Programming ARM TrustZone Architecture on the Xilinx Zynq-7000 All Programmable SoC

User Guide

UG1019 (v1.0) May 6, 2014

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
05/06/2014	1.0	Initial Xilinx release.

Table of Contents

Programming ARM TrustZone Architecture on the Xilinx Zynq-7000 All Programmable SoC

Introduction to ARM TrustZone Architecture

ARM TrustZone® architecture provides a solution that is able to "carve out" or segregate a hardware subset of the full System on a Chip (SoC).

It does this by defining processors, peripherals, memory addresses and even areas of L2 cache to run as secure or non-secure hardware.

An SoC that utilizes ARM TrustZone technology has the ability to dynamically, with only a few clock cycles delay, expose the full SoC to secure software, or to expose a subset of that SoC to normal software (see Figure 1).


Full SoC
(All HW)

HW Accessible to "Secure World" software
(All HW)

HW Accessible to "Normal World" software
(HW subset)
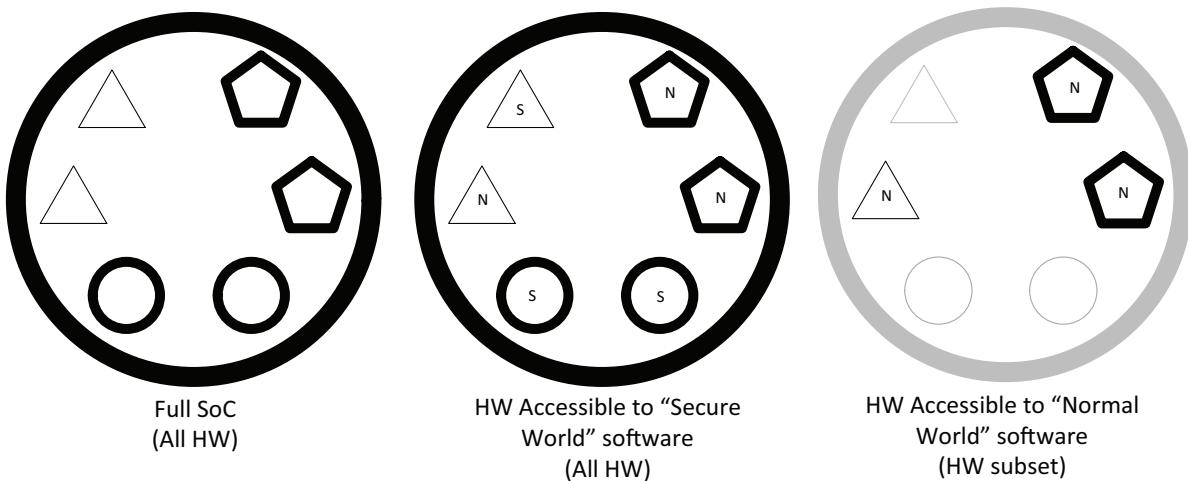
UG1019_01_082213

Figure 1: **ARM TrustZone Technology**

Normal World

The "normal world" (non-secure) that is created and enforced by TrustZone is typically a statically defined hardware subset of the SoC. TrustZone ensures that a non-secure processor or master can access only non-secure (NS) devices and receive only non-secure interrupts. For example, a normal world hardware subset might include the UART, Ethernet, and USB interfaces, but exclude CAN controller access. The CAN might instead be dedicated to the secure world where a separate RTOS or application runs for the sole purpose of managing CAN traffic, independent of the normal world software stack.

Software that runs in the normal world is assumed to be flawed from a safety and security perspective. That is, this normal world software is expected to contain bugs, exploits, hacks, faults, or irregularities that could expose sensitive information or functions. It is these assumptions that drive the value of TrustZone's ability to isolate the adjacent processing area (secure world) where sensitive data storage or functions are managed from the (flawed) normal world software.

Secure World

Unlike the hardware subset in which normal world software runs, software running within the secure world has complete access to the Zynq®-7000 All Programmable SoC (AP SoC) hardware. Thus, from the perspective of the secure software's execution, the system looks and acts nearly identical to what would be seen on a processor that does not have TrustZone. This means that secure software has access to all resources associated with both the secure and normal worlds.

Trusted software is the term chosen to represent software that runs within the secure world.

ARM TrustZone architecture contributes to the overall *system* security by preventing normal world software from accessing the secure world resources. It is important to understand also that TrustZone does little to improve the safety or security of the software that runs in the secure world except to prevent unwanted secure world access by normal world software. Therefore, it is the developer who determines that software is *trusted*, typically through rigorous development processes, testing and certification.

Because a trusted system can only be created when the software is known to be error-free and without exploits (verified as trusted software), and because smaller works of code are significantly easier to certify for safety and security needs, the trusted software typically includes only the minimum of functionality and device interfaces. If for example, trusted software does not include a TCP/IP stack, or a USB device, then security from such attacks is ensured based on the absence of such device drivers. absence of potentially exploitable software. Reference ARM's block diagrams found on the "System Architecture" tab on ARM's TrustZone website to view the types of applications that ARM envisages running from within the secure world.

Due to this assumption around the fundamental integrity of the trusted software, ARM does not intend TrustZone to support a general purpose operating system within the secure

world. The challenge of certifying any standard operating system (100K, or millions of lines of code) running as trusted software would be too great.

Trusted Execution Environment

A "trusted execution environment" (TEE) refers to software a stack running within the secure world and the communications that allow that secure software to interact with the normal world software. TEE software typically consists of a small operating system and its applications, and APIs which allow the secure software to communicate with the larger, user-centric software (Android, Linux, etc.).

Currently, Zynq devices are supported by two open source TEEs including Nagoya (Japan) University's TOPPERS-SafeG which can be described as an open source licensed RTOS/GPOS environment, and also the Sierraware SierraTEE which supports POSIX and GlobalPlatform APIs. The TEEs are available under GPL and commercial licenses.

GlobalPlatform is a cross industry association that publishes specifications to facilitate the secure and interoperable deployment and management of embedded applications. One of these specifications defines a Trusted Execution Environment offering what some might call "typical RTOS" APIs and functionality as well as additional capabilities and APIs that are well suited to the TEE use cases.

Xilinx TrustZone Documentation

As TrustZone is a system topic, several documents are necessary to fully understand the solution. In addition to this getting started guide, Xilinx recommends additional resources as described below.

White Paper 429: TrustZone Technology Support in Zynq-7000 All Programmable SoCs

White Paper 429 is intended to document how to extend Zynq-7000 TrustZone support to IP cores within the Programmable Logic.

White Paper 429 is provided under the same terms of confidentiality as this guide, so you should already have access to it. See http://www.xilinx.com/support/answers/54835.htm for details on obtaining these documents.

Zynq-7000 All Programmable SoC Technical Reference Manual

This user guide provides TrustZone-related register details and usage information for the Zynq-7000 AP SoC family to complement the primary technical information provided in the *Zynq-7000 All Programmable SoC Technical Reference Manual* (UG585) [Ref 1].

Third-Party IP Documentation

The Zynq-7000 family of devices include many different IP cores from ARM, Xilinx and third parties, most containing TrustZone architecture support. Although this user guide is not written in a manner that explicitly requires such additional third party information, documents linked from our web at http://www.xilinx.com/products/zynq-7000/third-party-documentation.htm) might provide additional context.

ARM TrustZone Architecture in the Zynq-7000 All Programmable SoC

ARM architecture supports multiple operating modes including supervisor, system, and user modes to provide different levels of protection at the application level. The architecture support for TrustZone technology helps to create a secure environment to run applications and protect their contents. TrustZone built into the ARM CPU processor and many peripherals enable a secure system to handle keys, private data, and encrypted information without allowing these secrets to leak to non-trusted programs or users. TrustZone security is summarized in Table 1.

Table 1:   **Zynq-7000 AP SoC TrustZone Security Summary**

PS7 Entity	TrustZone Security	Notes
ARM CPU System		
ARM A9 Core	Both	
L1 Cache Controller	Secure	
L1 Cache	Both	
Memory Management Unit	Secure	
SCU	Secure	
L2 Cache Controller	Secure	
SLCR	Secure	
Triple Timer-Counter0	Secure	
Triple Timer-Counter1	Configurable	Control using SLCR
Watch Dog	Secure	
SoC CoreSight Debug	Secure	
OCM	Secure and Non-secure	256 KB RAM can be divided into 4KB secure domains
DDR memory	Secure and Non-secure	Divided into 64MB secure domains. Pass AxPROTECT signals through
IOU Devices	Configurable	I2C, GPIO, SPI, Ethernet, SDIO, CAN, USB and UART, Quad-SPI, NOR

Support for TrustZone within the Zynq APU

For the system to transition from power-on-reset to a stable state with both secure and non-secure applications running simultaneously, it must go through the steps depicted in Figure 2.
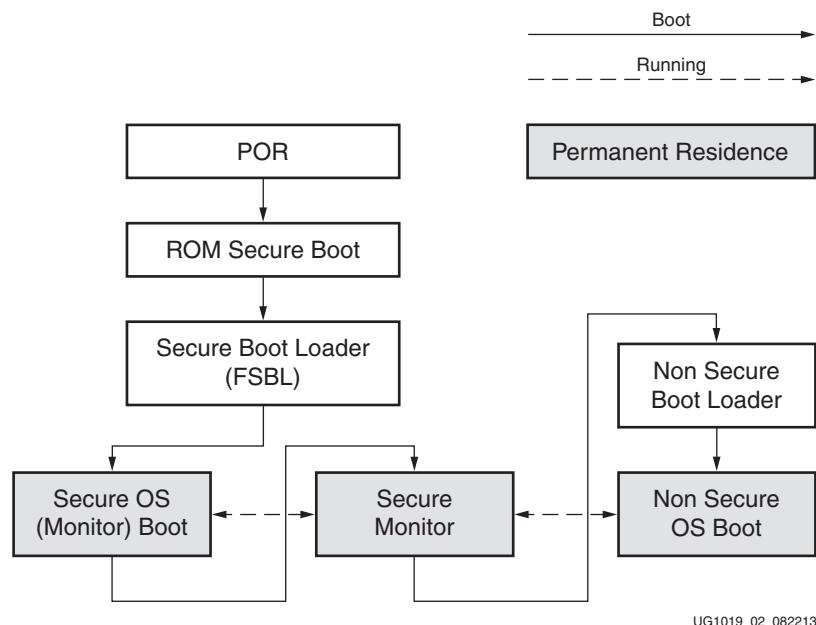
Figure 2: **TrustZone Boot Sequence**

In Figure 2, it is assumed that the Zynq secure boot sequence is used; however, this is not a necessary requirement to enable TrustZone secure boot. In this figure, solid lines are used to show the boot flow and dotted lines are used to indicate processing transition after the system is running. The shaded blocks are software functional blocks that remain running after the system boots. In the TrustZone boot flow, secure OS boots first and initiates a secure monitor as a secure gateway between the secure and non-secure operating systems. After the secure monitor starts, it can spawn a non-secure boot loader which in turn starts a non-secure OS. Before non-secure OS initiates, secure OS defines a set of events to force transition from non-secure OS to secure monitor. The possible events include SMC instruction, IRQ, FIQ, and external Data Abort, and external Prefetch Abort exceptions.

CPU Security Transition

Secure monitor call (SMC) causes a secure monitor exception which is available only in privileged modes. An attempt to execute this instruction in user mode causes an undefined instruction exception.

Other than entering the monitor mode through an SMC call, there are a few additional methods to allow users to switch back-and-forth between secure and non-secure worlds via the secure monitor, which acts as a gate-keeper between these two domains. These are all of the possible methods to enter the monitor mode:

- External abort handler
- FIQ handler
- IRQ handler

Refer to the *Zynq-7000 All Programmable SoC Technical Reference Manual* (UG585) [Ref 1]for more information.

In the secure monitor mode, the processor is always in the secure state, independent of the SCR.NS bit.

It is possible that Cortex-A9 TrustZone violations from normal world software can trigger erroneous asynchronous aborts in secure world. For example, a secure hypervisor runs some code with SCR.NS=1 and the mask asynchronous abort bit CPSR.A=1. This non-secure code attempts to read and write memory that is marked as secure-only, and receives AXI DECERR. At that time, no exceptions occur because CPSR.A=1 and the exceptions are masked. Once secure mode is re-entered, the hypervisor switches to other code, performs DSB/ISB, and clears CPSR.A to 0. In this case, the Cortex-A9 processor still remembers the pending asynchronous external abort and takes an exception as soon as CPSR.A=0. In addition, the non-secure code might have done multi-word stores to secure memory, causing data to be modified in the L1 cache. When this data is eventually cast out, an asynchronous external abort might also be taken in the secure mode.

To address the above operational aspects of the Cortex A9, software can to check the status of the Data Abort status register (see the *ARM Architecture Reference Manual* [Ref 2]) prior to switching from normal world to secure world.

If there no Data Abort is pending, then the switch can be executed.

If there is a Data Abort pending, then you might want to:

1. Enable CPSR.A (set to 0) - the pending abort immediately shows up to the non-secure world before the transition to secure world.

2. Ignore CPSR.A in the normal world and handle this in the secure world software. Such an approach ensures that secure software is aware that normal world software attempted to access secure address regions.

CPSR.A is, by default, accessible to normal world (non-secure) software and user-specific use cases and system designs dictate just how the CPSR.A should be handled. To designate this register as secure, you should refer to section B1.8.7 of the *ARM Architecture Reference Manual* [Ref 2] which describes how SCR.AW can be used to prevent the use of CPSR.A by non-secure software by disabling write-permissions.

CP15 Register Access Control

To support secure and non-secure functionality, the Zynq-7000 AP SoC provides a group of banked registers under CP15, indicating that the same register has two physical copies for

secure and non-secure modes. The physical register is automatically selected based on the SCR.NS bit when the system is not in the secure monitor mode. In the secure monitor mode, a secure version of the physical register is always selected. Table 2 shows part of the CP15 registers. (Refer to the *ARM Architecture Reference Manual* [Ref 2]for the entire list of the registers.)

Table 2: **Typical CP15 Registers**

CP15 Register	Banked Register	Permitted Accesses
c0	CSSELR, Cache Size Selection register	Read/write in privileged modes only
c1	SCTLR, System Control register	Read/write in privileged modes only
	ACTLR, Auxiliary Control register	Read/write in privileged modes only
c2	TTBR0, Translation Table Base 0	Read/write in privileged modes only
	TTBR0, Translation Table Base 1	Read/write in privileged modes only
	TTBCR, Translation Table Base Control	Read/write in privileged modes only
c3	DACR, Domain Access Control register	Read/write in privileged modes only
c5	DFSR, Data Fault Status register	Read/write in privileged modes only
	IFSR, Instruction Fault Status register	Read/write in privileged modes only
	ADFSR, Auxiliary Data Fault Status register	Read/write in privileged modes only
	AIFSR, Auxiliary Instruction Fault Status register	Read/write in privileged modes only
c6	DFAR, Data Fault Address register	Read/write in privileged modes only
	IFAR, Instruction Fault Address register	Read/write in privileged modes only
c7	PAR, Physical Address register (VA to PA translation)	Read/write in privileged modes only
c10	PRRR, Primary Region Remap register	Read/write in privileged modes only
	NMRR, Normal Memory Remap register	Read/write in privileged modes only
c12	VBAR Vector Base Address register	Read/write in privileged modes only

Table 3 shows additional CP15 Control registers with access control of secure and non-secure states.

Table 3: **Additional CP15 Control Registers with Access Control of Secure and Non-secure States**

CP15 Register	Banked Register	Permitted Accesses
c1	NSACR, Non-Secure Access Control register	Read/write in privileged modes only Read-only in non-secure privileged modes
	SCR, Secure Configuration register	Read/write in privileged modes only
	SDER, Secure Debug Enable register	Read/write in privileged modes only
c12	MVBAR, Monitor Vector Base Address register	Read/write in privileged modes only

MMU Security

The MMU in Cortex A9 is enhanced with TrustZone features to provide access permission checks in addition to address translation. In each of the secure and non-secure worlds, a single set of two-level page tables stored in main memory controls the contents of the instruction- and data-side Translation Look-aside Buffers (TLBs). The calculated physical address associated with the virtual address is placed in the TLB, accompanied by a non-secure table identifier (NSTID) that allows secure and non-secure entries to coexist. The TLBs are enabled in each world through a single bit in CP15 Control register c1, providing a single address translation and protection scheme for software.

The following describes how the TLB and Branch Target Access Control (BTAC) state is handled when transitioning between secure and non-secure worlds:

- Branch Predictor Invalidate All (BPIALL), when executed from the secure state, might or might not invalidate BTAC entries in the non-secure state.

- TLBIALL, when executed from the secure state, does not invalidate BTAC entries from the non-secure state.

- Writing to CONTEXTIDR from the secure state, but with SCR.NS=1, does invalidate BTC entries in the non-secure state.

Because any context switch writes to CONTEXTIDR and causes the non-secure BTAC entries to be invalidated, BPIALL from secure state does not affect non-secure BTAC entries and is not a problem.

L-1 Cache Security

Each cache line contains secure or non-secure data. The effect of an access attempting to violate data security causes a cache miss. On a miss, the next step is to go to the external memory which returns an abort if the NS attribute does not match access permissions.

Security Exception Control

When an exception is taken, processor execution is forced to an address that corresponds to the type of exception. These addresses are called the exception vectors. By default, the exception vectors are eight consecutive word-aligned memory addresses, starting at an exception base address, as follows:

There are three exception base addresses for Zynq-7000 AP SoC devices:

1. The non-secure exception base address is used for all exceptions processed in non-secure state.

2. The secure exception base address is used for all exceptions processed in secure state.

3. The monitor exception base address is used for all exceptions processed in monitor mode.

CPU Debug TrustZone Access Control

Four control signals control CPU debug status: DBGEN, NIDEN, SPIDEN, and SPNIDEN. These four control signals are part of a secure and protected register within the device configuration interface module. The most important modes supported are identified in Table 4.

Table 4: **Debug Status Control**

Mode	DBGEN	NIDEN	SPIDEN	SPNIDEN	Comment
No Debug	0	0	0	0	No CPU debug at all
Non-Secure Non Invasive Debug	1	1	0	0	Allows non-invasive debug, such as trace and performance monitor, in non-secure mode
Non-Secure Invasive Debug	1	1	0	0	Allows invasive debug, such as stop processor, in non-secure mode
Secure Non Invasive Debug	1	1	0	1	Allows CPU trace and profile under secure condition
Secure Invasive Debug	1	1	1	1	Allows invasive debug for secure mode

SCU Register Access Control

The SCU non-secure Access Control register (SNACR) controls the global non-secure accesses for each major component within the SCU. The Interrupt Controller Distributor Control register (ICDDCR) is a banked register for controlling secure and non-secure accesses.

TrustZone Support in the L2 Cache

The cache controller attaches an NS bit to all data stored in the L2 cache and in internal buffers. A non-secure transaction cannot access secure data. Therefore the controller treats secure and non-secure data as being part of two different memory spaces. The controller treats a non-secure access to secure data in the L2 cache as a miss. For a read transfer, the cache controller sends a line fill command to external memory, propagates any security errors from external memory to the processor, and does not allocate the line in L2.

These are a few notes about the TrustZone support in L2:

- The L2 Control register can only be written with an access tagged as secure, to enable or disable the L2 cache.

- The Auxiliary Control register can only be written with an access tagged as secure. Bit [26] in the Auxiliary Control register is for NS lockdown enable. This bit should be used to determine whether non-secure accesses can modify a lockdown register.

- Non-secure maintenance operations do not clean or invalidate secure data.

- The Zynq L2 cache controller does not need to access the Debug Control register. However, any attempt to access this register from the NS mode results in a DECERR.

Zynq TrustZone Support for DDR Memory

The TrustZone capabilities of the Zynq-700 SoC allow DDR memory to be independently configured as secure or non-secure in incremental sections of 64 MB. This configuration is provided via a system level control register.

- A `0` on a particular bit indicates a secure memory region for that particular memory segment.

- A `1` on a particular bit indicates a non-secure memory region for that particular memory segment.

In the case of a non-secure access to a secure region, a DECERR response is returned to the master. For writes to a secure region, the write data is masked out before being sent to the controller which results in no actual writes occurring in the DRAM. A read attempt of a secure region by non-secure access returns all zeros.

TrustZone Configuration Registers in the Zynq-7000 All Programmable SoC

This section describes how to configure peripherals included within the Zynq-7000 AP SoC for the instantiation of TrustZone enabled system designs.

The designation of a hardware item as secure indicates that this hardware is visible only within the secure world. Items marked as non-secure are visible within both the normal world and also the secure world.

Module Summary

Zynq-7000 AP SoC includes one TrustZone module comprising 22 registers.

Table 5: **Module Summary**

Module Name	Module Type	Base Address	Version	Description
trustzone	TrustZone	`E0200000, F8000000`	1.0.0	TrustZone control registers

Module trustzone

Module Name	trustzone
Base Address	`E0200000 and F8000000`
Description	TrustZone control registers
Version	1.0.0
Doc Version	1.0
Vendor Info	Xilinx

Register Access Legend

Access Type	Description
CLRONRD	Readable, clears value on read
CLRONWR	Readable, clears value on write
NSNSRO	During non-secure access, if thread is non-secure, it is read only
NSNSRW	During non-secure access, if thread is non-secure, it is read write
NSNSWO	During non-secure access, if thread is non-secure, it is write only
NSSRAZ	During non-secure access, if thread is secure, it is read as zero
RAZ	Read as zero
RO	Read-only
RS	w: no effect, r: sets all bits
RUD	Read undefined
RW	Normal read/write
RWSO	Read/write, set only
SRO	During secure access, it is read only
SRW	During secure access, it is read write

Access Type	Description
SWO	During secure access, it is write only
W0C	w: 1/0 no effect on/clears matching bit, r: no effect
W0CRS	w: 1/0 no effect on/clears matching bit, r: sets all bits
W0S	w: 1/0 no effect on/sets matching bit, r: no effect
W0SRC	w: 1/0 no effect on/sets matching bit, r: clears all bits
W0T	w: 1/0 no effect on/toggles matching bit, r: no effect
W1	w: first one after ~hard~ reset is as-is, other w have no effects, r: no effect
W1CRS	w: 1/0 clears/no effect on matching bit, r: sets all bits
W1SRC	w: 1/0 sets/no effect on matching bit, r: clears all bits
W1T	w: 1/0 toggles/no effect on matching bit, r: no effect
WAZ	Write as zero
WCRS	w: clears all bits, r: sets all bits
WO	Write-only
WO1	w: first one after ~hard~ reset is as-is, other w have no effects, r: error
WOC	w: clears all bits, r: error
WOS	w: sets all bits, r: error
WRC	w: as-is, r: clears all bits
WRS	w: as-is, r: sets all bits
WS	w: sets all bits, r: no effect
WSRC	w: sets all bits, r: clears all bits
WTC	Readable, write a 1 to clear
Z	Access (read or write) as zero

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
security2_sdio0	0xE0200008	1	WO	0x00000000	SDIO0 slave security setting.
security3_sdio1	0xE020000C	1	WO	0x00000000	SDIO1 slave security setting.
security4_qspi	0xE0200010	1	WO	0x00000000	QSPI slave security setting.
security6_apb_slaves	0xE0200018	15	WO	0x00000000	APB slave security setting.
security7_smc	0xE020001C	1	WO	0x00000000	SMC slave security setting.
DMAC_RST_CTRL	0xF800020C	32	RW	0x00000000	DMA Controller SW Reset Control
TZ_OCM_RAM0	0xF8000400	32	RW	0x00000000	OCM RAM TrustZone Config 0
TZ_OCM_RAM1	0xF8000404	32	RW	0x00000000	OCM RAM TrustZone Config 1
TZ_OCM	0xF8000408	32	RW	0x00000000	OCM ROM TrustZone Config
TZ_DDR_RAM	0xF8000430	32	RW	0x00000000	DDR RAM TrustZone Config

Register Name	Address	Width	Type	Reset Value	Description
TZ_DMA_NS	0xF8000440	32	RW	0x00000000	DMAC TrustZone Config
TZ_DMA_IRQ_NS	0xF8000444	32	RW	0x00000000	DMAC TrustZone Config for Interrupts
TZ_DMA_PERIPH_NS	0xF8000448	32	RW	0x00000000	DMAC TrustZone Config for Peripherals
TZ_GEM	0xF8000450	32	RW	0x00000000	Ethernet TrustZone Config
TZ_SDIO	0xF8000454	32	RW	0x00000000	SDIO TrustZone Config
TZ_USB	0xF8000458	32	RW	0x00000000	USB TrustZone Config
TZ_FPGA_M	0xF8000484	32	RW	0x00000000	FPGA master ports TrustZone Disable
TZ_FPGA_AFI	0xF8000488	32	RW	0x00000000	FPGA AFI AXI ports TrustZone Disable
security_fssw_s0	0xF890001C	1	WO	0x00000000	M_AXI_GP0 security setting
security_fssw_s1	0xF8900020	1	WO	0x00000000	M_AXI_GP1 security setting
security_apb	0xF8900028	6	WO	0x00000000	APB boot secure ports setting.

security2_sdio0

Name	security2_sdio0
Relative Address	0xE0200008
Absolute Address	0xE0200008
Width	1 bit
Access Type	WO
Reset Value	0x00000000
Description	SDIO0 slave security setting

Field Name	Bit	Type	Reset Value	Description
	0	WO	0x0	0: Secure 1: Non-secure

security3_sdio1

Name	security3_sdio1
Relative Address	0xE020000C
Absolute Address	0xE020000C
Width	1 bit
Access Type	WO
Reset Value	0x00000000
Description	SDIO1 slave security setting

Field Name	Bit	Type	Reset Value	Description
	0	WO	0x0	0: Secure 1: Non-secure

security4_qspi

Name	security4_qspi
Relative Address	0xE0200010
Absolute Address	0xE0200010
Width	1 bit
Access Type	WO
Reset Value	0x00000000
Description	QSPI slave security setting

Field Name	Bit	Type	Reset Value	Description
	0	WO	0x0	0: Secure 1: Non-secure

This Security4_qspi register defines whether access to the Linear QSPI address is a secure or non-secure function. See Chapter 12 of the *Zynq-7000 AP SoC Technical Reference Manual* (UG585) [Ref 1] for more information.

security6_apb_slaves

Name	security6_apb_slaves
Relative Address	0xE0200018
Absolute Address	0xE0200018
Width	15 bits
Access Type	WO
Reset Value	0x00000000
Description	APB slave security setting

Field Name	Bits	Type	Reset Value	Description
usb1_s_apb	14	WO	0x0	0: Secure 1: Non-secure
usb0_s_apb	13	WO	0x0	0: Secure 1: Non-secure
gem1_s_apb	12	WO	0x0	0: Secure 1: Non-secure
gem0_s_apb	11	WO	0x0	0: Secure 1: Non-secure

Field Name	Bits	Type	Reset Value	Description
smc_s_apb	10	WO	0x0	0: Secure 1: Non-secure
spi1_s_apb	9	WO	0x0	0: Secure 1: Non-secure
spi0_s_apb	8	WO	0x0	0: Secure 1: Non-secure
ua1_s_apb	7	WO	0x0	0: Secure 1: Non-secure
ua0_s_apb	6	WO	0x0	0: Secure 1: Non-secure
i2c1_s_apb	5	WO	0x0	0: Secure 1: Non-secure
i2c0_s_apb	4	WO	0x0	0: Secure 1: Non-secure
gpio_s_apb	3	WO	0x0	0: Secure 1: Non-secure
qspi_s_apb	2	WO	0x0	0: Secure 1: Non-secure
can1_s_apb	1	WO	0x0	0: Secure 1: Non-secure
can0_s_apb	0	WO	0x0	0: Secure 1: Non-secure

security7_smc

Name	security7_smc
Relative Address	0xE020001C
Absolute Address	0xE020001C
Width	1 bit
Access Type	WO
Reset Value	0x00000000
Description	SMC slave security setting

Field Name	Bit	Type	Reset Value	Description
	0	WO	0x0	0: Secure 1: Non-secure

This Security7_smc register defines whether access to the SMC (SRAM or NOR) is a secure or non-secure function. See Chapter 11 of the *Zynq-7000 AP SoC Technical Reference Manual* (UG585) [Ref 1] for more information.

DMAC_RST_CTRL

Name	DMAC_RST_CTRL
Relative Address	`0xF800020C`
Absolute Address	`0xF800020C`
Width	32 bits
Access Type	RW
Reset Value	`0x00000000`
Description	DMA controller SW reset control

Field Name	Bit	Type	Reset Value	Description
Reserved	31:1	RW	`0x0`	Reserved. Writes are ignored, read data is zero.
DMAC_RST	0	RW	`0x0`	DMA Controller software reset signal: • `0`: De-assert (DMA controller TrustZone register is read only) • `1`: Assert (DMA controller TrustZone register is writable)

Note: This DMAC_RST register is always writable, however, all writes are ignored until the DMAC comes out of reset. Therefore, the reset sequence required by the DMAC_RST_CTRL is: Assert followed by de-assert.

TZ_OCM_RAM0

Name	TZ_OCM_RAM0
Relative Address	`0xF8000400`
Absolute Address	`0xF8000400`
Width	32 bits
Access Type	RW
Reset Value	`0x00000000`
Description	OCM RAM TrustZone Config 0

Field Name	Bit	Type	Reset Value	Description

TZ_OCM_RAM0	31:0	RW	`0x0`	Each bit represents the TrustZone status for a 4 KB page:
				• `0`: Secure
				• `1`: Non-secure
				  Bit [0]: Page 0 (first 4 KB)
				  Bit [1]: Page 1
				  ...
				  Bit [31]: Page 31 (through 128 KB)

The OCM can be partitioned during run-time for secure and non-secure operation. This is possible because the OCM does not have strict configuration requirements based on reset and clock state. Xilinx recommends that the software should read back the contents of TZ_OCM_RAM registers prior to OCM access based on the new TZ settings.

TZ_OCM_RAM1

Name	TZ_OCM_RAM1
Relative Address	0xF8000404
Absolute Address	0xF8000404
Width	32 bits
Access Type	RW
Reset Value	0x00000000
Description	OCM RAM TrustZone Config 1

Field Name	Bit	Type	Reset Value	Description

TZ_OCM_RAM1	31:0	RW	0x0	Each bit represents the TrustZone status for a 4 KB page:
				• 0: Secure
				• 1: Non-secure
				  Bit [0]: Page 32 (starts at 128 KB)
				  Bit [1]: Page 33
				  …
				  Bit [31]: Page 63 (through 256 KB)

The OCM can be easily partitioned during run-time for secure and non-secure operation. This is possible because the OCM does not have strict configuration requirements based on reset and clock state. Xilinx recommends that the software should read back the contents of TZ_OCM_RAM registers prior to OCM access based on the new TZ settings.

TZ_OCM

Name	TZ_OCM
Relative Address	0xF8000408
Absolute Address	0xF8000408
Width	32 bits
Access Type	RW
Reset Value	0x00000000
Description	OCM TrustZone Config

Field Name	Bit	Type	Reset Value	Description

TZ_OCM	31:0	RW	0x0	Each bit represents the TrustZone status for a 4 KB page:
				• 0: Secure
				• 1: Non-secure
				   Bit [0]: Page 64 (starts at 256 KB)
				   Bit [1]: Page 65
				   ...
				   Bit [31]: Page 95 (through 512 KB)

TZ_DDR_RAM

Name	TZ_DDR_RAM
Relative Address	0xF8000430
Absolute Address	0xF8000430
Width	32 bits
Access Type	RW
Reset Value	0x00000000
Description	DDR RAM TrustZone Config

Field Name	Bit	Type	Reset Value	Description
TZ_DDR_RAM	0	RW	0x0	Each bit represents the TrustZone status for a 64 MB section n at nMB: • 0: Secure, reset value • 1: Non-secure

TZ_DMA_NS

Name	TZ_DMA_NS
Relative Address	0xF8000440
Absolute Address	0xF8000440
Width	32 bits
Access Type	RW
Reset Value	0x00000000
Description	DMAC TrustZone Config

Field Name	Bit	Type	Reset Value	Description
Reserved	31:1	RW	0x0	Should be zero.
DMAC_NS	0	RW	0x0	TZ security (connected to boot_manager_ns on DMAC): • 0: Secure, DMAC operates in the secure state • 1: Non-secure, DMAC operates in the non-secure state

TZ_DMA_IRQ_NS

Name	TZ_DMA_IRQ_NS
Relative Address	0xF8000444
Absolute Address	0xF8000444
Width	32 bits
Access Type	RW
Reset Value	0x00000000
Description	DMAC TrustZone Config for interrupts

Field Name	Bit	Type	Reset Value	Description
Reserved	31:16	RW	0x0	Should be zero.
DMA_IRQ_NS	15:0	RW	0x0	TZ security (connected to boot_irq_ns on DMAC): • 0: Secure, designated external interrupt operates in the secure state • 1: Non-secure, designated external interrupt operates in the non-secure state

TZ_DMA_PERIPH_NS

Name	TZ_DMA_PERIPH_NS
Relative Address	0xF8000448
Absolute Address	0xF8000448
Width	32 bits
Access Type	RW
Reset Value	0x00000000
Description	DMAC TrustZone Config for peripherals

Field Name	Bit	Type	Reset Value	Description
Reserved	31:4	RW	0x0	Should be zero.
DMAC_PERIPH_NS	3:0	RW	0x0	TZ security (connected to to boot_periph_ns on DMAC): • 0: Secure, peripheral request interface operates in the secure state • 1: Non-secure, reset value; peripheral request interface operates the non-secure state

TZ_GEM

Name	TZ_GEM
Relative Address	0xF8000450
Absolute Address	0xF8000450
Width	32 bits
Access Type	RW
Reset Value	0x00000000
Description	Ethernet TrustZone Config

Field Name	Bit	Type	Reset Value	Description
Reserved	31:2	RW	0x0	Should be zero.
E1	1	RW	0x0	TrustZone status for Gigabit Ethernet MAC 1: • 0: Secure, reset value • 1: Non-secure
E0	0	RW	0x0	TrustZone status for Gigabit Ethernet MAC 0: • 0: Secure, reset value • 1: Non-secure

TZ_SDIO

Name	TZ_SDIO
Relative Address	0xF8000454
Absolute Address	0xF8000454
Width	32 bits
Access Type	RW
Reset Value	0x00000000
Description	SDIO TrustZone Config

Field Name	Bit	Type	Reset Value	Description
Reserved	31:2	RW	0x0	Should be zero.
S1	1	RW	0x0	TrustZone status for SDIO Controller 1: • 0: Secure, reset value • 1: Non-secure
S0	0	RW	0x0	TrustZone status for SDIO Controller 0: • 0: Secure, reset value • 1: Non-secure

TZ_USB

Name	TZ_USB
Relative Address	0xF8000458
Absolute Address	0xF8000458
Width	32 bits
Access Type	RW
Reset Value	0x00000000
Description	USB TrustZone Config

Field Name	Bit	Type	Reset Value	Description
Reserved	31:2	RW	0x0	Should be zero.
U1	1	RW	0x0	TrustZone status for USB Controller 1: • 0: Secure, reset value • 1: Non-secure
U0	0	RW	0x0	TrustZone status for USB Controller 0: • 0: Secure, reset value • 1: Non-secure

TZ_FPGA_M

Name	TZ_FPGA_M
Relative Address	0xF8000484
Absolute Address	0xF8000484
Width	32 bits
Access Type	RW
Reset Value	0x00000000
Description	FPGA master ports TrustZone disable

Field Name	Bit	Type	Reset Value	Description
Reserved	31:2	RW	0x0	Should be zero.
M1	1	RW	0x0	Secure disable for PL AXI master port 1: • 0: Master port can make secure and non-secure accesses, reset value • 1: Master port can only make non-secure accesses
M0	0	RW	0x0	Secure disable for PL AXI master port 0: • 0: Master port can make secure and non-secure accesses, reset value • 1: Master port can only make non-secure accesses

TZ_FPGA_AFI

Name	TZ_FPGA_AFI
Relative Address	0xF8000488
Absolute Address	0xF8000488
Width	32 bits
Access Type	RW
Reset Value	0x00000000
Description	FPGA AFI AXI ports TrustZone disable

Field Name	Bit	Type	Reset Value	Description
Reserved	31:4	RW	0x0	Should be zero.
P3	3	RW	0x0	Secure disable for PL AXI_HP port 3: • 0: Port can make secure and non-secure accesses, reset value • 1: Port can only make non-secure accesses
P2	2	RW	0x0	Secure disable for PL AXI_HP port 2: • 0: Port can make secure and non-secure accesses, reset value • 1: Port can only make non-secure accesses
P1	1	RW	0x0	Secure disable for PL AXI_HP port 1 • 0: Port can make secure and non-secure accesses, reset value • 1: Port can only make non-secure accesses
P0	0	RW	0x0	Secure disable for PL AXI_HP port 0 • 0: Port can make secure and non-secure accesses, reset value • 1: Port can only make non-secure accesses

security_fssw_s0

Name	security_fssw_s0
Relative Address	0xF890001C
Absolute Address	0xF890001C
Width	1 bit
Access Type	RO
Reset Value	0x00000000
Description	M_AXI_GP0 security setting

Field Name	Bit	Type	Reset Value	Description
fssw_s0	0	RO	0x0	This register defines whether S or NS request propagates out to the logic: • 0: NS requests do not propagate to the logic • 1: Both NS and S requests are propagated to the logic

security_fssw_s1

Name	security_fssw_s0
Relative Address	0xF8900020
Absolute Address	0xF8900020
Width	1 bit
Access Type	RO
Reset Value	0x00000000
Description	M_AXI_GP0 security setting

Field Name	Bit	Type	Reset Value	Description
fssw_s1	0	RO	0x0	This register defines whether S or NS request propagates out to the logic: • 0: NS requests do not propagate to the logic • 1: Both NS and S requests are propagated to the logic

Also see the NIC301 PL Clocks and Inactive Resets section of this document.

security_apb

Name	security_apb
Relative Address	0xF8900028
Absolute Address	0xF8900028

Width	6 bits
Access Type	RW
Reset Value	`0x00000000`
Description	FPGA AFI AXI ports TrustZone disable

Field Name	Bit	Type	Reset Value	Description
ddrc_apb	5	WO	`0x0`	Controls the APB transactions to DDRC: • `0`: Always secure • `1`: Always non-secure
ttc1_apb	4	WO	`0x0`	Controls the APB transactions to TTC1: • `0`: Always secure • `1`: Always non-secure
afi3_apb	3	WO	`0x0`	Controls the APB transactions to HP3: • `0`: Always secure • `1`: Always non-secure
afi2_apb	2	WO	`0x0`	Controls the APB transactions to HP2: • `0`: Always secure • `1`: Always non-secure
afi1_apb	1	WO	`0x0`	Controls the APB transactions to HP1: • `0`: Always secure • `1`: Always non-secure
afi0_apb	0	WO	`0x0`	Controls the APB transactions to HP0: • `0`: Always secure • `1`: Always non-secure

Configuring and Programming TrustZone on the Zynq-7000 AP SoC

This section provides guidance necessary to program the TrustZone registers on the Zynq-7000 family of devices.

Active Clocks and Inactive Resets

The creation of a TrustZone-enabled solution on Zynq-7000 devices requires the configuration of several registers that are specific to various subsystems within the AP SoC.

IMPORTANT: *To ensure proper operation and avoid the PS hanging on a register access attempt, special attention should be paid to ensure that the proper clocks are active and that resets are deasserted.*

PL Clocks and Inactive Resets

Some configuration registers within the Zynq-7000 devices manipulate the Zynq-7000 GPV registers for the AXI interconnect. If the following are not addressed, the entire PS might hang on a GPV access attempt.

- Define all clocks within the Programmable Logic as *active*.

- Define all resets on all PS-PL AXI interfaces for the GPV as *deasserted*.

The above is true even if the AXI master (slave) interfaces are not connected to embedded AXI slaves (masters), and even if the remap feature is used to disable the FPGA AXI master interfaces (connecting to slaves).

See Section 5.6.2, *Clocks and Resets* of the *Zynq-7000 AP SoC Technical Reference Manual* (UG585) [Ref 1] for more information.

NIC301 PL Clocks and Inactive Resets

The Zynq NIC301 AMBA-AXI interconnect includes a few Address Region Control registers (nic301_addr_region_ctrl_registers) that require the PL clock to be active and the corresponding reset to be de-asserted. Thus, the FPGA_RST_CTL register must be configured per the Zynq-7000 AP SoC TRM recommendations (e.g., writing zeros to suggested bit fields). See sections B.21, *NIC301 Address Region Control* (nic301_addr_region_ctrl_registers) and B.28, *System Level Control Registers* (slcr)/*Register FPGA_RST_CTRL Details*) of the *Zynq-7000 AP SoC Technical Reference Manual* (UG585) [Ref 1].

APB Boot Secure Port Resets

The APB Boot Secure Ports for DDRC, TTC1, AFI0, AFI1, AFI2, and AFI3 also require deasserted resets at time of GPV configuration.

PL Master Security

The TrustZone security implementation on a PL master IP is design specific. The AXI Interconnect IP from Xilinx supports TrustZone security extension offered as an advanced configuration option in the IP configuration wizard in the Vivado IP catalog. For more information please refer to AXI Interconnect IP data sheet.

AXI bus Transactions as AxCACHE=3

To avoid possible memory coherency issues that appear under a specific set of conditions, all AXI bus transactions issued by the AXI master should be defined as AxCACHE=3.

Zynq-7000 DMAC Considerations

The Zynq-7000 family of devices uses an ARM PL330 DMA Controller which provides the ability to configure DMAs for operation within the secure and non-secure worlds.

Summary of DMA operations:

1. The DMAC is reset

   ◦ Security state of the DMAC is read

2. The DMA Channel is set-up with the DMAGO function

   ◦ With attention to the security state of the DMAC and the security state of the programming software

3. The PL330 fetches the instructions from the location specified by DMAGO

   ◦ DMA instructions are cached by the PL330

4. The specific DMA channel is started

   ◦ Channel-based memory operations execute until the last DMAC instruction is reached

Additional information related to how DMA accesses can be restricted to TrustZone secure and non-secure software follows. This information is intended to complement more detailed information available within the *Corelink DMA-330 Technical Reference Manual* [Ref 3] and within section 9.3 *Programming Guide for DMA Controller of the Zynq-7000 AP SoC Technical Reference Manual* (UG585) [Ref 1].

Step 1: Configure DMAC Security State

The Zynq-7000 device DMAC is treated just like any other TrustZone-enabled hardware within the Zynq-7000 AP SoC, it can be defined as either a secure or non-secure device.

IMPORTANT: *This setting defines the security status of the whole DMAC regardless of whether the non-secure or secure configuration port is used to communicate with it.*

The security for DMAC is set according to the value of the Zynq-7000 TZ_DMA_NS (referred to as: boot_manager_ns in ARM PL330 documentation) bit. This bit must be *set* with its intended value for secured or non-secured status, before the programmer de-asserts the reset to DMAC-core. Because this TZ_DMA_NS bit is set before de-assertion of the reset, all prior DMA channel set-ups are cleared.

When the DMAC is configured as secure, it will be allowed to set-up both non-secure and secure DMA channels, and depending on the security state of the master which is performing the DMA channel setup, it allows access to both secure and non-secure addresses. Such addresses include those where the DMAC instructions are located, and also where the resultant AXI channel Load/Store operations are to take place.

When the DMAC is configured as non-secure, it is allowed to access only NS addresses and create only NS DMA channels regardless of the security state of the CPU which is attempting to setup each DMA channel. A combination of S and NS DMA channels can only be provided when the DMAC is defined as secure.

Step 2: Set-up DMA Channels

Prior to use, each DMA channel must be setup by using a "DMAGO" instruction of the syntax:

```
DMAGO <CHANNEL_NUMBER>, <32-bit_immediate>,  [ns]
```

Note: [ns] can only be defined when the DMAC is defined as a secure device and secure software sets up the channel. When the DMAC is defined as a non-secure device, every DMA channel set-up is NS.

Depending on whether the DMAC has been defined as a secure or non-secure device (see above), the set-up must take place by means of either the secure or the non-secure APB port.

DMAC in Non-Secure State

When the DMAC is configured to the non-secure state, the configuration ports function as follows.

Non-Secure Configuration Port

The non-secure APB configuration port can be used by both secure and non-secure CPUs to set-up a NS DMA channel when the DMAC is in the non-secure state.

Regardless of the security state of the CPU that performs set-up, a DMA channel set-up from this port will be started in the non-secure state. Such non-secure DMA channels should take notice of the security aspects discussed in Step 3: DMA Channel Access.

Secure Configuration Port

The secure configuration port is accessible only to a secure CPU, regardless of the S or NS state of the DMAC.

When the DMAC is in the NS state, all DMA channels that are set-up through this port must be defined as NS channels. Any attempt to create a secure DMA channel [ns=0] results in a NOP. Such non-secure DMA channels should take notice of the security aspects discussed in Step 3: DMA Channel Access.

DMAC Channel Setup Summary

Table 6 identifies the conditions necessary to fulfill to ensure a successful DMA channel start. See sections 2.8.2 and 2.9 of the *Corelink DMA-330 Technical Reference Manual* [Ref 3] for details related to the applicable abort state.

- S = Secure

- NS = Non-secure

- X = Don't care

Table 6: **DMAC Channel Setup Summary**

Master (CPU) S/NS state	DMAC Access through S/NS APB port	DMAC S/NS state	Requested S/NS state of the channel to be started	Success?	Notes
NS	S	X	X	No	NS CPU cannot access secure APB port
X	NS	S	X	No	When the DMAC runs in the secure state, all data written to the non secure configuration port results in an abort
X	X	NS	S	No	NS DMAC cannot start an S channel
S	S	S	X	Yes	
S	X	NS	NS	Yes	
NS	NS	NS	NS	Yes	

Step 3: DMA Channel Access

Once a DMA channel has been started, access is controlled at two points based on the security settings of the DMA channel and the addresses to be accessed.

First, access from the DMAC to the DMA instruction code is based on the security state of the DMAC channel and the security state of address that contains that DMA instruction code. If the DMA channel is defined as NS and the instruction code resides at a secure address, then an abort is sent from the AXI interconnect to the DMAC. The DMAC responds to the CPU with an interrupt.

Second, the security state of the DMA channel must have sufficient privileges to access the data memory that is called out within that DMA instruction code. So, if the DMA channel is defined as non-secure and an address which the DMA channel is attempting to access is Secure, then an abort is sent from the AXI interconnect to the DMAC. The DMAC responds to the CPU with an interrupt.

- S = Secure

- NS = Non-secure

- X = Don't care

Table 7: **DMA Channel Access Summary**

S/NS Status of DMA Channel to be Accessed	S/NS Status of Instruction and Data Memory DMA Try to Access	Successful?
S	S	Yes
S	NS	Yes
NS	S	No
NS	NS	Yes

Interrupt and Peripheral Configurations

The preceding sections describe how the PL330 can be configured to restrict DMA channels to the secure and normal (non-secure) TrustZone worlds. The PL330 also includes two signals which define how external interrupts and peripherals interact with the system. These signals only define access to those interfaces, they do not affect the security access of the DMA.

TZ_DMA_IRQ_NS [Boot_irq_ns]

The boot_irq_ns[] bits define the operation of the external interrupts and whether they can be triggered by secure or non-secure operations. These bits also control ability for specific DMA channels to respond to WFI and WFE instructions.

TZ_DMA_PERIPH_NS [Boot_periph_ns]

The boot_periph_ns[] bits correspond to the peripheral handshake interface. The PL330 provides a configurable number of interfaces which can be provided with direct handshake with specific DMA channels. This handshake allows those peripherals to be controlled by the DMA – such as flush peripheral instruction and load and notify, etc. Signals applied at PL330 reset define whether these can be done in non-secure operations or not (so they are not tied to the boot_manager_ns.

Architecting Systems for Both Secure and Non-Secure DMA Channels

Consider a scenario where a platform must run two operating systems; one in the secure world and one in the normal (non-secure) world. The normal world OS requires DMA accesses, but it must be prevented from doing accessing secure world memory space. Such a scenario can be addressed by starting separate DMA channels, each running as secure and non-secure. There are, however, a couple of key system considerations that must be addressed.

Although the DMA controller, when running as secure, provides the ability to setup and start both secure and non-secure DMA channels, the fact that non-secure CPUs do not have the ability to directly manage the DMAC must be considered. That is, a non-secure DMA channel can be created side-by-side with a secure DMA channel, but that NS channel can only be set-up from a secure CPU.

> **RECOMMENDED:** *In general, the DMAC should be treated in much the same way that secure-only hardware (e.g. cache controller) are handled today.*

Secure-only devices such as the cache controller are directly controlled via software running on a secure CPU. When a non secure CPU uses of those services, it makes a secure monitor call (SMC) to the secure CPU. This SMC typically identifies the command that is requested to be executed along with all key attributes of that call.

In this scenario, just four custom SMCs can manage this process.

1. NS to S:

    a. Pass the parameters needed to setup the DMA channel to the secure world.

       i. The secure world creates a NS DMA channel on behalf of the NS CPU.

       ii. The DMAC begins execution of the DMAC instruction code, pausing at DMAWFE.

       iii. This SMC might look very similar to a standard DMAGO.

2. S to NS:

    a. The S CPU notifies the NS CPU when the DMA channel is ready.

    b. DMAC is parked at DMAWFE.

3. NS to S:

    a. The NS CPU notifies the S CPU to execute the DMA transaction.

4. S to NS:

    a. The S CPU notifies the NS CPU upon transaction completion.

Programming the Zynq-7000 Device PL330

Complementing the functional information provided, this section describes in detail how the DMAC should be programmed.

The DMAC provides configurable security through TrustZone registers. These registers (TZ_DMA_NS, TZ_DMA_IRQ_NS, TZ_DMA_PERIPH_NS) are external to the DMAC core (in SLCR) and these registers define the security conditions of the DMAC. When we are setting these bits in SLCR registers, we need to maintain these things in order:

1. The DMAC core gets "RESET" whenever DMAC_RST_CTRL=1 is set (@`0xF800020C`). All DMAC core registers get initialized on setting DMAC_RST_CTRL=1.

    Note: This activity does not reset the TrustZone DMAC registers.

2. The DMAC core expects the following interfaces *set* with their intended value for secured or non-secured status, before de-asserting the reset to the DMAC core. This means that when you program trustzone registers @`0xF8000440, 44, 48` for the

following settings (a,b,c), you need to keep the DMAC core under reset as described in the previous paragraph (1).

   a. Set TZ_DMA_NS = 0 if a secure DMA channel thread is required. TZ_DMA_NS = 1 indicates that a non-secure DMA channel thread is set.

      Note: When set, the security state of the DMA manager remains constant. It is not possible to alter the security state of the DMA manager by programming a register in the DMAC.

   b. Set TZ_DMA_IRQ_NS= 0 if a secure DMA irq is required. TZ_DMA_IRQ_NS = 1 indicates that a non-secure DMAC irq is set.

      Note: When set, the security state of the irq signals remains constant. It is not possible to alter the security state of an irq by programming a register in the DMAC.

   c. Set TZ_DMA_PERIF_NS= 0 if a secure DMA Perif is required. TZ_DMA_PERIF_NS = 1 indicates that a non-secure DMAC perif is set.

      Note: When set, the security state of the peripherals request remains constant. It is not possible to alter the security state of a peripherals request by programming a register in the DMAC.

3. Once the TrustZone registers are programmed to their intended value, de-assert the reset by writing DMAC_RST_CTRL = 0 (@0xF800020C).

4. You must ensure that you use the appropriate APB interface, depending on the security state in which the TZ_DMA_NS (boot_manager_ns - internal DMA signal) initializes the DMAC to operate. For example, if the DMAC is in the secure state, you must issue the instruction using the secure APB interface, otherwise the DMAC ignores the instruction. However, you can use either the secure APB interface or the non-secure APB interface to start or restart a DMA channel when the DMAC is in the non-secure state.

   Additional information can be found in ARM's PL330 specification Section 2.4 *Initializing the DMAC*.

TrustZone in Practice

The following sections provide additional details related to specific assets that can be leveraged for the improved safety or security of a Zynq-7000 based solution.

Securing CPSRs with CP15SDISABLE

The processor supports a primary input pin, CP15SDISABLE, to disable write access to the CP15 System Control Processor registers. When the CP15SDISABLE input is set to 1, any attempt by secure or normal world software to write to the secure version of the banked register, NS-bit is 0, or any non-banked register, NS-state is 0, results in an undefined instruction exception. Note that read accesses by secure world and normal world software are still possible when this signal is set, but those read attempts remain constrained by the security state of each specific CPSR bit. (See the *ARM Cortex-A9 MPCore Technical Reference*

Manual [Ref 4]for details).

Once this CP15DISABLE bit has been set, it cannot be modified, even by secure software, without a power-on-reset. Because of the foregoing, during product development and for those products which have less stringent security requirements, this bit can be set to zero on reset to allow the system to be configured.

This signal could be used within a Xilinx Zynq-7000 based system to prevent specific types of hardware attack by implementing this bit within a secure peripheral located wholly within the logic and inaccessible to external probing.

Securing Interrupts

The ARM Cortex A9-MP Interrupt Controller permits all implemented interrupts to be individually defined as secure or non-secure.

Non-secure interrupts are always signaled using the IRQ mechanism of a Cortex-A9 processor.

Secure interrupts can be programmed to use either the IRQ or FIQ interrupt mechanism of a Cortex-A9 processor through the FIQen bit in the ICPICR register. Note that it is not permissible to use both FIQ and IRQ within a secure system. You must choose one or the other. Refer to the *ARM Cortex-A9 MPCore Technical Reference Manual* [Ref 4] for details.

Interrupt Priority

The Cortex-A9 processor used in Zynq-7000 AP SoC devices implements a five-bit interrupt priority format as described in the *ARM Generic Interrupt Controller Architecture Specification*. Secure world software can access all five of these bits, while normal world software can only access four bits. Thus, it is possible to define the secure interrupt with a priority that is higher than every other normal world interrupt. Such an approach better ensures that safety or security critical applications run as needed and when needed, typically without risk of preemption by normal world software.

Securing the Interrupt Controller

The Interrupt Controller provides the facility to prevent write accesses to critical configuration registers when you assert CFGSDISABLE. This signal controls write behavior for the secure control registers in the distributor and Cortex-A9 processor interfaces, and the Lockable Shared Peripheral Interrupts (LSPIs) in the Interrupt Controller.

When CFGSDISABLE is High, the Interrupt Controller prevents write accesses to the following registers:

Distributor:

- ◦ The secure enable of the ICDDCR

Secure interrupts defined by LSPI field in the ICDICTR:

- Interrupt Security registers
- Interrupt Set-Enable registers
- Interrupt Clear-Enable registers
- Interrupt Set-Pending registers
- Interrupt Clear-Pending registers
- Interrupt Priority registers
- ICDIPTR
- Interrupt Configuration register

Cortex-A9 interrupt interfaces:

- The ICCICR, except for the EnableNS bit

After you assert CFGSDISABLE, it changes the register bits to read-only and therefore the behavior of these secure interrupts cannot change, even in the presence of rogue code executing in the secure domain. Once set, this register can only be cleared by a POR.

Securing Memory Access

As described at the beginning of this document, ARM TrustZone enforces the concept of the *secure world* and the *normal world*. The secure world has unrestricted access to all system resources, and the normal world has access only to a defined hardware subset.

As previously described, the Xilinx Zynq-7000 family of devices includes the configuration register TZ_DDR_RAM which allows developers to independently define 64 MB sections of the DDR physical address space as either secure or non-secure.

Although the secure world has the ability to access normal world physical memory locations, those accesses are effectively blind because the secure world knows nothing about the location of the normal world page table, nor the contents, nor the memory structure of the data within the physical/virtual memory locations. Except for memory regions that might be explicitly defined by developers as shared memory regions, all other normal world memory data appears as uncorrelated 1s and 0s to secure world software. While it should certainly be possible for secure world software to determine such details, this exercise is not trivial and would require the ability to modify the secure world software itself.

In consideration of this, concerns related to secure world access to normal world address locations can focus on ensuring that the secure world does not inadvertently over-write normal world locations.

The following section provides a short discussion on considerations for secure world access to shared memory regions.

Cache

The Zynq-7000 AP SoC includes an ARM PL310, providing key features that benefit software running in the secure world. This includes the ability to stop specified cache lines from becoming locked by normal world software. Such a precaution restricts the ability of normal world software to launch one type of denial of service attack against the secure world software.

To understand the proper usage of cache, it is important to understand two key points related to the PL310 implementation:

• The PL310 loads cache lines based on the physical memory address

• The PL310 treats the security bit as a 33rd physical address bit

With this implementation, it is possible for a single unique DDR memory location to occupy two different cache lines – one corresponding to the secure world software and another corresponding to the normal world. Such an ambiguous condition likely becomes problematic due to cache coherency problems in system designs. Fortunately, such coherency problems can be avoided by ensuring that secure world software modifies its page tables so that all access to locations that are also used by normal world software are performed by the secure world, as non-secure accesses.

Enabling Secure Access to the PL: A Case Study

describes how to configure a PL-based design for secure access.

In the following case study we use a block RAM-based design in the Programmable Logic of Zynq-7000 AP SoC with the clock source provided by the FCLK output from the Processing System-7 IP core. The AXI block RAM controller IP from the Vivado IP catalog runs at the FCLK clock frequency and the IP is connected to the Processing System-7 IP via the GP0 Master Port of Processing System-7 IP. To get this design working, you need to make the clock connections shown in Table 8.

Table 8:   **Case Study Clock Connections**

PS-PL Interface Name	Port Name	Clock Source
GP0/1 AXI Master	ACLK	User clock source
GP0/1 AXI Slave	ACLK	User clock source
HP0/1/2 AXI Slave	ACLK	User clock source
ACP port	ACLK	User clock source

The user clock source can be derived from FCLK or from an external clock source.

The secure/non-secure application needs to take care of a number of security-related settings for the GP port along with proper de-assertion of the GP port's reset signal.

In the control application, the software needs to do the following:

1. Write `0x1` to SECURITY_FSSW_Sx to allow non-secure access to propagate via the GP port in the non-secure mode.

2. De-assert the GP port reset using the FPGA_RST_CTRL register.

3. Mark non-secure region in DDR memory by writing into the TZ_DDR_RAM register.

4. Start the non-secure application.

Note that registers SECURITY_FSSW_S1, FPGA_RST_CTRL and TZ_DDR_RAM are System Level Control registers and can be accessed only in the secure mode of operation of the CPU.

The following procedures show the steps necessary to run the block RAM control application in non-secure mode. The steps have been verified on a Xilinx standalone OS and depending on user environment, all steps might not be necessary.

main():

- init_platform() Initialize platform
  - D-cache enable (L1, L2)
- pl_access_setup() Initialize PL access in non-secure mode
  - unlock SLCR register slcr_unlock (address: `F8000008`) to `0xDF0D`)
  - set FPGA Reset Control register FPGA_RST_CTRL (address: `F8000240`) to `0x0`
  - set register security_gp0_axi (address: `F890001C`) bit0 to `1`.
  - lock SLCR register slcr_unlock (address: `F8000004`) to `0x767B`
- configure_secure_world() Configure the system to non-secure mode
  - board_init
    - initialize TZ_DDR_RAM
      initialize TZ_SECURITY_SDIO2
      initialize TZ_SECURITY_SDIO1
      initialize TZ_SECURITY4_QSPI
      initialize TZ_SECURITY6_APBSL
      initialize TZ_SECURITY5_MIOU
      initialize TZ_SECURITY7_SMC
      initialize TZ_DMAC_RST_CTRL
      initialize TZ_DMA_NS
      initialize TZ_DMA_IRQ_NS

initialize TZ_DMA_PERIPH_NS

initialize TZ_GEM

initialize TZ_SDIO

initialize TZ_USB

initialize TZ_FPGA_M

initialize TZ_FPGA_AFI

initialize TZ_OCM_RAM_0

initialize TZ_OCM_RAM_1

initialize TZ_OCM_ROM

- mpcore_init

SCU initialization

GIC IRQ vectors initialization

GIC CPU base initialization

enable interrupts

- monitor_init

prepare monitor (assembly instructions, register Monitor handler)

invoke SMC instruction (callSMC() function)

◦ callSMC

- set destination address to non-secure application (ping or loopback or so)

- invoke SMC instruction

- this invokes the smc_handler

◦ smc_handler

- clear monitor

- setup TLB to allow non-secure DDR accesses

- configure TTBR to allow non-secure DDR accesses (banked registers)

- configure Domain access to allow non-secure DDR accesses (banked registers)

- read contents of banked registers while in secure mode

- temporarily switch to non-secure mode

- setup TLB in non-secure mode

- setup TTBR in non-secure mode (banked registers)

- configure Domain access to allow non-secure DDR accesses (banked registers)

- update contents of banked registers in non-secure mode from secure mode

- if selected non-secure mode, continue CPU in Non-secure mode

- else, switch CPU back to Secure mode

- invoke _custom_boot

◦ _custom_boot()

- set vector table in non-secure mode

- set stack areas for each of the vectors

- invoke _cstart (equivalent to calling main() function)
  ◦ _cstart()
    - clear BSS area
    - set initial stack frame
    - run global constructor (initial stack frame before invoking application)
    - invoke application code
  ◦ application()
    - perform application specific tasks

Enabling Non-secure Access for the PS GEM, SDIO, and USB Controller

GEM

To allow non-secure access to the GEM, the following TrustZone registers must be configured in secure world before starting any non-secure world operation for GEM.

1. Set the GEM slave security settings to 1 to allow non-secure access:

    a. security6_apb_slaves .gem0_s_apb = `0x1` // GEM controller 0

    b. security6_apb_slaves .gem1_s_apb = `0x1` // GEM controller 1

2. Set the GEM Trustzone configuration register to non-secure access mode:

    a. Slcr.TZ_GEM.E0 = `0x1` //Non-secure access for GEM controller 0

    b. Slcr.TZ_GEM.E1 = `0x1` //Non-secure access for GEM controller 1

The above settings allow non-secure memory accesses for the GEM master.

Apart from above settings, the clock and hard reset for the peripheral are also configured through the secure world only.

To enable/disable the GEM clock or to change the default clock settings, the following register must be configured in secure world:

◦ SLCR.GEMx_CLK_CTRL

   - CLKACT - To enable / disable the clock

   - iSRCSEL - Source of the clock domain

   - DIVISOR - First divisor for GEM controller

   - DIVISOR1 - Second divisor for GEM controller

To hard reset the GEM controller, the following register must be configured in the secure world.

  ◦ SLCR.GEM_RST_CTRL

All other GEM controller configuration and enabling should be done in the non-secure world.

Example

To enable GEM controller 0 in non-secure world with the default clock, the following registers must be configured in secure mode:

  ◦ security6_apb_slaves .gem0_s_apb

  ◦ SLCR.TZ_GEM.E0 = `0x1`

  ◦ SLCR.GEM0_CLK_CTRL.CLKACT = `0x1`

  ◦ Other GEM controller settings are done in non-secure mode (configuration, enabling, transfer trigger).

SDIO

To allow non-secure access to SDIO, the following TrustZone registers must be configured in secure world before starting any non-secure world operation for SDIO:

1. Set SDIO slave security settings to `1` to allow non-secure access:

   a. gpv_iou_switch.security2_sdio0 = `0x1`   // SDIO controller 0

   b. gpv_iou_switch.security3_sdio1 = `0x1`   // SDIO controller 1

2. Set the SDIO TrustZone configuration register to non-secure access mode:

   a. Slcr.TZ_SDIO.S0 = `0x1`   // Non-secure access for SDIO controller 0

   b. Slcr.TZ_SDIO.S1 = `0x1`   // Non-secure access for SDIO controller 1

Above settings allow non-secure memory accesses for the SDIO master.

Apart from the above settings, the clock and hard reset for the peripheral are also configured through secure world only.

To enable/disable SDIO clock or to change the default clock settings, following SLCR register must be configured in secure world:

  ◦ SLCR.SDIO_CLK_CTRL

    - CLKACT - To enable / disable the clock

    - SRCSEL - Source of the clock domain

- DIVISOR - Divisor for the SDIO controller

To hard reset the SDIO controller, the following register must be configured in secure world:

○ SLCR.SDIO_RST_CTRL

All other SDIO controller configuration and enabling should be done in non-secure world.

Example:

To enable SDIO controller 0 in non-secure world with the default clock, the following registers must be configured in secure mode:

- ○ gpv_iou_switch.security2_sdio0 = `0x1`
- ○ SLCR.TZ_SDIO.S0 = 0x1
- ○ SLCR.SDIO_CLK_CTRL.CLKACT0 = `0x1`
- ○ Other SDIO controller settings are done in non-secure mode (configuration, enabling, transfer trigger).

USB Controller

To allow non-secure access to the USB controller, the following TrustZone registers must be configured in secure world before starting any non-secure world operation for USB:

1. Set USB slave security settings to `1` to allow non-secure access:
   a. gpv_iou_switch. security6_apb_slaves. usb0_s_apb = `0x1` // USB controller 0
   b. gpv_iou_switch. security6_apb_slaves. Usb1_s_apb = `0x1`   // USB controller 1

2. Set the USB TrustZone configuration register to non-secure access mode:
   a. Slcr.TZ_USB.U0 = `0x1` //Non-secure access for USB controller 0
   b. Slcr.TZ_USB.U1 = `0x1` //Non-secure access for USB controller 1

The above settings allow non-secure memory accesses for USB.

Apart from above settings, the clock and hard reset for the peripheral are also configured through the secure world only.

To enable/disable the USB clock or to change the default clock settings, the following register must be configured in secure world:

- ○ SLCR.USBX_CLK_CTRL
  - CLKACT - To enable / disable the clock
  - SRCSEL - Source of the clock domain

- DIVISOR0 - First divisor for USB controller

- DIVISOR1 - Second divisor for USB controller

To hard reset the USB controller, the following register must be configured in secure world:

○ SLCR.USB_RST_CTRL

All other USB controller configuration and enabling should be done in non-secure world.

Example:

To enable non-secure access for USB controller 0 with the default clock, the following registers must be configured in secure mode:

○ gpv_iou_switch. security6_apb_slaves. usb0_s_apb = `0x1`

○ SLCR. TZ_USB.U0 = `0x1`

○ SLCR. USB0_CLK_CTRL. CLKACT = `0x1`

○ Other USB controller settings should be done in non-secure mode (configuration, enabling, transfer trigger).

Additional Resources

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see the xilinx support website.

For continual updates, add the Answer Record to your myAlerts.

For definitions and terms, see the Xilinx Glossary.

Solution Centers

See the Xilinx Solution Centers for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

References

These documents/websites provide supplemental material useful with this guide:

1. *Zynq-7000 AP SoC Technical Reference Manual* (UG585)

2. ARM Architecture Reference Manual

3. CoreLink DMA-330 DMA Controller Technical Reference Manual

4. ARM Cortex -A9 MPCore Technical Reference Manual

5. Zynq-7000 AP SoC - TrustZone Answer Record (AR54835)

6. Third-party documents

7. ARM's TrustZone website

8. ARM Architecture Reference Manual