Contents

Figures

Tables

4 System Control

4.1 SC

4.1.1 CRG

The clock and reset generator (CRG) is provides clocks and reset for each IP. This section describes only the AO CRG and PERIPH CRG functions. The functions of other CRGs are described in the clock reset section of the corresponding modules.

4.1.1.1 AOCRG

The AO CRG has the following functions:

- Provides clock reset for each IP in the system area.
- Implements software control and state query for the clock reset and configures the clock divider for each IP in the system area.
- Supports automatic frequency scaling of the always-on network on chip (AONOC).

4.1.1.2 PERICRG

The PERIPH CRG has the following functions:

- Accesses registers over the advanced peripheral bus (APB) interface.
- Isolates the securely-accessed register area from the non-securely-accessed register area.
- Provides clock reset for each IP and the NOC bus in the peripheral area.
- Supports software control and state query for the clock reset of each IP and the NOC bus in the peripheral area.
- Configures the clock divider for each IP and the NOC bus in the peripheral area.
- Provides the software configuration interfaces and state query interfaces for the A73_B, A53_L, LPM3, CCI500, ADB, and CoreSight modules.
- Supports the anti-suspension function of the APB and advanced high-performance bus (AHB) interfaces.
- Supports enable control for timer clocks and watchdog clocks in the peripheral area.

  - Enables the counting frequency to be independent of the system clock frequency. If the system clock changes, the counter maintains the original counting frequency.
  - Samples the input counting clocks to generate clock enable signals, and outputs the signals to the timers 4–7 in the peripheral area.
  - Selects the counting clock source for the timers in the peripheral area (32 kHz or 4.8 MHz).
  - Forcibly pulls the timers in the peripheral area up through software configuration. In this case, the operating clock of the timer is the bus clock.
  - Forcibly pulls the watchdog in the peripheral area up through software configuration. In this case, the operating clock of the watchdog is the bus clock.

    If the watchdog in the peripheral area is not forcibly pulled up, the counting clock for the watchdog in the peripheral area is 32 kHz.
- Controls the power-on/power-off state machine of the A73_B and A53_L cores.
  - Allows the software to enable or disable the state machine used to power on and power off A73_B and A53_L cores by configuring registers.
  - Allows the software to configure whether to report the A73_B/A53_L power-on/power-off completion interrupt and query the interrupt state when the state machine is enabled.
  - Responds to the generic interrupt controller (GIC) interrupt and powers on the corresponding cores of A73_B and A53_L cores when the state machine is enabled. Whether to respond to the GIC interrupt can be separately configured for each core of A73_B and A53_L.
  - Allows the software to independently power on each core of A73_B and A53_L by writing to corresponding registers when the power-on/power-off state machine is enabled.
  - Allows the software to independently power off each core of A73_B and A53_L by writing to corresponding registers when the power-on/power-off state machine is enabled.
  - Queries the state of the power-on/power-off state machine for each core of A73_B and A53_L in real time when the state machine is enabled.
  - Independently configures the reset time, reset deassertion time, MTCMOS power-on wait time, ISO wait time, ISO deassertion wait time, DBG configuration wait time during the power-on/power-off process of each core of A73_B and A53_L.
- Controls the A73_B ocldo state machine.
  - Allows the state machine to respond to the independent ocldo request signal of starting the state machine for the four big cores.
  - Allows the state machine to control the mtcmos control signal of the four big cores.
  - Allows the state machine to control the ISO clamping signal of the four big cores. This function can be statically bypassed.
  - Allows the state machine to control the ocldo enable signal of the four big cores.
  - Allows the software to query the state of the state machine.
- Supports automatic gating and frequency scaling for the double data rate (DDR) clocks.

4.1.2 SCTRL

The system controller (SC) provides system control interfaces and consists of the following modules:

- APB interface module
- System main control module
- Interrupt response mode request module
- Power-on/Power-off control module
- Universal counter module
- Crystal oscillator control module
- Low-power random access memory (LPRAM) low-power control module
- eFUSE control module

The SC has the following features:

- Bus interface

  The bus interface complies with the advanced microcontroller bus architecture 2.0 (AMBA 2.0) APB Slave Interface Specifications, and supports 32-bit data width and 12-bit address width.
- Reset
  - The SC supports preset_n reset.
  - Global soft reset is supported, and the global soft reset request is sent to the CRG.

4.1.3 PCTRL

The peripheral controller (PCTRL) provides configuration interfaces for some external IPs. The PCTRL takes effect only after the system enters the normal mode. Therefore, ensure that the system is in normal mode before using any PCTRL function.

The PCTRL has the following functions:

- Supports 3D LCD raster control.
- Provides configuration interfaces for some peripheral IPs.
- Supports resource locks.

4.2 RTC

4.2.1 Function Description

The real-time clock (RTC) is used to display the time in real time and periodically generate alarms. The Hi3660 provides three RTCs: RTC0 to RTC2.

The RTC works based on a 32-bit up counter. The initial count value is loaded from the RTCLR register. The count value is incremented by 1 on the rising edge of each counting clock. When the count value of RTCDR is the same as the value of RTCMR, the RTC generates an interrupt. Then the counter continues counting in incremental mode on the next rising edge of the counting clock.

The RTC uses a 1 Hz counting clock to convert the count value into a value in the format of year, month, day, hour, minute, and second.

The RTC has the following features:

- Allows you to configure the initial count value.
- Allows you to configure the match value.

- Generates the timeout interrupt.
- Supports soft reset.

4.2.2 Register Description

See the SP804 manual of the ARM timer IP (http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0270b/index.html).

4.3 Watchdog

4.3.1 Function Description

The watchdog is used to send a reset signal to reset the entire system within a period after exceptions occurs in the system. The Hi3660 provides two watchdogs: WD0 and WD1.

The watchdog has the following features:

- Supports a 32-bit down counter. The counting clock is configurable.
- Supports configurable timeout interval (initial count value).
- Locks registers to prevent them from being modified incorrectly.
- Generates timeout interrupts.
- Generates reset signals.
- Supports the debugging mode.

The watchdog works based on a 32-bit down counter. The initial count value is loaded from the WdogLoad register. When the watchdog clock is enabled, the count value is decremented by 1 on the rising edge of each counting clock. When the count value is decremented to 0, the watchdog generates an interrupt. On the next rising edge of the counting clock, the counter reloads the initial count value from WdogLoad, and then continues counting decreasingly.

If the count value is decremented to 0 for the second time and the CPU does not clear the watchdog interrupt, the watchdog sends a reset signal and the counter stops counting.

You can determine whether to allow the watchdog to generate interrupts and reset signals by configuring WdogControl.

- When interrupt generation is disabled, the counter stops counting.
- When interrupt generation is enabled again, the watchdog counter counts from the configured value of WdogLoad instead of the last count value. The initial count value can be reloaded before an interrupt is generated.

The counting clock of the watchdog can be a sleep clock or a bus clock, which provides different count time ranges.

You can disable the operation of writing to the watchdog registers by configuring WdogLock.

- Writing 0x1ACC_E551 to WdogLock enables the write permission for all watchdog registers.
- Writing any other value to WdogLock disables the write permission for all watchdog registers (except WdogLock).

The write permission control feature prevents watchdog registers from being incorrectly modified by the software. In this way, the watchdog operation will not be incorrectly terminated by the software when exceptions occur.

In ARM debugging mode, the watchdog is automatically disabled to avoid intervention to normal debugging operations.

CAUTION

The watchdog must be disabled before the system enters the sleep mode.

4.3.2 Register Description

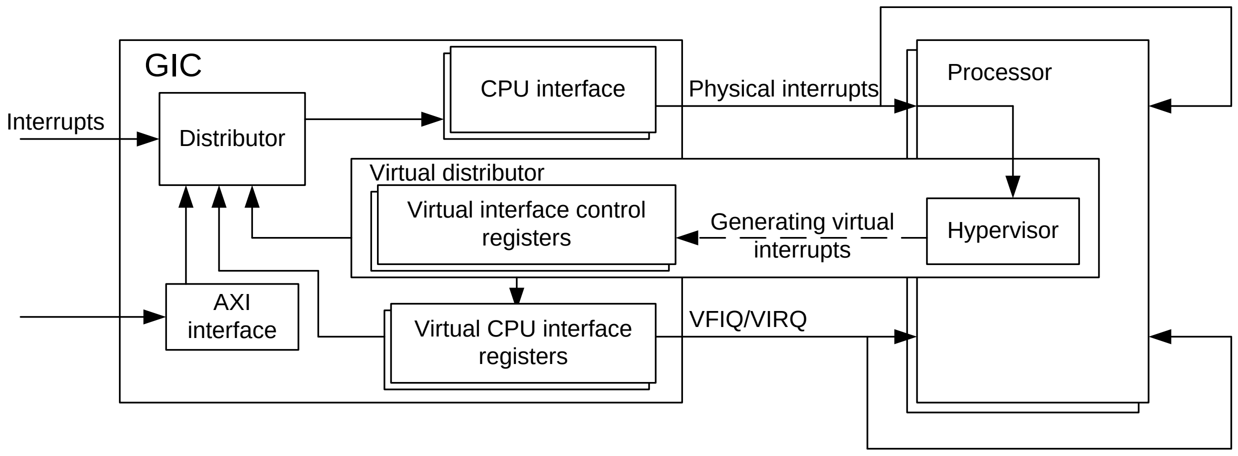
See the SP805*x* manual of the ARM watchdog IP (http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0270b/index.html).

4.4 GIC

4.4.1 Function Description

The GIC is used to detect, manage, and allocate external interrupts, internal interrupts, and software-generated interrupts (SGIs). The GIC supports security extension and virtualization extension.

Figure 4-1 GIC architecture



The GIC has the following features:

- Supports three types of interrupts: SGIs, private peripheral interrupts (PPIs), and shared peripheral interrupts (SPIs).
- Supports two interrupt trigger modes: level-sensitive mode and edge-triggered mode. The trigger mode of each interrupt source can be separately configured.

- Provides an interface for each CPU core to process the physical fast interrupt request (FIQ) and interrupt request (IRQ).
- Supports virtualization extension and virtualizes only the physical IRQs; generates the virtual FIQ (VFIQ) and virtual IRQ (VIRQ) through configuration and adds an interface for each CPU core to process the VFIQ and VIRQ.
- Supports interrupt routing. The physical interrupts are routed to the corresponding CPU core by querying the target processor register. The Hypervisor configures the target processor register based on the virtual machine ID (VMID) information. The virtual interrupts are routed to the corresponding CPU core based on the configured route target.
- Separately enables interrupts.
- Queries the interrupt states.
- Masks interrupts and configures the Mask Priority. An interrupt is reported only when its priority is greater than Mask Priority.
- Supports interrupt nesting.
  - For physical interrupts, the depth of interrupt nesting is determined by the Group Priority. Interrupt nesting is triggered only when the Group Priority is a large value. The number of Group Priority levels is configurable.
  - For virtual interrupts, the depth of interrupt nesting is 32.
- Provides the following for each CPU core:
  - Six PPIs of internal components. The PPIs of one core are independent of those of the other cores.
  - One internal virtual maintenance PPI
  - One BypassFIQ input and one BypassIRQ input
- Supports TrustZone extension. The input interrupts can be classified into the Secure group or the Non-Secure group. A smaller interrupt priority value indicates a higher interrupt priority.

  For the physical interrupts, the priority of all the Secure interrupts is higher than that of Non-Secure interrupts.
  - In Secure mode, the interrupt priority field is at least 5 bits, and can be extended to at most 8 bits.
  - In Non-Secure mode, the interrupt priority field is at least 4 bits, and can be extended to at most 7 bits.

  For the virtual interrupts, the number of priority levels is fixed at 32.
- Locks SPIs and support 32 lockable shared peripheral interrupts (LSPIs). The LSPIs must be Secure interrupts, and their configuration registers cannot be modified after these interrupts are locked.
- Supports interrupt wakeup events and reserves the nFIQOUT signal. Directly output over the port, the nIRQOUT signal can report interrupts as wakeup events when the GIC is disabled.

4.4.2 Register Description

See the ARM GIC-400 manual (http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0270b/index.html).

4.5 DMAC

4.5.1 Overview

The advanced eXtensible interface (AXI) direct memory access controller (DMAC) is a low-cost and high-performance IP targeted for the system-on-chip (SoC) chip. It is based on the AMBA AXI protocol. Direct memory access (DMA) is an operating mode in which data transfer is completely implemented by the hardware. In DMA mode, the DMAC transfers data using a master bus controller that is independent of the processor. Data is transferred between memories, or between a memory and a peripheral. The DMA mode applies to data block transfer at a high speed. After receiving a DMA request, the DMAC enables the master bus controller based on the CPU settings and transmits address and read/write control signals to memories or peripherals. The DMAC also counts the number of data transfers and reports data transfer completion to the CPU in interrupt mode.

The DMAC of the Hi3660 has the following features:

- A total of 16 logical channels
- Programmable logical channel priority
- An AXI master interface
- An APB 2.0 configuration interface
- A total of 32 sets of peripheral requests (Each set of peripheral request includes single requests and burst requests.)
- Data transfer from a memory to another memory, from a memory to a peripheral, and from a peripheral to a memory
- DMAC flow control when data is transferred between memories and peripherals
- Linked-list data transfer
- Link transfer of the logical channels
- One-dimensional, two-dimensional, and three-dimensional data transfer modes
- Four sets of interrupts
- Consistency operations and non-consistency operations
- Secure transfer, non-secure transfer, and interrupt reporting

4.5.2 Application Background

The DMA is an operating mode in which I/O switching is fully implemented by the hardware. In DMA mode, the DMAC directly transfers data between a memory and a peripheral, or between memories. This avoids the processor intervention and reduces the interrupt processing overhead of the processor.

The DMA mode applies to data block transfer at a high speed. After receiving a DMA transfer request, the DMAC enables the master bus controller based on the CPU settings and transmits address and read/write control signals to memories or peripherals. The DMAC also counts the number of data transfers and reports data transfer completion or errors to the CPU in interrupt mode.

The DMAC improves the system performance from the following aspects:

- The DMAC implements data transfer, reducing the CPU load.
- Channels are allocated to the I/O peripherals based on DMAC programming. As a result, large-amount data is transferred and the interrupt reporting frequency is reduced.

4.5.3 Function Description

The DMAC consists of the following components:

- A DMAC hardware request interface
- A total of 16 channels
- An arbiter
- An AXI master interface
- An APB slave interface

For a DMA data transfer, a logical channel of the DMAC must be triggered first. Then the master interface reads data from the source address and writes it to the destination address.

DMAC hardware request I/F is used by peripherals to request the DMAC to transfer data.

The following describes a data transfer triggered by the software:

The processor configures a logical channel through the APB slave interface, and specifies the source address, destination address, and transfer mode. Then the DMAC is triggered to perform the transfer operation. After the arbiter arbitrates channels, the DMAC reads data from the source address from the AXI master interface and writes data to the destination address based on the configured information. In this way, an interrupt is generated.

4.5.4 Application Description

4.5.4.1 Initialization Configuration

To configure the DMAC during system initialization, perform the following steps:

Step 1 Configure the channel parameter registers based on the required transfer, including CX_AXI_SPECIAL, CX_SRC_ADDR, CX_DES_ADDR, CX_LLI, CX_CNT0, CX_CNT1, CX_BINDX, and CX_CINDX.

For a one-dimensional non-linked-list transfer, if the AXI cache operation is not required, configure only CX_SRC_ADDR, CX_DES_ADDR, and CX_CNT0. This reduces the time of configuring registers.

For a one-dimensional non-linked-list transfer, configure only CX_SRC_ADDR, CX_DES_ADDR, CX_CNT0, and CX_CONFIG. This reduces the time of configuring registers. Ensure that other registers are set to default values.

Step 2 Specify the transfer mode, peripheral ID, and bus operation parameters by configuring CX_CONFIG.

Step 3 Set CX_CONFIG bit[0] to 1 to enable a channel to start the DMAC transfer.

----End

The sequence of step 1 and step 2 is exchangeable. Ensure that CX_CONFIG bit[0] is set to 1 to start the transfer.

4.5.4.2 Typical Configuration Processes

Memory Transfer

The following describes the one-dimensional INCR memory transfer(32-bit burst size and 8-bit burst length):

Step 1 Enable the clocks and deassert reset.

Step 2 Configure the interrupt mask register to enable all the interrupts.

Step 3 Determines the logical channel used for the transfer. The registers of the selected channel are configured in the following steps.

Step 4 Configure the channel parameter registers based on the required transfer, including CX_AXI_SPECIAL, CX_SRC_ADDR, CX_DES_ADDR, CX_LLI, and CX_CNT0.

For a one-dimensional non-linked-list transfer, if the AXI cache operation is not required, configure only CX_SRC_ADDR, CX_DES_ADDR, and CX_CNT0. This reduces the time of configuring registers. That is, configure the source address, destination address, and amount of the data to be transferred.

If there are requirements on the security attribute, configure CX_AXI_SPECIAL. Otherwise, the secure channel is used by default.

For a one-dimensional non-linked-list transfer, configure only CX_SRC_ADDR, CX_DES_ADDR, CX_CNT0, and CX_CONFIG. This reduces the time of configuring registers. Ensure that other registers are set to default values.

Step 5 Configure CX_CONFIG to specify the transfer mode and bus operation parameters. The peripheral ID is not required for memory transfer. The flow control mode must be set to "00: transfer between memories, DMAC flow control".

1. Set CX_CONFIG bit[31] and bit[30] to 1 to configure the operations of transferring the source address and destination address in DMA mode as INCR operations.
2. Set CX_CONFIG bit[27:24] and bit[23:20] to 0111 to configure the burst length for transferring the source address and destination address in DMA mode as 8.
3. Set CX_CONFIG bit[18:16] and bit[14:12] to 010 to configure the data width for transferring the source address and destination address in DMA mode as 32 bits.
4. Set CX_CONFIG bit[3:2] to 00 to configure the flow control mode of the DMA transfer as DMA flow control for memory transfer.

Step 6 Set CX_CONFIG bit[0] to 1 to enable a channel to start the DMAC transfer.

----End

Peripheral Transfer

The following describes the one-dimensional transfer from the memory to the peripheral (DMA flow control, 32-bit burst size, and 8-bit burst length):

Step 1 Repeat Step 1 and Step 2 in "Memory Transfer."

Step 2 Determines the logical channel used for the transfer. The registers of the selected channel are configured in the following steps.

Step 3 Configure the channel parameter registers based on the required transfer, including CX_AXI_SPECIAL, CX_SRC_ADDR, CX_DES_ADDR, CX_LLI, and CX_CNT0.

For a one-dimensional non-linked-list transfer, if the AXI cache operation is not required, configure only CX_SRC_ADDR, CX_DES_ADDR, and CX_CNT0 for the consistency operation. This reduces the time of configuring registers. That is, configure the source address, destination address, and amount of the data to be transferred.

If there are requirements on the security attribute, configure CX_AXI_SPECIAL. Otherwise, the secure channel is used by default.

For a one-dimensional non-linked-list transfer, configure only CX_SRC_ADDR, CX_DES_ADDR, CX_CNT0, and CX_CONFIG. This reduces the time of configuring registers. Ensure that other registers are set to default values.

Step 4 Configure CX_CONFIG to specify the transfer mode, peripheral ID, and bus operation parameters.

Step 5 Set CX_CONFIG bit[31] to 1 and bit[30] to 0 to configure the operation of transferring the source address in DMA mode as an INCR operation and the operation of transferring the destination address in DMA mode as an FIX operation because a peripheral is connected.

Set CX_CONFIG bit[27:24] and bit[23:20] to 0111 to configure the burst length for transferring the source address and destination address in DMA mode as 8.

Set CX_CONFIG bit[18:16] and bit[14:12] to 010 to configure the data width for transferring the source address and destination address in DMA mode as 32 bits.

Set CX_CONFIG bit[9:4] to the ID of the selected peripheral. The peripheral ID is unique to the corresponding peripheral and is allocated in advance.

Set CX_CONFIG bit[3:2] to 01 to configure the flow control mode of the DMA transfer as DMA flow control for peripheral transfer.

Step 6 Set CX_CONFIG bit[0] to 1 to enable a channel to start the DMAC transfer.

----End

4.5.5 Register Description

The base address of AP DMAC registers is 0xFDF30000.

Table 4-1 DMAC registers.

Offset	Register	Register Description
0x0000 + 0x40 x *in*	INT_STAT	Interrupt status register of processor *X*
0x0004 + 0x40 x *in*	INT_TC1	Channel transfer completion interrupt status register of processor *X*
0x0008 + 0x40 x *in*	INT_TC2	Linked list node transfer completion interrupt status register of processor *X*
0x000C + 0x40 x *in*	INT_ERR1	Configuration error interrupt status register of processor *X*
0x0010 + 0x40 x *in*	INT_ERR2	Data transfer error interrupt status register of processor *X*
0x0014 + 0x40 x *in*	INT_ERR3	Linked list read error interrupt status register of processor *X*

Offset	**Register**	**Register Description**
0x0018 + 0x40 x *in*	INT_TC1_MASK	Channel transfer completion interrupt mask register of processor *X*
0x001C + 0x40 x *in*	INT_TC2_MASK	Linked list node transfer completion interrupt mask register of processor *X*
0x0020 + 0x40 x *in*	INT_ERR1_MASK	Configuration error interrupt mask register of processor *X*
0x0024 + 0x40 x *in*	INT_ERR2_MASK	Data transfer error interrupt mask register of processor *X*
0x0028 + 0x40 x *in*	INT_ERR3_MASK	Linked list read error interrupt mask register of processor *X*
0x0600	INT_TC1_RAW	Raw channel transfer completion interrupt status register
0x0608	INT_TC2_RAW	Raw linked list node transfer completion interrupt status register
0x0610	INT_ERR1_RAW	Raw configuration error interrupt status register
0x0618	INT_ERR2_RAW	Raw data transfer error interrupt status register
0x0620	INT_ERR3_RAW	Raw linked list read error interrupt status register
0x660	SREQ	Single transfer request register
0x664	LSREQ	Last single transfer request register
0x668	BREQ	Burst transfer request register
0x66C	LBREQ	Last burst transfer request register
0x670	FREQ	Bulk transfer request register
0x674	LFREQ	Last bulk transfer request register
0x688	CH_PRI	Priority control register
0x690	CH_STAT	Global DMA status register
0x0694	SEC_CTRL	DMA global security control register
0x0698	DMA_CTRL	DMA global control register
0x0700 + 0x10 x *cn*	CX_CURR_CNT1	Remaining size status register for the three-dimensional transfer of channel *X*
0x0704 + 0x10 x *cn*	CX_CURR_CNT0	Remaining size status register for the one-dimensional and two-dimensional transfer of channel *X*
0x0708 + 0x10 x *cn*	CX_CURR_SRC_ADDR	Source address register of channel *X*

Offset	**Register**	**Register Description**
0x070C + 0x10 x *cn*	CX_CURR_DES_ADDR	Destination address register of channel *X*
0x0800 + 0x40 x *cn*	CX_LLI	Linked list address register of channel *X*
0x0804 + 0x40 x *cn*	CX_BINDX	Two-dimensional address offset configuration register of channel *X*
0x0808 + 0x40 x *cn*	CX_CINDX	Three-dimensional address offset configuration register of channel *X*
0x080C + 0x40 x *cn*	CX_CNT1	Transfer length 1 configuration register of channel *X*
0x0810 + 0x40 x *cn*	CX_CNT0	Transfer length configuration register of channel *X*
0x0814 + 0x40 x *cn*	CX_SRC_ADDR	Source address register of channel *X*
0x0818 + 0x40 x *cn*	CX_DES_ADDR	Destination address register of channel *X*
0x081C + 0x40 x *cn*	CX_CONFIG	Configuration register of channel *X*
0x0820 + 0x40 x *cn*	CX_AXI_CONF	AXI special operation configuration register of channel *X*