Qualcomm Technologies, Inc.

Ethernet Quality-of-Service (ETHQOS)

Hardware Programming Guide

80-PE986-11HG Rev. A

February 15, 2019

For additional information or to submit technical questions, go to: https://createpoint.qti.qualcomm.com

Confidential and Proprietary – Qualcomm Technologies, Inc.

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision history

Revision	Date	Description
A	February 2019	Initial release

Contents

Figures

1 Introduction

This document describes how to program the hardware for the Ethqos controller core. It is intended to be used in conjunction with the *SA8155 Hardware Register Description* (80-PE986-2X) or the *SA6155 Hardware Register Description* (80-PJ407-2X) as the case may be, which contains the registers of the engine.

1.1 Scope

This document is for customer distribution to engineering teams involved in the development, integration, and deployment of Qualcomm Technologies, Inc. (QTI) ASICs.

1.2 References

Ref.	Resource	DCN/reference source
QTI		
1.	*EMAC Controller data book*	Synopsys, Inc.
2.	*SA8155 Hardware Register Description*	80-PE986-2X
Standards		
3.	*Reduced Gigabit Media Independent Interface (RGMII)*	
4.	*IEEE802.3.2005*	

1.3 Functional description

The Ethqos IP is compliant with the Ethernet IEEE 802.3-2002 standard.

EMAC supports full-duplex mode at 10/100/1000Mbps.

The Ethqos core consists of two configurable FIFOs on the Tx and Rx sides. The FIFOs handle the application's latency during frame transmission and reception.

A Management Data Input or Output and Management Data Clock (MDIO/MDC) interface (clause 22) provides control and management functions to the external physical layer (PHY) device.

EMAC v1.1.0 provides a 10/100 Mbps RMII and a 1000 Mbps RGMII.

2 Programming sequence

The following resets apply to the top-level wrapper:

- cc_rgmii_clk_ares
- cc_eth_aclk_ares
- cc_eth_hclk_ares
- cc_sdcc_m_clk_ares

During power-on reset, all clocks are off. During SW_RESET, clocks are enabled, reset is synchronized to the appropriate clock, and the timing should be closed to flip-flops.

2.1 Initialization, configuration, and control sequence

The EMAC must be initialized (including the interrupts from the external PHY) before it can become functional.

2.1.1 EMAC controller initialization/programming

Refer to the *Synopsys controller data book* for detailed initialization/programming.

2.1.2 Programming guidelines for flexible pulse-per-second (PPS) output

Refer to the *Synopsys controller data book* for details on the programming guidelines for pps.

PPS Programming Sequence

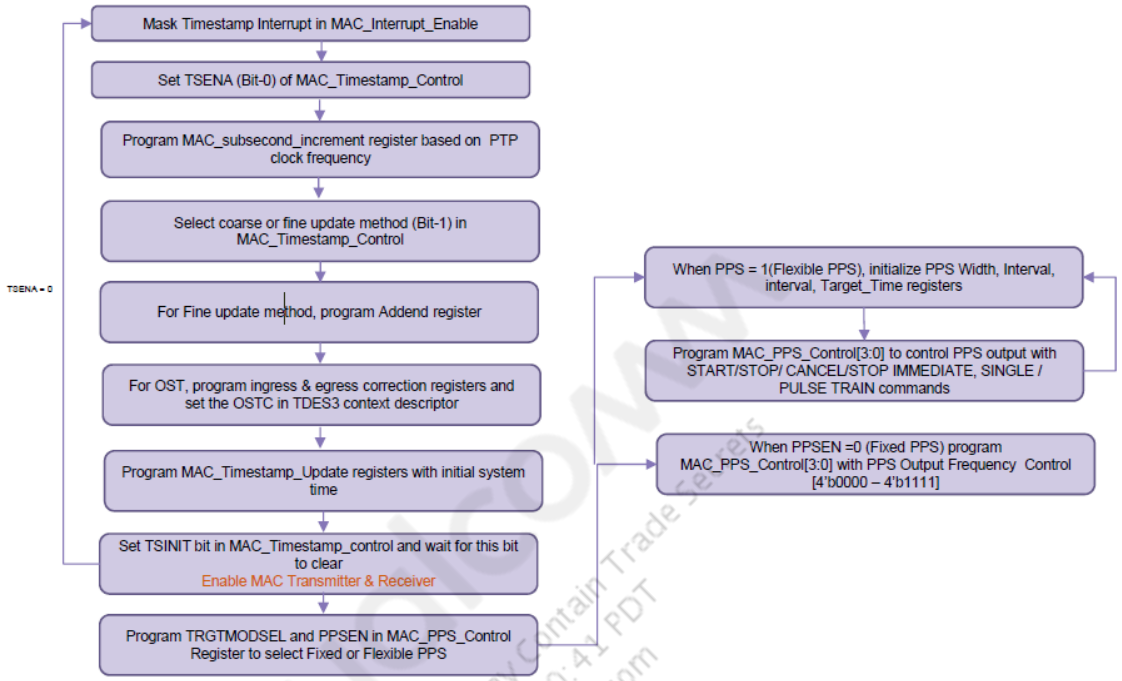

Figure 2-1 PPS programming sequence

This program values in the respective registers to achieve 19.2 MHz frequency output with 230.4 MHz PTP clock.

PTP_Clock	230.4
Expected PPS output frequency (MHz)	19.2
EMAC_MAC_SUB_SECOND_INCREMENT[MAC_SSINC] (digital to decimal)	4
EMAC_MAC_SUB_SECOND_INCREMENT [MAC_SNSINC] (digital -decimal)	76
MAC_PPS(#i)_Interval[PPSINT0]	11
MAC_PPS(#i)_Width[PPSWIDTH0]	5

where i = 0; i ≤ DWC_EQOS_PPS_OUT_NUM-1

This program values in the respective registers to achieve 9.6 MHz frequency output with 250 MHz PTP clock.

PTP_Clock	250
Expected PPS output frequency (MHz)	9.6
EMAC_MAC_SUB_SECOND_INCREMENT[MAC_SSINC] (digital to decimal)	4
EMAC_MAC_SUB_SECOND_INCREMENT [MAC_SNSINC] (digital -decimal)	0
MAC_PPS(#i)_Interval[PPSINT0]	25
MAC_PPS(#i)_Width[PPSWIDTH0]	12

Here are the calculations to get 9.6 MHz clock on PPS_O signal from 250MHz PTP Clock.

PTP clock = 250 MHz

MAC_SSINC = 4 ns [digital : 0x4, binary : 0x9]

To generate a PPS with output frequency of 9.6 MHz

Width = 52 ns, Interval = 104 ns

MAC_PPS_Width = (52/4) – 1 = 12

MAC_PPS_Interval = (104/4) – 1 = 25

2.2 RGMII IO macro initialization

DLL initialization is required irrespective of the interface (RGMII/RMII/MII) and speed selection as the RXC clock (comes from PHY) or TXC_FB (loopback TXC clock) propagates through the DLL and reaches the controller.

2.2.1 DLL initialization

1. Write 1 to DLL_RST bit of SDCC_HC_REG_DLL_CONFIG register.
2. Write 1 to PDN bit of SDCC_HC_REG_DLL_CONFIG register.
3. Write 0 to DLL_RST bit of SDCC_HC_REG_DLL_CONFIG register.
4. Write 0 to PDN bit of SDCC_HC_REG_DLL_CONFIG register.
5. Write 1 to DLL_EN
6. Write 1 to CK_OUT_EN
7. Wait until DLL_LOCK bit of SDC4_STATUS register is 1.

2.2.2 Putting DLL in the lowest power consumption mode

Write 1 to PDN bit of SDCC_HC_REG_DLL_CONFIG register.

2.2.3 DLL configuration

1. Set appropriate bit fields and write to SDCC_HC_REG_DLL_CONFIG register
   a. Set CDR_EN bit to 0.
   b. Set CDR_EXT_EN bit to 1.
   c. Set CK_OUT_EN bit to 0.
   d. Set DLL_EN bit to 1.
2. Wait until CK_OUT_EN bit of SDCC_HC_REG_DLL_CONFIG register is 0.
3. Set CK_OUT_EN bit of SDCC_HC_REG_DLL_CONFIG register to 1.
4. Wait until CK_OUT_EN bit of SDCC_HC_REG_DLL_CONFIG register is 1.
5. Write 1 to DDR_CAL_EN bit of SDCC_HC_REG_DLL_CONFIG_2 register.
6. Write 1 to PRG_DLY_EN bit of SDCC_HC_REG_DDR_CONFIG register.

2.2.4 SDCC_HC_REG_DDR_CONFIG

Set the correct delay for Rclk. This programming is for only RGMII –ID mode with 1000Mb/s mode.

For all other speeds, need to follow Bypass mode.

Example: -

SDCC_HC_REG_DDR_CONFIG [PRG_RCLK_DLY] = 69.

The rclk delay value is controlled by - SDCC_HC_REG_DDR_CONFIG [PRG_RCLK_DLY].

The calculation is done as follow:

$$\text{RCLK\_DLY} = \frac{\text{TCXO\_Period} \times \text{TCXO\_CYCLES\_CNT}}{2 \times (\text{PRG\_RCLK\_DLY})}$$

Assuming that TXCO is 19.2 MHz (Period = 52 ns) and TCXO_CYCLE_CNT is 4 (describe in bits [11:9] of SDCC_HC_REG_DDR_CONFIG).

To get delay of 1.5ns, PRG_RCLK_DLY should be set to 69:

$$\text{PRG}_{\text{RCLK}_{\text{DLY}}} = \frac{52\ \text{ns} \times 4}{2 \times (required\ delay)}$$

$$\text{PRG\_RCLK\_DLY} = \frac{104}{1.5}$$

$$\text{PRG\_RCLK\_DLY} = 69$$

S.No	RGMII ID mode – Delay	PRG_RCLK_DLY
1	1.5	69
2	1.6	65
3	1.7	61
4	1.8	60

2.2.5 DLL bypass mode programming

1. Write 1 to PDN bit of SDCC_HC_REG_DLL_CONFIG register.
2. Write 1 to bypass bit of SDCC_USR_CTL register. Default value of this register is 0x00010800.
3. Keep other register bits with the default values.

2.2.6 RGMII internal delay/external delay mode

2.2.6.1 RGMII ID mode (internal delay)

The GCC module must be programmed to generate 250 MHz/50 MHz/5 MHz (1 G/100 Mbps/10 Mbps) frequency on cc_eth_rgmii_clk clock signal. Data is driven at 90º phase shift of the clock.

Tx path

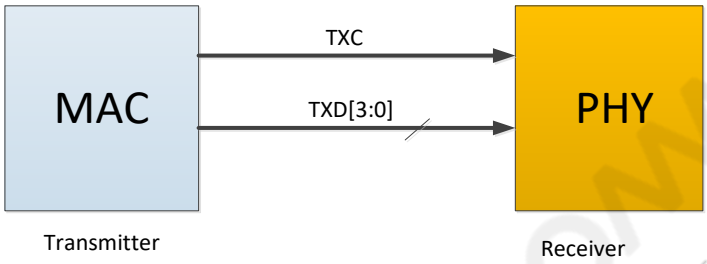

Figure 2-2 RGMII ID mode for Tx

In this mode, the clock will be delayed with data. This delay is inserted from the transmitter (MAC) itself. There is a 90° phase shift in the output clock. This introduces the required delay in the 1000/100/10 Mbps mode. So, the delayed clock will be transmitted from MAC to PHY.

For 1000Mb/s mode, this delay is 2 ns.

For 100 Mb/s mode, this delay is 10 ns.

For 10 Mb/s mode this delay is 100 ns.

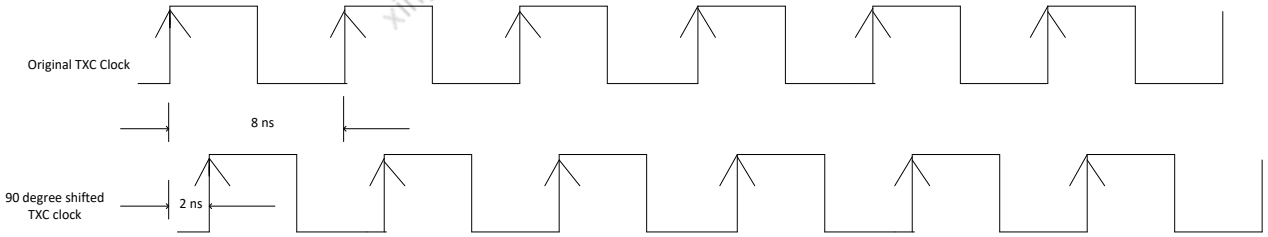

Figure 2-3 RGMII 90° shifted clock

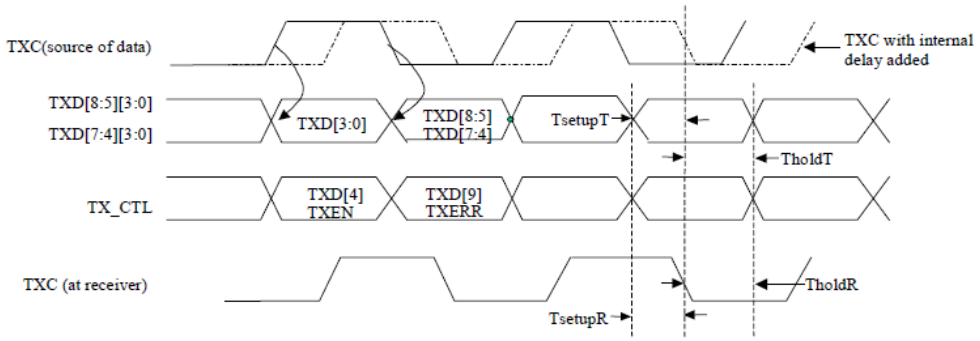

Figure 2-4 RGMII ID mode waveform for Tx

Rx path

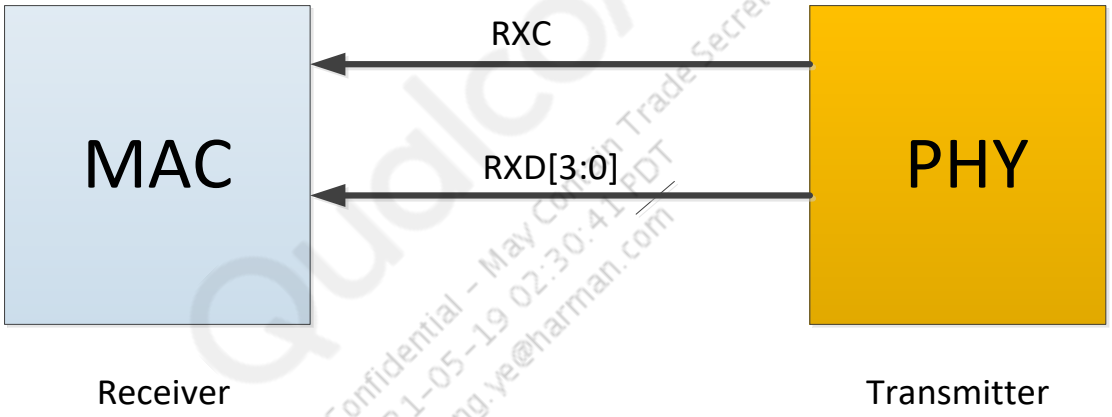

Figure 2-5 RGMII ID mode for Rx

In the RX path, the incoming clock will be delayed with respect to data. Transmitter(PHY) can introduce delay as per RGMII standard. Receiver(MAC) will receive delayed clock.

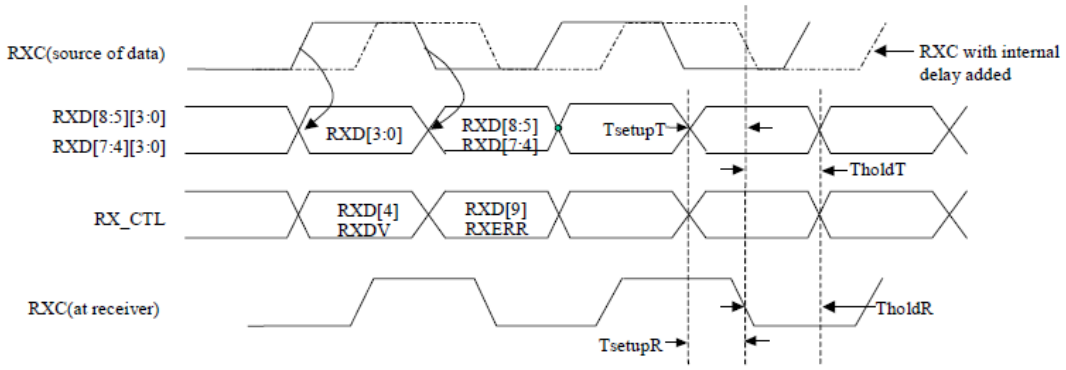

Figure 2-6 RGMII ID mode waveform for Rx

2.2.6.2 Non-ID mode (external delay)

The GCC module must be programmed to generate 250 MHz/25 MHz/2.5 MHz (1 G/100 Mbps/10 Mbps) frequency on the cc_eth_rgmii_clk clock signal. Data is synchronous to clock. The transmitter (MAC) will transmit the clock with the aligned data. There is no delay inserted from the MAC side.

Delay will be inserted in the middle of the board level between MAC and PHY. So, the receiver (PHY) will receive delayed clock TXC with respect to TXD [3:0].

Tx path

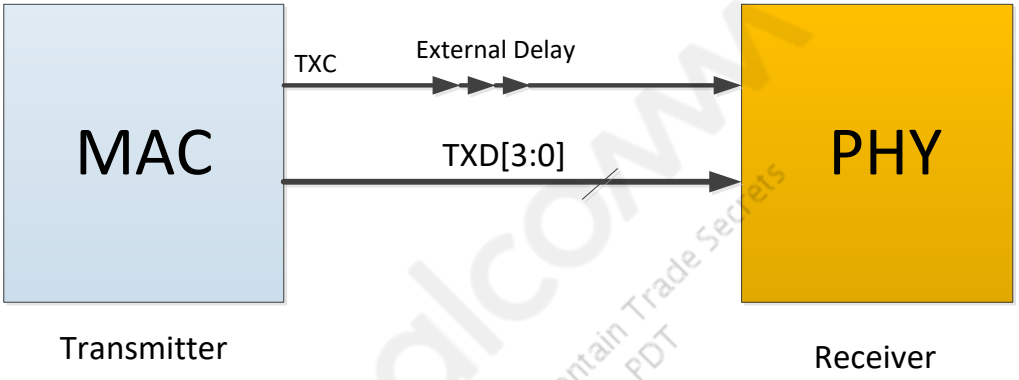

Figure 2-7 RGMII Non-ID mode for Tx

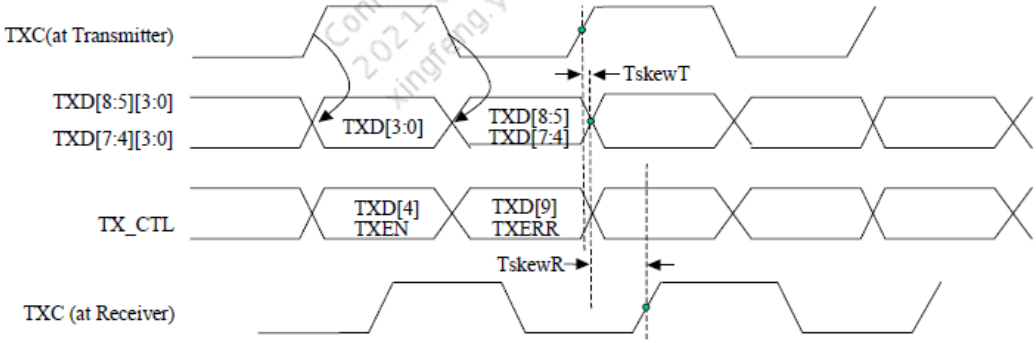

Figure 2-8 RGMII Non-ID mode waveform for TX

Rx path

The transmitter (PHY) will transmit clock with the aligned data. There is no delay inserted from the PHY side. Delay will be inserted in the middle of the board level between MAC and PHY. So, the receiver (MAC) will receive delayed clock RXC with respect to RXD [3:0].

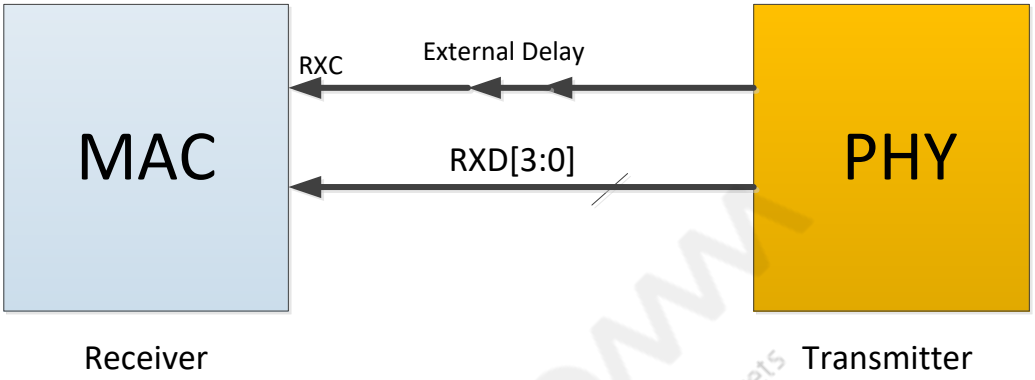

Figure 2-9 RGMII Non-ID mode for Rx

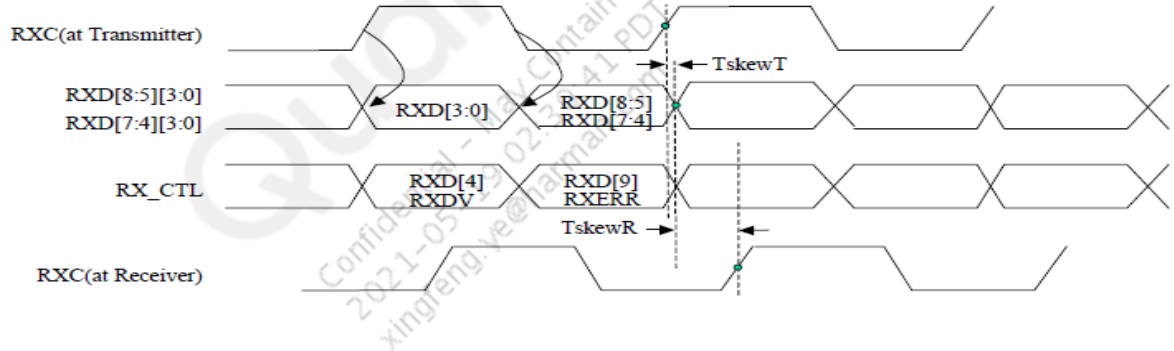

Figure 2-10 RGMII Non-ID mode waveform for Rx

2.2.7 Programming RGMII IO macro registers

Program the following fields with the appropriate value.

Write the func_clk_en bit as 1 in the RGMII_IO_MACRO_CONFIG register.

After enabling DLL, TX_TO_RX_LOOPBACK_EN bit in the RGMII_IO_MACRO_CONFIG_2 register should be set to 0.

2.2.7.1 RGMII – 1 Gbps

RGMII-ID mode Tx path

1. Write 1 to DDR_MODE bit of EMAC_RGMII_IO_MACRO_CONFIG register.
2. Write 0 to BYPASS_TX_ID_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register.
3. Write 1 to POS_NEG_DATA_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG register.
4. Write 1 to PROG_SWAP bit of EMAC_RGMII_IO_MACRO_CONFIG register.

5. Write 0 to DATA_DIVIDE_CLK_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
6. Write 1 to TX_CLK_PHASE_SHIFT_EN bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
7. Write 0 to RERVED_CONFIG_16[0] bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register

NOTE: This programming bit is renamed as RGMII_CLK_SEL_CFG for SA6155.

RGMII-ID mode Rx path

1. Write 0 for SA8155/1 for SA6155 to LOOPBACK_EN bit of EMAC_RGMII_IO_MACRO_CONFIG
2. If data arrives at positive edge or if clock is delayed by 1.5ns/ 2ns then write 1 to RX_PROG_SWAP bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
3. Write PRG_RCLK_DLY bits of register SDCC_HC_REG_DDR_CONFIG.

Bypass mode Tx path

1. Write 1 to DDR_MODE bit of EMAC_RGMII_IO_MACRO_CONFIG register.
2. Write 1 to BYPASS_TX_ID_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register.
3. Write 0 to POS_NEG_DATA_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG register.
4. Write 0 to PROG_SWAP bit of EMAC_RGMII_IO_MACRO_CONFIG register.
5. Write 0 to DATA_DIVIDE_CLK_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
6. Write 0 to TX_CLK_PHASE_SHIFT_EN bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
7. Write 0 to RERVED_CONFIG_16[0] bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register

NOTE: This programming bit is renamed as RGMII_CLK_SEL_CFG for SA6155.

Bypass mode Rx path

1. Write 0 for SA8155/1 for SA6155 to LOOPBACK_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register.
2. Write 0 to RX_PROG_SWAP bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
3. Program the DLL USER CTRL register.

2.2.7.2 RGMII – 100 Mbps

RGMII-ID mode Tx path

1. Write 1 to DDR_MODE bit of EMAC_RGMII_IO_MACRO_CONFIG register.
2. Write 1 to BYPASS_TX_ID_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register.

3. Write 0 to POS_NEG_DATA_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG register.
4. Write 0 to PROG_SWAP bit of EMAC_RGMII_IO_MACRO_CONFIG register.
5. Write 0 to DATA_DIVIDE_CLK_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
6. Write 1 to TX_CLK_PHASE_SHIFT_EN bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
7. Write 2'b01 to MAX_SPD_PRG_2 field of EMAC_RGMII_IO_MACRO_CONFIG register.
8. Write 0 to RERVED_CONFIG_16[0] bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.

NOTE: This programming bit is renamed as RGMII_CLK_SEL_CFG for SA6155.

RGMII-ID mode Rx path

1. Write 0 for SA8155/1 for SA6155 to LOOPBACK_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register.
2. Write 0 to RX_PROG_SWAP bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
3. Write 3'b101 to EXT_PRG_RCLK_DLY_CODE bit of SDCC_HC_REG_DDR_CONFIG
4. Write 6'b111111 to EXT_PRG_RCLK_DLY bit of SDCC_HC_REG_DDR_CONFIG
5. Write 1'b1 to EXT_PRG_RCLK_DLY_EN bit of SDCC_HC_REG_DDR_CONFIG

Bypass ID mode Tx path

1. Write 0 to DDR_MODE bit of EMAC_RGMII_IO_MACRO_CONFIG register.
2. Write 1 to BYPASS_TX_ID_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register.
3. Write 0 to POS_NEG_DATA_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG register.
4. Write 0 to PROG_SWAP bit of EMAC_RGMII_IO_MACRO_CONFIG register.
5. Write 0 to DATA_DIVIDE_CLK_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
6. Write 0 to TX_CLK_PHASE_SHIFT_EN bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
7. Write 2'b01 to MAX_SPD_PRG_2 field of EMAC_RGMII_IO_MACRO_CONFIG register.
8. Write 1 to RERVED_CONFIG_16[0] bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.

NOTE: This programming bit is renamed as RGMII_CLK_SEL_CFG for SA6155.

Bypass ID mode Rx path

1. Write 0 to RX_PROG_SWAP bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
2. Write 0 for SA8155/1 for SA6155 to LOOPBACK_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register.
3. Program DLL USER CTRL register.

2.2.7.3 RGMII – 10 Mbps

RGMII ID mode Tx path

1. Write 1 to DDR_MODE bit of EMAC_RGMII_IO_MACRO_CONFIG register.
2. Write 1 to BYPASS_TX_ID_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register.
3. Write 0 to POS_NEG_DATA_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG register.
4. Write 0 to PROG_SWAP bit of EMAC_RGMII_IO_MACRO_CONFIG register.
5. Write 0 to DATA_DIVIDE_CLK_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
6. Write 1 to TX_CLK_PHASE_SHIFT_EN bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.

   Write 0 for SA8155/1 for SA6155 to LOOPBACK_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register.
7. Write 19 to MAX_SPD_PRG_9 field of EMAC_RGMII_IO_MACRO_CONFIG register.
8. Write 0 to RERVED_CONFIG_16[0] bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register

NOTE: This programming bit is renamed as RGMII_CLK_SEL_CFG for SA6155.

RGMII ID mode Rx path

1. Write 0 to RX_PROG_SWAP bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
2. Write 3’b101 to EXT_PRG_RCLK_DLY_CODE bit of SDCC_HC_REG_DDR_CONFIG register.
3. Write 6’b111111 to EXT_PRG_RCLK_DLY bit of SDCC_HC_REG_DDR_CONFIG register.
4. Write 1’b1 to EXT_PRG_RCLK_DLY_EN bit of SDCC_HC_REG_DDR_CONFIG register.

Bypass ID mode Tx path

1. Write 0 to DDR_MODE bit of EMAC_RGMII_IO_MACRO_CONFIG register.
2. Write 1 to BYPASS_TX_ID_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register.
3. Write 0 to POS_NEG_DATA_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG register.
4. Write 0 to PROG_SWAP bit of EMAC_RGMII_IO_MACRO_CONFIG register.
5. Write 0 to DATA_DIVIDE_CLK_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
6. Write 0 to TX_CLK_PHASE_SHIFT_EN bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
7. Write 0 for SA8155/1 for SA6155 to LOOPBACK_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register.
8. Write 19 to MAX_SPD_PRG_9 field of EMAC_RGMII_IO_MACRO_CONFIG register.

9. Write 1 to RERVED_CONFIG_16[0] bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register

NOTE: This programming bit is renamed as RGMII_CLK_SEL_CFG for SA6155.

Bypass ID mode Rx path

1. Write 0 to RX_PROG_SWAP bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
2. Program DLL USER CTRL register.

RMII Tx transfer should happen in the falling edge of TXC. This is needed to meet the setup and hold time in the RMII mode. This RMII TXC clock needs to be looped back to the IO MACRO inside TXC pad as RXC clock is not available (comes from the external PHY) in RMII interface.

2.2.7.4 RMII mode – 100 Mbps

Bypass ID mode Tx path

1. Write 2'b01 to INTF_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG register.
2. Write 0 to DDR_MODE bit of EMAC_RGMII_IO_MACRO_CONFIG register.
3. Write 1 to BYPASS_TX_ID_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register.
4. Write 0 to POS_NEG_DATA_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG register for TXD toggling to align with falling edge of TXC or write 1 to align with rising edge of TXC.
5. Write 1 to PROG_SWAP bit of EMAC_RGMII_IO_MACRO_CONFIG register.
6. Write 1 to DATA_DIVIDE_CLK_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
7. Write 0 to TX_CLK_PHASE_SHIFT_EN bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
8. Write 1 to CLK_DIVIDE_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
9. Write 2'b01 to MAX_SPD_PRG_2 field of EMAC_RGMII_IO_MACRO_CONFIG register.
10. Write default to MAX_SPD_PRG_9 field of EMAC_RGMII_IO_MACRO_CONFIG register.
11. Write 0 to RERVED_CONFIG_16[0] bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.

NOTE: This programming bit is renamed as RGMII_CLK_SEL_CFG for SA6155.

Bypass ID mode Rx path

1. Write 1 for SA8155/0 for SA6155 to LOOPBACK_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register.
2. Program DLL USER CTRL register.
3. Write default to EXT_PRG_RCLK_DLY_CODE bit of SDCC_HC_REG_DDR_CONFIG register.

4. Write default to EXT_PRG_RCLK_DLY bit of SDCC_HC_REG_DDR_CONFIG register.
5. Write default to EXT_PRG_RCLK_DLY_EN bit of SDCC_HC_REG_DDR_CONFIG register.

2.2.7.5 RMII Mode – 10 Mbps

Bypass ID mode Tx path

1. Write 2'b01 to INTF_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG register.
2. Write 0 to DDR_MODE bit of EMAC_RGMII_IO_MACRO_CONFIG register.
3. Write 1 to BYPASS_TX_ID_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register.
4. Write 0 to POS_NEG_DATA_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG register for TXD toggling to align with falling edge of TXC or write 1 to align with rising edge of TXC.
5. Write 1 to PROG_SWAP bit of EMAC_RGMII_IO_MACRO_CONFIG register.
6. Write 1 to DATA_DIVIDE_CLK_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
7. Write 0 to TX_CLK_PHASE_SHIFT_EN bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
8. Write 1 to CLK_DIVIDE_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
9. Write 9'b19 to MAX_SPD_PRG_9 field of EMAC_RGMII_IO_MACRO_CONFIG register.
10. Write default to MAX_SPD_PRG_2 field of EMAC_RGMII_IO_MACRO_CONFIG register.
11. Write 0 to RERVED_CONFIG_16[0] bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register

NOTE: This programming bit is renamed as RGMII_CLK_SEL_CFG for SA6155.

Bypass ID mode Tx path Rx path

1. Write 1 for SA8155/0 for SA6155 to LOOPBACK_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register.
2. Program DLL USER CTRL register.
3. Write default to EXT_PRG_RCLK_DLY_CODE bit of SDCC_HC_REG_DDR_CONFIG register.
4. Write default to EXT_PRG_RCLK_DLY bit of SDCC_HC_REG_DDR_CONFIG register
5. Write default to EXT_PRG_RCLK_DLY_EN bit of SDCC_HC_REG_DDR_CONFIG register.

2.2.7.6 MII mode – 100/10 Mbps

1. Write 2'b10 to INTF_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG register.
2. Write 0 to DDR_MODE bit of EMAC_RGMII_IO_MACRO_CONFIG register.
3. Write 1 to BYPASS_TX_ID_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register.
4. Write 0 to POS_NEG_DATA_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG register.
5. Write 0 to PROG_SWAP bit of EMAC_RGMII_IO_MACRO_CONFIG register.
6. Write 1 to DATA_DIVIDE_CLK_SEL bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
7. Write 0 to TX_CLK_PHASE_SHIFT_EN bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.
8. Write 1 to RERVED_CONFIG_16[0] bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.

NOTE: This programming bit is renamed as RGMII_CLK_SEL_CFG for SA6155.

4. Program the DLL USER CTRL register.
5. Write 0 for SA8155/1 for SA6155 to LOOPBACK_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register. This setting ensures EMAC uses the RXC clock that comes from PHY to process the Rx data.

2.2.8 Enabling DLL loopback

NOTE: This DLL loopback is applicable to SA6155 only.

In the functional mode, the design drives the Rx path signals (RXC, RXD and RX_CTL) on DLL inputs. In the loopback mode, the design drives the Tx path signals (TXC_FB, TXD and TX_CTL) on DLL inputs. These Tx path signals are same as those that go from MAC to PHY..

1. Write to 1 DLL_TX_RX_LOOPBACK_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register.
2. Write 0 to LOOPBACK_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register.
3. Other programming bits are same as above for their respective mode and speed.

2.2.9 Enabling RGMII IO macro loopback

1. This loop back is internal to the IO macro and routes Tx path data/clock on to the Rx path.
2. Write to 1 TX_TO_RX_LOOPBACK bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register to enable IO macro loopback.

   RMII mode does not support this IO Macro loopback.

   Other programming bits are same as above for respective mode and speed except for the following bit: prog_swap bit must be set to 1 in both ID and non-ID modes of RGMII in 1G speed case.

2.2.10 Enabling controller MAC loopback

To enable this feature, set the LM bit of MAC_Configuration register. When this bit is set, MAC operates in a loopback mode. The Rx clock input (clk_rx_i) is required for the loopback to work properly. This is because of TX clock is not internally looped back.

Program the DLL USER CTRL register.

2.2.10.1 RGMII IO macro in RGMII and MII mode

1. Write 0 for SA8155/1 for SA6155 to LOOPBACK_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register.
2. Write to 1 TX_TO_RX_LOOPBACK bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.

2.2.10.2 RGMII IO macro in RMII mode

1. Write 1 for SA8155/0 for SA6155 to LOOPBACK_EN bit of EMAC_RGMII_IO_MACRO_CONFIG register.
2. Write to 0 TX_TO_RX_LOOPBACK bit of EMAC_RGMII_IO_MACRO_CONFIG_2 register.

2.2.11 Programming the receive programmable burst length (RxPBL)

These bits indicate the maximum number of bits to be transferred in one DMA data transfer. This is the maximum value that is used in a single block read or write. The DMA always attempts to burst as specified in the PBL each time it starts a burst transfer on the application bus. You can program the PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior.

To transfer more than 32 bits, these are the steps:

1. Set the PBLx8 mode in the DMA_CH0_Control register.
2. Set the PBL.

NOTE: The maximum value of RxPBL must be less than or equal to half the Rx Queue size (RQS field of MTL_RxQ[n]_Operation_Mode register) in terms of bits when Rx Queue is operated in the store and forward mode.

In the threshold mode, it must be programmed to minimum of the above mentioned condition and the programmed threshold value (RTC field of MTL_RxQ[n]_Operation_Mode register). This is required so that the application can start reading the packet when threshold numbers of packet bytes are received.

3 Power sequences

3.1 Power collapse

EMAC IP is used in automotive applications. By default, EMAC IP is kept under power collapsed mode. The software brings out the IP of the power collapsed mode when automotive applications are enabled.

RGMII IO macro is connected to the always ON (AON) domain. The EMAC wrapper block which includes the EMAC controller completely is collapsible. There is no retention logic present inside the EMAC IP. The software must reprogram the EMAC IP after power collapse.

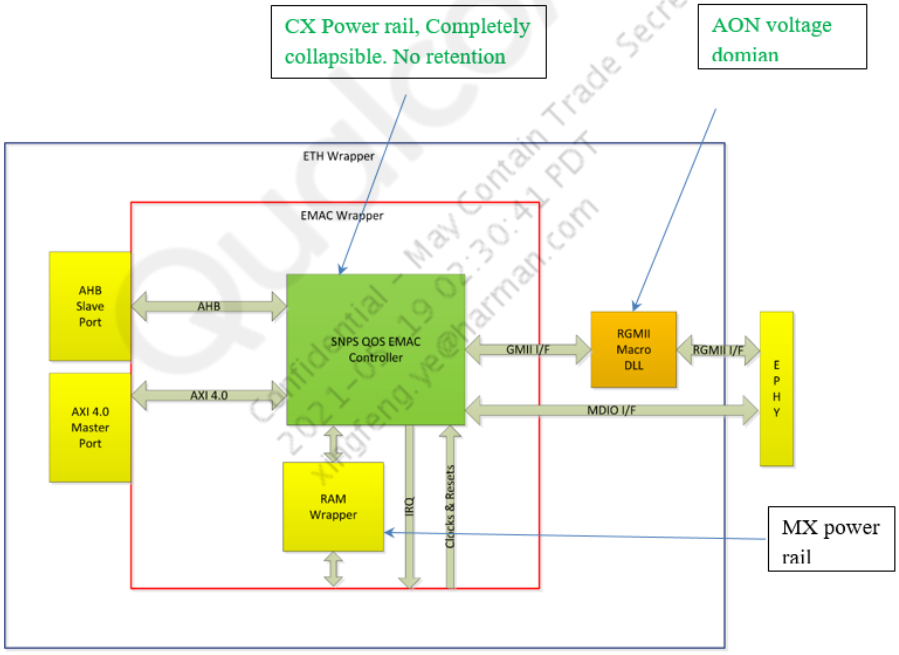

Figure 3-1 Power collapse

3.2 MDIO

To send an MDIO transaction (clause 22) the MAC initiates the management write or read operation with respect to the MDC clock. The MDC clock is a divided clock from the CSR clock. The divide factor depends on the clock range setting in the MAC_MDIO_Address register.

The MDC clock is selected as follows:

Selection	CSR Clock	MDC Clock
0000	60–100 MHz	CSR clock/42
0001	100–150 MHz	CSR clock/62
0010	20–35 MHz	CSR clock/16
0011	35–60 MHz	CSR clock/26
0100	150–250 MHz	CSR clock/102
0101	250–300 MHz	CSR clock/124
0110, 0111	Reserved	

The data exchange between the MAC and the PHY is performed through mdi_i, mdo_o, and mdo_oe signals. This signal group is passed through a three-state buffer and brought out as MDIO line connected to the PHY.

MDIO packet structure

Field	Description
IDLE	The mdio line is three-state; there is no clock on gmii_mdc_o.
PREAMBLE	32 continuous bits of value 1
START	Start of packet is 2'01
OPCODE	2'b10 for Read and 2'b01 for Write
PHY ADDR	5-bit address select for one of 32 PHYs
REG ADDR	Register address in the selected PHY
TA	Turnaround is 2'bZ0 for Read and 2'b10 for Write
DATA	Any 16-bit value. In a Write operation, the MAC drives mdio. In a Read operation, the PHY drives it.

3.2.1 GMII/MII management write operation

When you set bit[3:2] to 2'b01and bit 0 in the MAC_MDIO_Address register, the MAC CSR module transfers the PHY address, the register address in PHY, and the write data (MAC_MDIO_Data register) to the SMA to initiate a write operation into the PHY registers. At this point, the SMA module starts a write operation on the GMII management interface using the management packet format specified in the GMII specifications (as per *IEEE 802.3-2002, Section 22.2.4.5*).

When the SMA module starts a write operation, the write data packet is transmitted on the MDIO line. The MAC drives the MDIO line for complete duration of the packet. The Busy bit is set high until the write operation is complete. The CSR ignores the write operations performed to the MAC_MDIO_Address register or the MAC_MDIO_Data register during this period (the Busy bit

is high). When the write operation is complete, the SMA module indicates this to the CSR, and the CSR resets the Busy bit.

3.2.2 GMII/MII management read operation

When you set bit[3:2] to 2'b11 and bit 0 in the MAC_MDIO_Address register, the MAC CSR module transfers the PHY address and the register address in PHY to the SMA to initiate a read operation in the PHY registers. At this point, the SMA module starts a read operation on the GMII management interface using the management packet format specified in the GMII specifications (as per *IEEE 802.3-2002, Section 22.2.4.5*).

When the SMA module starts a read operation on the MDIO, the CSR ignores the write operations to the MAC_MDIO_Address or MAC_MDIO_Data register during this period (the Busy bit is high) and the transaction is completed without any error on the MCI interface. When the read operation is complete, the SMA indicates this to the CSR. The CSR resets the Busy bit and updates the MAC_MDIO_Data register with the data read from the PHY.

The MAC drives the MDIO line for the complete duration of the frame except during the Data fields when the PHY is driving the MDIO line. For more information about the communication from the application to the PHYs, see the *Reconciliation Sublayer and Media Independent Interface Specifications sections of the IEEE 802.3z, 1000BASE Ethernet.*