIO[9:8]
		23	Spi0_en	0	RW	1=SPI0 pin mux to GPIO[7:4]
		24	SPI1_en	0	RW	1=SPI1 pin mux to GPIO[13:10]
		25	CONFIG_SPI_en	0	RW	1=CONFIG_SPI pin mux to GPIO[16:14]
		26	SEL_RISCV_UART	0	RW	1=RISCV's UART1 to GPIO[1:0]
		27	Reserved	0	RW	
		31:28	FLC_SLICE_RST[3:0]	0	RW	1=FLC slice [3:0] hardware reset. Note: IPM_PHY and CSR does not reset by this bit.
0x26000008	SYS_CTRL	0	LOCKUPRESETN	0	RW	
		1	WDOGIRQNMI	0	RW	
		2	Simulation_speed_up	0	RW	"1"=speed up 32KHz clock for simulation.
		3	FORCE_DM_ACTIVE	0	RW	"1" = force Core Debug Module active
		13:4	AHB_NS_ACCESS_DIS	0	RW	"1" = Disable AHB0 device access in non-secure mode. Bit: 4=ahb0, 5=sfctrl, 6=l1c, 7=otp, 8=ahb1 (flc1/2, axim…), 9=sysctrl, 10=dmac, 11=security engine, 12=boot rom, 13=xip nor flash.
		14	SPI1 master mode	0	RW	1=set SPI1 IO pads as master mode
		23:15	NOC_CFG	15:0, 16:1, 17:1, 18:1, 19:1, 23:20:4	RW	NOC configuration: Bit 15=DualMemChEn Bit 16=GFH0En Bit 17=GFH1En Bit 18=GFH2En Bit 19=GFH3En Bit[23:20]= MemIntlvWays
		27:24	ELBI_SRAM_BASE	0xf	RW	ELBI access base address in SRAM with 8KB allocation.
		31:28	ELBI_ACC_FLAG[3:0]	0	RWc	Set to "1" when ELBI accessed. Write 1 to clear.
0x2600000c	(FPGA test only)		FPGA_REG			
		0	APP_LTSSM_EN		RO	EP MAC's APP_LTSSM_EN port status
		1	PERST_N		RO	EP PCIe reset status

		2	HOT_RESET_REQ		RO	EP MAC hot reset active flag
		3	CORE_CLK_RUNNING		RO	1=EP MAC's core clock is active.
		4	smlh_req_rst_disable	0	RW	1= pcie wrapper's "smlh_req_rst" does not reset "ltssm_en" port of MAC
		5	lip_smlh_req_rst_not_disable	0	RW	1=disable MAC's "smlh_req_rst_not" output
		6	lip_link_req_rst_not_disable	0	RW	1=disable MAC's "link_req_rst_not" output
		7	pcie_ep_prst_	0	RW	In RC mode, 0=ep device perst_ asserted.
		8	DDR_init_complete	0	RO	
		23:9	Reserved	0		
		27:24	Pcie_ep_linkup	0	RO	
		31:28	Pcie_rc_linkup	0	RO	28=smlh_linkup 29=rdlh_linkup
0x26000010	Reset Information	0	SYSRESETREQ	0	R/Wc	
		1	WDOGRESETREQ	0	R/Wc	
		2	LOCKUPRESET	0	R/Wc	
		3	BOOT_XIP	X	R	Returns pin value of "BOOT_XIP"
		4	OTP_ready	0	R	
		15:5	Reserved	0	RW	
		30:16	riscv_etm[30:16]	0	RW	RISCV-ETM AXI access address [30:16]. Address [15:14] selects etm#.
		31	riscv_etm_enable	0	RW	1=enable riscv_etm mapping.
0x26000014	sysctrl_riscv	0	riscv_resetn	0		0=reset
		1	riscv_irq	0		RISCV irq[31]
		2	riscv_wakeup	0		
		3	riscv_X_om	0		Operational mode
		11:4	Reserved	0		
		12	riscv_h0_rstvec_l_en	0		1=enable h0_rstvec_l
		13	riscv_h1_rstvec_l_en	0		1=enable h1_rstvec_l

		31:14	h0_ rstvec_l[19:2]			RISCV core0 low reset vector A[19:2].
0x26000018	sysctrl_riscv_rstvec	11:0	h0_rstvec_l[31:20]	0		RISCV core0 low reset vector A[31:20].
		26:12	riscv_rstvec_h	0		RISCV high reset vector A[46:32] (both cores)
		31:27	Reserved	0		
0x2600001c	sysctrl_riscv_rstvec1	1:0	Reserved	0		
		31:2	h1_ rstvec_l[31:2]	0		RISCV core1 low reset vector A[31:2].
0x26000020	bmx_config	9:0	bmx_master_hsel[9:0]	0	RW	(1 bit per master) Ignore waiting for slave's HREADYOUT for better timing
		13:10	bmx_arb_mode	0	RW	Arbitration priority
		17:14	bmx_slave_timeout_en	4'h0	RW	(1 bit per slave) bus timeout enable
		18	bmx_err_dis	0	RW	Disable error response
		19	bmx_busy_option_dis	0	RW	Ignore HTRANS busy
		23:20	bmx_dbg_sel	0	RW	bmx debug mux select
		24	bmx_err_addr_dis	0	RW	
		25	bmx_err_dec	0	R	
		26	bmx_timeout	0	R/Wc	bmx time out latch, write clear.
		31:27	Reserved			
0x26000024	bmx_err_addr	31:0	bmx_err_addr	0	R	Access error address
0x26000028	dbl_rc0_x4_remap	19:0	dbl_rc0_x4_a_43_24	0	RW	dbl_rc0_x4 access window [0x58xx_xxxx] upper address bit
		31:20	Reserved			

0x2600002c	uc_rc0_x4_remap	19:0	uc_rc0_x4_a_43_24	0	RW	uc_rc0_x4 access window [0x59xx_xxxx] upper address bit
		31:20	Reserved			
0x26000030	dbl_rc1_x4_remap	19:0	dbl_rc1_x4_a_43_24	0	RW	dbl_rc1_x4 access window [0x5cxx_xxxx] upper address bit
		31:20	Reserved			
0x26000034	uc_rc1_x4_x8_remap	19:0	uc_rc1_x4_a_43_24	0	RW	uc_rc1_x4 access window [0x5dxx_xxxx] upper address bit
		27:20	uc_rc1_x8_a_43_36	0	RW	uc_rc1_x8 access window [0x5bxx_xxxx] upper address bit. A[43:36]
		31:28	Reserved			
0x26000038	Interface_sel	3:0	pcie_sel	0	RW	Select CXL/PCIe interface for configuration. 0: RCx8 MAC [NVMe host/CXL RC] 1-4: host side slice[1:4]. 5: RCx4 MAC 6: RC1_x8 MAC 7: RC1_x4 MAC 13-8: Reserved. 14: Broadcast: slice 1-4 and RC, RC1 15: Broadcast (slice 1-4)
		7:4	Ipm_phy_sel	0	RW	Select IPM PHY for configuration. (register: 0x26000064) 0:FLC slice 0, ipm0 1:FLC slice 0, ipm1 2:FLC slice 1, ipm0 3:FLC slice 1, ipm1 4:FLC slice 2, ipm0 5:FLC slice 2, ipm1 6:FLC slice 3, ipm0 7:FLC slice 3, ipm1 8: ~~ipm9~~ 9-14: Reserved. 15: Broadcast (slice 0-3)
		11:8	d2d_sel	0	RW	Select D2D for configuration. 0-3: 1st 1x16 DDR[1:4] 4: Reserved 5: Reserved 6~9: Reserved. 10~14: Reserved. 15: Broadcast (slice 0-5)

						This field combines with address windows 0x27xxxxxx for D2D decoding.
		19:12	D2d_io_a31x24	0x2a	RW	D2D APB (A5L) access upper address bits: A[31:24]. Default points to apbx (0x2axxxxxx)
		31:20	Reserved			
0x2600003c	Alias to 0x2601003c					
0x26000040..0x2600005f	se_aes_key	255:0		0	RW	Security engine key
0x26000060	d2d_irq					
		9:0	d2d_irq_o[9:0]	0	RW	D2D Chiplet IRQ output. Bit 0..3 = Goldfinch [3:0] slice#0 Bit 4 = MKB D2D0 Bit 5 = MKB D2D1 Bit 6..9 = Reserved.
		19:10	d2d_irq_i[9:0]	0	R	D2D Chiplet IRQ input status. Corresponding to E21 IRQ[47:40, 37:36] Bit 0..3 = Goldfinch [3:0] slice#0 Bit 4 = MKB D2D0 Bit 5 = MKB D2D1 Bit 6..9 = Reserved
		31:20	Reserved			
0x26000064	ipm_phyconfig	0	ipm_mcu_core_rstn	0	RW	[FLC slice, IPM PHY] selected by pcie_flc_d2d_sel[7:4]
		1	ipm_mcu_uncore_rstn	1	RW	
		2	ipm_mcu_mem_load_en	0	RW	
		3	ipm_mcu_apb_en	0	RW	
		4	ipm_mcu_addr_offset	0	RW	0:mcu_addr_offset=0x20001: mcu_addr_offset=0x1000
		5	Boot_i	1	RW	“boot_i” pin of PHY
		6	Powerdn	0	RW	1=power down PHY
		7	Ipm_mcu_scratchpad	0	RW	1:mcu_addr_offset=0x3000

		31:8	Reserved			
0x26000068	dbl_rc0_x8_remap	19:0	dbl_rc0_x8_a_43_24	0	RW	dbl_rc0_x8 access window [0x50xx_xxxx] upper address bit
		31:20	Reserved			
0x2600006c	uc_rc0_x8_remap	19:0	uc_rc0_x8_a_43_24	0	RW	uc_rc0_x8 access window [0x51xx_xxxx] upper address bit
		31:20	Reserved			
0x26000070	uc_ep01_remap	15:0	uc_ep0_a_39_24	0	RW	uc_pcie access window [0x52xxxxxx] upper address bit
		31:16	uc_ep1_a_39_24	0	RW	uc_pcie access window [0x53xxxxxx] upper address bit
0x26000074	uc_ep23_remap	15:0	uc_ep2_a_39_24	0	RW	uc_pcie access window [0x54xxxxxx] upper address bit
		31:16	uc_ep3_a_39_24	0	RW	uc_pcie access window [0x55xxxxxx] upper address bit
0x26000078	dbl_rc1_x8_remap	19:0	dbl_rc1_x8_a_43_24	0	RW	dbl_rc1_x8 access window [0x5axx_xxxx] upper address bit
		31:20	uc_rc1_x8_a_35_24	0	RW	uc_rc1_x8 access window [0x5bxx_xxxx] upper address bit. A[35:24]
0x2600007c	uc_nic_remap	0	valid	0	RW	1=remap is valid
		3:1	Reserved	0		
		27:4	uc_nic_addr_47x24	0x500000	RW	A[47:24] of uc to NIC access
		31:28	uc_nic_addr_51x48	0	RW	A[51:48] of uc to NIC access
0x26000080~0x260000ff	FLC address remap					Offset: S0:80~9f, S1:a0~bf, S2:C0~df, S3:e0~ff.
	FLC_REMAP_L	0	valid	0	RW	
		5:1	length	0	RW	0=16M, 1=32M, 2=64M...16=1T

		7:6	Reserved			
		31:8	Base_address	0	RW	A[47:24]. Remap block size: 16MB
	FLC_REMAP_H	3:0	New_base_address_h	0	RW	A[51:48] after remap
		7:4	Reserved			
		31:8	New_base_address	0	RW	A[47:24] after remap
0x26000110~0x260003FF	Alias to 0x26010110~0x260103FF					

MKB PLL controller register

~~0x26000724~~ 0x26000100	Mockingbird_pll_ctrl1					Configure Mockingbird PLL1
		0	Pll_en	0	RW	PLL enable
		1	Dac_en	0	RW	Enabling fractional noise canceling DAC
		2	dskewcalbyp	0	RW	Deskew calibration bypass
		5:3	Dskewcalcnt[2:0]	2	RW	Counter for deskew calibration loop
		6	Dskewcalen	0	RW	Deskew calibration enable
		18:7	Dskewcalin[11:0]	0	RW	Dskewcalbyp=0: initial value for deskew calibration. Dskewcalbyp=1: override value of deskew calibration
		19	Dskewfastcal	0	RW	Deskew fast calibration enable.
		20	Dmsen	0	RW	Enable delta-sigma modulator. 0->DSM is powered down (integer mode). 1->DSM is powered up (fractional mode)
		21	Foutcmlen	0	RW	PLL bypass
		22	Foutdiffen	0	RW	
		27:23	foutvcobyp	0	RW	
		28	foutvcoen	0	RW	
		29	frefcmlen	0	RW	0=25Mhz SE reference clock
		30	PLL_LOCKED		R	PLL locked
		31	DESKEW_PLL_LOCKED		R	Deskew calibration settled.

~~0x26000728~~ 0x26000104	Mockingbird_pll_ctrl2	5:0	Refdiv[5:0]	1		Reference clock divider
		17:6	Fbdiv[11:0]	80		Feedback clock divider, 25MHz*80=2GHz
		19:18	Posdiv4	0		2G/4 -> fout CML (not used)
		23:20	Posdiv0	1		2G/2 -> logic_1g_1
		27:24	Posdiv1	1		2G/2 -> clk_riscv
		31:28	Posdiv2	7		2G/10 -> clk_uc_1
~~0x2600072c~~ 0x26000108	Mockingbird_pll_ctrl3	23:0	Frac[23:0]	0	RW	Fractional value of feed- back divider
		27:24	Posdiv3	0	RW	Ref_clk bypass (25MHz)
		31:28	fouten	0xf	RW	{fouten3,fouten2,fouten1,fouten0}

AXI4 Traffic Generator register

0x26000418	AXI4TG_CTRL	0	Axi4tg_en	RW		1=enable AXI4TG
		1	Axi4tg_start	RW		1=start AXI4TG
		6:2	Axi4tg_sel	RW		DUT selection. 0={slice0, ipm0}, 1={slcie0, ipm1}, 2={slice1, ipm0}, 3={slice1, ipm1}, 4={slice2, ipm0}, 5={slcie2, ipm1}, 6={slice3, ipm0}, 7={slice3, ipm1}, ~~8=ipm9,~~ 8={Goldfinch, d2d0, slice0}, 9={Goldfinch, d2d1, slice0}, 10={Goldfinch, d2d2, slice0}, 11={Goldfinch, d2d3, slice0}. 12=MKB D2D0 13=MKB D2D1
		15:7	Reserved	RW		
		16	Axi4tg_irq	R		
		17	Axi4tg_err	R		
		31:18	Reserved			

AXI4TG Register Base address = 0x2a020000

For more information, see axi-traffic-gen.pdf.

I3C controller register

0x2600041c~0x2600042c	I3C					I3C control registers

Register: I3C_DEBUG_PORT0

Address: 0x2600041C

Bits	Bit Name	Default	Type	Comment
31:0	i3cm_debug_port [31:0]	32'h8000555A	R	I3C master debug ports

Register: I3C_DEBUG_PORT1

Address: 0x26000420

Bits	Bit Name	Default	Type	Comment
31	wakeup	1'b0	R	I3C Slave rxc wakeup
30	i2c_glitch_filter_en	1'b1	R	I3C Slave I2C glitch filter enable
29:23	Reserved	N/A	N/A	Reserved
22:0	i3cm_debug_port [54:32]	23'h0	R	I3C master debug ports

Register: I3CS_CTRL1

Address: 0x26000424

Bits	Bit Name	Default	Type	Comment
31	Reserved	N/A	N/A	Reserved
30:23	i3cs_slv_dcr	8'b0	R/W	
22:20	i3cs_slv_max_wr_speed	3'b0	R/W	
19:17	i3cs_slv_max_rd_speed	3'b0	R/W	
16:10	i3cs_static_addr	7'b0	R/W	
9	i3cs_static_addr_en	1'b0	R/W	
8:5	i3cs_pending_int	4'b0	R/W	
4:3	i3cs_act_mode	2'b0	R/W	
2	i3cs_slv_test_mode	1'b0	R/W	
1	i3cs_mode_i2c	1'b0	R/W	
0	legacy_i2c_xfer	1'b0	R	

Register: I3CS_CTRL2

Address: 0x26000428

Bits	Bit Name	Default	Type	Comment
31:0	i3c_slave_slv_pid [31:0]	32'h0	R/W	I3C slave PID

Register: I3CS_CTRL3

Address: 0x2600042C

Bits	Bit Name	Default	Type	Comment
31:23	Reserved	N/A	N/A	Reserved
22:20	i3cs_slv_clk_data_turn_time	3'b0	R/W	
19:16	Reserved	N/A	N/A	Reserved
15:0	i3c_slave_slv_pid [47:32]	16'h0	R/W	I3C slave PID

I3C Register

I3C Master Register Base address = 0x2a000000

I3C Slave Register Base address = 0x2a010000

For more information, see Chapter 5 of DWC_mipi_i3c_databook.pdf

D2D PLL controller register

0x26000700	D2D_DDR0_pll_ctrl1					Configure D2D_DDR[0] PLL
		0	Pll_en	1	RW	PLL enable
		1	Dac_en	0	RW	Enabling fractional noise canceling DAC
		2	dskewcalbyp	0	RW	Deskew calibration bypass
		5:3	Dskewcalcnt[2:0]	2	RW	Counter for deskew calibration loop
		6	Dskewcalen	0	RW	Deskew calibration enable
		18:7	Dskewcalin[11:0]	0	RW	Dskewcalbyp=0: initial value for deskew calibration. Dskewcalbyp=1: override value of deskew calibration
		19	Dskewfastcal	0	RW	Deskew fast calibration enable.
		20	Dmsen	0	RW	Enable delta-sigma modulator. 0->DSM is powered down (integer mode). 1->DSM is powered up (fractional mode)
		21	Foutcmlen	0	RW	PLL bypass
		22	Foutdiffen	0	RW	
		27:23	foutvcobyp	0	RW	
		28	foutvcoen	1	RW	Vco (8G) drives BOW PHY
		29	frefcmlen	0	RW	0=100Mhz se reference clock
		31:30	Reserved			
0x26000704	D2D_DDR0_pll_ctrl2	5:0	Refdiv[5:0]	1		Reference clock divider
		17:6	Fbdiv[11:0]	80		Feedback clock divider, 100MHz*80=8GHz
		19:18	Posdiv4	0		8G/4 -> fout CML (not used)

		23:20	Posdiv0	0		Ref_clk bypass
		27:24	Posdiv1	0		Ref_clk bypass
		31:28	Posdiv2	0		Ref_clk bypass
0x26000708	D2D_DDR0_pll_ctrl3	23:0	Frac[23:0]	0	RW	Fractional value of feed- back divider
		24	BC_PHY_SYSRST	0	RW	
		25	PHY_EXT_LPBK	0	RW	
		26	LL_EXT_LPBK	0	RW	
		27	Reserved	0	RW	
		31:28	Reserved	0	R	
0x26000710	D2D_DDR0_pll_sts	0	PLL_LOCKED	0	R	GFH0 PLL status
		1	DESKEW_PLL_LOCKED		R	Deskew calibration settled.
		31:2	Reserved			
0x26000714	D2D_DDR1_pll_sts	0	PLL_LOCKED	0	R	GFH1 PLL status
		1	DESKEW_PLL_LOCKED		R	Deskew calibration settled.
		31:2	Reserved			
0x26000718	D2D_DDR2_pll_sts	0	PLL_LOCKED	0	R	GFH2 PLL status
		1	DESKEW_PLL_LOCKED		R	Deskew calibration settled.
		31:2	Reserved			
0x2600071C	D2D_DDR3_pll_sts	0	PLL_LOCKED	0	R	GFH3 PLL status
		1	DESKEW_PLL_LOCKED		R	Deskew calibration settled.
		31:2	Reserved			
0x26000720	MKB_D2D0_pll_sts	0	PLL_LOCKED0	0	R	MKB_D2D0 PLL status
		1	DESKEW_PLL_LOCKED0	0	R	Deskew calibration settled.
		2	PLL_LOCKED1	0	R	MKB_D2D1 PLL status
		3	DESKEW_PLL_LOCKED1	0	R	Deskew calibration settled.
		31:4	Reserved			

0x26000730	D2D_DDR1_pll_ctrl1					Configure D2D_DDR[1] PLL
		0	Pll_en	1	RW	PLL enable
		1	Dac_en	0	RW	Enabling fractional noise canceling DAC
		2	dskewcalbyp	0	RW	Deskew calibration bypass
		5:3	Dskewcalcnt[2:0]	2	RW	Counter for deskew calibration loop
		6	Dskewcalen	0	RW	Deskew calibration enable
		18:7	Dskewcalin[11:0]	0	RW	Dskewcalbyp=0: initial value for deskew calibration. Dskewcalbyp=1: override value of deskew calibration
		19	Dskewfastcal	0	RW	Deskew fast calibration enable.
		20	Dmsen	0	RW	Enable delta-sigma modulator. 0->DSM is powered down (integer mode). 1->DSM is powered up (fractional mode)
		21	Foutcmlen	0	RW	PLL bypass
		22	Foutdiffen	0	RW	
		27:23	foutvcobyp	0	RW	
		28	foutvcoen	1	RW	Vco (8G) drives BOW PHY
		29	frefcmlen	0	RW	0=25Mhz SE reference clock
		31:30	Reserved			
0x26000734	D2D_DDR1_pll_ctrl2	5:0	Refdiv[5:0]	1		Reference clock divider
		17:6	Fbdiv[11:0]	80		Feedback clock divider, 100MHz*80=8GHz
		19:18	Posdiv4	0		8G/4 -> fout CML (not used)
		23:20	Posdiv0	0		Ref_clk bypass
		27:24	Posdiv1	0		Ref_clk bypass
		31:28	Posdiv2	0		Ref_clk bypass

0x26000738	D2D_DDR1_pll_ctrl3	23:0	Frac[23:0]	0	RW	Fractional value of feed- back divider
		24	BC_PHY_SYSRST	0	RW	
		25	PHY_EXT_LPBK	0	RW	
		26	LL_EXT_LPBK	0	RW	
		27	Reserved	0	RW	
		31:28	Reserved	0	R	
0x2600073c	D2D_DDR2_pll_ctrl1					Configure D2D_DDR[2] PLL
		0	Pll_en	1	RW	PLL enable
		1	Dac_en	0	RW	Enabling fractional noise canceling DAC
		2	dskewcalbyp	0	RW	Deskew calibration bypass
		5:3	Dskewcalcnt[2:0]	2	RW	Counter for deskew calibration loop
		6	Dskewcalen	0	RW	Deskew calibration enable
		18:7	Dskewcalin[11:0]	0	RW	Dskewcalbyp=0: initial value for deskew calibration. Dskewcalbyp=1: override value of deskew calibration
		19	Dskewfastcal	0	RW	Deskew fast calibration enable.
		20	Dmsen	0	RW	Enable delta-sigma modulator. 0->DSM is powered down (integer mode). 1->DSM is powered up (fractional mode)
		21	Foutcmlen	0	RW	PLL bypass
		22	Foutdiffen	0	RW	
		27:23	foutvcobyp	0	RW	
		28	foutvcoen	1	RW	Vco (8G) drives BOW PHY
		29	frefcmlen	0	RW	0=25Mhz SE reference clock
		31:30	Reserved			
0x26000740	D2D_DDR2_pll_ctrl2	5:0	Refdiv[5:0]	1		Reference clock divider
		17:6	Fbdiv[11:0]	80		Feedback clock divider, 100MHz*80=8GHz
		19:18	Posdiv4	0		8G/4 -> fout CML (not used)

		23:20	Posdiv0	0		Ref_clk bypass
		27:24	Posdiv1	0		Ref_clk bypass
		31:28	Posdiv2	0		Ref_clk bypass
0x26000744	D2D_DDR2_pll_ctrl3	23:0	Frac[23:0]	0	RW	Fractional value of feed- back divider
		24	BC_PHY_SYSRST	0	RW	
		25	PHY_EXT_LPBK	0	RW	
		26	LL_EXT_LPBK	0	RW	
		27	Reserved	0	RW	
		31:28	Reserved	0	R	
0x26000748	D2D_DDR3_pll_ctrl1					Configure D2D_DDR[3] PLL
		0	Pll_en	1	RW	PLL enable
		1	Dac_en	0	RW	Enabling fractional noise canceling DAC
		2	dskewcalbyp	0	RW	Deskew calibration bypass
		5:3	Dskewcalcnt[2:0]	2	RW	Counter for deskew calibration loop
		6	Dskewcalen	0	RW	Deskew calibration enable
		18:7	Dskewcalin[11:0]	0	RW	Dskewcalbyp=0: initial value for deskew calibration. Dskewcalbyp=1: override value of deskew calibration
		19	Dskewfastcal	0	RW	Deskew fast calibration enable.
		20	Dmsen	0	RW	Enable delta-sigma modulator. 0->DSM is powered down (integer mode). 1->DSM is powered up (fractional mode)
		21	Foutcmlen	0	RW	PLL bypass
		22	Foutdiffen	0	RW	
		27:23	foutvcobyp	0	RW	
		28	foutvcoen	1	RW	Vco (8G) drives BOW PHY
		29	frefcmlen	0	RW	0=25Mhz SE reference clock

		31:30	Reserved			
0x2600074c	D2D_DDR3_pll_ctrl2	5:0	Refdiv[5:0]	1		Reference clock divider
		17:6	Fbdiv[11:0]	80		Feedback clock divider, 100MHz*80=8GHz
		19:18	Posdiv4	0		8G/4 -> fout CML (not used)
		23:20	Posdiv0	0		Ref_clk bypass
		27:24	Posdiv1	0		Ref_clk bypass
		31:28	Posdiv2	0		Ref_clk bypass
0x26000750	D2D_DDR3_pll_ctrl3	23:0	Frac[23:0]	0	RW	Fractional value of feed- back divider
		24	BC_PHY_SYSRST	0	RW	
		25	PHY_EXT_LPBK	0	RW	
		26	LL_EXT_LPBK	0	RW	
		27	Reserved	0	RW	
		31:28	Reserved	0	R	

0x26000754	MKB_D2D_pll_ctrl					Configure MKB_D2D PLL
		0	Pll_en	1	RW	PLL enable
		1	Dac_en	0	RW	Enabling fractional noise canceling DAC
		2	dskewcalbyp	0	RW	Deskew calibration bypass
		5:3	Dskewcalcnt[2:0]	2	RW	Counter for deskew calibration loop
		6	Dskewcalen	0	RW	Deskew calibration enable
		18:7	Dskewcalin[11:0]	0	RW	Dskewcalbyp=0: initial value for deskew calibration. Dskewcalbyp=1: override value of deskew calibration
		19	Dskewfastcal	0	RW	Deskew fast calibration enable.
		20	Dmsen	0	RW	Enable delta-sigma modulator. 0->DSM is powered down (integer mode). 1->DSM is powered up (fractional mode)

		21	Foutcmlen	0	RW	PLL bypass
		22	Foutdiffen	0	RW	
		27:23	foutvcobyp	0	RW	
		28	foutvcoen	1	RW	Vco (8G) drives BOW PHY
		29	frefcmlen	0	RW	0=25Mhz SE reference clock
		31:30	Reserved			
0x26000758	MKB_D2D_pll_ctrl2	5:0	Refdiv[5:0]	1		Reference clock divider
		17:6	Fbdiv[11:0]	80		Feedback clock divider, 100MHz*80=8GHz
		19:18	Posdiv4	0		8G/4 -> fout CML (not used)
		23:20	Posdiv0	0		Ref_clk bypass
		27:24	Posdiv1	0		Ref_clk bypass
		31:28	Posdiv2	0		Ref_clk bypass
0x2600075c	MKB_D2D_pll_ctrl3	23:0	Frac[23:0]	0	RW	Fractional value of feed- back divider
		24	BC_PHY_SYSRST	0	RW	
		25	PHY_EXT_LPBK	0	RW	
		26	LL_EXT_LPBK	0	RW	
		27	Reserved	0	RW	
		31:28	Reserved	0	R	

Temperature Sensor

0x26000760	TS_CTRL					
		0	Rstn	0	RW	
		1	Run	0	RW	
		9:2	cfg[7:0]	0	RW	
		13:10	An_sel[3:0]	0	RW	
		14	Pd_0	1	RW	
		15	Cload_0	0	RW	
		16	An_en_0	0	RW	
		17	Pd_1	1	RW	
		18	Cload_1	0	RW	
		19	An_en_1	0	RW	
		20	Pd_2	1	RW	
		21	Cload_2	0	RW	
		22	An_en_2	0	RW	
		23	Pd_3	1	RW	
		24	Cload_3	0	RW	
		25	An_en_3	0	RW	
		26	Pd_4	1	RW	
		27	Cload_4	0	RW	
		28	An_en_4	0	RW	
		29	Pd_5	1	RW	
		30	Cload_5	0	RW	
		31	An_en_5	0	RW	

0x26000764	TS_OUT_1	11:0	Dout_0[11:0]	0	R	
		12	Rdy_latched_0	0	R	
		13	Dout_type_0	0	R	
		14	Faultn_0	0	R	Fault flag

		15	Reserved			
		27:16	Dout_1[11:0]	0	R	
		28	Rdy_latched_1	0	R	
		29	Faultn_1	0	R	Fault flag
		30	Dout_type_1	0	R	
		31	Reserved			

0x26000768	TS_OUT_2	11:0	Dout_2[11:0]	0	R	
		12	Rdy_latched_2	0	R	
		13	Faultn_2	0	R	Fault flag
		14	Dout_type_2	0	R	
		15	Reserved			
		27:16	Dout_3[11:0]	0	R	
		28	Rdy_latched_3	0	R	
		29	Faultn_3	0	R	Fault flag
		30	Dout_type_3	0	R	
		31	Reserved			
0x2600076c	TS_OUT_3	11:0	Dout_4[11:0]	0	R	
		12	Rdy_latched_4	0	R	
		13	Faultn_4	0	R	Fault flag
		14	Dout_type_4	0	R	
		15	Reserved			
		27:16	Dout_5[11:0]	0	R	
		28	Rdy_latched_5	0	R	
		29	Faultn_5	0	R	Fault flag
		30	Dout_type_5	0	R	
		31	Reserved			

Voltage Sensor

0x26000780	VS_CTRL					
		0	Rstn	0	RW	
		1	Run	0	RW	
		9:2	Cfg1[7:0]	0	RW	
		17:10	Cfg2[7:0]	0	RW	
		18:21	An_sel[3:0]	0	RW	
		22	Pd	1	RW	
		23	Cload	0	RW	
		31:24	Reserved	0		

0x26000784	VS_OUT	13:0	Dout[13:0]	0	R	
		14	Rdy_latched	0	R	
		15	Faultn	0	R	Fault flag
		16	Dout_type	0	R	
		31:17	Reserved			

LVDS clock buffer

0x26000790	LVDS_CTRL					
		0	RXEN	1	RW	
		1	RTERM_EN	0	RW	
		2	RXCM_EN	0	RW	
		3	BIAS_EN	1	RW	
		4	~~BIAS_SEL~~ Reserved	0	RW	Always use int bias
		7:5	RTERM_VAL[2:0]	100	RW	
		31:8	Reserved			

MKB_DRO

0x26000800	MKB_DRO_0_1					
		0	DRO0_START	0	RW	
		1	DRO0_READY	0	R	
		7:2	Reserved	0	R	
		15:8	DRO0_MEA[7:0]	0	R	
		16	DRO1_START	0	RW	
		17	DRO1_READY	0	R	
		23:18	Reserved	0	R	
		31:24	DRO1_MEA[7:0]	0	R	

0x26000804	MKB_DRO_2_3					
		0	DRO2_START	0	RW	
		1	DRO2_READY	0	R	
		7:2	Reserved	0	R	
		15:8	DRO2_MEA[7:0]	0	R	
		16	DRO3_START	0	RW	
		17	DRO3_READY	0	R	
		23:18	Reserved	0	R	
		31:24	DRO3_MEA[7:0]	0	R	

0x26000808	MKB_DRO4_5					
		0	DRO4_START	0	RW	
		1	DRO4_READY	0	R	
		7:2	Reserved	0	R	
		15:8	DRO5_MEA[7:0]	0	R	
		16	DRO5_START	0	RW	
		17	DRO5_READY	0	R	
		23:18	Reserved	0	R	

		31:24	DRO5_MEA[7:0]	0	R	

Chip ID (read only)

0x26000fe0	UC_SYSCTRL_PID0	15:0	PID[15:0]	0x2	R	Part ID: 02-Bouffalo Lab IP
		26:16	JEPID[10:0]	0x489	R	JTAG ID
		30:27	REVISION[3:0]	0x1	R	Revision
		31	Reserved			
0x26000fe4	UC_SYSCTRL_PID1	3:0	ECOREVNUM	0x0	R	ECO revision
		31:4	Reserved	0x0		
0x26000ff0	UC_SYSCTRL_CID0	31:0	CID0	0x4249 5244	R	Chip ID0 ("BIRD")
0x26000ff4	UC_SYSCTRL_CID1	31:0	CID1	0x4b49 4e47	R	Chip ID1 ("KING")
0x26000ff8	UC_SYSCTRL_CID2	31:0	CID2	0x2020 4d4F	R	Chip ID2 (" MO")
0x26000ffc	UC_SYSCTRL_CID3	31:0	CID3	0x464c 4354	R	Chip ID3 ("FLCT")

7.2 GPIO Control Register

GPIO CSR Base address = 0x20005000

Gpio_func_sel: 0=SW control (SWGPIO), 1=UART, 2=SPI0, 3=I2C

Register: GPIO_MISC

Address: 0x20005080

Bits	Bit Name	Default	Type	Comment
31:21	Reservedreg_spi_swap	N/A	N/AR/W	ReservedSPI config: swap miso/mosi
1	reg_spi_swap	1'b0	R/W	SPI config: swap miso/mosi
0	reg_spi_master_mode	1'b0	R/W	SPI config: master/slave mode

Register: GPIO_CFGCTL0

Address: 0x20005100

Bits	Bit Name	Default	Type	Comment
31:28	Reserved	N/A	N/A	Reserved
27:24	reg_gpio_1_func_sel	4'h0	R/W	GPIO function select (Default: SWGPIO)
23:22	Reservedreg_spi_swap	N/A	N/AR/W	ReservedSPI config: swap miso/mosi
21	reg_gpio_1_pd	1'b0	R/W	GPIO pull down control
20	reg_gpio_1_pu	1'b0	R/W	GPIO pull up control
19:18	reg_gpio_1_drv	2'b0	R/W	GPIO driving control
17	reg_gpio_1_smt	1'b1	R/W	GPIO SMT control
16	reg_gpio_1_ie	1'b1	R/W	GPIO input enable
15:12	Reserved	N/A	N/A	Reserved
11:8	reg_gpio_0_func_sel	4'h0	R/W	GPIO function select (Default: SWGPIO)
7:6	Reserved	N/A	N/A	Reserved
5	reg_gpio_0_pd	1'b0	R/W	GPIO pull down control
4	reg_gpio_0_pu	1'b0	R/W	GPIO pull up control
3:2	reg_gpio_0_drv	2'b0	R/W	GPIO driving control
1	reg_gpio_0_smt	1'b1	R/W	GPIO SMT control
0	reg_gpio_0_ie	1'b1	R/W	GPIO input enable

Register: GPIO_CFGCTL1

Address: 0x20005104

Bits	Bit Name	Default	Type	Comment
31:28	Reserved	N/A	N/A	Reserved
27:24	reg_gpio_3_func_sel	4'h0	R/W	GPIO function select (Default: SWGPIO)

Commented [SC1]: This should be 31:1, right?

Commented [SC2]: Is this correct?

Formatted Table

Commented [SC3]: Is this correct?

Commented [SC4]: Is this correct?

Commented [SC5]: Is this correct?

Commented [SC6]: Is this correct?

Commented [SC7]: Is this correct?

Bits	Bit Name	Default	Type	Comment
23:22	Reservedreg_spi_swap	N/A	N/AR/W	ReservedSPI config: swap miso/mosi
21	reg_gpio_3_pd	1'b0	R/W	GPIO pull down control
20	reg_gpio_3_pu	1'b0	R/W	GPIO pull up control
19:18	reg_gpio_3_drv	2'b0	R/W	GPIO driving control
17	reg_gpio_3_smt	1'b1	R/W	GPIO SMT control
16	reg_gpio_3_ie	1'b1	R/W	GPIO input enable
15:12	Reserved	N/A	N/A	Reserved
11:8	reg_gpio_2_func_sel	4'h0	R/W	GPIO function select (Default: SWGPIO)
7:6	Reserved	N/A	N/A	Reserved
5	reg_gpio_2_pd	1'b0	R/W	GPIO pull down control
4	reg_gpio_2_pu	1'b0	R/W	GPIO pull up control
3:2	reg_gpio_2_drv	2'b0	R/W	GPIO driving control
1	reg_gpio_2_smt	1'b1	R/W	GPIO SMT control
0	reg_gpio_2_ie	1'b1	R/W	GPIO input enable

Commented [SC8]: Is this correct?

Commented [SC9]: Is this correct?

Commented [SC10]: Is this correct?

Register: GPIO_CFGCTL2

Address: 0x20005108

Bits	Bit Name	Default	Type	Comment
31:28	Reserved	N/A	N/A	Reserved
27:24	reg_gpio_5_func_sel	4'h0	R/W	GPIO function select (Default: SWGPIO)
23:22	Reservedreg_spi_swap	N/A	N/AR/W	ReservedSPI config: swap miso/mosi
21	reg_gpio_5_pd	1'b0	R/W	GPIO pull down control
20	reg_gpio_5_pu	1'b0	R/W	GPIO pull up control
19:18	reg_gpio_5_drv	2'b0	R/W	GPIO driving control
17	reg_gpio_5_smt	1'b1	R/W	GPIO SMT control
16	reg_gpio_5_ie	1'b1	R/W	GPIO input enable
15:12	Reserved	N/A	N/A	Reserved
11:8	reg_gpio_4_func_sel	4'h0	R/W	GPIO function select (Default: SWGPIO)
7:6	Reserved	N/A	N/A	Reserved
5	reg_gpio_4_pd	1'b0	R/W	GPIO pull down control
4	reg_gpio_4_pu	1'b0	R/W	GPIO pull up control
3:2	reg_gpio_4_drv	2'b0	R/W	GPIO driving control
1	reg_gpio_4_smt	1'b1	R/W	GPIO SMT control
0	reg_gpio_4_ie	1'b1	R/W	GPIO input enable

Commented [SC11]: Is this correct?

Commented [SC12]: Is this correct?

Commented [SC13]: Is this correct?

Register: GPIO_CFGCTL3

Address: 0x2000510C

Bits	Bit Name	Default	Type	Comment
31:28	Reserved	N/A	N/A	Reserved
27:24	reg_gpio_7_func_sel	4'h0	R/W	GPIO function select (Default: SWGPIO)
23:22	Reservedreg_spi_swap	N/A	N/AR/W	ReservedSPI config: swap miso/mosi
21	reg_gpio_7_pd	1'b0	R/W	GPIO pull down control
20	reg_gpio_7_pu	1'b0	R/W	GPIO pull up control
19:18	reg_gpio_7_drv	2'b0	R/W	GPIO driving control
17	reg_gpio_7_smt	1'b1	R/W	GPIO SMT control
16	reg_gpio_7_ie	1'b1	R/W	GPIO input enable
15:12	Reserved	N/A	N/A	Reserved
11:8	reg_gpio_6_func_sel	4'h0	R/W	GPIO function select (Default: SWGPIO)
7:6	Reserved	N/A	N/A	Reserved
5	reg_gpio_6_pd	1'b0	R/W	GPIO pull down control
4	reg_gpio_6_pu	1'b0	R/W	GPIO pull up control
3:2	reg_gpio_6_drv	2'b0	R/W	GPIO driving control
1	reg_gpio_6_smt	1'b1	R/W	GPIO SMT control
0	reg_gpio_6_ie	1'b1	R/W	GPIO input enable

Commented [SC14]: Is this correct?

Commented [SC15]: Is this correct?

Commented [SC16]: Is this correct?

Register: GPIO_CFGCTL4

Address: 0x20005110

Bits	Bit Name	Default	Type	Comment
31:28	Reserved	N/A	N/A	Reserved
27:24	reg_gpio_9_func_sel	4'h0	R/W	GPIO function select (Default: SWGPIO)
23:22	Reservedreg_spi_swap	N/A	N/AR/W	ReservedSPI config: swap miso/mosi
21	reg_gpio_9_pd	1'b0	R/W	GPIO pull down control
20	reg_gpio_9_pu	1'b0	R/W	GPIO pull up control
19:18	reg_gpio_9_drv	2'b0	R/W	GPIO driving control
17	reg_gpio_9_smt	1'b1	R/W	GPIO SMT control
16	reg_gpio_9_ie	1'b1	R/W	GPIO input enable
15:12	Reserved	N/A	N/A	Reserved
11:8	reg_gpio_8_func_sel	4'h0	R/W	GPIO function select (Default: SWGPIO)
7:6	Reserved	N/A	N/A	Reserved
5	reg_gpio_8_pd	1'b0	R/W	GPIO pull down control
4	reg_gpio_8_pu	1'b0	R/W	GPIO pull up control
3:2	reg_gpio_8_drv	2'b0	R/W	GPIO driving control
1	reg_gpio_8_smt	1'b1	R/W	GPIO SMT control
0	reg_gpio_8_ie	1'b1	R/W	GPIO input enable

Commented [SC17]: Is this correct?

Commented [SC18]: Is this correct?

Commented [SC19]: Is this correct?

Register: GPIO_CFGCTL5

Address: 0x20005114

Bits	Bit Name	Default	Type	Comment
31:28	Reserved	N/A	N/A	Reserved
27:24	reg_gpio_11_func_sel	4'h0	R/W	GPIO function select (Default: SWGPIO)
23:22	Reserved~~reg_spi_swap~~	N/A	N/A~~R/W~~	Reserved~~SPI config: swap miso/mosi~~
21	reg_gpio_11_pd	1'b0	R/W	GPIO pull down control
20	reg_gpio_11_pu	1'b0	R/W	GPIO pull up control
19:18	reg_gpio_11_drv	2'b0	R/W	GPIO driving control
17	reg_gpio_11_smt	1'b1	R/W	GPIO SMT control
16	reg_gpio_11_ie	1'b1	R/W	GPIO input enable
15:12	Reserved	N/A	N/A	Reserved
11:8	reg_gpio_10_func_sel	4'h0	R/W	GPIO function select (Default: SWGPIO)
7:6	Reserved	N/A	N/A	Reserved
5	reg_gpio_10_pd	1'b0	R/W	GPIO pull down control
4	reg_gpio_10_pu	1'b0	R/W	GPIO pull up control
3:2	reg_gpio_10_drv	2'b0	R/W	GPIO driving control
1	reg_gpio_10_smt	1'b1	R/W	GPIO SMT control
0	reg_gpio_10_ie	1'b1	R/W	GPIO input enable

Commented [SC20]: Is this correct?

Commented [SC21]: Is this correct?

Commented [SC22]: Is this correct?

Register: GPIO_CFGCTL6

Address: 0x20005118

Bits	Bit Name	Default	Type	Comment
31:28	Reserved	N/A	N/A	Reserved
27:24	reg_gpio_13_func_sel	4'h0	R/W	GPIO function select (Default: SWGPIO)
23:22	Reserved~~reg_spi_swap~~	N/A	N/A~~R/W~~	Reserved~~SPI config: swap miso/mosi~~
21	reg_gpio_13_pd	1'b0	R/W	GPIO pull down control
20	reg_gpio_13_pu	1'b0	R/W	GPIO pull up control
19:18	reg_gpio_13_drv	2'b0	R/W	GPIO driving control
17	reg_gpio_13_smt	1'b1	R/W	GPIO SMT control
16	reg_gpio_13_ie	1'b1	R/W	GPIO input enable
15:12	Reserved	N/A	N/A	Reserved
11:8	reg_gpio_12_func_sel	4'h0	R/W	GPIO function select (Default: SWGPIO)
7:6	Reserved	N/A	N/A	Reserved
5	reg_gpio_12_pd	1'b0	R/W	GPIO pull down control
4	reg_gpio_12_pu	1'b0	R/W	GPIO pull up control
3:2	reg_gpio_12_drv	2'b0	R/W	GPIO driving control
1	reg_gpio_12_smt	1'b1	R/W	GPIO SMT control
0	reg_gpio_12_ie	1'b1	R/W	GPIO input enable

Commented [SC23]: Is this correct?

Commented [SC24]: Is this correct?

Commented [SC25]: Is this correct?

Register: GPIO_CFGCTL7

Address: 0x2000511C

Bits	Bit Name	Default	Type	Comment
31:28	Reserved	N/A	N/A	Reserved
27:24	reg_gpio_15_func_sel	4'h0	R/W	GPIO function select (Default: SWGPIO)
23:22	Reservedreg_spi_swap	N/A	N/AR/W	ReservedSPI config: swap miso/mosi
21	reg_gpio_15_pd	1'b0	R/W	GPIO pull down control
20	reg_gpio_15_pu	1'b0	R/W	GPIO pull up control
19:18	reg_gpio_15_drv	2'b0	R/W	GPIO driving control
17	reg_gpio_15_smt	1'b1	R/W	GPIO SMT control
16	reg_gpio_15_ie	1'b1	R/W	GPIO input enable
15:12	Reserved	N/A	N/A	Reserved
11:8	reg_gpio_14_func_sel	4'h0	R/W	GPIO function select (Default: SWGPIO)
7:6	Reserved	N/A	N/A	Reserved
5	reg_gpio_14_pd	1'b0	R/W	GPIO pull down control
4	reg_gpio_14_pu	1'b0	R/W	GPIO pull up control
3:2	reg_gpio_14_drv	2'b0	R/W	GPIO driving control
1	reg_gpio_14_smt	1'b1	R/W	GPIO SMT control
0	reg_gpio_14_ie	1'b1	R/W	GPIO input enable

Commented [SC26]: Is this correct?

Commented [SC27]: Is this correct?

Commented [SC28]: Is this correct?

Register: GPIO_CFGCTL8

Address: 0x20005120

Bits	REGISTER NAME	Default	Type	Comment
31:28	Reserved	N/A	N/A	Reserved
27:24	reg_gpio_17_func_sel	4'h0	R/W	GPIO function select (Default: SWGPIO)
23:22	Reservedreg_spi_swap	N/A	N/AR/W	ReservedSPI config: swap miso/mosi
21	reg_gpio_17_pd	1'b0	R/W	GPIO pull down control
20	reg_gpio_17_pu	1'b0	R/W	GPIO pull up control
19:18	reg_gpio_17_drv	2'b0	R/W	GPIO driving control
17	reg_gpio_17_smt	1'b1	R/W	GPIO SMT control
16	reg_gpio_17_ie	1'b1	R/W	GPIO input enable
15:12	Reserved	N/A	N/A	Reserved
11:8	reg_gpio_16_func_sel	4'h0	R/W	GPIO function select (Default: SWGPIO)
7:6	Reserved	N/A	N/A	Reserved
5	reg_gpio_16_pd	1'b0	R/W	GPIO pull down control
4	reg_gpio_16_pu	1'b0	R/W	GPIO pull up control
3:2	reg_gpio_16_drv	2'b0	R/W	GPIO driving control
1	reg_gpio_16_smt	1'b1	R/W	GPIO SMT control
0	reg_gpio_16_ie	1'b1	R/W	GPIO input enable

Commented [SC29]: Is this correct?

Commented [SC30]: Is this correct?

Commented [SC31]: Is this correct?

Register: GPIO_CFGCTL9

Address: 0x20005124

Bits	Bit Name	Default	Type	Comment
31:28	Reserved	N/A	N/A	Reserved
27:24	reg_gpio_19_func_sel	4'h0	R/W	GPIO function select (Default: SWGPIO)
23:22	Reserved~~reg_spi_swap~~	N/A	N/A~~R/W~~	Reserved~~SPI config: swap miso/mosi~~
21	reg_gpio_19_pd	1'b0	R/W	GPIO pull down control
20	reg_gpio_19_pu	1'b0	R/W	GPIO pull up control
19:18	reg_gpio_19_drv	2'b0	R/W	GPIO driving control
17	reg_gpio_19_smt	1'b1	R/W	GPIO SMT control
16	reg_gpio_19_ie	1'b1	R/W	GPIO input enable
15:12	Reserved	N/A	N/A	Reserved
11:8	reg_gpio_18_func_sel	4'h0	R/W	GPIO function select (Default: SWGPIO)
7:6	Reserved	N/A	N/A	Reserved
5	reg_gpio_18_pd	1'b0	R/W	GPIO pull down control
4	reg_gpio_18_pu	1'b0	R/W	GPIO pull up control
3:2	reg_gpio_18_drv	2'b0	R/W	GPIO driving control
1	reg_gpio_18_smt	1'b1	R/W	GPIO SMT control
0	reg_gpio_18_ie	1'b1	R/W	GPIO input enable

Commented [SC32]: Is this correct?

Commented [SC33]: Is this correct?

Commented [SC34]: Is this correct?

Register: GPIO_CFGCTL30

Address: 0x20005180

Bits	Bit Name	Default	Type	Comment
31:20	Reserved	N/A	N/A	Reserved
19	reg_gpio_19_i	1'b0	R	Register controlled GPIO input value
18	reg_gpio_18_i	1'b0	R	Register controlled GPIO input value
17	reg_gpio_17_i	1'b0	R	Register controlled GPIO input value
16	reg_gpio_16_i	1'b0	R	Register controlled GPIO input value
15	reg_gpio_15_i	1'b0	R	Register controlled GPIO input value
14	reg_gpio_14_i	1'b0	R	Register controlled GPIO input value
13	reg_gpio_13_i	1'b0	R	Register controlled GPIO input value
12	reg_gpio_12_i	1'b0	R	Register controlled GPIO input value
11	reg_gpio_11_i	1'b0	R	Register controlled GPIO input value
10	reg_gpio_10_i	1'b0	R	Register controlled GPIO input value
9	reg_gpio_9_i	1'b0	R	Register controlled GPIO input value
8	reg_gpio_8_i	1'b0	R	Register controlled GPIO input value
7	reg_gpio_7_i	1'b0	R	Register controlled GPIO input value
6	reg_gpio_6_i	1'b0	R	Register controlled GPIO input value
5	reg_gpio_5_i	1'b0	R	Register controlled GPIO input value
4	reg_gpio_4_i	1'b0	R	Register controlled GPIO input value
3	reg_gpio_3_i	1'b0	R	Register controlled GPIO input value

Bits	Bit Name	Default	Type	Comment
2	reg_gpio_2_i	1'b0	R	Register controlled GPIO input value
1	reg_gpio_1_i	1'b0	R	Register controlled GPIO input value
0	reg_gpio_0_i	1'b0	R	Register controlled GPIO input value

Register: GPIO_CFGCTL32

Address: 0x20005188

Bits	Bit Name	Default	Type	Comment
31:20	Reserved	N/A	N/A	Reserved
19	reg_gpio_19_o	1'b0	R/W	Register controlled GPIO output Value
18	reg_gpio_18_o	1'b0	R/W	Register controlled GPIO output Value
17	reg_gpio_17_o	1'b0	R/W	Register controlled GPIO output Value
16	reg_gpio_16_o	1'b0	R/W	Register controlled GPIO output Value
15	reg_gpio_15_o	1'b0	R/W	Register controlled GPIO output Value
14	reg_gpio_14_o	1'b0	R/W	Register controlled GPIO output Value
13	reg_gpio_13_o	1'b0	R/W	Register controlled GPIO output Value
12	reg_gpio_12_o	1'b0	R/W	Register controlled GPIO output Value
11	reg_gpio_11_o	1'b0	R/W	Register controlled GPIO output Value
10	reg_gpio_10_o	1'b0	R/W	Register controlled GPIO output Value
9	reg_gpio_9_o	1'b0	R/W	Register controlled GPIO output Value
8	reg_gpio_8_o	1'b0	R/W	Register controlled GPIO output Value
7	reg_gpio_7_o	1'b0	R/W	Register controlled GPIO output Value
6	reg_gpio_6_o	1'b0	R/W	Register controlled GPIO output Value
5	reg_gpio_5_o	1'b0	R/W	Register controlled GPIO output Value
4	reg_gpio_4_o	1'b0	R/W	Register controlled GPIO output Value
3	reg_gpio_3_o	1'b0	R/W	Register controlled GPIO output Value
2	reg_gpio_2_o	1'b0	R/W	Register controlled GPIO output Value
1	reg_gpio_1_o	1'b0	R/W	Register controlled GPIO output Value
0	reg_gpio_0_o	1'b0	R/W	Register controlled GPIO output Value

Register: GPIO_CFGCTL34

Address: 0x20005190

Bits	Bit Name	Default	Type	Comment
31:20	Reserved	N/A	N/A	Reserved
19	reg_gpio_19_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
18	reg_gpio_18_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
17	reg_gpio_17_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)

Bits	Bit Name	Default	Type	Comment
16	reg_gpio_16_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
15	reg_gpio_15_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
14	reg_gpio_14_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
13	reg_gpio_13_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
12	reg_gpio_12_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
11	reg_gpio_11_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
10	reg_gpio_10_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
9	reg_gpio_9_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
8	reg_gpio_8_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
7	reg_gpio_7_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
6	reg_gpio_6_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
5	reg_gpio_5_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
4	reg_gpio_4_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
3	reg_gpio_3_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
2	reg_gpio_2_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
1	reg_gpio_1_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
0	reg_gpio_0_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)

Register: GPIO_INT_MASK1

Address: 0x200051A0

Bits	Bit Name	Default	Type	Comment
~~19:0~~	~~reg_gpio_int_mask1~~	~~{20{1'b1}}~~	~~R/W~~	~~reg_gpio_int_mask [19:0]~~
31:20	Reserved	N/A	N/A	Reserved
19:0	reg_gpio_int_mask1	20'hFFFFF	R/W	reg_gpio_int_mask [19:0]

Commented [SC35]: Why is this one 19:10?

Commented [SC36]: Why is this one 19:10?

Register: GPIO_INT_STAT1

Address: 0x200051A8

Bits	Bit Name	Default	Type	Comment
31:20	Reserved	N/A	N/A	Reserved
19:0	gpio_int_stat1	20'b0	R	gpio_int_stat [19:0]

Formatted: Font: (Default) +Body (Times New Roman), Not Bold

Formatted: Left

Formatted Table

Register: GPIO_INT_CLR1

Address: 0x200051B0

Bits	Bit Name	Default	Type	Comment
31:20	Reserved	N/A	N/A	Reserved
19:0	reg_gpio_int_clr1	20'b0	R/W	reg_gpio_int_clr [19:0]

Formatted: Font: (Default) +Body (Times New Roman), Not Bold

Formatted: Left

Register: GPIO_INT_MODE_SET1

Address: 0x200051C0

Bits	Bit Name	Default	Type	Comment
31:30	Reserved	N/A	N/A	Reserved
29:0	reg_gpio_int_mode_set1	20'b0	R/W	{reg_gpio9_int_mode [2:0], ... reg_gpio0_int_mode [2:0]}

Register: GPIO_INT_MODE_SET2

Address: 0x200051C4

Bits	Bit Name	Default	Type	Comment
31:30	Reserved	N/A	N/A	Reserved
29:0	reg_gpio_int_mode_set2	20'b0	R/W	{reg_gpio19_int_mode [2:0], ... reg_gpio10_int_mode [2:0]}

7.3 Serial Flash Control Register

Serial Flash CSR Base address = 0x26001000

Register: SF_CTRL_0

Address: 0x26001000

Bits	Bit Name	Default	Type	Comment
31:24	sf_id	8'h1A	R/W	ID
23	sf_aes_iv_endian	1'b1	R/W	0: Little-endian 1: Big-endian
22	sf_aes_key_endian	1'b1	R/W	0: Little-endian 1: Big-endian
21	sf_aes_din_endian	1'b1	R/W	0: Little-endian 1: Big-endian
20	sf_aes_dout_endian	1'b1	R/W	0: Little-endian 1: Big-endian
19	sf_if_32b_adr_en	1'b0	R/W	1: Enable 32-bit address
18	sf_if_int_set	1'b0	R/W	1: Set interrupt
17	sf_if_int_clr	1'b1	R/W	1: Clear interrupt
16	sf_if_int	1'b0	R	Interrupt value
15:12	Reserved	N/A	N/A	Reserved
11	sf_if_read_dly_en	1'b0	R/W	1: Enable flash controller read delay mode
10:8	sf_if_read_dly_n	3'b0	R/W	Flash controller read delay cycle = n + 1
7:5	Reserved	N/A	N/A	Reserved
4	sf_clk_out_inv_sel	1'b1	R/W	1: Select inverted clock out
3	sf_clk_out_gate_en	1'b1	R/W	1: Hardware auto gated clock out
2	sf_clk_sf_rx_inv_sel	1'b1	R/W	1: Select inverted flash Rx clock
1:0	Reserved	N/A	N/A	Reserved

Formatted Table

Formatted: Left

Register: SF_CTRL_1

Address: 0x26001004

Bits	Bit Name	Default	Type	Comment
31	sf_ahb2sram_en	1'b1	R/W	1: Enable SRAM
30	sf_ahb2sif_en	1'b1	R/W	1: Enable IAHB to flash interface
29	sf_if_en	1'b1	R/W	0: Disable `sf_if` 1: enable `sf_if`
28	sf_if_fn_sel	1'b1	R/W	0: System AHB 1: icache AHB
27	sf_ahb2sif_stop	1'b0	R/W	1: Stop CPU access flash

Formatted Table

Bits	Bit Name	Default	Type	Comment
26	sf_ahb2sif_stopped	1'b0	R	1: CPU stopped
25	sf_if_reg_wp	1'b1	R/W	Write protect
24	sf_if_reg_hold	1'b1	R/W	Hold
23	sf_ahb2sif_diswrap	1'b0	R/W	1: Disable IAHB to flash wrap access for XTS mode
22:20	sf_if_0_ack_lat	3'b110	R/W	ACK latency cycles
19	ReservedReserved	N/A	N/A	ReservedReserved
18	sf_if_sr_int_set	1'b0	R/W	1: Set interrupt
17	sf_if_sr_int_en	1'b0	R/W	1: Status read interrupt enable
16	sf_if_sr_int	1'b0	R	Interrupt value
15:8	sf_if_sr_pat	8'b0	R/W	Interrupt pattern
7:0	sf_if_sr_pat_mask	1'b0	R/W	Interrupt mask

Formatted Table

Formatted: Not Highlight

Register: SF_IF_SAHB_0

Address: 0x26001008

Bits	Bit Name	Default	Type	Comment
31	sf_if_0_qpi_mode_en	1'b0	R/W	0: Normal SPI 1: QPI mode enable
30:28	sf_if_0_spi_mode	3'b000	R/W	000: Normal SPI 001: Dual output 010: Quad output 011: Dual IO 100: Quad IO
27	sf_if_0_cmd_en	1'b1	R/W	1: Command enable
26	sf_if_0_adr_en	1'b1	R/W	1: Address enable
25	sf_if_0_dmy_en	1'b0	R/W	1: Dummy enable
24	sf_if_0_dat_en	1'b1	R/W	1: Data enable
23	sf_if_0_dat_rw	1'b0	R/W	0: Read 1: write
22:20	sf_if_0_cmd_byte	3'b000	R/W	Number of command bytes minus 1
19:17	sf_if_0_adr_byte	3'b010	R/W	Number of address bytes minus 1
16:12	sf_if_0_dmy_byte	5'b0	R/W	Number of dummy bytes minus 1
11:2	sf_if_0_dat_byte	10'hFF	R/W	Number of command byte minus 1
1	sf_if_0_trig	1'b0	R/W	1: Trigger sf_if FSM
0	sf_if_busy	1'b0	R	1: Busy

Register: SF_IF_SAHB_1

Address: 0x2600100C

Bits	Bit Name	Default	Type	Comment
31:0	sf_if_0_cmd_buf_0	32'h03000000	R/W	Command buffer 0

Register: SF_IF_SAHB_2

Address: 0x26001010

Bits	Bit Name	Default	Type	Comment
31:0	sf_if_0_cmd_buf_1	32'h0	R/W	Command buffer 1

Register: SF_IF_IAHB_0

Address: 0x26001014

Bits	Bit Name	Default	Type	Comment
31	sf_if_1_qpi_mode_en	1'b0	R/W	0: Normal SPI 1: QPI mode enable
30:28	sf_if_1_spi_mode	3'b000	R/W	000: Normal SPI 001: Dual output 010: Quad output 011: Dual IO 100: Quad IO
27	sf_if_1_cmd_en	1'b1	R/W	1: Command enable
26	sf_if_1_adr_en	1'b1	R/W	1: Address enable
25	sf_if_1_dmy_en	1'b0	R/W	1: Dummy enable
24	sf_if_1_dat_en	1'b1	R/W	1: Data enable
23	sf_if_1_dat_rw	1'b0	R/W	0: Read 1: write
22:20	sf_if_1_cmd_byte	3'b000	R/W	Number of command bytes minus 1
19:17	sf_if_1_adr_byte	3'b010	R/W	Number of address bytes minus 1
16:12	sf_if_1_dmy_byte	5'b0	R/W	Number of dummy bytes minus 1
11:0	Reserved	N/A	N/A	Reserved

Register: SF_IF_IAHB_1

Address: 0x26001018

Bits	Bit Name	Default	Type	Comment
31:0	sf_if_1_cmd_buf_0	32'h03000000	R/W	Command buffer 0

Register: SF_IF_IAHB_2

Address: 0x2600101C

Bits	Bit Name	Default	Type	Comment
31:0	sf_if_1_cmd_buf_1	32'h0	R/W	Command buffer 1

Register: SF_IF_STATUS_0

Address: 0x26001020

Bits	Bit Name	Default	Type	Comment
31:0	sf_if_status_0	32'h0	R	

Register: SF_IF_STATUS_1

Address: 0x26001024

Bits	Bit Name	Default	Type	Comment
31:0	sf_if_status_1	32'h20000000	R	

Register: SF_AHB2SIF_STATUS

Address: 0x2600102C

Bits	Bit Name	Default	Type	Comment
31:0	sf_ahb2sif_status	32'h1010003	R	

Register: SF_IF_IO_DLY_0

Address: 0x26001030

Bits	Bit Name	Default	Type	Comment
31:30	sf_dqs_do_dly_sel	2'b0	R/W	IO delay selection
29:28	sf_dqs_di_dly_sel	2'b0	R/W	IO delay selection
27:26	sf_dqs_oe_dly_sel	2'b0	R/W	IO delay selection
25:10	Reserved	N/A	N/A	Reserved
9:8	sf_clk_out_dly_sel	2'b0	R/W	IO delay selection
7:4	Reserved	N/A	N/A	Reserved
3:2	sf_cs2_dly_sel	2'b0	R/W	IO delay selection
1:0	sf_cs_dly_sel	2'b0	R/W	IO delay selection

Register: SF_IF_IO_DLY_1

Address: 0x26001034

Bits	Bit Name	Default	Type	Comment
31:18	Reserved	N/A	N/A	Reserved
17:16	sf_io_0_do_dly_sel	2'b0	R/W	IO delay selection
15:10	Reserved	N/A	N/A	Reserved
9:8	sf_io_0_di_dly_sel	2'b0	R/W	IO delay selection

Bits	Bit Name	Default	Type	Comment
7:2	Reserved	N/A	N/A	Reserved
1:0	sf_io_0_oe_dly_sel	2'b0	R/W	IO delay selection

Register: SF_IF_IO_DLY_2

Address: 0x26001038

Bits	Bit Name	Default	Type	Comment
31:18	Reserved	N/A	N/A	Reserved
17:16	sf_io_1_do_dly_sel	2'b0	R/W	IO delay selection
15:10	Reserved	N/A	N/A	Reserved
9:8	sf_io_1_di_dly_sel	2'b0	R/W	IO delay selection
7:2	Reserved	N/A	N/A	Reserved
1:0	sf_io_1_oe_dly_sel	2'b0	R/W	IO delay selection

Register: SF_IF_IO_DLY_3

Address: 0x2600103C

Bits	Bit Name	Default	Type	Comment
31:18	Reserved	N/A	N/A	Reserved
17:16	sf_io_2_do_dly_sel	2'b0	R/W	IO delay selection
15:10	Reserved	N/A	N/A	Reserved
9:8	sf_io_2_di_dly_sel	2'b0	R/W	IO delay selection
7:2	Reserved	N/A	N/A	Reserved
1:0	sf_io_2_oe_dly_sel	2'b0	R/W	IO delay selection

Register: SF_IF_IO_DLY_4

Address: 0x26001040

Bits	Bit Name	Default	Type	Comment
31:18	Reserved	N/A	N/A	Reserved
17:16	sf_io_3_do_dly_sel	2'b0	R/W	IO delay selection
15:10	Reserved	N/A	N/A	Reserved
9:8	sf_io_3_di_dly_sel	2'b0	R/W	IO delay selection
7:2	Reserved	N/A	N/A	Reserved
1:0	sf_io_3_oe_dly_sel	2'b0	R/W	IO delay selection

Register: SF_RESERVED

Address: 0x26001044

Bits	Bit Name	Default	Type	Comment
31:0	sf_reserved	32'hFFFF	R/W	~~31:0~~

Register: SF_CTRL_2

Address: 0x26001070

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	N/A	N/A	Reserved
7	sf_id_offset_lock	1'b0	R/W	
6:0	Reserved	N/A	N/A	Reserved

Register: SF_CTRL_3

Address: 0x26001074

Bits	Bit Name	Default	Type	Comment
31:29	sf_if_1_ack_lat	3'b001	R/W	ACK latency cycles
28:21	Reserved	N/A	N/A	Reserved
20	sf_cmds_core_en	1'b1	R/W	1: Enable command splitter core
19:18	sf_cmds_1_wrap_mode [1:0]	2'b0	R/W	[0] 0: cmds bypass wrap commands to macro 1: cmds handle wrap commands. [1] 0: original mode 1: cmds force wrap16x4 is split into two wrap 8x4
17	sf_cmds_1_en	1'b0	R/W	1: Enable command splitter
16:13	sf_cmds_1_wrap_len	4'd6	R/W	Wrap length 0:8, 1:16, 2:32, 3:64, 4:128, 5:256, 6:512, 7:1024, 8:2048, 9:4096
12:0	Reserved	N/A	N/A	Reserved

Register: SF_ID0_OFFSET

Address: 0x260010A0

Bits	Bit Name	Default	Type	Comment
31:28	Reserved	N/A	N/A	Reserved
27:0	sf_id0_offset	28'b0	R/W	

Register: SF_DBG

Address: 0x260010B0

Bits	Bit Name	Default	Type	Comment
31:7	Reserved	N/A	N/A	Reserved
6	sf_autoload_st_done	1'b0	R	
4:0	sf_autoload_st	5'h1	R	

Register: SF_CTRL_PROT_EN_RD

Address: 0x26001100

Bits	Bit Name	Default	Type	Comment
31	sf_dbg_dis	1'b0	R	
30	sf_if_0_trig_wr_lock	1'b0	R	
29	Reserved	N/A	N/A	Reserved
28	sf_sec_tzsid_lock	1'b0	R	
27:0	Reserved	N/A	N/A	Reserved

Register: SF SRAM

Address: 0x26001600~0x2600167F

Bits	Bit Name	Default	Type	Comment
31:0	SF SRAM	N/A	R/W	Address [6:0] = SF SRAM address 128x8x4

7.4 L1C Control Register

L1C CSR Base address = 0x26002000

Register: L1C_CONFIG

Address: 0x26002000

Bits	Bit Name	Default	Type	Comment
31:14	Reserved	N/A	N/A	Reserved
13:12	sf_clk_sel [1:0]	2'b00	R/W	00 = 25MHz 01 = 50MHz 1X = 100MHz
11:8	l1c_way_dis	4'b0	R/W	Disable part of cache ways & used as AHB SRAM
7:2	Reserved	N/A	N/A	Reserved
1	l1c_cnt_en	1'b0	R/W	Cache performance counter enable
0	l1c_cacheable	1'b0	R/W	Cacheable regions enable

Register: L1C_HIT_CNT_LSB

Address: 0x26002004

Bits	Bit Name	Default	Type	Comment
31:0	l1c_hit_cnt_lsb	32'h0	R	Low 32-bit hit couter

Register: L1C_HIT_CNT_MSB

Address: 0x26002008

Bits	Bit Name	Default	Type	Comment
31:0	l1c_hit_cnt_msb	32'h0	R	High 32-bit hit counter

Register: L1C_MISS_CNT

Address: 0x2600200C

Bits	Bit Name	Default	Type	Comment
31:0	l1c_miss_cnt	32'h0	R	Miss counter

Register: L1C_AUX_CONFIG

Address: 0x26002010

Bits	Bit Name	Default	Type	Comment
31	l1c_invalid_done	1'b0	R	
30:4	Reserved	N/A	N/A	Reserved
3	early_resp_dis	1'b0	R/W	
2	xip_2t_access	1'b0	R/W	Set 1 for ROM 2T access if CPU freq >72MHz
1	l1c_invalid	1'b0	R/W	
0	l1c_en	1'b0	R/W	1: burst=3'b100 0: burst=3'b001

7.5 Security Engine Control Register

Security Engine CSR Base address = 0x26005000

Register: SE_SHA_0_CTRL

Address: 0x26005000

Bits	Bit Name	Default	Type	Comment
31:16	se_sha_0_msg_len	16'h0	R/W	Number of 512-bit block
15:12	Reserved	NA	NA	Reserved
11	se_sha_0_int_mask	1'b0	R/W	se_sha_0 interrupt mask
10	se_sha_0_int_set_1t	1'b0	W1p	1: Set Interrupt
9	se_sha_0_int_clr_1t	1'b0	W1p	1: Clear Interrupt
8	se_sha_0_int	1'b0	R	Interrupt value
7	Reserved	NA	NA	Reserved
6	se_sha_0_hash_sel	1'b0	R/W	0: New Hash 1: Accumulate Last Hash
5	se_sha_0_en	1'b0	R/W	1: Enable sha Engine
4:2	se_sha_0_mode	3'b000	R/W	000: SHA-256 001: SHA-224 010: SHA-1 011: SHA-1 100: SHA-512 101: SHA-384 110: SHA-512/224 111: SHA-512/256
1	se_sha_0_trig_1t	1'b0	W1p	1: Trigger sha Engine
0	se_sha_0_busy	1'b0	R	1: sha Engine Busy

Register: SE_SHA_0_MSA

Address: 0x26005004

Bits	Bit Name	Default	Type	Comment
31:0	se_sha_0_msa	32'h0	R/W	Message Source Address

Register: SE_SHA_0_STATUS

Address: 0x26005008

Bits	Bit Name	Default	Type	Comment
31:0	se_sha_0_status	32'h41	R	"TODO"

Register: SE_SHA_0_ENDIAN

Address: 0x2600500C

Bits	Bit Name	Default	Type	Comment
31:1	Reserved	N/A~~0x1~~	N/A~~R/W~~	Reserved
0	se_sha_0_dout_endian	1'b1	R/W	0: Little-Endian 1: Big-Endian

Formatted: Indent: Left: 0"

Formatted: Indent: Left: 0"

Register: SE_SHA_0_HASH_L_0

Address: 0x26005010

Bits	Bit Name	Default	Type	Comment
31:0	se_sha_0_hash_l_0	32'h0	R	Big-Endian Hash 0 (MSB)

Register: SE_SHA_0_HASH_L_1

Address: 0x26005014

Bits	Bit Name	Default	Type	Comment
31:0	se_sha_0_hash_l_1	32'h0	R	Big-Endian Hash 1

Register: SE_SHA_0_HASH_L_2

Address: 0x26005018

Bits	Bit Name	Default	Type	Comment
31:0	se_sha_0_hash_l_2	32'h0	R	Big-Endian Hash 2

Register: SE_SHA_0_HASH_L_3

Address: 0x2600501C

Bits	Bit Name	Default	Type	Comment
31:0	se_sha_0_hash_l_3	32'h0	R	Big-Endian Hash 3

Register: SE_SHA_0_HASH_L_4

Address: 0x26005020

Bits	Bit Name	Default	Type	Comment
31:0	se_sha_0_hash_l_4	32'h0	R	Big-Endian Hash 4

Register: SE_SHA_0_HASH_L_5

Address: 0x26005024

Bits	Bit Name	Default	Type	Comment
31:0	se_sha_0_hash_l_5	32'h0	R	Big-Endian Hash 5

Register: SE_SHA_0_HASH_L_6

Address: 0x26005028

Bits	Bit Name	Default	Type	Comment
31:0	se_sha_0_hash_l_6	32'h0	R	Big-Endian Hash 6

Register: SE_SHA_0_HASH_L_7

Address: 0x2600502C

Bits	Bit Name	Default	Type	Comment
31:0	se_sha_0_hash_l_7	32'h0	R	Big-Endian Hash 7 (LSB)

Register: SE_SHA_0_CTRL_PROT

Address: 0x260050FC

Bits	Bit Name	Default	Type	Comment
31:3	Reserved	NA	NA	Reserved
2	se_sha_id1_en	1'b1	R/W	id1 Access Right
1	se_sha_id0_en	1'b1	R/W	id0 Access Right
0	Reserved	NA	NA	Reserved

Register: SE_AES_0_CTRL

Address: 0x26005100

Bits	Bit Name	Default	Type	Comment
31:16	se_aes_0_msg_len	16'h0	R/W	Number of 128-bit Block
15	Reserved	NA	NA	Reserved
14	se_aes_0_iv_sel	1'b0	R/W	0: New iv 1: Same iv as Last One
13:12	se_aes_0_block_mode	2'b00	R/W	00: ECB mode 01: CTR mode 10: CBC mode 11: XTS mode
11	se_aes_0_int_mask	1'b0	R/W	se_aes_0 interrupt mask
10	se_aes_0_int_set_1t	1'b0	W1p	1: Set Interrupt
9	se_aes_0_int_clr_1t	1'b0	W1p	1: Clear Interrupt

Bits	Bit Name	Default	Type	Comment
8	se_aes_0_int	1'b0	R	Interrupt Value
7	se_aes_0_hw_key_en	1'b0	R/W	0: sw Key 1: hw Key
6	se_aes_0_dec_key_sel	1'b0	R/W	0: New Key 1: Same Key as Last One
5	se_aes_0_dec_en	1'b0	R/W	0: Encode 1: Decode
4:3	se_aes_0_mode	2'b00	R/W	00: 128-bit Mode 01: 256-bit Mode 10: 192-bit Mode 11: 128-bit Double Key Mode
2	se_aes_0_en	1'b0	R/W	1: Enable
1	se_aes_0_trig_1t	1'b0	W1p	1: Trigger aes Engine
0	se_aes_0_busy	1'b0	R	1: aes Engine Busy

Register: SE_AES_0_MSA

Address: 0x26005104

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_msa	32'h0	R/W	Message Source Address

Register: SE_AES_0_MDA

Address: 0x26005108

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_mda	32'h0	R/W	Message Destination Address

Register: SE_AES_0_STATUS

Address: 0x2600510C

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_status	32'h4100	R	*"TODO"*

Register: SE_AES_0_IV_0

Address: 0x26005110

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_iv_0	32'h0	R/W	Big Endian Initial Vector (MSB)

Register: SE_AES_0_IV_1

Address: 0x26005114

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_iv_1	32'h0	R/W	Big Endian Initial Vector

Register: SE_AES_0_IV_2

Address: 0x26005118

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_iv_2	32'h0	R/W	Big Endian Initial Vector

Register: SE_AES_0_IV_3

Address: 0x2600511C

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_iv_3	32'h0	R/W	Big Endian Initial Vector (LSB) (CTR Mode: 32-bit Counter Initial Value)

Register: SE_AES_0_KEY_0

Address: 0x26005120

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_key_0	32'h0	R/W	Big Endian aes Key (aes-128/256 Key MSB)

Register: SE_AES_0_KEY_1

Address: 0x26005124

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_key_1	32'h0	R/W	Big Endian aes Key

Register: SE_AES_0_KEY_2

Address: 0x26005128

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_key_2	32'h0	R/W	Big Endian aes Key

Register: SE_AES_0_KEY_3

Address: 0x2600512C

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_key_3	32'h0	R/W	Big Endian aes Key (aes-128 key LSB)

Register: SE_AES_0_KEY_4

Address: 0x26005130

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_key_4	32'h0	R/W	Big Endian aes Key

Register: SE_AES_0_KEY_5

Address: 0x26005134

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_key_5	32'h0	R/W	Big Endian aes Key

Register: SE_AES_0_KEY_6

Address: 0x26005138

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_key_6	32'h0	R/W	Big Endian aes Key

Register: SE_AES_0_KEY_7

Address: 0x2600513C

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_key_7	32'h0	R/W	Big Endian aes Key (aes-256 key LSB)

Register: SE_AES_0_ENDIAN

Address: 0x26005148

Bits	Bit Name	Default	Type	Comment
31:30	se_aes_0_ctr_len	2'b00	R/W	00:4-byte Counter 01:1-byte Counter 10:2-byte Counter 11:3-byte Counter
29:5	Reserved	N/A	N/A	Reserved
4	se_aes_0_twk_endian	1'b1	R/W	0: Little-Endian 1: Big-Endian, default 1 for XTS

Bits	Bit Name	Default	Type	Comment
3	se_aes_0_iv_endian	1'b1	R/W	0: Little-Endian 1: Big-Endian
2	se_aes_0_key_endian	1'b1	R/W	0: Little-Endian 1: Big-Endian
1	se_aes_0_din_endian	1'b1	R/W	0: Little-Endian 1: Big-Endian
0	se_aes_0_dout_endian	1'b1	R/W	0: Little-Endian 1: Big-Endian

Register: SE_TRNG_0_CTRL_0

Address: 0x26005200

Bits	Bit Name	Default	Type	Comment
31:12	Reserved	N/A	N/A	Reserved
11	se_trng_0_int_mask	1'b0	R/W	se_trng_0 interrupt mask
10	se_trng_0_int_set_1t	1'b0	W1p	1: Set Interrupt
9	se_trng_0_int_clr_1t	1'b0	W1p	1: Clear Interrupt
8	se_trng_0_int	1'b0	R	Interrupt Value
7:5	Reserved	N/A	N/A	Reserved
4	se_trng_0_ht_error	1'b0	R	1: Health Test Error
3	se_trng_0_dout_clr_1t	1'b0	W1p	1: Clear trng_dout to Zero
2	se_trng_0_en	1'b0	R/W	1: Enable
1	se_trng_0_trig_1t	1'b0	W1p	1: Trigger trng Engine
0	se_trng_0_busy	1'b0	R	1: trng Engine Busy

Register: SE_TRNG_0_STATUS

Address: 0x26005204

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_status	32'h100020	R	*"TODO"*

Register: SE_TRNG_0_DOUT0

Address: 0x26005208

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_dout_0	32'h0	R	random value

Register: SE_TRNG_0_DOUT1

Address: 0x2600520C

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_dout_1	32'h0	R	random value

Register: SE_TRNG_0_DOUT2

Address: 0x26005210

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_dout_2	32'h0	R	random value

Register: SE_TRNG_0_DOUT3

Address: 0x26005214

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_dout_3	32'h0	R	random value

Register: SE_TRNG_0_DOUT4

Address: 0x26005218

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_dout_4	32'h0	R	random value

Register: SE_TRNG_0_DOUT5

Address: 0x2600521C

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_dout_5	32'h0	R	random value

Register: SE_TRNG_0_DOUT6

Address: 0x26005220

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_dout_6	32'h0	R	random value

Register: SE_TRNG_0_DOUT7

Address: 0x26005224

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_dout_7	32'h0	R	random value

Register: SE_TRNG_0_TEST

Address: 0x26005228

Bits	Bit Name	Default	Type	Comment
31:2	Reserved	N/A	N/A	Reserved
1	se_trng_0_cp_test_en	1'b0	R/W	1: Enable trng Conditional Component Test Mode
0	se_trng_0_test_en	1'b0	R/W	1: Enable trng Test Mode

Register: SE_TRNG_0_CTRL_1

Address: 0x2600522C

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_reseed_n_lsb	32'hFFFF	R/W	Reload Seed When Number of Used Random Value is Larger than reseed_n

Register: SE_TRNG_0_CTRL_2

Address: 0x26005230

Bits	Bit Name	Default	Type	Comment
31:16	Reserved	N/A	N/A	Reserved
15:0	se_trng_0_reseed_n_msb	16'hFF	R/W	Reload Seed When Number of Used Random Value is Larger than reseed_n

Register: SE_TRNG_0_CTRL_3

Address: 0x26005234

Bits	Bit Name	Default	Type	Comment
31	se_trng_0_rosc_en	1'b1	R/W	trng rosc Enable
30:0	Reserved	N/A	N/A	Reserved

Register: SE_TRNG_0_TEST_OUT_0

Address: 0x26005240

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_test_out_0	32'h0	R	*"TODO"*

Register: SE_TRNG_0_TEST_OUT_1

Address: 0x26005244

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_test_out_1	32'h0	R	*"TODO"*

Register: SE_TRNG_0_TEST_OUT_2

Address: 0x26005248

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_test_out_2	32'h0	R	*"TODO"*

Register: SE_TRNG_0_TEST_OUT_3

Address: 0x2600524C

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_test_out_3	32'h0	R	*"TODO"*

Register: SE_CTRL_RESERVED_0

Address: 0x26005F04

Bits	Bit Name	Default	Type	Comment
31:0	se_ctrl_reserved_0	32'h0	R/W	Reserved

Register: SE_CTRL_RESERVED_1

Address: 0x26005F08

Bits	Bit Name	Default	Type	Comment
31:0	se_ctrl_reserved_1	32'hFFFFFFFF	R/W	Reserved

Register: SE_CTRL_RESERVED_2

Address: 0x26005F0C

Bits	Bit Name	Default	Type	Comment
31:0	se_ctrl_reserved_2	32'h0	R	Reserved Read SE_CTRL_RESERVED_0

7.6 UART Control Register

UART CSR Base address = 0x20004000

Register: UTX_CONFIG

Address: 0x20004000

Bits	Bit Name	Default	Type	Comment
31:16	cr_utx_lentxlen	16'b0	R/W	Length of UART TX data transfer (Unit: character/byte) (Don't-care if cr_utx_frm_en is enabled)Length of TX data (unit: character/byte). Utx_end_int will assert when this length is reached
15:13	cr_utx_bit_cnt_btxbcntp	3'b100	R/W	UART TX BREAK bit count (for LIN protocol) Note: Additional 8 bit times will be added since LIN Break field requires at least 13 bit timesTX BREAK bit count (for LIN protocol) Note: Additional 8 bit times will be added since LIN break field requires at least 13-bit.
12:11	cr_utx_bit_cnt_ptxscntp	2'b01	R/W	UART TX STOP bit count (unit: 0.5 bit)TX stop bit count (unit: 0.5 bit)
10:8	cr_utx_bit_cnt_dtxbcntd	3'b111	R/W	TX data bit count for each characterTX data bit count for each character
7	cr_utx_ir_invirtxinv	1'b00	R/W	Inverse signal of TX output in IR modeInverse signal of TX output in IR mode
6	cr_utx_ir_enirtxen	1'b00	R/W	Enable signal of TX IR modeEnable signal of TX IR mode
5	cr_utx_prt_sel	1'b0	R/W	Select signal of TX parity bit 1: Odd parity 0: Even parity
5	txprsel	0	R/W	Select signal of TX parity bit 1: Odd parity 0: Even parity
4	cr_utx_prt_entxpren	1'b00	R/W	Enable signal of TX parity bitEnable signal of TX parity bit
3	cr_utx_lin_entlinen	1'b00	R/W	Enable signal of TX LIN mode (LIN header will be sent before sending data)Enable signal of TX LIN mode (LIN header will be sent before sending data)
2	cr_utx_frm_enFrmen	1'b00	R/W	Enable signal of TX free run mode (utx_end_int will be disabled)Enable signal of TX free run mode (utx_end_int will be disabled)
1	cr_utx_cts_enCtsen	1'b00	R/W	Enable signal of TX CTS flow control function.Enable signal of TX CTS flow control function.
0	cr_utx_enen	1'b00	R/W	Enable signal of TX functionEnable signal of TX function

Register: URX_CONFIG

Address: 0x20004004

Bits	Bit Name	Default	Type	Comment
31:16	cr_urx_lenrxlen	16'b0	R/W	Length of RX data (unit: character/byte). Urx_end_int will assert when this length is reached
15:12	cr_urx_deg_cntdegcnt	4'b0	R/W	De-glitch function cycle count

Formatted Table

Formatted: Font: 9 pt

Formatted Table

Formatted Table

Formatted: Font: (Default) Arial

Bits	Bit Name	Default	Type	Comment
11	cr_urx_deg_endegen	1'b00	R/W	Enable signal of RXD input de-glitch function
10:8	cr_urx_bit_cnt_drxbcntd	3'3'b111d7	R/W	RX data bit count for each character
7	cr_urx_ir_invirrxinv	1'b00	R/W	Inverse signal of RX output in IR mode
6	cr_urx_ir_enirrxen	1'b00	R/W	Enable signal of RX IR mode
5	cr_urx_prt_selrxprsel	1'b0	R/W	Select signal of RX parity bit 1: Odd parity 0: Even parity
4	cr_urx_prt_enrxpren	1'b00	R/W	Enable signal of RX parity bit
3	cr_urx_lin_enrlinen	1'b00	R/W	Enable signal of RX LIN mode (LIN header will be required and checked before receiving data)
2	Reservedrsvd	N/A0	N/AR/W	Reserved
1	cr_urx_abr_enarben	1'b00	R/W	Enable signal of RX auto baud rate detection function.
0	cr_urx_enen	1'b00	R/W	Enable signal of RX function

Formatted Table

Register: UART_BIT_PRD

Address: 0x20004008

Bits	Bit Name	Default	Type	Comment
31:16	cr_urx_bit_prdrbitprd	16'h00FF	R/W	Period of each UART RX bit, related to baud rate.
15:0	cr_utx_bit_prdtbitprd	16'h00FF	R/W	Period of each UART TX bit, related to baud rate.

Formatted Table

Register: DATA_CONFIG

Address: 0x2000400C

Bits	Bit Name	Default	Type	Comment
31:1	Reserved	N/A	N/A	Reserved
0	cr_uart_bit_invbitinv	1'b0	R/W	Bit inverse signal for each data byte 0: Each byte is sent LSB-first 1: Each byte is sent MSB-first

Formatted Table

Register: UTX_IR_POSITION

Address: 0x20004010

Bits	Bit Name	Default	Type	Comment
31:16	cr_utx_ir_pos_ptxirpp	16'h009F	R/W	Stop position of UART TX IR pulse.
15:0	cr_utx_ir_pos_stxirps	16'h0070	R/W	Start position of UART TX IR pulse.

Formatted Table

Register: URX_IR_POSITION

Address: 0x20004014

Bits	Bit Name	Default	Type	Comment
31:16	ReservedRSVD	N/A	N/A	Reserved
15:0	cr_urx_ir_pos_srxirps	16'd111 16'h6F	R/W	Start position of UART RXD pulse recovered from IR signal.

Formatted Table

Register: URX_RTO_TIMER

Address: 0x20004018

Bits	Bit Name	Default	Type	Comment
31:8	ReservedRSVD	N/A	N/A	ReservedReserved
7:0	cr_urx_rto_valuerxrtova	8'd158'h F	R/W	Time-out value for triggering RTO interrupt (unit: bit time)

Formatted Table

Register: UART_SW_MODE

Address: 0x2000401C

Bits	Bit Name	Default	Type	Comment
31:4	ReservedRSVD	N/A	N/A	Reserved
3	cr_urx_rts_sw_valrrv	1'b00	R/W	UART RX RTS output SW control value
2	cr_urx_rts_sw_moderrm	1'b0	R/W	UART RX RTS output SW control mode
1	cr_utx_txd_sw_valttv	1'b0	R/W	UART TX TXD output SW control value
0	cr_utx_txd_sw_modettm	1'b0	R/W	UART TX TXD output SW control mode

Register: UART_INT_STS

Address: 0x20004020

Bits	Bit Name	Default	Type	Comment
31:129	ReservedRSVD	N/A	N/A	Reserved
811	urx_ad5_intRLSEINT	1'b00	R	UART RX ABR Detection finish interrupt using codeword 0x55UART RX LIN mode sync field error interrupt
10	urx_ads_int	1'b0	R	UART RX ABR Detection finish interrupt using START bit
9	urx_bcr_int	1'b0	R	UART RX byte count reached interrupt
8	urx_lse_int	1'b0	R	UART RX LIN mode sync field error interrupt
7	urx_fer_intRFERINT	1'b00	R	UART RX FIFO error interrupt. Auto cleared when FIFO overflow/underflow error flag is cleared.
6	utx_fer_intTFERINT	1'b00	R	UART TX FIFO error interrupt. Auto cleared when FIFO overflow/underflow error flag is cleared.
5	urx_pce_intRPCEINT	1'b00	R	UART RX parity check error interrupt.
4	urx_rto_intRRTOINT	1'b00	R	UART RX timeout interrupt
3	urx_frdy_intRFIN	1'b0	R	UART RX FIFO ready (rx_fifo_cnt > rx_fifo_th) interrupt, auto cleared when data is popped.
2	utx_frdy_intTFIN	1'b1	R	UART TX FIFO ready (tx_fifo_cnt > tx_fifo_th) interrupt, auto cleared when data is pushed.
1	urx_end_intREIN	1'b0	R	UART RX transfer end interrupt (set according to cr_urx_len)
0	utx_end_intTEIN	1'b0	R	UART TX transfer end interrupt (set according to cr_utx_len)

Register: UART_INT_MASK

Address: 0x20004024

Bits	Bit Name	Default	Type	Comment
31:129	ReservedRSVD	N/A	N/A	Reserved
11	cr_urx_ad5_mask	1'b11	R/W	Interrupt mask of urx_ad5_int
10	cr_urx_ads_mask	1'b11	R/W	Interrupt mask of urx_ads_int
9	cr_urx_bcr_mask	1'b11	R/W	Interrupt mask of urx_bcr_int
8	cr_urx_lse_maskRLSEMASK	1'b1	R/WR/W	Interrupt mask of urx_lse_int

Formatted Table
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial
Formatted Table
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial

Bits	Bit Name	Default	Type	Comment
7	cr_urx_fer_maskRFERMASK	1'b1	R/WR/W	Interrupt mask of urx_fer_int
6	cr_utx_fer_maskTFERMASK	1'b1	R/WR/W	Interrupt mask of utx_fer_int
5	cr_urx_pce_maskRPCEMASK	1'b1	R/WR/W	Interrupt mask of urx_pce_int
4	cr_urx_rto_maskRRTOMASK	1'b1	R/WR/W	Interrupt mask of urx_rto_int
3	cr_urx_frdy_maskRFMS	1'b1	R/WR/W	Interrupt mask of urx_fifo_int
2	cr_utx_frdy_maskTFMS	1'b1	R/WR/W	Interrupt mask of utx_fifo_int
1	cr_urx_end_maskREMS	1'b1	R/WR/W	Interrupt mask of urx_end_int
0	cr_utx_end_maskTEMS	1'b1	R/WR/W	Interrupt mask of utx_end_int

Register: UART_INT_CLR

Address: 0x20004028

Bits	Bit Name	Default	Type	Comment
31:129	ReservedRSVD	N/A	N/A	Reserved
11	cr_urx_ad5_clr	1'b0	W1c	Interrupt clear of urx_ad5_int
10	cr_urx_ads_clr	1'b0	W1c	Interrupt clear of urx_ads_int
9	cr_urx_bcr_clr	1'b0	W1c	Interrupt clear of urx_bcr_int
8	cr_urx_lse_clrRLSECLR	1'b00	W1cW1C	Interrupt clear of urx_lse_intInterrupt clear of urx_lse_int
7:6	ReservedRSVD	N/A	N/A	Reserved
5	cr_urx_pce_clrRPCECLR	1'b00	W1cW1C	Interrupt clear of urx_pce_int
4	cr_urx_rto_clrRRTOCLR	1'b00	W1cW1C	Interrupt clear of urx_rto_int
3:2	ReservedRSVD	N/A	N/A	Reserved
1	cr_urx_end_clrRECL	1'b00	W1cW1C	Interrupt clear of urx_end_int
0	cr_utx_end_clrTECL	1'b00	W1cW1C	Interrupt clear of utx_end_int

Register: UART_INT_EN

Address: 0x2000402C

Bits	Bit Name	Default	Type	Comment
31:129	ReservedRSVD	N/A	N/A	Reserved
11	cr_urx_ad5_en	1'b1	R/W	Interrupt enable of urx_ad5_int
10	cr_urx_ads_en	1'b1	R/W	Interrupt enable of urx_ads_int
9	cr_urx_bcr_en	1'b1	R/W	Interrupt enable of urx_bcr_int
8	cr_urx_lse_enRLSE	1'b1	R/W	Interrupt enable of urx_lse_int

Formatted Table
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial
Formatted Table
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial
Formatted Table
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial, 9 pt
Formatted: Font: (Default) Arial

Bits	Bit Name	Default	Type	Comment
7	cr_urx_fer_en~~RFER~~	1'b1	R/W	Interrupt enable of urx_fer_int
6	cr_utx_fer_en~~TFER~~	1'b1	R/W	Interrupt enable of utx_fer_int
5	cr_urx_pce_en~~RPCE~~	1'b1	R/W	Interrupt enable of urx_pce_int
4	cr_urx_rto_en~~RRTO~~	1'b1	R/W	Interrupt enable of urx_rto_int
3	cr_urx_frdy_en~~RFIF~~	1'b1	R/W	Interrupt enable of urx_fifo_int
2	cr_utx_frdy_en~~TFIF~~	1'b1	R/W	Interrupt enable of utx_fifo_int
1	cr_urx_end_en~~REND~~	1'b1	R/W	Interrupt enable of urx_end_int
0	cr_utx_end_en~~TEND~~	1'b1	R/W	Interrupt enable of utx_end_int

Formatted Table

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial

Register: UART_STATUS

Address: 0x20004030

Bits	Bit Name	Default	Type	Comment
31:2	ReservedRSVD	N/A	N/A	Reserved
1	sts_urx_bus_busyRBB	1'b0	R	Indicator of UART RX bus busyUART RX bus busy
0	sts_utx_bus_busyTBB	1'b0	R	Indicator of UART TX bus busyUART TX bus busy

Register: STS_URX_ABR_PRD URX_RTO_TIMER

Address: 0x20004034

Bits	Bit Name	Default	Type	Comment
31:16	sts_urx_abr_prd_0x55 ABRPRD	16'b0	R	Bit period of Auto Baud Rate detection using codeword 0x55Bit period of auto baud rate detection using codeword 0x55
15:0	sts_urx_abr_prd_start ABRPRDS	16'b0	R	Bit period of Auto Baud Rate detection using START bitBit period of auto baud rate detection using start bit

Register: URX_ABR_PRD_B01

Address: 0x20004038

Bits	Bit Name	Default	Type	Comment
31:16	sts_urx_abr_prd_bit1	16'b0	R	Bit period of Auto Baud Rate detection - bit[1]
15:0	sts_urx_abr_prd_bit0	16'b0	R	Bit period of Auto Baud Rate detection - bit[0]

Register: URX_ABR_PRD_B23

Address: 0x2000403C

Bits	Bit Name	Default	Type	Comment
31:16	sts_urx_abr_prd_bit3	16'b0	R	Bit period of Auto Baud Rate detection - bit[3]
15:0	sts_urx_abr_prd_bit2	16'b0	R	Bit period of Auto Baud Rate detection - bit[2]

Register: URX_ABR_PRD_B45

Address: 0x20004040

Bits	Bit Name	Default	Type	Comment
31:16	sts_urx_abr_prd_bit5	16'b0	R	Bit period of Auto Baud Rate detection - bit[5]
15:0	sts_urx_abr_prd_bit4	16'b0	R	Bit period of Auto Baud Rate detection - bit[4]

Register: URX_ABR_PRD_B67

Address: 0x20004044

Formatted Table

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial, 9 pt

Formatted Table

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial, 9 pt

Bits	Bit Name	Default	Type	Comment
31:16	sts_urx_abr_prd_bit7	16'b0	R	Bit period of Auto Baud Rate detection - bit[7]
15:0	sts_urx_abr_prd_bit6	16'b0	R	Bit period of Auto Baud Rate detection - bit[6]

Register: URX_ABR_PW_TOL

Address: 0x20004048

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	N/A	N/A	Reserved
7:0	cr_urx_abr_pw_tol	8'h3	R/W	Auto Baud Rate detection pulse-width tolerance for using codeword 0x55

Register: URX_BCR_INT_CFG

Address: 0x20004050

Bits	Bit Name	Default	Type	Comment
31:16	sts_urx_bcr_count	16'b0	R	Current byte count of urx_bcr_int counter, auto-cleared bt cr_urx_bcr_clr
15:0	cr_urx_bcr_value	16'hFFFF	R/W	Byte count setting for urx_bcr_int counter

Register: UTX_RS485_CFG

Address: 0x20004054

Bits	Bit Name	Default	Type	Comment
31:2	Reserved	N/A	N/A	Reserved
1	cr_utx_rs485_pol	1'b1	R/W	
0	cr_utx_rs485_en	1'b0	R/W	

Register: UART_FIFO_CONFIG_0

Address: 0x20004080

Bits	Bit Name	Default	Type	Comment
31:83 1:8	ReservedRSVD	N/AN/A	N/AN/A	ReservedReserved
77	rx_fifo_underflowRFIU	1'b00	RR	Underflow flag of RX FIFO, can be cleared by rx_fifo_clrUnderflow flag of RX FIFO, cleared by rx_fifo_clr
66	rx_fifo_overflowRFIO	1'b00	RR	Overflow flag of RX FIFO, can be cleared by rx_fifo_clrOverflow flag of RX FIFO, cleared by rx_fifo_clr
55	tx_fifo_underflowTFIU	1'b00	RR	Underflow flag of TX FIFO, can be cleared by tx_fifo_clrUnderflow flag of TX FIFO, cleared by tx_fifo_clr

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial, 9 pt

Formatted Table

Bits	Bit Name	Default	Type	Comment
44	tx_fifo_overflowTFIO	1'b00	RR	Overflow flag of TX FIFO, can be cleared by tx_fifo_clrOverflow flag of TX FIFO, cleared by tx_fifo_clr
33	rx_fifo_clrRFICLR	1'b00	W1c W1C	Clear signal of RX FIFOClear signal of RX FIFO
22	tx_fifo_clrTFICLR	1'b00	W1c W1C	Clear signal of TX FIFOClear signal of TX FIFO
11	uart_dma_rx_enUDRE N	1'b00	R/WR /W	Enable signal of dma_rx_req/ack interfaceEnable signal of dma_rx_req/ack interface
00	uart_dma_tx_enUDTE N	1'b00	R/WR /W	Enable signal of dma_tx_req/ack interfaceEnable signal of dma_tx_req/ack interface

Formatted Table

Register: UART_FIFO_CONFIG_1

Address: 0x20004084

Bits	Bit Name	Default	Type	Comment
31	ReservedRSVD	N/A	N/A	Reserved
30:24	rx_fifo_thRFITH	7'b0	R/W	RX FIFO threshold, dma_rx_req will not be asserted if rx_fifo_cnt is less than this valueRX FIFO threshold, dma_rx_req will not assert if rx_fifo_cnt is less than this value
23	ReservedRSVD	N/A	N/A	Reserved
22:16	tx_fifo_thTFITH	7'b0	R/W	TX FIFO threshold, dma_tx_req will not be asserted if tx_fifo_cnt is less than this valueTX FIFO threshold, dma_tx_req will not assert if tx_fifo_cnt is less than this value
15:8	rx_fifo_cntRFICNT	8'b0	RAW	RX FIFO available countRX FIFO available count
7:0	tx_fifo_cntTFICNT	8'h80	RAW	TX FIFO available countTX FIFO available count

Register: UART_FIFO_WDATA

Address: 0x20004088

Bits	Bit Name	Default	Type	Comment
31:8	ReservedRSVD	N/A	N/A	Reserved
7:0	uart_fifo_wdataUFIWD	8'b0	W	UART FIFO write data

Register: UART_FIFO_RDATA

Address: 0x2000408C

Bits	Bit Name	Default	Type	Comment
31:8	ReservedRSVD	N/A	N/A	Reserved
7:0	uart_fifo_rdataUFIRD	8'b0	R	UART FIFO read data

Formatted Table

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: (Default) Arial, 9 pt

Formatted Table

Formatted: Font: (Default) Arial, 9 pt

Formatted Table

7.7 SPI Control Register

SPI0 CSR Base address = 0x20000000

SPI1 CSR Base address = 0x20001000

Register: SPI_CONFIG

Address: 0x2000[1:0]000

Bits	Bit Name	Default	Type	Comment
31:16	31:18	Reserved	N/A	N/A
15:12	cr_spi_deg_cnt	4'b0	R/W	De-glitch function cycle count.
11	cr_spi_deg_en	1'b0	R/W	Enable signal of de-glitch function.
10	cr_spi_s_3pin_mode	1'b0	R/W	SPI slave 3-pin mode 0: 4-pin mode (SS_n is enabled) 1: 3-pin mode (SS_n is disabled / don't care)
9	cr_spi_m_cont_en	1'b0	R/W	Enable signal of master continuous transfer mode 0: Disabled. SS_n will de-assert between each data frame. 1: Enabled. SS_n will stay asserted between each consecutive data frame if next data is valid in FIFO.
8	cr_spi_rxd_ignr_en	1'b0	R/W	Enable signal of RX data ignore function.
7	cr_spi_byte_inv	1'b0	R/W	Byte inverse signal for each FIFO entry. 0: Byte [0] is sent first. 1: Byte [3] is sent first.
6	cr_spi_bit_inv	1'b0	R/W	Bit inverse signal for each FIFO entry. 0: Each byte is sent MSB first. 1: Each byte [3] is sent LSB first.
5	cr_spi_sclk_ph	1'b0	R/W	SCLK clock phase inverse signal
4	cr_spi_sclk_pol	1'b0	R/W	SCLK polarity 0: SCLK output LOW at IDLE state 1: SCLK output HIGH at IDLE state
3:2	cr_spi_frame_size	2'b00	R/W	SPI frame size (also the valid width for each FIFO entry) 00: 8-bit 01: 16-bit 10: 24-bit 11: 32-bit
1	cr_spi_s_en	1'b0	R/W	Enable signal of SPI slave function. Master and slave should not be both enabled at same time. This bit becomes don't-care if "cr_spi_m_en" is enabled
0	cr_spi_m_en	1'b0	R/W	Enable signal of SPI Maser function. Asserting this bit will trigger the transaction and should be de-asserted after finish.

Register: SPI_INT_STS

Address: 0x2000[1:0]004

Bits	Bit Name	Default	Type	Comment
31:30	Reserved	N/A	N/A	Reserved
29	cr_spi_fer_en	1'b1	R/W	Interrupt enable of spi_fer_int
28	cr_spi_txu_en	1'b1	R/W	Interrupt enable of spi_txu_int
27	cr_spi_sto_en	1'b1	R/W	Interrupt enable of spi_sto_int
26	cr_spi_rxf_en	1'b1	R/W	Interrupt enable of spi_rxv_int
25	cr_spi_txf_en	1'b1	R/W	Interrupt enable of spi_txe_int
24	cr_spi_end_en	1'b1	R/W	Interrupt enable of spi_end_int
23:21	Reserved	N/A	N/A	Reserved
20	cr_spi_txu_clr	1'b0	W1c	Interrupt clear of spi_txu_int
19	cr_spi_sto_clr	1'b0	W1c	Interrupt clear of spi_sto_int
18:17	Reserved	N/A	N/A	Reserved
16	cr_spi_end_clr	1'b0	W1c	Interrupt clear of spi_end_int
15:14	Reserved	N/A	N/A	Reserved
13	cr_spi_fer_mask	1'b1	R/W	Interrupt mask of spi_fer_int
12	cr_spi_txu_mask	1'b1	R/W	Interrupt mask of spi_txu_int
11	cr_spi_sto_mask	1'b1	R/W	Interrupt mask of spi_sto_int
10	cr_spi_rxf_mask	1'b1	R/W	Interrupt mask of spi_rxv_int
9	cr_spi_txf_mask	1'b1	R/W	Interrupt mask of spi_txe_int
8	cr_spi_end_mask	1'b1	R/W	Interrupt mask of spi_end_int
7:6	Reserved	N/A	N/A	Reserved
5	spi_fer_int	1'b0	R	SPI TX/RX FIFO error interrupt. Auto cleared when FIFO overflow/underflow error flag is cleared.
4	spi_txu_int	1'b0	R	SPI slave mode TX underrun error flag. Triggered when TXD is not ready during transfer in slave mode.
3	spi_sto_int	1'b0	R	SPI slave mode transfer time-out interrupt. Triggered when SPI bus is idle for a given value
2	spi_rxf_int	1'b0	R	SPI RX FIFO ready (rx_fifo_cnt>rx_fifo_th) interrupt, auto cleared when data is popped.
1	spi_txf_int	1'b1	R	SPI TX FIFO ready (tx_fifo_cnt>tx_fifo_th) interrupt, auto cleared when data is pushed.
0	spi_end_int	1'b0	R	SPI transfer end interrupt. Shared by both master and slave mode.

Register: SPI_BUS_BUSY

Address: 0x2000[1:0]008

Bits	Bit Name	Default	Type	Comment
31:1	Reserved	N/A	N/A	Reserved
0	sts_spi_bus_busy	1'b0	R	Indicator of SPI bus busy

Register: SPI_PRD_0

Address: 0x2000[1:0]010

Bits	Bit Name	Default	Type	Comment
31:24	cr_spi_prd_d_ph_1	8'h0F	R/W	Length of DATA phase 1 (unit: SPI source clock period)
23:16	cr_spi_prd_d_ph_0	8'h0F	R/W	Length of DATA phase 0 (unit: SPI source clock period))
15:8	cr_spi_prd_p	8'h0F	R/W	Length of STOP condition (unit: SPI source clock period)
7:0	cr_spi_prd_s	8'h0F	R/W	Length of START condition (unit: SPI source clock period)

Register: SPI_PRD_1

Address: 0x2000[1:0]014

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	N/A	N/A	Reserved
7:0	cr_spi_prd_i	8'h0F	R/W	Length of INTERVAL between frame (unit: SPI source clock period)

Register: SPI_RXD_IGNR

Address: 0x2000[1:0]018

Bits	Bit Name	Default	Type	Comment
31:21	Reserved	N/A	N/A	Reserved
20:16	cr_spi_rxd_ignr_s	5'b0	R/W	Starting point of RX data ignore function (unit: bit)
15:5	Reserved	N/A	N/A	Reserved
4:0	cr_spi_rxd_ignr_p	5'b0	R/W	Stopping point of RX data ignore function (unit: bit)

Register: SPI_STO_VALUE

Address: 0x2000[1:0]01C

Bits	Bit Name	Default	Type	Comment
31:12	Reserved	N/A	N/A	Reserved
11:0	cr_spi_sto_value	12'hFFF	R/W	Time-out value for spi_sto_int triggering

Register: SPI_FIFO_CONFIG_0

Address: 0x2000[1:0]080

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	N/A	N/A	Reserved
7	rx_fifo_underflow	1'b0	R	Underflow flag of RX FIFO, can be cleared by rx_fifo_clr
6	rx_fifo_overflow	1'b0	R	Overflow flag of RX FIFO, can be cleared by rx_fifo_clr
5	tx_fifo_underflow	1'b0	R	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	tx_fifo_overflow	1'b0	R	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
3	rx_fifo_clr	1'b0	W1c	Clear signal of RX FIFO
2	tx_fifo_clr	1'b0	W1c	Clear signal of TX FIFO
1	spi_dma_rx_en	1'b0	R/W	Enable signal of dma_rx_req/ack interface
0	spi_dma_tx_en	1'b0	R/W	Enable signal of dma_tx_req/ack interface

Register: SPI_FIFO_CONFIG_1

Address: 0x2000[1:0]084

Bits	Bit Name	Default	Type	Comment
31:26	Reserved	N/A	N/A	Reserved
25:24	rx_fifo_th	2'b00	R/W	RX FIFO threshold, dma_rx_req will not be asserted if rx_fifo_cnt is less than this value
23:18	Reserved	N/A	N/A	Reserved
17:16	tx_fifo_th	2'b00	R/W	TX FIFO threshold, dma_tx_req will not be asserted if tx_fifo_cnt is less than this value
15:11	Reserved	N/A	N/A	Reserved
10:8	rx_fifo_cnt	3'b000	R	RX FIFO available count, means byte count of data received in RX FIFO (unit: byte)
7:3	Reserved	N/A	N/A	Reserved
2:0	tx_fifo_cnt	3'b100	R	TX FIFO available count, means empty space remained in TX FIFO (unit: byte)

Register: SPI_FIFO_WDATA

Address: 0x2000[1:0]088

Bits	Bit Name	Default	Type	Comment
31:0	spi_fifo_wdata	32'h0	W	SPI FIFO write data port

Register: SPI_FIFO_RDATA

Address: 0x2000[1:0]08C

Bits	Bit Name	Default	Type	Comment
31:0	spi_fifo_rdata	32'h0	R	SPI FIFO read data port

Register: BACKUP_IO_EN

Address: 0x2000[1:0]0FC

Bits	Bit Name	Default	Type	Comment
31:1	Reserved	N/A	N/A	Reserved
0	backup_io_en	1'b0	R/W	Enable IO backup function

7.8 I2C Control Register

I2C CSR Base address = 0x20002000

Bits	Bit Name	Default	Type	Comment
31:28	cr_i2c_deg_cnt	4'b0	R/W	De-glitch function cycle count.
27:20	cr_i2c_pkt_len	8'h0	R/W	Packet length (unit: byte)
19:18	Reserved	N/A	N/A	Reserved
17:8	cr_i2c_slv_addr	10'h0	R/W	Slave address for I²C transaction (target address)
7	cr_i2c_10b_addr_en	1'b0	R/W	10 bit address enable
6:5	cr_i2c_sub_addr_bc	2'b00	R/W	Slave address field byte count 00: 1 byte 01: 2 byte 10: 3 byte 11: 4 byte
4	cr_i2c_sub_addr_en	1'b0	R/W	Enable signal of I²C sub-address field
3	cr_i2c_scl_sync_en	1'b1	R/W	Enable signal of I²C SCL synchronization. Should be enabled to support Multi-Master and Clock-Stretching (normally should be turned off)
2	cr_i2c_deg_en	1'b0	R/W	Enable signal of I²C de-glitch (for all input pins)
1	cr_i2c_pkt_dir	1'b1	R/W	Transfer direction of the packet. 0: write 1: read.
0	cr_i2c_m_en	1'b0	R/W	Enable signal of I²C Maser function. Asserting this bit will trigger the transaction and should be de-asserted after finish.

Register: I2C_INT_STS

Address: 0x20002004

Bits	Bit Name	Default	Type	Comment
31:30	Reserved	N/A	N/A	Reserved
29	cr_i2c_fer_en	1'b1	R/W	Interrupt enable of i2c_fer_int
28	cr_i2c_arb_en	1'b1	R/W	Interrupt enable of i2c_arb_int
27	cr_i2c_nak_en	1'b1	R/W	Interrupt enable of i2c_nak_int
26	cr_i2c_rxf_en	1'b1	R/W	Interrupt enable of i2c_rxf_int
25	cr_i2c_txf_en	1'b1	R/W	Interrupt enable of i2c_txf_int
24	cr_i2c_end_en	1'b1	R/W	Interrupt enable of i2c_end_int
23:22	Reserved	N/A	N/A	Reserved
21	rsvd_I2C_INT_STS_21	1'b0	R/W	
20	cr_i2c_arb_clr	1'b0	W1c	Interrupt clear of i2c_arb_int
19	cr_i2c_nak_clr	1'b0	W1c	Interrupt clear of i2c_nak_int
18	rsvd_I2C_INT_STS_18	1'b0	R	

Bits	Bit Name	Default	Type	Comment
17	rsvd_I2C_INT_STS_17	1'b0	R	
16	cr_i2c_end_clr	1'b0	W1c	Interrupt clear of i2c_end_int
15:14	Reserved	N/A	N/A	Reserved
13	cr_i2c_fer_mask	1'b1	R/W	Interrupt mask of i2c_fer_int
12	cr_i2c_arb_mask	1'b1	R/W	Interrupt mask of i2c_arb_int
11	cr_i2c_nak_mask	1'b1	R/W	Interrupt mask of i2c_nak_int
10	cr_i2c_rxf_mask	1'b1	R/W	Interrupt mask of i2c_rxf_int
9	cr_i2c_txf_mask	1'b1	R/W	Interrupt mask of i2c_txf_int
8	cr_i2c_end_mask	1'b1	R/W	Interrupt mask of i2c_end_int
7:6	Reserved	N/A	N/A	Reserved
5	i2c_fer_int	1'b0	R	I^2C TX/RX FIFO error interrupt. Auto-cleared when FIFO overflow/underflow error flag is cleared.
4	i2c_arb_int	1'b0	R	I^2C arbitration lost interrupt
3	i2c_nak_int	1'b0	R	I^2C NAK received interrupt
2	i2c_rxf_int	1'b0	R	I2C RX FIFO ready (rx_fifo_cnt > rx_fifo_th) interrupt, auto cleared when data is popped
1	i2c_txf_int	1'b1	R	I2C TX FIFO ready (tx_fifo_cnt > tx_fifo_th) interrupt, auto cleared when data is pushed
0	i2c_end_int	1'b0	R	I^2C transfer end interrupt.

Register: I2C_SUB_ADDR

Address: 0x20002008

Bits	Bit Name	Default	Type	Comment
31:24	cr_i2c_sub_addr_b3	8'h0	R/W	Sub address field byte 3
23:16	cr_i2c_sub_addr_b2	8'h0	R/W	Sub address field byte 2
15:8	cr_i2c_sub_addr_b1	8'h0	R/W	Sub address field byte 1
7:0	cr_i2c_sub_addr_b0	8'h0	R/W	Sub address field byte 0

Register: I2C_BUS_BUSY

Address: 0x2000200C

Bits	Bit Name	Default	Type	Comment
31:2	Reserved	N/A	N/A	Reserved
1	cr_i2c_bus_busy_clr	1'b0	W1c	Clear signal of bus_busy status, not for normal usage (in case I^2C bus hangs.)
0	sts_i2c_bus_busy	1'b0	R	I^2C bus busy

Register: I2C_PRD_START

Address: 0x20002010

Bits	Bit Name	Default	Type	Comment
31:24	cr_i2c_prd_s_ph_3	8'h0F	R/W	Length of START condition phase 3
23:16	cr_i2c_prd_s_ph_2	8'h0F	R/W	Length of START condition phase 2
15:8	cr_i2c_prd_s_ph_1	8'h0F	R/W	Length of START condition phase 1
7:0	cr_i2c_prd_s_ph_0	8'h0F	R/W	Length of START condition phase 0

Register: I2C_PRD_STOP

Address: 0x20002014

Bits	Bit Name	Default	Type	Comment
31:24	cr_i2c_prd_p_ph_3	8'h0F	R/W	Length of STOP condition phase 3
23:16	cr_i2c_prd_p_ph_2	8'h0F	R/W	Length of STOP condition phase 2
15:8	cr_i2c_prd_p_ph_1	8'h0F	R/W	Length of STOP condition phase 1
7:0	cr_i2c_prd_p_ph_0	8'h0F	R/W	Length of STOP condition phase 0

Register: I2C_PRD_DATA

Address: 0x20002018

Bits	Bit Name	Default	Type	Comment
31:24	cr_i2c_prd_d_ph_3	8'h0F	R/W	Length of DATA phase 3
23:16	cr_i2c_prd_d_ph_2	8'h0F	R/W	Length of DATA phase 2
15:8	cr_i2c_prd_d_ph_1	8'h0F	R/W	Length of DATA phase 1
7:0	cr_i2c_prd_d_ph_0	8'h0F	R/W	Length of DATA phase 0

Register: I2C_FIFO_CONFIG_0

Address: 0x20002080

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	N/A	N/A	Reserved
7	rx_fifo_underflow	1'b0	R	Underflow flag of RX FIFO, cleared by rx_fifo_clr
6	rx_fifo_overflow	1'b0	R	Overflow flag of RX FIFO, cleared by rx_fifo_clr
5	tx_fifo_underflow	1'b0	R	Underflow flag of TX FIFO, cleared by tx_fifo_clr
4	tx_fifo_overflow	1'b0	R	Overflow flag of TX FIFO, cleared by tx_fifo_clr
3	rx_fifo_clr	1'b0	W1c	Clear signal of RX FIFO
2	tx_fifo_clr	1'b0	W1c	Clear signal of TX FIFO

Bits	Bit Name	Default	Type	Comment
1	i2c_dma_rx_en	1'b0	R/W	Enable signal of dma_rx_req/ack interface
0	i2c_dma_tx_en	1'b0	R/W	Enable signal of dma_tx_req/ack interface

Register: I2C_FIFO_CONFIG_1

Address: 0x20002084

Bits	Bit Name	Default	Type	Comment
31:25	Reserved	N/A	N/A	Reserved
24	rx_fifo_th	1'b0	R/W	RX FIFO threshold, dma_rx_req will not assert if rx_fifo_cnt is less than this value
23:17	Reserved	N/A	N/A	Reserved
16	tx_fifo_th	1'b0	R/W	TX FIFO threshold, dma_tx_req will not assert if tx_fifo_cnt is less than this value
15:10	Reserved	N/A	N/A	Reserved
9:8	rx_fifo_cnt	2'b00	R	RX FIFO available count
7:2	Reserved	N/A	N/A	Reserved
1:0	tx_fifo_cnt	2'b10	R	TX FIFO available count

Register: I2C_FIFO_WDATA

Address: 0x20002088

Bits	Bit Name	Default	Type	Comment
31:0	i2c_fifo_wdata	32'h0	W	I^2C FIFO write data

Register: I2C_FIFO_RDATA

Address: 0x2000208C

Bits	Bit Name	Default	Type	Comment
31:0	i2c_fifo_rdata	32'h0	R	I^2C FIFO read data

7.9 Local Timer Register

Local timer CSR Base address = 0x20003000

Register: TCCR

Address: 0x20003000

Bits	Bit Name	Default	Type	Comment
31:10	Reserved	N/A	N/A	Reserved
9:8	cs_wdt	2'b00	R/W	Clock source for timer WDT 00: fclk 01: f32_clk 10: 1 KHz 11: PLL 32 MHz
7	Reserved	N/A	N/A	Reserved
6:5	cs_2	2'b00	R/W	Clock source for timer #3 00: fclk 01: f32_clk 10: 1 KHz 11: PLL 32 MHz
4	Reserved	N/A	N/A	Reserved
3:2	cs_1	2'b00	R/W	Clock source for timer #2 00: fclk 01: f32_clk 10: 1 KHz 11: PLL 32 MHz
1:0	Reserved	N/A	N/A	Reserved

Register: TMR2_0

Address: 0x20003010

Bits	Bit Name	Default	Type	Comment
31:0	tmr_2_0	32'hFFFFFFFF	R/W	Timer2 match register 0

Register: TMR2_1

Address: 0x20003014

Bits	Bit Name	Default	Type	Comment
31:0	tmr_2_1	32'hFFFFFFFF	R/W	Timer2 match register 1

Register: TMR2_2

Address: 0x20003018

Bits	Bit Name	Default	Type	Comment
31:0	tmr_2_2	32'hFFFFFFFF	R/W	Timer2 match register 2

Register: TMR3_0

Address: 0x2000301C

Bits	Bit Name	Default	Type	Comment
31:0	tmr_3_0	32'hFFFFFFFF	R/W	Timer3 match register 0

Register: TMR3_1

Address: 0x20003020

Bits	Bit Name	Default	Type	Comment
31:0	tmr_3_1	32'hFFFFFFFF	R/W	Timer3 match register 1

Register: TMR3_2

Address: 0x20003024

Bits	Bit Name	Default	Type	Comment
31:0	tmr_3_2	32'hFFFFFFFF	R/W	Timer3 match register 2

Register: TCR2

Address: 0x2000302C

Bits	Bit Name	Default	Type	Comment
31:0	tcr2_counter	32'h0	R	Timer2 counter register

Register: TCR3

Address: 0x20003030

Bits	Bit Name	Default	Type	Comment
31:0	tcr3_counter	32'h0	R	Timer3 counter register

Register: TMSR2

Address: 0x20003038

Bits	Bit Name	Default	Type	Comment
31:3	Reserved	N/A	N/A	Reserved
2	tmsr2_2	1'b0	R	Timer2 match register 2 status. Clear interrupt would also clear this bit.

Bits	Bit Name	Default	Type	Comment
1	tmsr2_1	1'b0	R	Timer2 match register 1 status. Clear interrupt would also clear this bit.
0	tmsr2_0	1'b0	R	Timer2 match register 0 status. Clear interrupt would also clear this bit.

Register: TMSR3

Address: 0x2000303C

Bits	Bit Name	Default	Type	Comment
31:3	Reserved	N/A	N/A	Reserved
2	tmsr3_2	1'b0	R	Timer3 match register 2 status. Clear interrupt would also clear this bit.
1	tmsr3_2	1'b0	R	Timer3 match register 1 status. Clear interrupt would also clear this bit.
0	tmsr3_2	1'b0	R	Timer3 match register 0 status. Clear interrupt would also clear this bit.

7.10 DMA Control Register

DMA CSR Base address = 0x26004000

Register: DMA_INTSTATUS

Address: 0x26004000

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	N/A	N/A	Reserved
7:0	INTSTA	8'b0	R	Status of DMA interrupts after masking

Register: DMA_INTTCSTATUS

Address: 0x26004004

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	N/A	N/A	Reserved
7:0	INTTCSTA	8'b0	R	Interrupt terminal count request status

Register: DMA_INTTCCLEAR

Address: 0x26004008

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	N/A	N/A	Reserved
7:0	TCRC	8'b0	W	Terminal count request clear

Register: DMA_INTERRORSTATUS

Address: 0x2600400C

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	N/A	N/A	Reserved
7:0	IES	8'b0	R	Interrupt error status

Register: DMA_INTERRCLR

Address: 0x26004010

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	N/A	N/A	Reserved
7:0	IEC	8'b0	W	Interrupt error clear

Register: DMA_RAWINTTCSTATUS

Address: 0x26004014

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	N/A	N/A	Reserved
7:0	SOTCIPTM	8'b0	R	Status of terminal counter interrupt prior to masking

Register: DMA_RAWINTERRORSTATUS

Address: 0x26004018

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	N/A	N/A	Reserved
7:0	SOTEIPTM	8'b0	R	Status of error interrupts prior to masking

Register: DMA_ENBLDCHNS

Address: 0x2600401C

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	N/A	N/A	Reserved
7:0	CES	8'b0	R	Channel enable status

Register: DMA_SOFTBREQ

Address: 0x26004020

Bits	Bit Name	Default	Type	Comment
31:0	SBR	32'h0	R/W	Software burst request

Register: DMA_SOFTSREQ

Address: 0x26004024

Bits	Bit Name	Default	Type	Comment
31:0	SSR	32'h0	R/W	Software single request

Register: DMA_SOFTLBREQ

Address: 0x26004028

Bits	Bit Name	Default	Type	Comment
31:0	SLBR	32'h0	R/W	Software last burst request

Register: DMA_SOFTLSREQ

Address: 0x2600402C

Bits	Bit Name	Default	Type	Comment
31:0	SLSR	32'h0	R/W	Software last single request

Register: DMA_CONFIG

Address: 0x26004030

Bits	Bit Name	Default	Type	Comment
31:2	Reserved	N/A	N/A	Reserved
1	AHBMEC	1'b0	R/W	AHB Master endianness configuration: 0=little-endian, 1=big-endian.
0	SMDMAEN	1'b0	R/W	SMDMA enable

Register: DMA_SYNC

Address: 0x26004034

Bits	Bit Name	Default	Type	Comment
31:0	DSLFDRS	32'h0	R/W	DMA synchronization logic for DMA request signals. 0=enable, 1=disable.

Register: DMA_C [3:0] SRCADDR

Address: 0x26004[4:1]00

Bits	Bit Name	Default	Type	Comment
31:0	DMASA	32'h0	R/W	DMA source address

Register: DMA_C [3:0] DSTADDR

Address: 0x26004 [4:1]04

Bits	Bit Name	Default	Type	Comment
31:0	DMADA	32'h0	R/W	DMA destination address

Register: DMA_C [3:0] LLI

Address: 0x26004[4:1]08

Bits	Bit Name	Default	Type	Comment
31:0	FLLI	32'h0	R/W	First linked list item. Bits [1:0] must be 0

Register: DMA_C [3:0] CONTROL

Address: 0x26004[4:1]0C

Bits	Bit Name	Default	Type	Comment
31	TCIEN	1'b0	R/W	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	PROTECT	3'b000	R/W	Protection
27	DI	1'b1	R/W	Destination increment. When set the destination address is incremented after each transfer
26	SI	1'b1	R/W	Source Increment. When set the source address is incremented after each transfer
25	Reserved	N/A	N/A	Reserved
24:23	FIXCNT	2'b00	R/W	Only effect when dst_min_mode = 1 Destination transfer cnt = (total src byte cnt - (fix_- cnt«DWidth))«DWidth
22:21	DTW	2'b10	R/W	Destination transfer width: 8/16/32
20	Reserved	N/A	N/A	Reserved
19:18	STW	2'b10	R/W	Source transfer width: 8/16/32
17	ADDMODE	1'b0	R/W	Add mode: issue remain destination traffic
16:15	DBS	2'b01	R/W	Destination burst size: 1/4/8/16
14	MINMODE	1'b0	R/W	Minus mode. Not issue all destination traffic.
13:12	SBS	2'b01	R/W	Source burst size: 1/4/8/16. Note CH FIFO Size is 16Bytes and SBSize*Swidth should <= 16B
11:0	TS	12'h0	R/W	Transfer size: 0:4095: Number of data transfer left to complete when SMDMA is the flow controller.

Register: DMA_C [3:0]CONFIG

Address: 0x26004[4:1]10

Bits	Bit Name	Default	Type	Comment
31:30	Reserved	N/A	N/A	Reserved
29:20	LLICOUNT	10'b0	R	LLI counter: Increased 1 each LLI run. Cleared 0 when config control.
19	Reserved	N/A	N/A	Reserved
18	HALT	1'b0	R/W	Halt. 0=enable DMA requests, 1=ignore subsequent source DMA requests.
17	ACTIVE	1'b0	R	Active. 0=no data in FIFO of the channel, 1=FIFO of the channel has data.
16	LOCK	1'b0	R/W	Lock
15	TCIM	1'b0	R/W	Terminal count interrupt mask

Bits	Bit Name	Default	Type	Comment
14	IEM	1'b0	R/W	Interrupt error mask
13:11	FLOWCTRL	3'b000	R/W	000: Memory to memory DMA 001: Memory to peripheral DMA 010: Peripheral to memory DMA 011: Source peripheral to destination peripheral DMA.
10:6	DSTPH	5'h0	R/W	Destination peripherals: 0: UART TX 1: UART RX 2: SPI0 TX 3: SPI0 RX 4: SPI1 TX 5: SPI1 RX 6: I2C TX 7: I2C RX Others: Reserved
5:1	SRCPH	5'h0	R/W	Source peripherals: 0: UART TX 1: UART RX 2: SPI0 TX 3: SPI0 RX 4: SPI1 TX 5: SPI1 RX 6: I2C TX 7: I2C RX Others: Reserved
0	CHEN	1'b0	R/W	Channel enable.

EMAC REGISTER (base address: 0x2b000000)

Name	Description
MODE	EMAC configuration
INT_SOURCE	EMAC transmit control
INT_MASK	EMAC interrupt mask
IPGT	Inter packet gap
PACKETLEN	Frame length
COLLCONFIG	Collision configuration
TX_BD_NUM	TX buffer descriptors number
MIIMODE	Management Data configuration
MIICOMMAND	Trigger command
MIIADDRESS	Register address
MIITX_DATA	Control data to be written to PHY
MIIRX_DATA	Received data from PHY
MIISTATUS	MIIM I/F status
MAC_ADDR0	Ethernet MAC address0
MAC_ADDR1	Ethernet MAC address1
HASH0_ADDR	Lower 32-bit of HASH register
HASH1_ADDR	Upper 32-bit of HASH register
TXCTRL	TX control

MODE

Address: 0x0

Bit	Name	Type	Reset	Description
31:18	RSVD			
17	RMII_EN	r/w	1'b0	RMII mode enable 0: MII PHY I/F is used 1: RMII PHY I/F is used
16	RECSMALL	r/w	1'b0	Receive small frame enable 0: Frames smaller than MINFL are ignored. 1: Frames smaller than MINFL are accepted.
15	PAD	r/w	1'b1	Padding enable 0: Do not add pads to frames shorter than MINFL. 1: Add pads to short frames, until the length equals MINFL.
14	HUGEN	r/w	1'b0	Huge frames enable 0: The maximum frame length is MAXFL. All additional bytes are dropped. 1: Frame size is not limited by MAXFL and can be up to 64K bytes.
13	CRCEN	r/w	1'b1	CRC Enable 0: TX MAC does not append CRC field. 1: TX MAC will append CRC field to every frame.
12:11	RSVD			
10	FULLD	r/w	1'b0	Full duplex 0: Half duplex mode. 1: Full duplex mode.
9:7	RSVD			
6	IFG	r/w	1'b0	Inter frame gap check 0: IFG is verified before each frame be received. 1: All frames are received regardless to IFG requirement.
5	PRO	r/w	1'b0	Promiscuous mode enable 0: The destination address is checked before receiving. 1: All frames received regardless of the address.
4	RSVD			
3	BRO	r/w	1'b1	Broadcast address enable 0: Reject all frames containing the broadcast address unless the PRO bit is asserted. 1: Receive all frames containing broadcast address.
2	NOPRE	r/w	1'b0	No preamble mode 0: 7-byte preamble will be sent. 1: No preamble will be sent.
1	TXEN	r/w	1'b0	Transmit enable 0: Transmitter is disabled. 1: Transmitter is enabled. If TX_BD_NUM equals 0x0 (zero buffer descriptors are used), then the transmitter is disabled regardless of TXEN.
0	RXEN	r/w	1'b0	Receiver enable

Bit	Name	Type	Reset	Description
				0: Receiver is disabled. 1: Receiver is enabled. If TX_BD_NUM equals 0x80 (all buffer descriptors are used for TX), then the receiver is disabled regardless of RXEN.

INT_SOURCE

Address: 0x4

Bit	Name	Type	Reset	Description
31:7	RSVD			
6	RXC	r/w	1'b0	Receive control frame This bit indicates that the control frame was received. It is cleared by writing 1 to it. Bit RXFLOW in the CTRLMODE register must be set to 1 in order to get the RXC bit set.
5	TXC	r/w	1'b0	Transmit control frame This bit indicates that a control frame was transmitted. It is cleared by writing 1 to it. Bit TXFLOW in the CTRLMODE register must be set to 1 in order to get the TXC bit set.
4	BUSY	r/w	1'b0	Busy This bit indicates that RX packet is being received and there is no empty buffer descriptor to use. It iscleared by writing 1 to it. This bit appears regardless to the IRQ bits in the Receive Buffer Descriptor.
3	RXE	r/w	1'b0	Receive error This bit indicates that an error occurred while receiving data (overrun, receiver error, dribble nibble, too long, >64K, CRC error, bus error or late collision. It is cleared by writing 1 to it. This bit appears only when IRQ bit is set in the Receive Buffer Descriptor.
2	RXB	r/w	1'b0	Receive frame This bit indicates that a frame was received. It is cleared by writing 1 to it. This bit appears only when IRQ bit is set in the Receive Buffer Descriptor.
1	TXE	r/w	1'b0	Transmit error This bit indicates that a buffer was not transmitted due to a transmit error (underrun, retransmission limit, late collision, bus error or defer timeout). It is cleared by writing 1 to it. This bit appears only when IRQ bit is set in the Transmit Buffer Descriptor.
0	TXB	r/w	1'b0	Transmit buffer This bit indicates that a buffer has been transmitted. It is cleared by writing 1 to it. This bit appears only when IRQ bit is set in the Transmit Buffer Descriptor.

INT_MASK

Address: 0x8

Bit	Name	Type	Reset	Description
31:7	RSVD			
6	RXC_M	r/w	1'b1	Receive control frame mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked
5	TXC_M	r/w	1'b1	Transmit control frame mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked
4	BUSY_M	r/w	1'b1	Busy mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked
3	RXE_M	r/w	1'b1	Receive error mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked
2	RXB_M	r/w	1'b1	Receive frame mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked
1	TXE_M	r/w	1'b1	Transmit error mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked
0	TXB_M	r/w	1'b1	Transmit buffer mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked

IPGT

Address: 0x4000d00c

Bit	Name	Type	Reset	Description
31:7	RSVD			
6:0	IPGT	r/w	7'h18	Inter packet gap The recommended value is 0x18 (24 clock cycles), which equals 9.6 us for 10 Mbps and 0.96 us for 100 Mbps mode

PACKETLEN

Address: 0x18

Bit	Name	Type	Reset	Description
31:16	MINFL	r/w	16'h40	Minimum frame length The minimum Ethernet packet is 64 bytes long (0x40). To receive small packets, assert the RECSMALL bit or change the MINFL value. To transmit small packets, assert the PAD bit or change the MINFL value.
15:0	MAXFL	r/w	16'h600	Maximum frame length

Bit	Name	Type	Reset	Description
				The maximum Ethernet packet is 1518 bytes long. To support this and to have some additional space for tags, a default maximum packet length equals to 1536 bytes (0x600). For bigger packets, you can assert the HUGEN bit or increase the value of MAXFL field.

COLLCONFIG

Address: 0x1c

Bit	Name	Type	Reset	Description
31:20	RSVD			
19:16	MAXRET	r/w	4'hF	Maximum retry This field specifies the maximum number of consequential retransmission attempts after the collision is detected. When the maximum number has been reached, the TX MAC reports an error and stops transmitting the current packet. According to the Ethernet standard, the MAXRET default value is set to 0xf (15).
15:6	RSVD			
5:0	COLLVALID	r/w	6'h3F	Collision valid This field specifies a collision time window. A collision that occurs later than the time window is reported as a "Late Collisions" and transmission of the current packet is aborted.

TX_BD_NUM

Address: 0x20

Bit	Name	Type	Reset	Description
31	RSVD			
30:24	RXBDPTR	r	7'h0	RX buffer descriptors (BD) pointer, pointing at the RXBD currently being used
23	RSVD			
22:16	TXBDPTR	r	7'h0	TX buffer descriptors (BD) pointer, pointing at the TXBD currently being used
15:8	RSVD			
7:0	TXBDNUM	r/w	8'h40	TX buffer descriptors (BD) number Number of TX BD. TX and RX share 128 (0x80) descriptors, so the number of RX BD equals 0x80 - TXBDNUM. The maximum number of TXBDNUM is 0x80. Values greater then 0x80 cannot be written into this register.

MIIMODE

Address: 0x28

Bit	Name	Type	Reset	Description
31:9	RSVD			
8	MIINOPRE	r/w	1'b0	No preamble for Management Data (MD) 0: 32-bit preamble will be sent. 1: No preamble will be sent.
7:0	CLKDIV	r/w	8'h64	Clock divider for Management Data Clock (MDC) The source clock is bus clock and can be divided by any even number.

MIICOMMAND

Address: 0x2c

Bit	Name	Type	Reset	Description
31:3	RSVD			
2	WCTRLDATA	r/w	1'b0	Write control data, setting this bit to 1 will trigger the command (auto cleared) Note: [2]/[1]/[0] cannot be asserted at the same time, execute one command at a time
1	RSTAT	r/w	1'b0	Read status, setting this bit to 1 will trigger the command (auto cleared) Note: [2]/[1]/[0] cannot be asserted at the same time, execute one command at a time
0	SCANSTAT	r/w	1'b0	Scan status, setting this bit to 1 will trigger the command (auto cleared) Note: [2]/[1]/[0] cannot be asserted at the same time, execute one command at a time

MIIADDRESS

Address: 0x30

Bit	Name	Type	Reset	Description
31:13	RSVD			
12:8	RGAD	r/w	5'h0	Register Address
7:5	RSVD			
4:0	FIAD	r/w	5'h0	PHY Address

MIITX_DATA

Address: 0x34

Bit	Name	Type	Reset	Description
31:16	RSVD			
15:0	CTRLDATA	r/w	16'h0	Control Data to be written to PHY

MIIRX_DATA

Address: 0x38

Bit	Name	Type	Reset	Description
31:16	RSVD			
15:0	PRSD	r	16'h0	Received Data from PHY

MIISTATUS

Address: 0x3c

Bit	Name	Type	Reset	Description
31:2	RSVD			
1	MIIM_BUSY	R	1'b0	MIIM I/F busy signal 0: The MIIM I/F is ready. 1: The MIIM I/F is busy.
0	MIIM_LINKFAIL	R	1'b0	MIIM I/F link fail signal

MAC_ADDR0

Address: 0x40

Bit	Name	Type	Reset	Description
31:24	MAC_B2	r/w	8'd0	Ethernet MAC address byte 2
23:16	MAC_B3	r/w	8'd0	Ethernet MAC address byte 3
15:8	MAC_B4	r/w	8'd0	Ethernet MAC address byte 4
7:0	MAC_B5	r/w	8'd0	Ethernet MAC address byte 5

MAC_ADDR1

Address: 0x44

Bit	Name	Type	Reset	Description
31:16	RSVD			
15:8	MAC_B0	r/w	8'd0	Ethernet MAC address byte 0
7:0	MAC_B1	r/w	8'd0	Ethernet MAC address byte 1

HASH0_ADDR

Address: 0x48

Bit	Name	Type	Reset	Description
31:0	HASH0	r/w	32'h0	Lower 32-bit of HASH register

HASH1_ADDR

Address: 0x4c

Bit	Name	Type	Reset	Description
31:0	HASH1	r/w	32'h0	Upper 32-bit of HASH register

TXCTRL

Address: 0x50

Bit	Name	Type	Reset	Description
31:17	RSVD			
16	TXPAUSERQ	r/w	1'b0	TX Pause Request Writing 1 to this bit starts sending control frame and is automatically cleared to zero.
15:0	TXPAUSETV	r/w	16'h0	TX Pause Timer Value The value that is sent in the pause control frame.

uC-PCIE MAC-PHY interface CSR

- E21 address window: 0x2601xxxx.
- Pcie_sel (0x26000038, bit[3:0]): 0 = RCx8, 1..4=Endpoint 1..4, 5=RCx4

Address	Register Name	Bits	R/W	Default	Description
0x003C	**pcie_phyconfig**				**PHY configuration**
	phy_apb_write_done	[31]	RO	0	Status bit of setting uC_PHY_apb_write_done (0->1), PHY configuration completed and pipe_reset released.
	RSVD	[30:26]	RW	0	
	x4_bifurcation_en	[25]	RW	0	1=enable 2x(4x1) bifurcation mode Note: UNUSED in EP
	pclk_state_poower	[24]	RW	1	
	perst	[23]	RW	0	1=Drive PERSTn=0 at RC port.
	ictl_pma_ref_ls_ena_a	[22]	RW	0	**PHY port "ictl_pma_ref_ls_ena_a"**
	flc1_axi_slv_config	[21:20]	RW	0	00 = CXL.mem -> flc1 01 = CXL.io -> flc1 1x = D2D (host) -> flc1
	tx_lane_flip_en	[19]	RW	0	
	rx_lane_flip_en	[18]	RW	0	
	outband_pwrup_cmd	[17]	RW	0	
	margining_software_ready	[16]	RW	0	
	margining_ready	[15]	RW	0	
	pm_xmt_pme	[14]	RW	0	Drives 'apps_pm_xmt_pme'. assert this will request the controller to wakeup from L2
	dbi_ro_wr_disable	[13]	RW	0	
	pf_req_retry_en	[12]	RW	0	
	req_retry_en	[11]	RW	0	
	xfer_pending	[10]	RW	0	Asserting this will stop ASPM L1 entry

	req_exit_l1	[9]	RW	0	Write 1 and then 0 will trigger exit from L1.
	ready_entr_l23	[8]	RW	0	**This must be driven high once radm_pm_turnoff interrupt is received to enter L2**
	req_entr_l1	[7]	RW	0	Write 1 and then 0 will trigger entry into L1 when traffic is not there.
	init_rst	[6]	RW	0	Set this to 0 for EP (no functionality in EP controller)
	clk_pm_en	[5]	RW	1	
	clk_req_n	[4]	RW	1	
	hold_phy_rst	[3]	RW	0	
	ltssm_enable	[2]	RW	0	Should be enabled after writing all DBI interface registers
	sris_mode	[1]	RW	0	
	uc_phy_apb_write_done	[0]	RW	0	Firmware set this bit after PHY programming completed

0x0118	**mstr_armisc_0**				
	mstr_armisc_31_0	[31:0]	RO	0	mstra_armisc 0
0x011C	**mstr_armisc_1**				
	mstr_armisc_63_32	[31:0]	RO	0	mstra_armisc 1

0x0120	**mstr_awmisc_0**				
	mstr_awmisc_31_0	[31:0]	RO	0	mstra_awmisc 0
0x0124	**mstr_awmisc_1**				
	mstr_awmisc_63_32	[31:0]	RO	0	mstra_awmisc 1

0x0128	**mstr_awmisc_2**				
	mstr_awmisc_95_64	[31:0]	RO	0	mstra_awmisc 2
0x012C	**mstr_awmisc_3**				**Reading this register will pop awmisc FIFO after reading above 0,1,2, Note: RD_DONE signal unused**
	mstr_awmisc_127_96	[31:0]	RO	0	mstra_awmisc 3

0x0130	**mstr_resp**				**Response misc signals driven to the EP Controller**
	mstr_resp	[31:0]	RW	0	mstr_resp, [23:19] -> msg type in awmisc_fifo (This FIFO fills data based on type, wr condition is added) [18:6] -> mstr_rmisc_info [5:3] -> mstr_rmisc_info_cpl_stat [2:0] -> mstr_bmisc_info_cpl_stat
0x0134	**slv_addrh**	slv_addrh	[31:0]	RW	0

0x0138	**slv_armisc**				
	RSVD	[31:22]	RW	0	reserved field
	slv_armisc_info	[21:0]	RW	0	slv_armisc_info[21:0]
0x013c	**slv_resp**				
	slv_resp	[31:0]	RO	0	{10'h0, slv_bmisc_info, slv_rmisc_info}

0x0140	**slv_awmisc_0**				
	slv_awmisc_31_0	[31:0]	RW	0	miscellanious slv_awmisc slv_awmisc_info_hdr_34dw[31:0]

0x0144	**slv_awmisc_1**				
	slv_awmisc_63_32	[31:0]	RW	0	miscellanious slv_awmisc slv_awmisc_info_hdr_34dw[63:32]
0x0148	**slv_awmisc_2**				
	slv_awmisc_95_64	[31:0]	RW	0	{slv_awmisc_info_p_tag[95:86],slv_awmisc_info[85:64]}
0x014C	**slv_awmisc_3**				
	slv_awmisc_127_96	[31:0]	RW	0	Not connected as of now.

0x0158	**vmi_31_0**				Ideally we should write this registers first and then VMI_63_32 so that {vmi_63_32,vmi_31_0} data gets into FIFO
	vmi_31_0	[31:0]	RW	0	
0x015C	**vmi_63_32**				Writing into this register will also given pulse output which is used for writing {vmi_63_32,vmi_31_0} into async FIFO
	vmi_63_47	[31:15]	RW	0	
	vmi_gnt	[14]	RO	0	should be connected to vmi_gnt
	vmi_45_32	[13:0]	RW	0	vmi_45_32 bits
0x0160	**vmi_data_31_0**				
	vmi_data_31_0	[31:0]	RW	0	vmi data will be writtien to the FIFO when vmi_data_63_32 also written
0x0164	**vmi_data_63_32**				Writing into this register will also given pulse output which is used for writing {vmi_data_63_32,vmi_data_31_0} into async FIFO

	vmi_data_63_32	[31:0]	RW	0	

0x0170	**mac_status_1_0**				**MAC Status 0**
	mac_status_31_0	[31:0]	RO	0	MAC status 0
0x0174	**mac_status_1_1**				**MAC Status 1**
	mac_status_63_32	[31:0]	RO	0	MAC status 1
0x0178	**mac_status_1_2**				**MAC Status 2**
	mac_status_95_64	[31:0]	RO	0	MAC status 2
0x017C	**mac_status_1_3**				**MAC Status 3**
	mac_status_127_96	[31:0]	RO	0	MAC status 3

0x0180	**mac_cfg**				**MAC configuration**
	sys_int	[31]	RW	0	assert INTx interrupt from EP which was configured to the particular FUNC
	RSVD	[30]	RW	0	reserved
	ndr_rsp_meminv_thrgh_e21	[29]	RW	0	Mem Inv response from E21 will be selected on asserting this bit.
	cfg_vdm_1_wren	[28]	RW	0	Enable Writing into VDM_1 FIFO
	cfg_vdm_0_wren	[27]	RW	0	Enable Writing into VDM_0 FIFO
	Hdm_locked	[26:17]	RW	0	
	Pclk_state_rate	[16:8]	RW	9'h20	
	Cxl_mem_l1_entry_latency	[7:5]	RW	3'h3	8us Idle time will trigger L1 entry on L1 entry_en asserted below, when traffic is not present
	Cxl_mem_l1_entry_en	[4]	RW	0	ASPM entry enable, for mem interface. Similar to CXL.io (PCIe)

	Cfg_poison_on_dec_error	[3]	RW	0	asserting poisons the data when decoding error
	Cfg_hdm_dec_en	[2]	RW	0	HDM decoders gets into action
	Cxl_pm_init_cmpl	[1]	RW	0	CXL PM initialization is completed (Exchange of CXL PM VDMs)
	Sys_aux_pwr_det	[0]	RW	0	MAC configuration

0x0184	**mac_ctl**				**MAC control Register**
	mac_ctl_write	[31:8]	RW	0	For Future use (Currently Reserved)
	IO_remappers_prog_done	[7]	RW	0	IO remappers programming done by E21
	vdm_1_fifo_pop	[6]	RW	0	Write "1" to pop(read) vdm_1_fifo pcie_sel=1..4 selects endpoints 1..4
	vdm_0_fifo_pop	[5]	RW	0	Write "1" to pop(read) vdm_0_fifo pcie_sel=1..4 selects endpoints 1..4
	RSVD	[4:2]	RW	0	Reserved
	ndr_resp_en	[1]	RW	0	NDR_response enable for M2S Req, Type 2 response for M2S Req (Required for Intel SPR)
	mstr_aw_pop	[0]	RW	0	Writing to this register 1'b1 will read(pop) mstr_aw FIFO used to read the FIFO

0x0188	**mac_status_2_0**				**MAC Status 4**
	mac_status_159_128	[31:0]	RO	0	MAC status 4 [7:0] -> iip_cfg_int_pin **[11:8] -> hdm_num** [21:12] -> lbc_write_addr [26:22] -> iip_cfg_pbus_dev_num [31:27] -> iip_cfg_pbus_num[4:0]
0x018C	**mac_status_2_1**				**MAC Status 5**
	mac_status_191_160	[31:0]	RO	0	MAC status 5 [3:0] -> iip_cfg_pbus_num[7:5]

0x0190	**mstr_awaddr_0**				
	mstr_awaddr_31_0	**[31:0]**	**RO**	**0**	**mstra_awaddr 0**
0x0194	**mstr_awaddr_1**				
	RSVD	[31:20]	RO	0	Reserved field
	mstr_awaddr_51_32	[19:0]	RO	0	mstra_awaddr 1

0x01a0	**SpecRd_Req_0**				Speculative Read Address request FIFO lower 32 bits data [31:0]
	cxl_mst_m2s_specrd_addr_31_0	[31:0]	RO	0	
0x01a4	**SpecRd_Req_1**				Speculative Read Address request FIFO higher 16 bits, [47:32], Reading this register will issue read_enable of the SPECULATIVE FIFO
	RSVD	[31:16]	RO	0	Reserved
	cxl_mst_m2s_specrd_addr_47_32	[15:0]	RO	0	Address 47:32 bits

0x01a8	**MemInv_Req_0**				MemInv Requests received data 0
	cxl_mst_m2s_MemInv_NT_fifo_data_31_0	[31:0]	RO	0	

0x01ac	**MemInv_Req_1**				MemInv Requests received data 1
	cxl_mst_m2s_MemInv_NT_fifo_data_63_32	[31:0]	RO	0	
0x01b0	**MemInv_Req_3**				MemInv Requests received data 2, Rd_en may be driven to the memINV_NT FIFO, Ideally processor should read Above two registers befofe reading this register

	RSVD	[31:4]	RO	0	
	cxl_mst_m2s_MemInv_NT_fifo_data_67_64	[3:0]	RO	0	

0x01B4	**MemInv_RSP**				MemInv Response, Wr_done is used for writing FIFO
	RSVD	[31:20]	RW	0	Reserved field
	cxl_mst_m2s_MemInv_NT_rsp_fifo_data	[19:0]	RW	0	MemInv Rsp data

0x01c0	**cxl_io_remap_size_0**				
	cxl_io_remap_size_0	[31:0]	RW	0	CXL io AXI remapping size 0

0x01c4	**cxl_io_remap_base_0_0**				
	cxl_io_remap_base_0	[31:0]	RW	0	CXL io AXI remapping base address 0, [32+28-1: 28], As BAR0/1 are programmed as 64 bit, after removing 28 bits (min 256MB address remap done by io_remappers

0x01c8	**cxl_io_remap_dpabase_0**				
	cxl_io_remap_dpabase_0	[31:0]	RW	0	CXL io AXI remapping dpabase address 0

0x01d0	**cxl_io_remap_size_1**				
	cxl_io_remap_size_1	[31:0]	RW	0	CXL io AXI remapping size 1

0x01d4	**cxl_io_remap_base_1_0**				
	cxl_io_remap_base_1	[31:0]	RW	0	CXL io AXI remapping base address 1, [32+28-1: 28], As BAR0/1 are programmed as 64 bit, after removing 28 bits (min 256MB address remap done by io_remappers

0x01d8	**cxl_io_remap_dpabase_1**				
	cxl_io_remap_dpabase_1	[31:0]	RW	0	CXL io AXI remapping dpabase address 1

0x1dc	**radm_slot_pwr_payload**				
	radm_slot_pwr_payload	[31:0]	RO	0	used to capture radm_slot_pwr limit payload message
0x01e0	**cxl_io_remap_base_0_1**				
	RSVD	**[31:4]**	RW	**0**	**Reserved**
	cxl_io_remap_base_35_32_0	[3:0]	RW	0	CXL io AXI remapping base address 0, [63:60], As BAR0/1 are programmed as 64 bit, after removing 28 bits (min 256MB address remap done by io_remappers
0x01e4	**cxl_io_remap_base_1_1**				
	RSVD	**[31:4]**	RW	**0**	**Reserved**
	cxl_io_remap_base_35_32_1	[3:0]	RW	0	CXL io AXI remapping base address 0, [63:60], As BAR4/5 are programmed as 64 bit, after removing 28 bits (min 256MB address remap done by io_remappers
0x1e8	**interrupt_Control_Register**				
	ICR_31_15	**[31:27]**	**RW**	**0**	**May be used in future (Reserved for now)**
	intr_clr_elbi_15	**[26]**	**RW**	**0**	Interrupt clear from ELBI 15 (16 ranges are given)

	intr_clr_elbi_14	**[25]**	**RW**	**0**	Interrupt clear from ELBI 14 (16 ranges are given)
	intr_clr_elbi_13	**[24]**	**RW**	**0**	Interrupt clear from ELBI 13 (16 ranges are given)
	intr_clr_elbi_12	**[23]**	**RW**	**0**	Interrupt clear from ELBI 12 (16 ranges are given)
	intr_clr_elbi_11	**[22]**	**RW**	**0**	Interrupt clear from ELBI 11 (16 ranges are given)
	intr_clr_elbi_10	**[21]**	**RW**	**0**	Interrupt clear from ELBI 10 (16 ranges are given)
	intr_clr_elbi_9	**[20]**	**RW**	**0**	Interrupt clear from ELBI 9 (16 ranges are given)
	intr_clr_elbi_8	**[19]**	**RW**	**0**	Interrupt clear from ELBI 8 (16 ranges are given)
	intr_clr_elbi_7	**[18]**	**RW**	**0**	Interrupt clear from ELBI 7 (16 ranges are given)
	intr_clr_elbi_6	**[17]**	**RW**	**0**	Interrupt clear from ELBI 6 (16 ranges are given)
	intr_clr_elbi_5	**[16]**	**RW**	**0**	Interrupt clear from ELBI 5 (16 ranges are given)
	intr_clr_elbi_4	**[15]**	**RW**	**0**	Interrupt clear from ELBI 4 (16 ranges are given)
	intr_clr_elbi_3	**[14]**	**RW**	**0**	Interrupt clear from ELBI 3 (16 ranges are given)
	intr_clr_elbi_2	**[13]**	**RW**	**0**	Interrupt clear from ELBI 2 (16 ranges are given)
	intr_clr_elbi_1	**[12]**	**RW**	**0**	Interrupt clear from ELBI 1 (16 ranges are given)
	intr_clr_elbi_0	**[11]**	**RW**	**0**	Interrupt clear from ELBI 0 (16 ranges are given)
	cfg_send_f_clr	[10]	RW	0	write 1 to clear fatal error interrupt
	cfg_send_nf_clr	[9]	RW	0	write 1 to clear non-fatal error interrupt
	cfg_send_corr_err_clr	[8]	RW	0	write 1 to clear correctable error interrupt
	radm_qoverflow_clr	[7]	RW	0	write 1 to clear radm_q_overflow interrupt
	trgt_cpl_timeout_clr	[6]	RW	0	write 1 to clear trgt_cpl_timeout interrupt
	mem_space_en_ack	[5]	RW	0	Writing 1 will generate a pulse and will clear mem_space_en interrupt

	iip_radm_pm_turnoff_ack	[4]	RW	0	Writing this bit to 1'b1 will de-assert radm_pm_turnoff interrupt to the E21,
	intx_assert_ack	[3]	RW	0	Writing 1 will generate a pulse will clear deassert INTx and gets auto clear
	intx_deassert_ack	[2]	RW	0	Writing 1 will generate a pulse will clear assert INTx and gets auto clear
	hot_reset_ack	[1]	RW	0	Writing this bit to 1'b1 and 1'b0 deassert hot_reset_req interrupt to the E21, This signal anded with wr_en of this register is used in logic
	radm_pme_or_err_ack	[0]	RW	0	write 1 to clear rc pme or err interrupt, pcie_sel = selects rcx8, pcie_sel=5 selects rcx4 **Note: RSVD for EP**
0x1ec	**mac_status_future_use**				**Reseved register for future use**
	mac_status_rsvd	[31:0]	RO	0	Reseved register for future use
0x1f0	**size_of_mmio_cfg_registers**				Size of MMIO and CFG register can be programmed here
	size_of_bar0_mem_mapped_registers	[31:16]	RW	16'h1f0	BAR0 MMIO registers (512-16 = 496 bytes)
	size_of_cfg_mapped_registers	[15:0]	RW	16'h100	Config Mapped Registers (256 bytes default value)

0x1f4	**Interrupt_Control_Register_2**				
	ICR_63_32	[31:0]	RW	0	0th bit , clear interrupt due to rdlh_link_up
0x1f8	**Interrupt_Mask_Register_1**				Mask the interrupt Register 1 (Writing 1 to this Register will disable the interrupt)
	IMR_1	[31:0]	RW	0	0th- 1st - 2nd - 3rd- 4th -

					5th - 6th - ISR_6 (ELBI range 15) 7th - 8th - 9th- intx_deasserted 10th - intx_asserted 11th - trgt_cpl_timeout 12th - radm_qoverflow 13th - iip_cfg_send_cor_err 14th - iip_cfg_send_nf_err 15th - iip_cfg_send_f_err 16th - 17th -31st - ISR_31_17 (0th range to 14th range)
0x1fc	**Interrupt_Mask_Register_2**				Mask the interrupt Register 2 (Writing 1 to this Register will disable the interrupt)
	IMR_2	[31:0]	RW	0	0th - rdlh_link_up masking

0x0200	**cxl_mem_remap_size_0**				
	cxl_mem_remap_size_0	[31:0]	RW	0	CXL Mem AXI remapping size 0

0x0204	**cxl_mem_remap_base_0**				
	cxl_mem_remap_base_0	[31:0]	RW	0	CXL Mem AXI remapping base address 0

0x0208	**cxl_mem_remap_dpabase_0**				
	cxl_mem_remap_dpabase_0	[31:0]	RW	0	CXL Mem AXI remapping DPA address 0 [19:0] DPA base -> Final DPA base = {dpa_base[19:0],28'b0} [23:20] - IW [27:24]- IG

0x020C	Interrupt_Status_Register				Interrupt Status Register
	i_ISR_31_17	[31:17]	RO	0	For Future use(currently Reserved)- Should be connected to 0 for now. 31:17 is Interrupt Stutus from ELBI, 0p7 has used [20:17] bits only
	iip_mem_space_en	[16]	RO	0	Mem Space is enabled
	iip_cfg_send_f_err	[15]	RO	0	controller has sent fatal error message
	iip_cfg_send_nf_err	[14]	RO	0	controller has sent non-fatal error message
	iip_cfg_send_cor_err	[13]	RO	0	controller has sent correctable error message
	radm_qoverflow	[12]	RO	0	radm q overflow occurred
	trgt_cpl_timeout	[11]	RO	0	EP completion timeout occurred
	intx_asserted	[10]	RO	0	Intx asserted for the corresponding Function
	intx_deasserted	[9]	RO	0	Intx de-asserted for the corresponding Function
	m2sreq_memInv_NT_rsp_fifo_full	[8]	RO	0	MemInv Rsp FIFO full
	hot_reset_req	[7]	RO	0	hot reset is triggered from the Host, This has to be acknowledged by writing bit hot_reset_ack in the 'mac_ctl' register.
	i_ISR_6	[6]	RW	0	interrupt Status 15th from ELBI
	m2sreq_memInv_NT_fifo_empty_n	[5]	RO	0	MemInv FIFO not empty(data to be read)
	m2sreq_specrd_fifo_empty_n	[4]	RO	0	Spec Rd FIFO not empty(data to be read)
	mstr_aw_empty_n	[3]	RO	0	Master aw misc FIFO not empty (data to be read)
	radm_pm_turnoff	[2]	RO	0	PM Turnoff Message received from the Host
	vdm_1_fifo_empty_n	[1]	RO	0	VDM 1 FIFO not empty
	vdm_0_fifo_empty_n	[0]	RO	0	VDM 0 FIFO not empty

0x0210	**cxl_mem_remap_size_1**				
	cxl_mem_remap_size_1	[31:0]	RW	0	CXL Mem AXI remapping size 1

0x0214	**cxl_mem_remap_base_1**				
	cxl_mem_remap_base_1	[31:0]	RW	0	CXL Mem AXI remapping base address 1

0x0218	**cxl_mem_remap_dpabase_1**				
	cxl_mem_remap_dpabase_1	[31:0]	RW	0	CXL Mem AXI remapping DPA address 1 [19:0] DPA base -> Final DPA base = {cxl_mem_remap_dpabase_1[19:0],28'b0} [23:20] - IW [27:24]- IG

0x0220	**cxl_mem_remap_size_2**				
	cxl_mem_remap_size_2	[31:0]	RW	0	CXL Mem AXI remapping size 2

0x0224	**cxl_mem_remap_base_2**				
	cxl_mem_remap_base_2	[31:0]	RW	0	CXL Mem AXI remapping base address 2

0x0228	**cxl_mem_remap_dpabase_2**				
	cxl_mem_remap_dpabase_2	[31:0]	RW	0	CXL Mem AXI remapping DPA address 2

0x0230	**cxl_mem_remap_size_3**				
	cxl_mem_remap_size_3	[31:0]	RW	0	CXL Mem AXI remapping size 3

0x0234	cxl_mem_remap_base_3				
	cxl_mem_remap_base_3	[31:0]	RW	0	CXL Mem AXI remapping base address 3

0x0238	cxl_mem_remap_dpabase_3				
	cxl_mem_remap_dpabase_3	[31:0]	RW	0	CXL Mem AXI remapping DPA address 3

0x0240	cxl_mem_remap_size_4				
	cxl_mem_remap_size_4	[31:0]	RW	0	CXL Mem AXI remapping size 4

0x0244	cxl_mem_remap_base_4				
	cxl_mem_remap_base_4	[31:0]	RW	0	CXL Mem AXI remapping base address 4

0x0248	cxl_mem_remap_dpabase_4				
	cxl_mem_remap_dpabase_4	[31:0]	RW	0	CXL Mem AXI remapping DPA address 4

0x0250	cxl_mem_remap_size_5				
	cxl_mem_remap_size_5	[31:0]	RW	0	CXL Mem AXI remapping size 5

0x0254	cxl_mem_remap_base_5				
	cxl_mem_remap_base_5	[31:0]	RW	0	CXL Mem AXI remapping base address 5

0x0258	cxl_mem_remap_dpabase_5				
	cxl_mem_remap_dpabase_5	[31:0]	RW	0	CXL Mem AXI remapping DPA address 5

0x0260	**cxl_mem_remap_size_6**				
	cxl_mem_remap_size_6	[31:0]	RW	0	CXL Mem AXI remapping size 6

0x0264	**cxl_mem_remap_base_6**				
	cxl_mem_remap_base_6	[31:0]	RW	0	CXL Mem AXI remapping base address 6

0x0268	**cxl_mem_remap_dpabase_6**				
	cxl_mem_remap_dpabase_6	[31:0]	RW	0	CXL Mem AXI remapping DPA address 6

0x0270	**cxl_mem_remap_size_7**				
	cxl_mem_remap_size_7	[31:0]	RW	0	CXL Mem AXI remapping size 7

0x0274	**cxl_mem_remap_base_7**				
	cxl_mem_remap_base_7	[31:0]	RW	0	CXL Mem AXI remapping base address 7

0x0278	**cxl_mem_remap_dpabase_7**				
	cxl_mem_remap_dpabase_7	[31:0]	RW	0	CXL Mem AXI remapping DPA address 7

0x0280	**cxl_mem_remap_size_8**				
	cxl_mem_remap_size_8	[31:0]	RW	0	CXL Mem AXI remapping size 8

0x0284	**cxl_mem_remap_base_8**				
	cxl_mem_remap_base_8	[31:0]	RW	0	CXL Mem AXI remapping base address 8

0x0288	**cxl_mem_remap_dpa base_8**				
	cxl_mem_remap_dpa base_8	[31:0]	RW	0	CXL Mem AXI remapping DPA address 8

0x0290	**cxl_mem_remap_size _9**				
	cxl_mem_remap_size _9	[31:0]	RW	0	CXL Mem AXI remapping size 9

0x0294	**cxl_mem_remap_bas e_9**				
	cxl_mem_remap_base _9	[31:0]	RW	0	CXL Mem AXI remapping base address 9

0x0298	**cxl_mem_remap_dpa base_9**				
	cxl_mem_remap_dpa base_9	[31:0]	RW	0	CXL Mem AXI remapping DPA address 9

0x02a0	**vdm_0_0**				
	vdm_0_31_0	[31:0]	RO	0	vdm0_31_0
0x02a4	**vdm_0_1**				
	vdm_0_63_32	[31:0]	RO	0	vdm0_63_32
0x02a8	**vdm_0_2**				
	RSVD	[31:17]			
	vdm_0_fifo_empty	[16]	RO	0	vdm_0_fifo_empty
	vdm_0_79_64	[15:0]	RO	0	vdm0_79_64, Read enable to the vdm fifo will be driven from generated signal while reading from this Register, Order for getting data from VDM FIFO is read vdm_0_0, vdm_0_1, vdm_0_2

0x02b0	**vdm_1_0**				
	vdm_1_31_0	[31:0]	RO	0	vdm1_31_0, lower 32 bits of VDM
0x02b4	**vdm_1_1**				
	vdm_1_63_32	[31:0]	RO	0	vdm1_63_32
0x02b8	**vdm_1_2**				
	RSVD	[31:17]	RO		
	vdm_1_fifo_empty	[16]	RO	0	vdm_1_fifo_empty
	vdm_1_79_64	[15:0]	RO	0	vdm1_79_64, Read enable to the vdm fifo will be driven from generated signal while reading from this Register, Order for getting data from VDM FIFO is read vdm_0_0, vdm_0_1, vdm_0_2
0x300	**start_addr_range_1**				**Address Range 1 for routing memory space registers to SRAM**
	start_addr_range_1	[31:0]	RW	32'h0002_0000	
0x304	**range1_properties**				**Size of Range1**
	RSVD	[31:20]	RW	0	Reserved
	bar_range1	[19:16]	RW	4'h0	bar_number_of_range1, default is BAR0
	size_range1	[15:0]	RW	16'h40	Size of Range1 , default is 64bytes
0x308	**SRAM_target_address_1**				

	RSVD	**[31:16]**	**RW**	**0**	
	range1_sram_dest_addr	[15:0]	RW	16'h0400	sram range1 destination address
0x310	**start_addr_range_2**				**Address Range 2 for routing memory space registers to SRAM**
	start_addr_range_2	[31:0]	RW	32'h0002_1000	
0x314	**range2_properties**				**Size of Range2**
	RSVD	[31:20]	RW	0	Reserved
	bar_range2	[19:16]	RW	4'h0	bar_number_of_range2
	size_range2	[15:0]	RW	16'h10	Size of Range2
0x318	**SRAM_target_address_2**				
	RSVD	**[31:16]**	**RW**	**0**	
	range2_sram_dest_addr	[15:0]	RW	16'h0440	sram range2 destination address
0x320	**start_addr_range_3**				**Address Range 3 for routing memory space registers to SRAM**
	start_addr_range_3	[31:0]	RW	32'h0002_2000	
0x324	**range3_properties**				**Size of Range3**
	RSVD	[31:20]	RW	0	Reserved
	bar_range3	[19:16]	RW	4'h0	bar_number_of_range3

	size_range3	[15:0]	RW	16'h220	Size of Range3

0x328	**SRAM_target_address_3**				
	RSVD	**[31:16]**	**RW**	**0**	
	range3_sram_dest_addr	[15:0]	RW	16'h450	sram range3 destination address
0x330	**start_addr_range_4**				**Address Range 4 for routing memory space registers to SRAM**
	start_addr_range_4	[31:0]	RW	32'h0012_3000	
0x334	**range4_properties**				**Size of Range4**
	RSVD	[31:20]	RW	0	Reserved
	bar_range4	[19:16]	RW	4'h0	bar_number_of_range4
	size_range4	[15:0]	RW	16'h10	Size of Range4
0x338	**SRAM_target_address_4**				
	RSVD	**[31:16]**	**RW**	**0**	
	range4_sram_dest_addr	[15:0]	RW	16'h670	sram range4 destination address
0x33c	**start_addr_range_5**				**Address Range 5 for routing memory space registers to SRAM**
	start_addr_range_5	[31:0]	RW	32'h0012_3000	

0x340	**range5_properties**				**Size of Range5**
	RSVD	[31:20]	RW	0	Reserved
	bar_range5	[19:16]	RW	4'h0	bar_number_of_range5
	size_range5	[15:0]	RW	16'h10	Size of Range5
0x344	**SRAM_target_address_5**				
	RSVD	**[31:16]**	**RW**	**0**	
	range5_sram_dest_addr	[15:0]	RW	16'h6700	sram range5 destination address
0x348	**start_addr_range_6**				**Address Range 6 for routing memory space registers to SRAM**
	start_addr_range_6	[31:0]	RW	32'h0012_3000	
0x34c	**range6_properties**				**Size of Range6**
	RSVD	[31:20]	RW	0	Reserved
	bar_range6	[19:16]	RW	4'h0	bar_number_of_range6
	size_range6	[15:0]	RW	16'h10	Size of Range6
0x350	**SRAM_target_address_6**				
	RSVD	**[31:16]**	**RW**	**0**	
	range6_sram_dest_addr	[15:0]	RW	16'h6700	sram range6 destination address

0x354	**start_addr_range_7**				**Address Range 7 for routing memory space registers to SRAM**
	start_addr_range_7	[31:0]	RW	32'h0012_3000	
0x358	**range7_properties**				**Size of Range7**
	RSVD	[31:20]	RW	0	Reserved
	bar_range7	[19:16]	RW	4'h0	bar_number_of_range7
	size_range7	[15:0]	RW	16'h10	Size of Range7
0x35c	**SRAM_target_address_7**				
	RSVD	**[31:16]**	**RW**	**0**	
	range7_sram_dest_addr	[15:0]	RW	16'h670	sram range7 destination address
0x360	**start_addr_range_8**				**Address Range 8 for routing memory space registers to SRAM**
	start_addr_range_8	[31:0]	RW	32'h0012_3000	
0x364	**range8_properties**				**Size of Range8**
	RSVD	[31:20]	RW	0	Reserved
	bar_range8	[19:16]	RW	4'h0	bar_number_of_range8
	size_range8	[15:0]	RW	16'h10	Size of Range8
0x368	**SRAM_target_address_8**				

	RSVD	**[31:16]**	**RW**	**0**	
	range8_sram_dest_addr	[15:0]	RW	16'h6700	sram range8 destination address
0x36c	**start_addr_range_9**				**Address Range 8 for routing memory space registers to SRAM**
	start_addr_range_9	[31:0]	RW	32'h0012_3000	
0x370	**range9_properties**				**Size of Range9**
	RSVD	[31:20]	RW	0	Reserved
	bar_range9	[19:16]	RW	4'h0	bar_number_of_range9
	size_range9	[15:0]	RW	16'h10	Size of Range9
0x374	**SRAM_target_address_9**				
	RSVD	**[31:16]**	**RW**	**0**	
	range9_sram_dest_addr	[15:0]	RW	16'h6700	sram range9 destination address
0x378	**start_addr_range_10**				**Address Range 8 for routing memory space registers to SRAM**
	start_addr_range_10	[31:0]	RW	32'h0012_3000	
0x37c	**range10_properties**				**Size of Range10**
	RSVD	[31:20]	RW	0	Reserved
	bar_range10	[19:16]	RW	4'h0	bar_number_of_range10

	size_range10	[15:0]	RW	16'h10	Size of Range10
0x380	**SRAM_target_address_10**				
	RSVD	**[31:16]**	**RW**	**0**	
	range10_sram_dest_addr	[15:0]	RW	16'h670	sram range10 destination address
0x384	**start_addr_range_11**				**Address Range 8 for routing memory space registers to SRAM**
	start_addr_range_11	[31:0]	RW	32'h0012_3000	
0x388	**range11_properties**				**Size of Range11**
	RSVD	[31:20]	RW	0	Reserved
	bar_range11	[19:16]	RW	4'h0	bar_number_of_range11
	size_range11	[15:0]	RW	16'h10	Size of Range11
0x38c	**SRAM_target_address_11**				
	RSVD	**[31:16]**	**RW**	**0**	
	range11_sram_dest_addr	[15:0]	RW	16'h670	sram range11 destination address
0x390	**start_addr_range_12**				**Address Range 8 for routing memory space registers to SRAM**
	start_addr_range_12	[31:0]	RW	32'h0012_3000	

0x394	**range12_properties**				**Size of Range12**
	RSVD	[31:20]	RW	0	Reserved
	bar_range12	[19:16]	RW	4'h0	bar_number_of_range12
	size_range12	[15:0]	RW	16'h10	Size of Range12
0x398	**SRAM_target_address_12**				
	RSVD	**[31:16]**	**RW**	**0**	
	range12_sram_dest_addr	[15:0]	RW	16'h6700	sram range12 destination address
0x39c	**start_addr_range_13**				**Address Range 8 for routing memory space registers to SRAM**
	start_addr_range_13	[31:0]	RW	32'h0012_3000	
0x3a0	**range13_properties**				**Size of Range13**
	RSVD	[31:20]	RW	0	Reserved
	bar_range13	[19:16]	RW	4'h0	bar_number_of_range13
	size_range13	[15:0]	RW	16'h10	Size of Range13
0x3a4	**SRAM_target_address_13**				
	RSVD	**[31:16]**	**RW**	**0**	
	range13_sram_dest_addr	[15:0]	RW	16'h6700	sram range13 destination address

0x3a8	**start_addr_range_14**				**Address Range 8 for routing memory space registers to SRAM**
	start_addr_range_14	[31:0]	RW	32'h0012_3000	
0x3ac	**range14_properties**				**Size of Range14**
	RSVD	[31:20]	RW	0	Reserved
	bar_range14	[19:16]	RW	4'h0	bar_number_of_range14
	size_range14	[15:0]	RW	16'h10	Size of Range14
0x3b0	**SRAM_target_address_14**				
	RSVD	**[31:16]**	**RW**	**0**	
	range14_sram_dest_addr	[15:0]	RW	16'h670	sram range14 destination address
0x3b4	**start_addr_range_15**				**Address Range 15 for routing memory space registers to SRAM**
	start_addr_range_15	[31:0]	RW	32'h0012_3000	
0x3b8	**range15_properties**				**Size of Range15**
	RSVD	[31:20]	RW	0	Reserved
	bar_range15	[19:16]	RW	4'h0	bar_number_of_range15
	size_range15	[15:0]	RW	16'h10	Size of Range15
0x3bc	**SRAM_target_address_15**				

	RSVD	**[31:16]**	**RW**	**0**	
	range15_sram_dest_addr	[15:0]	RW	16'h670	sram range15 destination address
0x3c0	**start_addr_range_16**				**Address Range 16 for routing memory space registers to SRAM**
	start_addr_range_16	[31:0]	RW	32'h0012_3000	
0x3c4	**range16_properties**				**Size of Range16**
	RSVD	[31:20]	RW	0	Reserved
	bar_range16	[19:16]	RW	4'h0	bar_number_of_range16
	size_range16	[15:0]	RW	16'h10	Size of Range16
0x3c8	**SRAM_target_address_16**				
	RSVD	**[31:16]**	**RW**	**0**	
	range16_sram_dest_addr	[15:0]	RW	16'h670	sram range16 destination address
0x400	**Interrupt_properties_range1_1**				**Interrupt_properties for 1st range**
	addr_int_range1	[31:0]	RW	0	Address for which interrupt to be raised
0x404	**Interrupt_properties_range1_2**				
	RSVD	[31:5]	RW	0	Reserved
	rd_int_range1	[4]	RW	0	1 for read based on interrupt
	wr_int_range1	[3]	RW	0	1 for write based interrupt

	bar_int_range1	[2:0]	RW	0	bar number for which interrupt will be raised, 3'b111 for CFG,
0x408	**Interrupt_properties_range2_1**				**Interrupt_properties for 2nd range**
	addr_int_range2	[31:0]	RW	0	Address for which interrupt to be raised
0x40c	**Interrupt_properties_range2_2**				
	RSVD	[31:5]	RW	0	Reserved
	rd_int_range2	[4]	RW	0	1 for read based on interrupt
	wr_int_range2	[3]	RW	0	1 for write based interrupt
	bar_int_range2	[2:0]	RW	0	bar number for which interrupt will be raised
0x410	**Interrupt_properties_range3_1**				**Interrupt_properties for 3rd range**
	addr_int_range3	[31:0]	RW	0	Address for which interrupt to be raised
0x414	**Interrupt_properties_range3_2**				
	RSVD	[31:5]	RW	0	Reserved
	rd_int_range3	[4]	RW	0	1 for read based on interrupt
	wr_int_range3	[3]	RW	0	1 for write based interrupt
	bar_int_range3	[2:0]	RW	0	bar number for which interrupt will be raised
0x418	**Interrupt_properties_range4_1**				**Interrupt_properties for 4th range**
	addr_int_range4	[31:0]	RW	0	Address for which interrupt to be raised

0x41c	**Interrupt_properties_range4_2**				
	RSVD	[31:5]	RW	0	Reserved
	rd_int_range4	[4]	RW	0	1 for read based on interrupt
	wr_int_range4	[3]	RW	0	1 for write based interrupt
	bar_int_range4	[2:0]	RW	0	bar number for which interrupt will be raised
0x420	**Interrupt_properties_range5_1**				**Interrupt_properties for 4th range**
	addr_int_range5	[31:0]	RW	0	Address for which interrupt to be raised
0x424	**Interrupt_properties_range5_2**				
	RSVD	[31:5]	RW	0	Reserved
	rd_int_range5	[4]	RW	0	1 for read based on interrupt
	wr_int_range5	[3]	RW	0	1 for write based interrupt
	bar_int_range5	[2:0]	RW	0	bar number for which interrupt will be raised
0x428	**Interrupt_properties_range6_1**				**Interrupt_properties for 4th range**
	addr_int_range6	[31:0]	RW	0	Address for which interrupt to be raised
0x42c	**Interrupt_properties_range6_2**				
	RSVD	[31:5]	RW	0	Reserved
	rd_int_range6	[4]	RW	0	1 for read based on interrupt
	wr_int_range6	[3]	RW	0	1 for write based interrupt
	bar_int_range6	[2:0]	RW	0	bar number for which interrupt will be raised

0x430	**Interrupt_properties_range7_1**				**Interrupt_properties for 4th range**
	addr_int_range7	[31:0]	RW	0	Address for which interrupt to be raised
0x434	**Interrupt_properties_range7_2**				
	RSVD	[31:5]	RW	0	Reserved
	rd_int_range7	[4]	RW	0	1 for read based on interrupt
	wr_int_range7	[3]	RW	0	1 for write based interrupt
	bar_int_range7	[2:0]	RW	0	bar number for which interrupt will be raised
0x438	**Interrupt_properties_range8_1**				**Interrupt_properties for 4th range**
	addr_int_range8	[31:0]	RW	0	Address for which interrupt to be raised
0x43c	**Interrupt_properties_range8_2**				
	RSVD	[31:5]	RW	0	Reserved
	rd_int_range8	[4]	RW	0	1 for read based on interrupt
	wr_int_range8	[3]	RW	0	1 for write based interrupt
	bar_int_range8	[2:0]	RW	0	bar number for which interrupt will be raised
0x440	**Interrupt_properties_range9_1**				**Interrupt_properties for 4th range**
	addr_int_range9	[31:0]	RW	0	Address for which interrupt to be raised
0x444	**Interrupt_properties_range9_2**				

	RSVD	[31:5]	RW	0	Reserved
	rd_int_range9	[4]	RW	0	1 for read based on interrupt
	wr_int_range9	[3]	RW	0	1 for write based interrupt
	bar_int_range9	[2:0]	RW	0	bar number for which interrupt will be raised
0x448	**Interrupt_properties_range10_1**				**Interrupt_properties for 4th range**
	addr_int_range10	[31:0]	RW	0	Address for which interrupt to be raised
0x44c	**Interrupt_properties_range10_2**				
	RSVD	[31:5]	RW	0	Reserved
	rd_int_range10	[4]	RW	0	1 for read based on interrupt
	wr_int_range10	[3]	RW	0	1 for write based interrupt
	bar_int_range10	[2:0]	RW	0	bar number for which interrupt will be raised
0x450	**Interrupt_properties_range11_1**				**Interrupt_properties for 4th range**
	addr_int_range11	[31:0]	RW	0	Address for which interrupt to be raised
0x454	**Interrupt_properties_range11_2**				
	RSVD	[31:5]	RW	0	Reserved
	rd_int_range11	[4]	RW	0	1 for read based on interrupt
	wr_int_range11	[3]	RW	0	1 for write based interrupt
	bar_int_range11	[2:0]	RW	0	bar number for which interrupt will be raised

0x458	**Interrupt_properties_range12_1**				**Interrupt_properties for 4th range**
	addr_int_range12	[31:0]	RW	0	Address for which interrupt to be raised
0x45c	**Interrupt_properties_range12_2**				
	RSVD	[31:5]	RW	0	Reserved
	rd_int_range12	[4]	RW	0	1 for read based on interrupt
	wr_int_range12	[3]	RW	0	1 for write based interrupt
	bar_int_range12	[2:0]	RW	0	bar number for which interrupt will be raised
0x460	**Interrupt_properties_range13_1**				**Interrupt_properties for 4th range**
	addr_int_range13	[31:0]	RW	0	Address for which interrupt to be raised
0x464	**Interrupt_properties_range13_2**				
	RSVD	[31:5]	RW	0	Reserved
	rd_int_range13	[4]	RW	0	1 for read based on interrupt
	wr_int_range13	[3]	RW	0	1 for write based interrupt
	bar_int_range13	[2:0]	RW	0	bar number for which interrupt will be raised
0x468	**Interrupt_properties_range14_1**				**Interrupt_properties for 4th range**
	addr_int_range14	[31:0]	RW	0	Address for which interrupt to be raised
0x46c	**Interrupt_properties_range14_2**				
	RSVD	[31:5]	RW	0	Reserved

	rd_int_range14	[4]	RW	0	1 for read based on interrupt
	wr_int_range14	[3]	RW	0	1 for write based interrupt
	bar_int_range14	[2:0]	RW	0	bar number for which interrupt will be raised
0x470	**Interrupt_properties_range15_1**				**Interrupt_properties for 4th range**
	addr_int_range15	[31:0]	RW	0	Address for which interrupt to be raised
0x474	**Interrupt_properties_range15_2**				
	RSVD	[31:5]	RW	0	Reserved
	rd_int_range15	[4]	RW	0	1 for read based on interrupt
	wr_int_range15	[3]	RW	0	1 for write based interrupt
	bar_int_range15	[2:0]	RW	0	bar number for which interrupt will be raised
0x478	**Interrupt_properties_range16_1**				**Interrupt_properties for 4th range**
	addr_int_range16	[31:0]	RW	0	Address for which interrupt to be raised
0x47c	**Interrupt_properties_range16_2**				
	RSVD	[31:5]	RW	0	Reserved
	rd_int_range16	[4]	RW	0	1 for read based on interrupt
	wr_int_range16	[3]	RW	0	1 for write based interrupt
	bar_int_range16	[2:0]	RW	0	bar number for which interrupt will be raised
0x500	**doe_mailbox_register_addr_1**				

	RSVD	[31:24]	RW	0	Reserved
	doe_cfg_size_1	[23:12]	RW	0x18	doe_cfg_size_1
	base_addr_doe_capab_1	[11:0]	RW	0	base address of doe mail box regiset set 1
0x504	**doe_mailbox_register_addr_2**				
	RSVD	[31:24]	RW	0	Reserved
	doe_cfg_size_2	[23:12]	RW	0x18	doe_cfg_size_2
	base_addr_doe_capab_2	[11:0]	RW	0	base address of doe mail box regiset set 2
0x508	**doe_mailbox_register_addr_3**				
	RSVD	[31:24]	RW	0	Reserved
	doe_cfg_size_3	[23:12]	RW	0x18	doe_cfg_size_3
	base_addr_doe_capab_3	[11:0]	RW	0	base address of doe mail box regiset set 3

NEO OTP CSR Base address = 0x2608_0000

Register: OFS_VERSION

Address: 0x26083400

Bits	Bit Name	Default	Type	Comment
31:0	WRAPPER_VER	32'h20230613	R	

Register: OFS_PART_NUM

Address: 0x26083404

Bits	Bit Name	Default	Type	Comment
31:0	NEO_PART_NUM	32'h4547512D	R	

Register: OFS_INTRPT

Address: 0x26083408

Bits	Bit Name	Default	Type	Comment
31:18	Reserved	N/A	N/A	Reserved
17:16	intrpt_en [1:0]	2'b00	R/W	
15:2	Reserved	N/A	N/A	Reserved
1:0	intrpt_st [1:0] (R) intrpt_clr [1:0] (W)	2'b00	R/W1c	Write 1'b1=intrpt_clr

Register: OFS_STATUS

Address: 0x26083500

Bits	Bit Name	Default	Type	Comment
31:14	Reserved	N/A	N/A	Reserved
13	watch_dog_active	1'b0	R	
12	bist_mode	1'b0	R	write 0x3508[31:24] = 8'hA5
11:8	bist_fail_flag [3:0]	4'b0	R	
7:1	Reserved	N/A	N/A	Reserved
0	ctl_busy	1'b0	R	

Register: OFS_DEEP

Address: 0x26083504

Bits	Bit Name	Default	Type	Comment
31	bist_blank_check	1'b0	R/W	
30	bist_wtd_enable	1'b1	R/W	
29:1	Reserved	N/A	N/A	Reserved
0	ctl_pdstb	1'b1	R/W	

Register: OFS_CONFIG

Address: 0x26083508

Bits	Bit Name	Default	Type	Comment
31:24	set_bist_mode	8'h0	W	Write 8'hA5 to set bist_mode(0x3500[12])
23:11	Reserved	N/A	N/A	Reserved
10:8	ctl_rdmd	3'b000	R/W	read modes: 3'b000=CTL_RD 3'b001=CTL_INIT_MG_RD 3'b010=CTL_PGM_MG_RD 3'b011=CTL_HT_INIT_MG_RD 3'b100=CTL_HT_PGM_MG_RD 3'b101=CTL_LT_PGM_MG_RD
7:3	Reserved	N/A	N/A	Reserved
2	ctl_pgm_prt	1'b0	R/W	
1	ctl_pgm_ign	1'b0	R/W	
0	Reserved	N/A	N/A	Reserved