CHAPTER 3

Zynq-7000 Adaptive SoC Architecture

3.1 Introduction

Chapter 2 provided an overview of the ARM Cortex-A9 MPCore, the fundamental processing unit in the Zynq-7000, but this versatile SoC contains three other major elements — the Application Processing Unit (APU), the Processing System (PS), and the Programmable Logic (PL). The interface between the PS and PL is also of critical importance, as it ensures that there are no performance bottlenecks between each section. All of these components will have been evident when the device was introduced as a simple embedded platform in Chapter 1 (Section 1.3), but in this chapter each will be presented in much more detail. The principle concepts of Zynq-7000 boot and configuration will also be discussed, followed by a thorough overview of the debug system, which is based on ARM CoreSight technology.

The discussion starts with the block diagram of the Zynq-7000 SoC in Figure 3.1. Each major element is summarised below:

1. **Application Processing Unit**: The ARM Cortex-A9 MPCore is integrated in the Application Processing Unit, along with on-chip memory, Level 2 cache, two triple timer counters, a system watchdog timer, an 8-channel DMA, and a system-level control register block. The APU is covered in Section 3.2.
2. **Processing System**: The next level in the architecture is the Processing System. This contains the APU, memory controllers for DRAM and SRAM/Flash, an I/O Peripheral Unit, and a range of logic blocks for clocking, reset, configuration, boot, and debug. The PS is covered in Section 3.3.
3. **Programmable Logic**: The programmable logic is the next major element in the Zynq-7000. An overview of key Xilinx 7-Series FPGA features will be presented in Section 3.4, including the CLB, clock management, Block RAM, DSP, SelectIO, and Serial Transceivers.

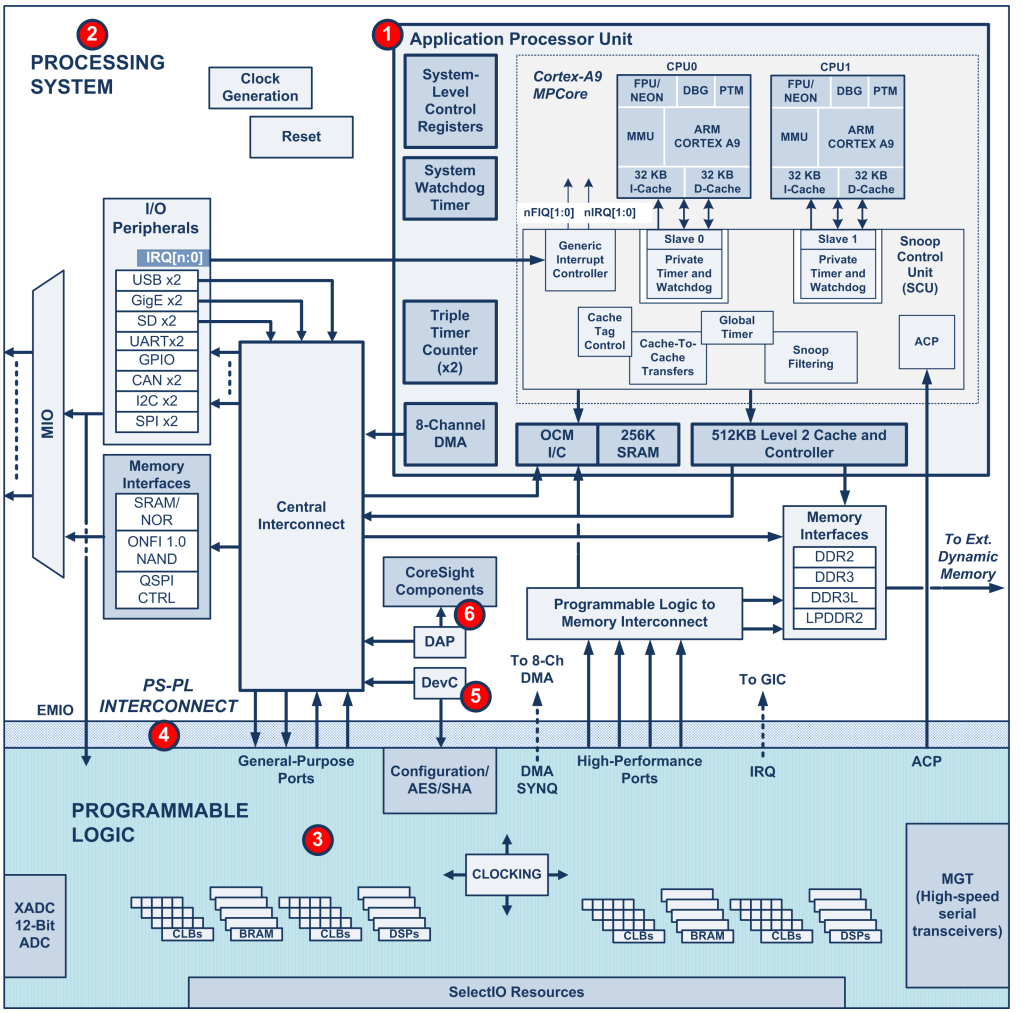

Figure 3.1. Simplified block diagram of the Zynq-7000 Adaptive SoC, illustrating the topics to be covered in this chapter.

4. **PS-PL Interconnect:** To achieve the high-performance advantages offered by the Zynq-7000, the processing system and programmable logic must be tightly interconnected. The PS-PL interface consists of a wide range of ports for high-performance and general-purpose computing, as well as clock/reset signals, interrupt routing, extended I/O signals, and debug functionality. The PS-PL Interconnect is covered in Section 3.5.
5. **Boot and Configuration**: This important topic is covered in Section 3.6, with three boot options being considered: (a) Booting an operating system; (b) Booting a bare-metal application; (c) JTAG development mode.
6. **Zynq-7000 Debug System**: The debug system in the Zynq-7000 is based on Arm CoreSight technology, and this highly-effective (but complex) approach is discussed in detail in Section 3.7.

The first of these elements to be discussed is the Application Processing Unit.

3.2 Application Processing Unit

3.2.1 ARM Cortex-A9 MPCore

The APU integrates the ARM Cortex-A9 MPCore with on-chip memory, L2 Cache, DMA, timers, and system control registers to become the centre of the Zynq-7000 processing system. The MPCore (labelled '1' in Figure 3.2) implements all the functionality discussed in the previous chapter: cache coherency; a private timer and WDT for each core; a shared global timer; a generic interrupt controller; and an ACP port. Some of the major MPCore configuration options for the Zynq-7000 family are as follows:

1. Single-core devices (e.g. Z-7012S) have one Cortex-A9 core enabled, and dual-core devices (e.g. Z-7020) have two Cortex-A9 cores enabled.
2. Each core includes 32 KB L1 instruction and data caches with parity.
3. Each core includes the NEON coprocessor (media processing engine), and the VFPv3 floating-point unit.

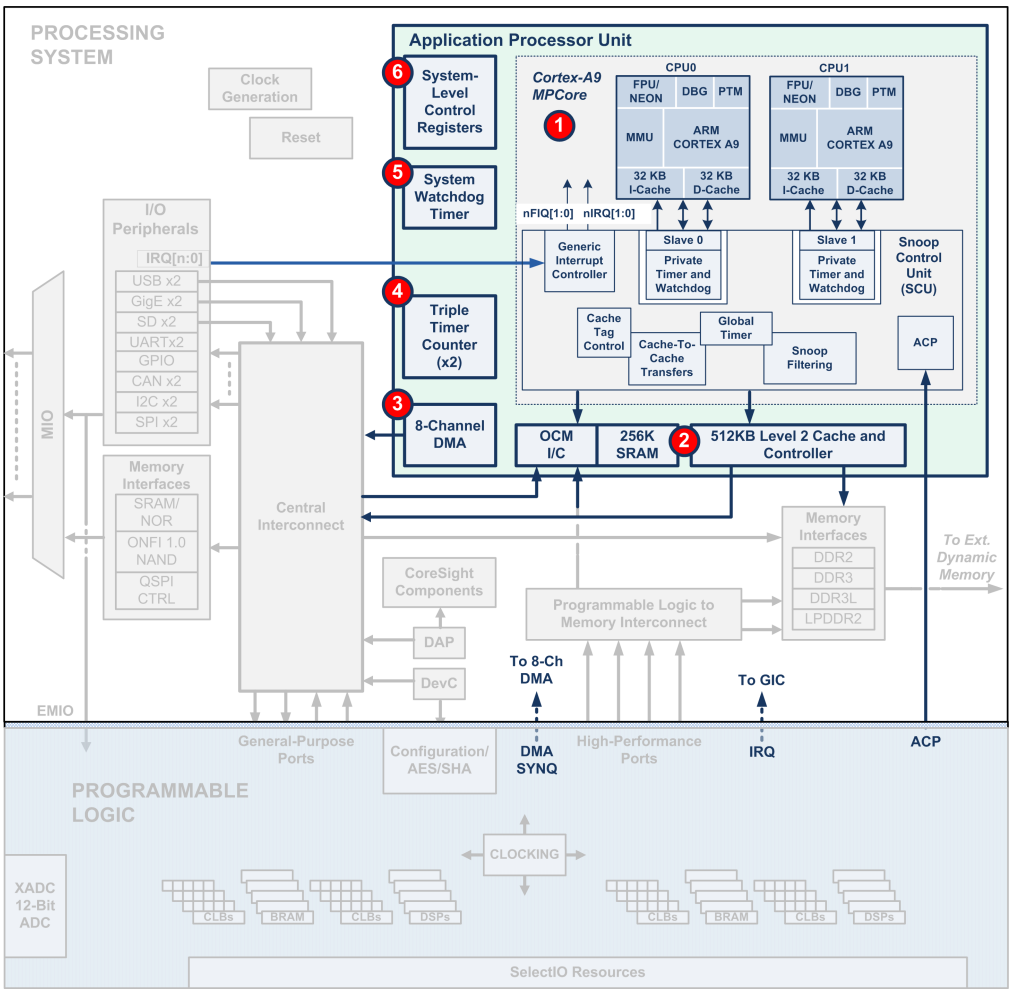

Figure 3.2. Zynq-7000 Application Processing Unit

4. Two master interface ports are available: M0 connects to the APU OCM, and M1 connects to the APU L2 Cache.
5. The Accelerator Coherency Port (ACP) is included in the solution.
6. The Generic Interrupt Controller is based on version 1.0 of the GIC specification, and it supports 64 shared peripheral interrupts, 5 private interrupts and 16 software generated interrupts. The GIC is covered in detail in Chapter 11.
7. The Programmable Trace Macrocell (PTM) is included for integration with CoreSight systems.

3.2.2 APU Memory Features

The following memory functionality (labelled '2' in Figure 3.2) is incorporated in the APU:

- **L2-Cache controller**: Based on the ARM PL310 [38], the 512 KB L2 cache operates in an 8-way set-associative manner.
- **256KB on-chip memory RAM**: This is initially used during the boot process, with the accessible memory reduced to 192 KB. The full 256 KB is then available when the boot sequence is complete.
- **128KB on-chip memory ROM**: Also critical to the boot process, the 128 KB ROM is effectively invisible to the user (and is not shown in Figure 3.2).

3.2.3 Direct Memory Access

An 8-channel DMA, based on the ARM PL330 [74], is available to move large amounts of data to and from memory without processor intervention. The interface paths are memory-to-memory, memory-to-peripheral, and peripheral-to-memory. The interconnect type is 64-bit AXI, and the controller is clocked at CPU_2x (which is 222 MHz in the hardware configuration in this text, see Section 3.3.3.1). Four of the eight channels are dedicated to the programmable logic.

3.2.4 2x Triple Timer Counters

Two triple-timer counters are provided in the APU, adding to the private and global timers already available in the MPCore. Each module consists of three 16-bit up-down counters which generate individual wave output and interrupt signals; the interrupts can be configured in interval, match, or overflow mode.

In contrast to the private and global timers in the MPCore, which are always available, each TTC must be enabled in Vivado when building the hardware design — this will be seen in Chapter 6. The TTC will also be covered in much greater detail in Chapter 12, where it is the focus of the entire chapter.

3.2.5 System Watchdog Timer

Along with the triple timer counters, a 24-bit system-level watchdog timer (SWDT) is integrated in the APU. This is more flexible than the private WDTs in the MPCore, as it can be clocked by three different sources (PS CPU_1x, external MIO, or external EMIO), and there are more output signal options on timeout (PS, PL or MIO output, and/or PS interrupt)

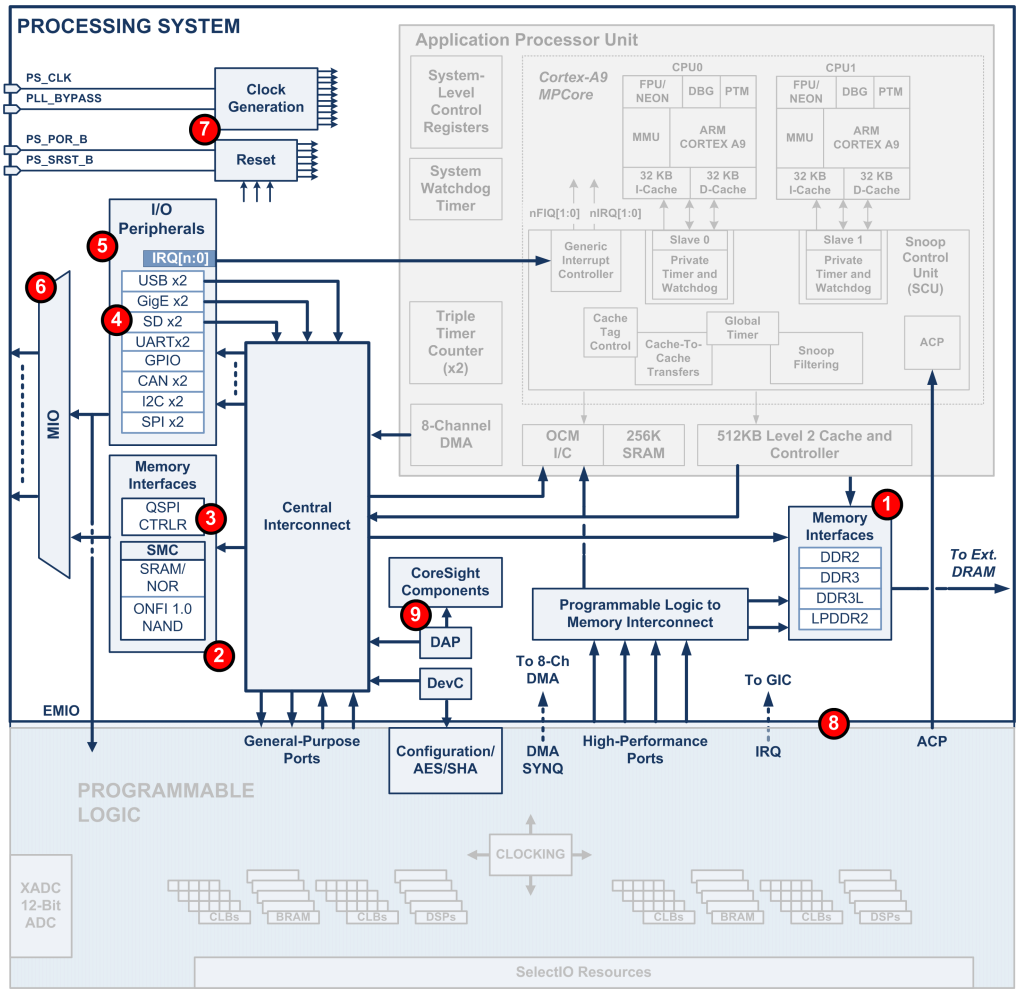

Figure 3.3. Zynq-7000 Processing System

3.2.6 System-Level Control Registers

Finally, a bank of system-level control registers is integrated in the APU. These are used to configure many fundamental hardware features in the device, such as clock and reset behaviour, multiplexed I/O (MIO) routing, and DDR controller settings. Note, however, that in the simple projects in this text, the user does not have to manually manipulate these registers; the relevant settings are configured transparently during the design process in Vivado.

3.3 Processing System

The heart of the Processing System is the APU, just discussed above, but the PS also contains the following major components and interfaces (see also Figure 3.3):

- Memory interface blocks for dynamic and static RAM.
- An I/O Peripheral unit and related I/O pin multiplexer.

- Clock and reset logic.
- PS-PL interconnect.
- CoreSight debug components.

The memory interfaces, IO peripheral unit, and clock/reset functionality will be discussed in this section. The PS-PL interconnect (labelled '8' in Figure 3.3) will be covered in Section 3.5, and the debug functionality (labelled '9' in Figure 3.3) will be covered in Section 3.7.

The discussion starts with the memory interfaces in the processing system.

3.3.1 Memory Interfaces

The typical memory hierarchy for an application-profile processor was summarised in Section 1.3.2. As a brief reminder, SRAM is high-density memory used where speed is critical, such as the processor register set and on-chip memory/caches. In contrast, DRAM is slower, but it offers high capacity at a low cost per bit, making it a common choice for main memory. The third fundamental storage type is non-volatile memory, used for system boot, OS file systems, virtual memory, and general data storage. The memory options in the Zynq-7000 processing system are summarised in this section, starting with the DRAM controller.

3.3.1.1 Dynamic DDR Controller

The DDR memory controller (labelled '1' in Figure 3.3) supports DDR3, DDR3L, DDR2 and LPDDR2 memory, with 32-bit or 16-bit data buses (ECC can be enabled in the latter). The addressable space is 1 GB, and the maximum speed is 666.67 MHz (for DDR3 on the fastest grade devices). There are three main access paths to this shared memory:

1. The ARM CPU has access via the L2 cache controller in the MPCore.
2. Two 64-bit AXI ports are available for programmable logic access.
3. A 64-bit AXI interface links the Central Interconnect with the DDR controller, providing access to relevant AXI masters in the system.

3.3.1.2 Static Memory Controller

The Static Memory Controller (labelled '2' in Figure 3.3) is based on the ARM PL353 SMC [75], providing two main memory interfaces in the system. The first interface supports both asynchronous SRAM (which is volatile) and NOR flash (which is non-volatile), while the second interface supports NAND flash (also non-volatile). The SRAM option on interface 1 can be used as an alternative to DRAM in the system, but, because the capacity of SRAM is low compared to DRAM (64 MB versus 1 GB, for example), most systems will use the DRAM interface. However, the option is there should the user require it. The non-volatile SMC options are discussed below.

3.3.1.3 Non-Volatile Memory

As just mentioned, two flash memory options are provided by the static memory controller: NOR flash on Interface 1, and NAND flash on Interface 2. NOR flash has lower capacity than NAND flash, but the technology is more reliable (as discussed in Section 1.3.2). In the Zynq-7000, the NOR flash interface supports up to two 32 MB devices on an 8-bit data bus. The NAND flash interface, on the other hand, supports flash sizes up to 1GB, with bus widths of 8- or 16-bit. NAND flash can also be configured with 1-bit ECC, and it is compatible with Open NAND Flash Interface, Specification 1 (ONFi 1.0).

Aside from the SMC, other more practical non-volatile memory options are available in the processing system. The first of these is the Quad-SPI interface (labelled '3' in Figure 3.3), , which can access up to two 16MB serial flash devices in a range of modes: 4-bit single slave-select (SS), 8-bit parallel dual SS, 4-bit dual SS stacked, and single legacy SPI (1x, 2x, 4x, 8x).

Two SD/SDIO controllers (labelled '4' in Figure 3.3) are also available in the IOP, allowing SD and MMC memory cards to be used in the system. eMMC devices are also nominally supported, but they cannot be used for primary boot. The SD/SDIO controllers have an integrated DMA engine for enhanced performance, and they are compatible with SDIO Specification 2.0 and MMC3.31.

3.3.2 I/O Peripherals and MIO/EMIO

Table 3.1 summarises the full range of I/O peripherals available in the PS. (The IOP unit is labelled '5' in Figure 3.3.) An important point to note is that only 54 I/O pins are available in the processing system, meaning that the pins must be shared in a multiplexed I/O (MIO) fashion (labelled '6' in Figure 3.3).

To compensate for the lack of pins, many of the peripheral I/O signals can also be routed to the programmable logic — this is known as extended multiplexed I/O (EMIO). In the PL, they can be used in the digital design or routed to top-level FPGA pins. Note that while the MIO/EMIO functionality is completely configurable, decisions regarding peripheral use and MIO pin designation should be taken early in the project (certainly at or before board design time).

Peripheral	Comments	DMA	EMIO
GigE 0, 1	10/100/1000 EMAC, IEEE 802.3, IEEE 1588 (2.0)	Yes	Partial
USB 0, 1	USB 2.0 OTG, high-speed/full-speed, host/device/OTG	Yes	No
SDIO 0, 1	SD/SDIO 2.0 compliant	Yes	Yes
SPI 0, 1	4-wire; up to 3 slave-selects; 50MHz (MIO); 25MHz (EMIO)	No	Yes
I2C 0, 1	I2C Bus Specification 2.0; up to 400 Kb/s	No	Yes
UART 0, 1	Full-duplex asynchronous receiver/transmitter	No	Yes
CAN 0, 1	CAN 2.0-A; CAN 2.0-B; ISO 11898-1	No	Yes
GPIO	Two banks, up to 118 I/O	No	Yes

Table 3.1. Overview of PS7 peripherals

Some general points about the I/O peripherals and MIO/EMIO functionality are listed below (the reader should refer to the TRM [39] for complete details):

- The SD/SDIO controller, USB, and Gigabit Ethernet peripherals have built-in DMA.
- The static memory (NAND/NOR/SRAM) and QSPI controllers cannot be routed to EMIO.
- There are two PS MIO voltage banks: Bank 0 (MIO[15:0]), and bank 1 (MIO[53:16]). Each bank can be independently configured for either 1.8V or 2.5V/3.3V signalling, using voltage pin-strapping.
- The MIO[8:2] pins are dual-purpose pins. Initially, they are used as inputs pins at board power-up, defining options such as the boot device (e.g. JTAG or flash) and MIO bank voltage settings. After the boot process, the pins revert to their designated peripheral I/O function.

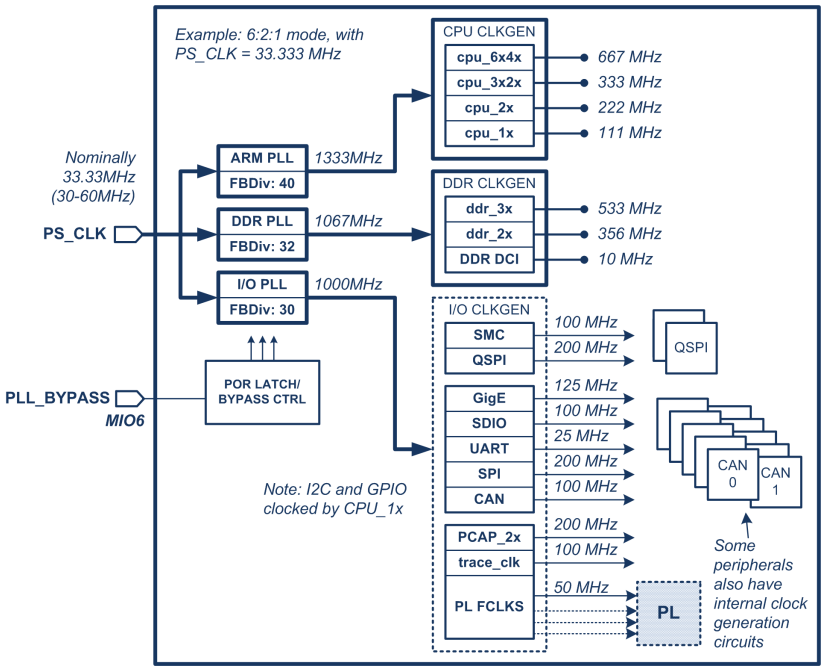

Figure 3.4. Typical processing system clock details for a 33.33MHz input clock and 6:2:1 frequency ratio settings. The PLLs are enabled (i.e. PLL_BYPASS is low).

- In normal operation (i.e. after the boot sequence), MIO pins 7 and 8 can only be used as outputs.
- If the SD Card boot option is required in the system, MIO[45:40] must be used for this interface. Card detect functionality can be supported using any other available MIO pin.

Several of these topics will be revisited in Section 3.6.4

3.3.3 Clock and Reset Generation

The processing system clock and reset blocks are labelled '7' in Figure 3.3. The PS clock logic will be discussed first in Section 3.3.3.1, followed by the reset sources in Section 3.3.3.2.

3.3.3.1 Clock Generation

The processing system requires a single-ended LVCMOS clock in the range of 30-60 MHz, which connects to the *PS_CLK* input as shown in Figure 3.4. This clock feeds dedicated PLLs for each of the major sub-systems in the processing system: ARM (i.e. CPU), DDR, and I/O. The PLLs can also be bypassed if necessary, resulting in a much lower PS operating frequency; this might be required in low-power designs, or to support debugging (for example). The PLL Bypass pin, MIO6, is sampled during the power-on reset sequence, and the PLLs will be bypassed if the pin reads as a logic '1'. Normally this option is not selected, though, and the typical system clock frequencies for a 33.33 MHz clock are also shown in Figure 3.4. Clock generation for each subsystem can be summarised as follows:

- The ARM PLL feeds a clock generator which outputs four CPU clocks ranging from 667 MHz down to 111 MHz. The higher frequency clocks are used to drive main processing elements such as the CPUs, SCU, OCM and L2 cache. The lower frequency clocks are used mainly for interconnect clocking in the PS.
- The DDR clocks are used mainly by the DDR DRAM controller, although they are also used for some HP AXI interfaces in the processing system.
- The I/O clocks are used as the base clock for most peripherals.

The PS also supplies four independent, asynchronous clocks to the programmable logic (*FCLKCLK[3:0]*). These clocks are typically generated by the I/O PLL, but the DDR or CPU PLL can also be used. Of course, the programmable logic can also be clocked using dedicated pins in the SelectIO banks, just like any Xilinx FPGA.

The Zynq clocking system supports two different frequency ratios, 6:2:1 and 4:2:1. A ratio of 6:2:1 provides higher CPU performance in the system, as the CPU and related components are clocked at a faster rate. Conversely, the 4:2:1 ratio may be a better choice when increased interconnect bandwidth is desired, as the interconnect clocks run faster at this setting (at the expense of lower CPU frequencies).

Finally, while the clock system in the Zynq-7000 may seem somewhat complex, the settings can be left at their default values for simple projects (such as we have in this text). Also, if the user really does need to change any settings, this can be easily carried out when configuring the Zynq-7000 IP in Vivado.

3.3.3.2 Reset System

Figure 3.5 summarises the reset sources in the Zynq-7000. The chip master-reset is *PS_POR_B*, which initiates the power-on reset sequence when asserted (low) — this resets all registers and RAM in the device, clears the programmable logic and the debug environment, and starts the boot routine. This reset should only be released when the platform voltages and clocks are stable; a power-good signal generated by the power distribution network (PDN) is often used for this purpose. (This will be seen in Section 4.2 when the PDN for the Zybo-Z7-20 is discussed).

The other external reset signal is System Reset, *PS_SRST_B* , which initiates a modified reset sequence — the boot-strap pins are not (re-)sampled, for example, and the debug environment and other persistent registers are not cleared.

In addition to the external reset signals, there are also a range of internal reset sources; these are briefly summarised below. Refer to Chapter 26 in the Zynq-7000 TRM [39] for more comprehensive details.

System Software Reset

The system software reset has the same effect as *PS_SRST_B*. The user can control this reset signal using bit 0 (*SOFT_RST*) of the *PSS_RST_CTRL* register in the SLCR.

Watchdog Timers

The CPU private WDTs (Section 2.6.5) can be configured to carry out a full system reset, or just a reset of the associated CPU. The system watchdog timer (Section 3.2.5) can be used to reset the system. (As a reminder, a system reset means that persistent registers and the debug logic are *not* cleared.)

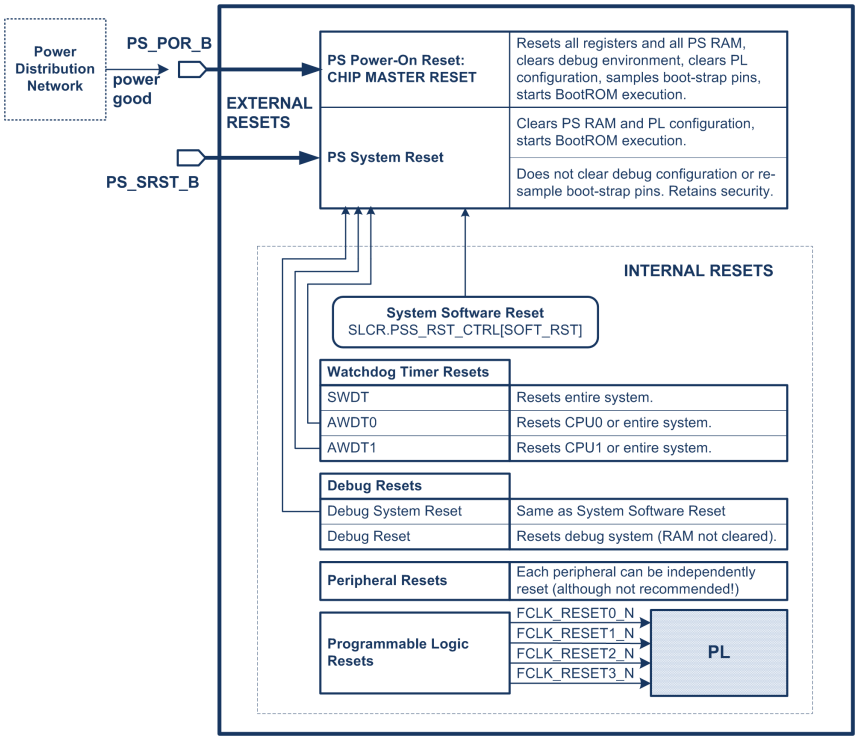

Figure 3.5. Processing System Reset Details

Debug Resets

There are two debug resets, each of which originate in the ARM debug access port (see Section 3.7.1.1.1). The Debug System Reset has the same effect as *PS_SRST_B*, while the Debug Reset just clears the debug environment.

Peripheral Resets.

Each peripheral can be reset using associated settings in the SLCR. However, the results are unpredictable when a program is running, and this functionality is not recommended for inexperienced users.

Programmable Logic Resets

Four reset signals (*FCLKRESETN[3:0]*) are routed to the programmable logic from the PS, each of which is loosely associated with the PL clock of the same number (*FCLKCLK[3:0]*). For a reset to be propagated out of the PS, the corresponding FCLK must be toggling. Note, though, that the resets are asynchronous to the associated clock, and the user must synchronise the reset(s) in the PL. This is an easy task when using IP Integrator in Vivado, as a predefined module called the Processor System Reset IP can be instantiated in the PL design. This will be seen in Section 6.7.6.

3.3.4 Processing System Interconnect

The fundamental elements in an embedded system have always been the processor(s), memory components and I/O peripherals. Early computing systems used simple buses to

connect all of these components together, but modern embedded devices require a more standardised solution. With this in mind, Arm has been providing the Advanced Microcontroller Bus Architecture (AMBA) for designers since the mid-1990s. Originally consisting of just a peripheral and system bus (APB and ASB respectively), the architecture has evolved to include a wide range of high-performance buses, along with other products such as interconnect switches, memory controllers and cache controllers. The interconnect framework in the Zynq-7000 SoC is based on Version 3.0 of the AMBA standard, along with some AXI4 additions. The main points are summarised below:

- The ARM NIC-301 interconnect solution [76] is a fundamental component in the Zynq-7000 processing system. The main element is the central interconnect block, but it also includes peripheral, memory, and OCM interfaces, along with AHB and APB bridges.
- The L2 cache controller is based on the ARM PL310 cache controller [38], which is AMBA 3.0 compliant.
- The Advanced eXtensible Interface (AXI) bus [77] is widely used in the Zynq-7000 architecture. On the PS-side, 64-bit AXI3 buses are used where high performance is required, and 32-bit AXI3 buses are used for general-purpose/lower-performance tasks. In contrast, AXI4 is the primary interconnect in the programmable logic, where its main function is to connect the components in the user design together. AXI4 (full), AXI4-Lite, and AXI4-Streaming interfaces are available; the latter allows high-bandwidth streaming data transfers to be carried out.
- The Advanced High-performance Bus (AHB) [78] is used for high-bandwidth PS peripherals such as USB, Ethernet and SDIO.
- The remaining lower-speed peripherals in the processing system use the Advanced Peripheral Bus (APB) [78].
- Finally, the Advanced Trace Bus (ATB) [79] is central to the CoreSight Trace and Debug implementation, where it is used to collect and transfer trace data to memory or off-chip. This interface bus will be seen again in Section 3.7.

This completes the discussion on the processing system; next we move to the other primary section in the Zynq-7000, the programmable logic.

3.4 Programmable Logic

A history of Xilinx programmable logic devices was presented in Chapter 1, where, even in the early days, the fundamental characteristics of Xilinx programmable logic could be clearly seen. For example, the XC2064 (the first Xilinx device) contained three main programmable features: a configurable logic block, an I/O block, and programmable interconnect. These features were still a fundamental part of the FPGA when the 7-Series was released in 2011, but the device had evolved significantly in many other areas. The CLBs contained carry, memory, and wide-multiplexer logic, and the I/O blocks supported a vast array of modern standards. Dedicated BlockRAM memory was available on-chip, along with advanced DSP slices that featured a 25x18 multiplier and a 48-bit accumulator. Complex clocking structures had become the backbone of the device, with highly-configurable PLL circuits providing wide-fanout clock signals for the digital design. In addition, the more powerful devices in the 7-Series contained advanced high-speed serial I/O blocks capable of

running at 28.5 Gbps. And, of course, devices such as the Zynq-7000 featured powerful dual-core ARM Cortex-A9 processors

Xilinx FPGAs became even more advanced after the 7-Series, with the introduction of UltraScale+ and Versal devices; however, this book is about the Zynq-7000, and so the 7-Series will be used to illustrate the main characteristics of Xilinx programmable logic. Also, because the target platform in this text is the Digilent Zybo-Z7-20, the focus will be on the Zynq XC7Z020. In comparison to its 7-Series counterparts, the XC7Z020 lies somewhere between two higher-end Artix-7 devices, the XC7A75T and the XC7A100T (see Table 3.2). The features listed in the resource column of this table will become clearer as the discussion progresses.

Resource	Artix XC7A75T	Zynq XC7Z020	Artix XC7A100T
Logic Cells	75520	85120	101440
Slices	11800	13300	15850
SLICEL	8232	8950	11100
SLICEM	3568	4350	4750
6-Input LUT	47200	53200	63400
Distributed RAM (Kb)	892	1088	1188
Shift Register (Kb)	446	544	594
CLB Flip-Flops	94400	106400	126800
Block RAM/FIFO (36 Kb each)	105	140	135
Total Block RAM (Kb)	3780	4900	4860
CMTs (1 MMCM + 1 PLL)	6	4	6
Maximum Single-Ended I/O	300	200	300
DSP Slices	180	220	240

Table 3.2. Fundamental characteristics of the Zynq XC7Z020, which lies somewhere between the Artix XC7A75T and XC7A100T

3.4.1 Fundamental XC7Z020 Structure

Figure 3.6 shows a conceptual floor-plan for the Zynq XC7Z020. The layout is dominated by the programmable logic, with a smaller area reserved for the PS7 processing system. At a broad level, 7-Series FPGAs can be viewed as a collection of clock regions, divided by a vertical clocking backbone and a horizontal centre (although the horizontal centre isn't necessarily in the physical centre of the architecture, as can be seen). Each clock region also has a local horizontal clock row, dividing each region into an upper and lower part. The vertical clocking backbone can supply global clocks to any region, while the horizontal pathways have more limited (but still very flexible) routing options.

To implement the user design, each clock region contains the following fundamental resources::

- A SelectIO bank containing 50 I/Os, with 25 each in the upper and lower halves.
- Multiple CLB columns, each having 25 CLBs in each half, just like the I/O arrangement.

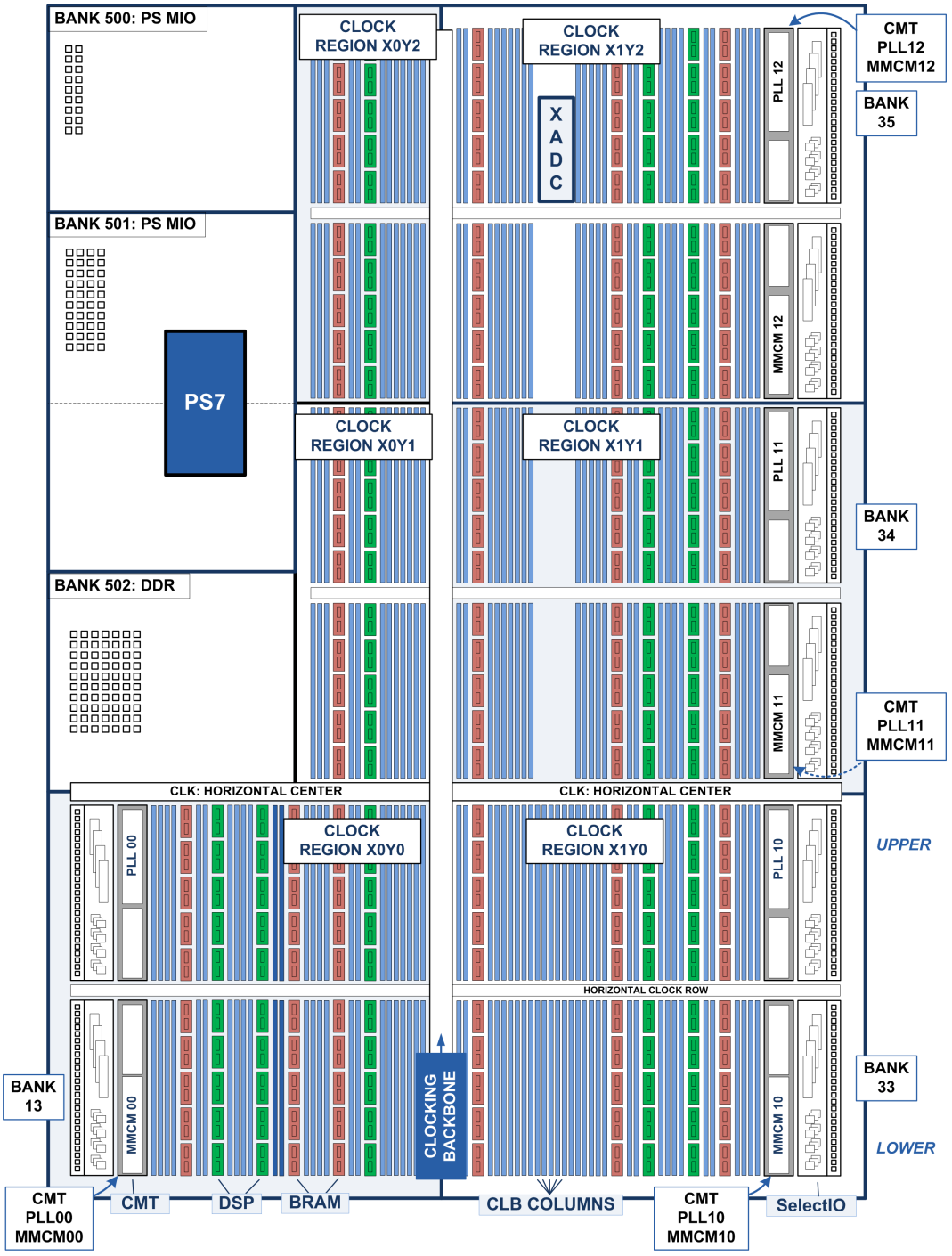

Figure 3.6. Xilinx XC7Z020 floor-plan

- A clock management tile (CMT) which contains a mixed-mode clock manager (MMCM) and a phase-locked-loop (PLL). The PLL is very similar to the MMCM, with the latter offering slightly more advanced functionality, to be discussed later.
- Two or three BRAM columns, with a total of ten 36 Kb BRAMs in each column. (Each BRAM can also be divided into two independent 18 Kb BRAMs.)
- Two or three DSP columns, with a total of ten DSP tiles in each column; each tile contains two DSP48E1 slices.

The logic for the user circuit will generally span multiple clock regions, although experienced FPGA designers can optimise their designs by constraining their circuits to make the best use of the closest resources. Returning to Figure 3.6, we can see that the Zynq XC7Z020 has six clock regions, although regions X0Y1 and X0Y2 are reduced in size to accommodate the hard processor core of the PS7. The full clock regions contain a SelectIO block and associated I/O bank, plus a CMT (with MMCM and PLL), as summarised in Table 3.3. The XADC is located in region X1Y2 (or, to be more specific, the XADC auxiliary inputs are in this region).

Clock Region	I/O Bank	CMT
X0Y0	13	PLL00 / MMCM00
X1Y0	33	PLL10 / MMCM10
X1Y1	34	PLL11 / MMCM11
X1Y2	35	PLL012 / MMCM12

Table 3.3. CMTs and related clock regions/IO banks in the Zynq XC7Z020

As just mentioned, clock regions X0Y1 and X0Y2 are reduced in size, freeing up the required space for the Zynq processing system. In this area, three I/O banks are available for PS pin connections: Banks 500 and 501 contain the PS MIO pins, and Bank 502 contains the DDR pins.

Depending on the package, not all I/O pads will be bonded out, even for the same version of silicon. For example, the Zynq-7000 device on the Zybo-Z7-20 platform is packaged as a CLG400 device, with I/O Bank 13 only partially bonded out, and Bank 33 not bonded out at all. In contrast, the Digilent/Avnet Zedboard uses a CLG484-packaged device, with all four banks fully bonded out. Some devices also contain GTP transceiver blocks for high-speed serial applications, which further affects the floor-plan (this will be covered in more detail in Section 3.4.7).

Each programmable logic feature in the XC7Z020 will be summarised next, starting with the configurable logic block.

3.4.2 Configurable Logic Block

The configurable logic block (CLB) is the main digital logic resource in Xilinx FPGAs, being used to implement sequential and combinatorial logic. They are the most plentiful resource in the device, generally occupying around 30 columns in a 7-series clock region, with 50 CLBS in each column. Taking into account the four full and two partial clock regions in the Zynq XC7Z020, this equates to 6650 CLBS. The CLB itself isn't the most fundamental component in the device, though; it actually contains two smaller elements called slices, as shown in the simplified illustration in Figure 3.7. Each slice contains the following:

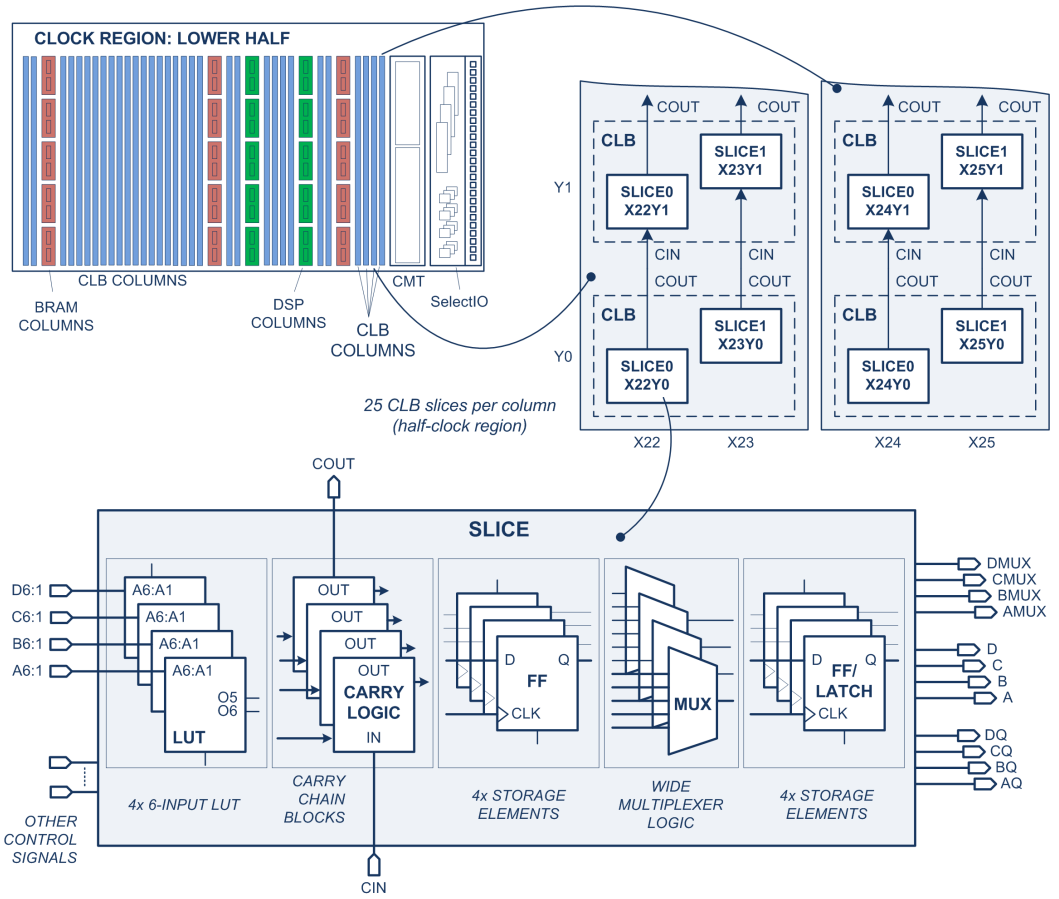

Figure 3.7. Simplified illustration of the 7-Series configurable logic block

Four Look-up Tables

The look-up tables (LUTs), sometimes known as function generators, are the main components used to implement user logic. They can be configured as 6-input LUTs with one output, or as dual LUTs with five (or fewer) inputs and two outputs. In the latter configuration, the outputs can also be registered in the slice storage elements (see next).

Eight Storage Elements

The eight storage elements in the slice are effectively flip-flops that can store one bit of information. Four of these elements can be configured as latches, but in that scenario the other four flip-flops cannot be used in the slice.

High-Speed Carry Chain

A high-speed carry chain is available for the implementation of efficient arithmetic logic or counters.

Wide Multiplexer Logic

A CLB slice also contains wide multiplexer logic with the following options: a 4:1 multiplexer using one LUT, an 8:1 multiplexer using two LUTs, or a 16:1 multiplexer using four LUTs.

In addition, the function generators in some dedicated slices can be used as distributed RAM, or as 32-bit shift registers. These enhanced slices are known as SLICEM, while the standard slice is called SLICEL. Approximately one third of slices are SLICEM — note that these slices are always paired with a SLICEL (that is, a CLB never contains two SLICEMs).

3.4.3 Clock Management

A brief summary of the clocking architecture in 7-Series devices was given earlier (Section 3.4.1), where it was seen that there are 4 clock management tiles (CMTs) in the XC7Z020. Some devices have many more CMTs — the Virtex-7 XC7V2000T has 24, for example. The fundamental concept of this resource is shown in Figure 3.8. . The CMT contains a multi-mode clock manager (MMCM) and a phase-locked loop (PLL), both of which are capable of generating a wide range of frequencies for the FPGA design. Also, because both circuits are PLL-based, they can filter out jitter in external and internal clocks, resulting in

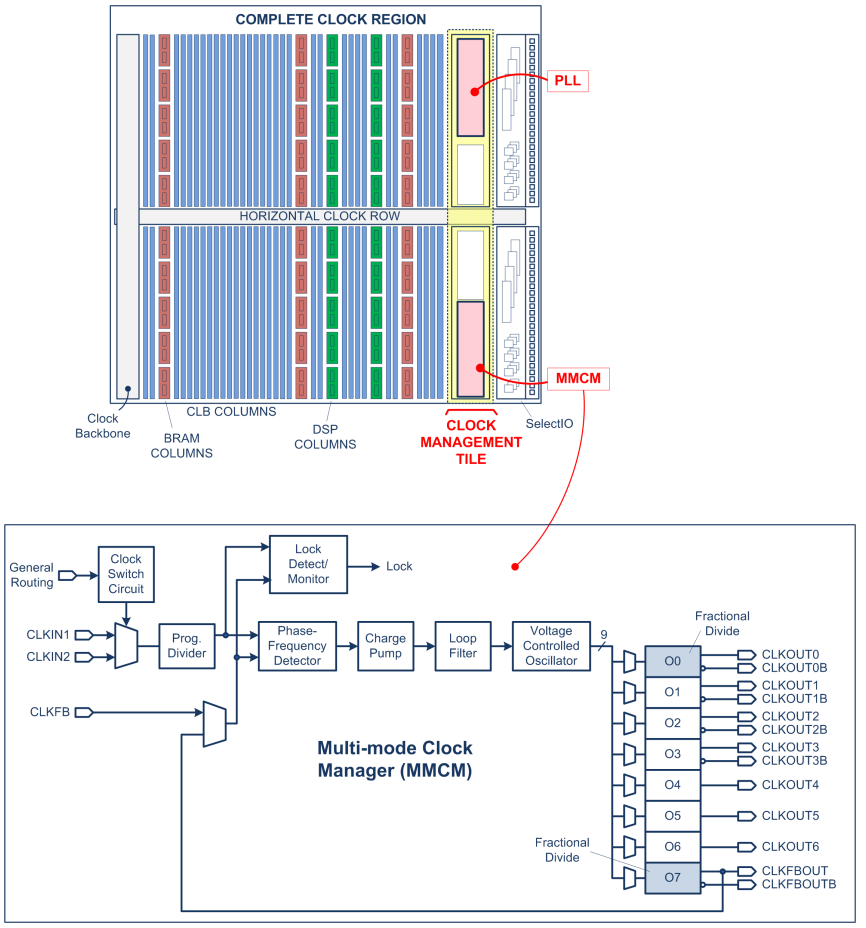

Figure 3.8. Xilinx 7-Series Clock Management Tile (CMT), which includes a PLL and MMCM. The MMCM is shown in the bottom part of the figure.

cleaner clocking for the design. (This is due to a fundamental characteristic of PLLs where jitter cannot be tracked above the loop bandwidth, meaning that the circuit is effectively a low-pass filter.)

The MMCM and PLL are very similar — in fact, the MMCM is really just a PLL with slightly more features, including (but not limited to):

- Four inverted clock outputs for CLKOUT[0:3]B
- An extra clock output (CLKOUT6)
- Fractional divide for CLKOUT0
- Fractional multiply for the feedback clock
- Fine phase shifting
- Dynamic phase shifting

3.4.3.1 Clock Pins

In 7-Series FPGAs, a range of multi-function pins are available in each I/O bank, which, if not used as regular I/Os, can be used as clock input pins. The pins are designated as either single-region clock-capable (SRCC) or multi-region clock capable (MRCC). Each type can drive FPGA clocking resources such as BUFRs, BUFIOs, BUFGs, and, of course, MMCMs/PLLs. The SRCC type is limited to driving clock resources in a single region, while the MRCC pin provides multi-region clocking. Both types of pin are available as differential pairs (meaning they can be driven by a differential clock source), but when a single-ended clock is required, the positive pin of the differential pair should be used. On the Zybo-Z7-20 platform, for example, a single-ended 125 MHz clock is connected to an MRCC pin (K17) in bank 35 — this pin has the designation **`IO_L12P_T1_MRCC_35`**, where the "**`L`**" in "**`LxxP`**" indicates a differential pair, and the "**`P`**" indicates the positive side of the pair.

3.4.4 Block RAM

As mentioned earlier, the SLICEM components in the CLB offer basic memory functionality in the form of distributed RAM, although the storage capacity is quite low (just 1,088 Kb in the XC7Z020, for example). Another disadvantage of using CLBs for on-chip memory is that a slice is no longer available for its primary function, which, of course, is to implement user logic.

With this in mind, Xilinx FPGAs also contain higher-capacity dedicated memory structures called Block RAM (BRAM). In 7-Series FPGAs, the storage capacity of the BRAM is 36 Kb, which can be configured as a single RAM or as two independent 18 Kb RAMs. In an XC7Z020 clock region, there are three BRAM columns, each containing ten 36 Kb BRAMs; this means that when all full and partial clock regions are taken into account, there are a total of 140 BRAMs in the XC7Z020, with an equivalent memory capacity of 4.9 Mb. Figure 3.9 illustrates how BRAMs fit into the overall 7-Series floor-plan, and it also shows some example configurations.

In its main form, the BRAM is a synchronous dual-port memory with 36 Kb (or 18 Kb) storage space, and completely independent access ports. BRAMs can also be configured as simple dual-port RAM, where the port width doubles to 72 bits for the 36 Kb RAM, and 36 bits for the 18 Kb RAM. In this case, there is only one read port and one write port (as distinct to two independent ports in true dual-port mode). Note that memory collisions can occur in some configurations, which the developer should try to guard against (see [80] for further information).

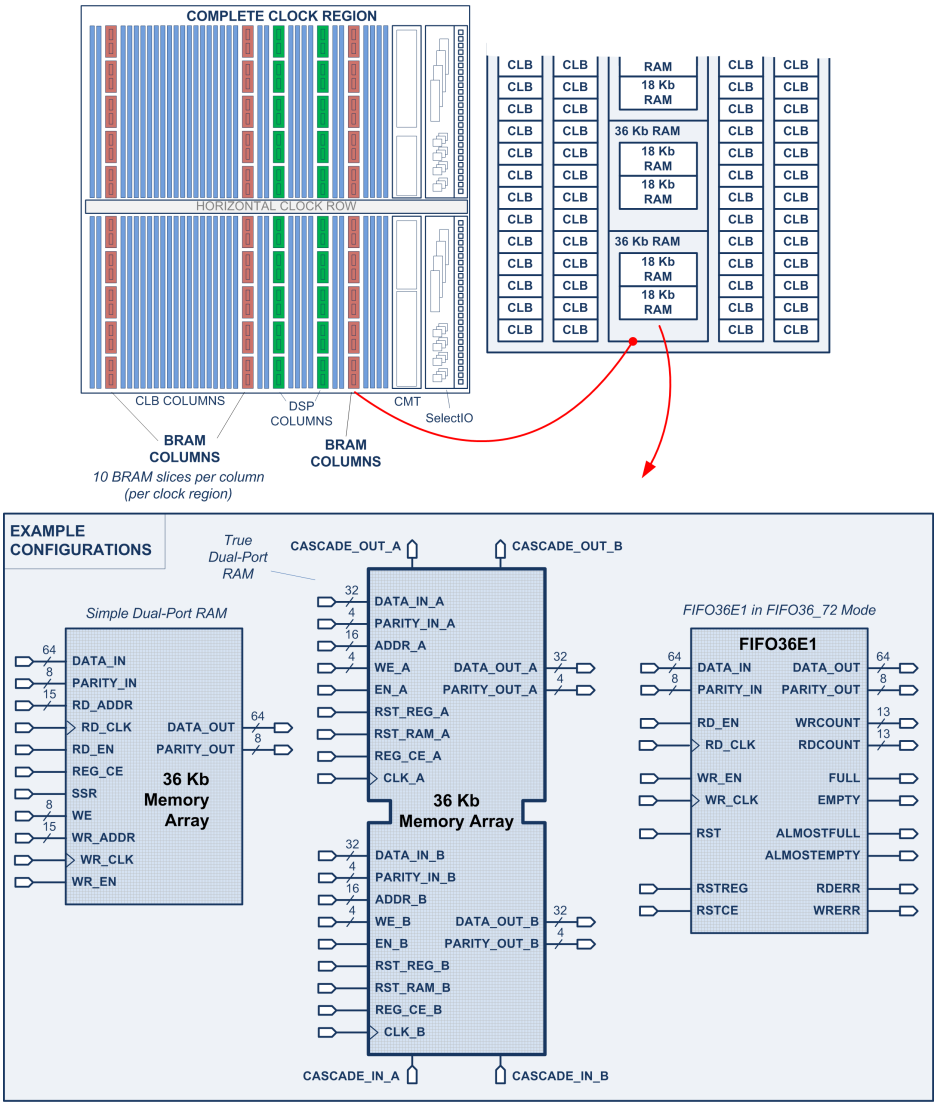

Figure 3.9. Xilinx 7-Series BRAM, with some example configurations.

BRAMs can also be cascaded to form a 64Kx1 RAM, and they include ECC capability. Finally, they can be used to implement synchronous or asynchronous (dual-clock) FIFOs, resulting in a much more efficient structure than when CLBs are used.

3.4.5 DSP

Digital Signal Processing (DSP) techniques have been established in a wide variety of consumer, multimedia and communications applications for several decades now. The associated computing algorithms are characterised by repetitive multiply-and-accumulate (MAC) operations on wide-bit-input operands, and while DSP logic can be implemented using traditional digital design techniques (or directly in software), superior performance can be achieved by using dedicated MAC hardware units.

In Xilinx 7-series FPGAs, the DSP48E1 is the unit dedicated to this purpose, and, just like the CLB and BRAM resources, the DSP slices are arranged as columns in the FPGA floor-plan (see Figure 3.10). There are generally two or three DSP columns per clock region, with 20 DSP48E1 slices per column. In the Zynq XC7Z020, this adds up to a total of 220 DSP slices.

The DSP48E1 follows the multiply-and-accumulate model closely, with the main components being a 25x18 two's complement multiplier and a 48-bit accumulator. There is also a pre-adder which improves performance and reduces slice count in relevant applications (e.g symmetrical IIR/FIR filters, or alpha blending in image processing).

As well as operating as a high performance DSP MAC unit, the DSP48E1 can be used for other functionality, with some examples being:

- SIMD operations
- Adder, subtracter, or boolean logic operations
- Pattern detection
- Wide bus multiplexers
- Memory-mapped I/O registers

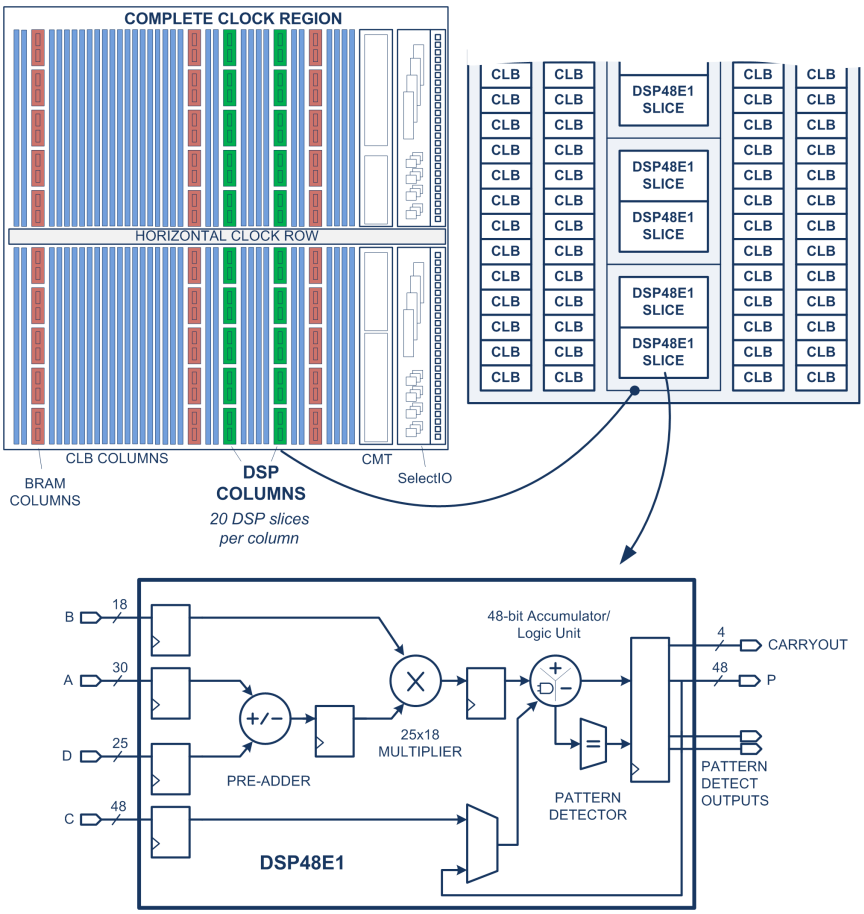

Figure 3.10. Simplified illustration of Xilinx 7-Series DSP, showing just fundamental logic and I/O ports.

3.4.6 SelectIO

Each clock region contains a SelectIO block, allowing the FPGA to support a wide range of I/O standards. Table 3.4 gives a broad overview of the available standards in Xilinx 7-Series FPGAs, although many of these can be broken down again into different class and voltage options.

Supported I/O Standards
Low Voltage TTL (LVTTL)
Low Voltage CMOS (LVCMOS)
High-Speed Transceiver Logic (HSTL)
Stub-Series Terminated Logic (SSTL)
High-Speed Unterminated Logic (HSUL)
Mobile DDR (LPDDR)
Low Voltage Differential Signalling (LVDS)
Reduced Swing Differential Signalling (RSDS)
Mini-LVDS
Point-to-Point Differential Signalling (PPDS)
Transition Minimised Differential Signalling (TMDS)
Bus-LVDS

Table 3.4. Broad overview of Xilinx 7-Series I/O standards.

In terms of the FPGA pin-out, there are 50 pins per clock region, with 25 in each half (Figure 3.11). The top and bottom pins in each region are single-ended only, but all other (adjacent) pins can be configured as differential pairs.

Xilinx 7-Series FPGAs were the first to support two different types of I/O banks, namely high-range (HR) banks and high-performance (HP) banks. The HR option is intended for general-purpose interfacing, and it can operate across a wide voltage range (1.2V to 3.3V). These banks can also handle higher output current, for example 24mA for LVTTL and LVCMOS18. The HP banks, on the other hand, are intended for high-speed memory or chip-to-chip interfaces, and they operate at a reduced voltage range of 1.2V to 1.8V. They tend to be available only on higher-performance 7-Series devices, which, for the Zynq-7000 range, starts with the XC7Z030.

Because HP pins are more likely to be used for high speed interfacing (where signal integrity is of critical importance), these banks also include digitally-controlled impedance (DCI) technology. This feature is used to internally adjust the output impedance of SelectIO drivers or add a parallel termination to drivers/receivers, ensuring that high-speed networks are impedance-matched. This reduces the number of external termination resistors on the PCB, saving on BOM costs and board space.

Logic resources associated with the I/O drivers/receivers also expand the functionality of the SelectIO banks. For example, ILOGIC/OLOGIC blocks can be configured to capture/transmit data in combinatorial, registered or DDR mode, and tri-state options are available for output pins. Also, IDELAY and ODELAY blocks can be used to finely adjust I/O path delays,

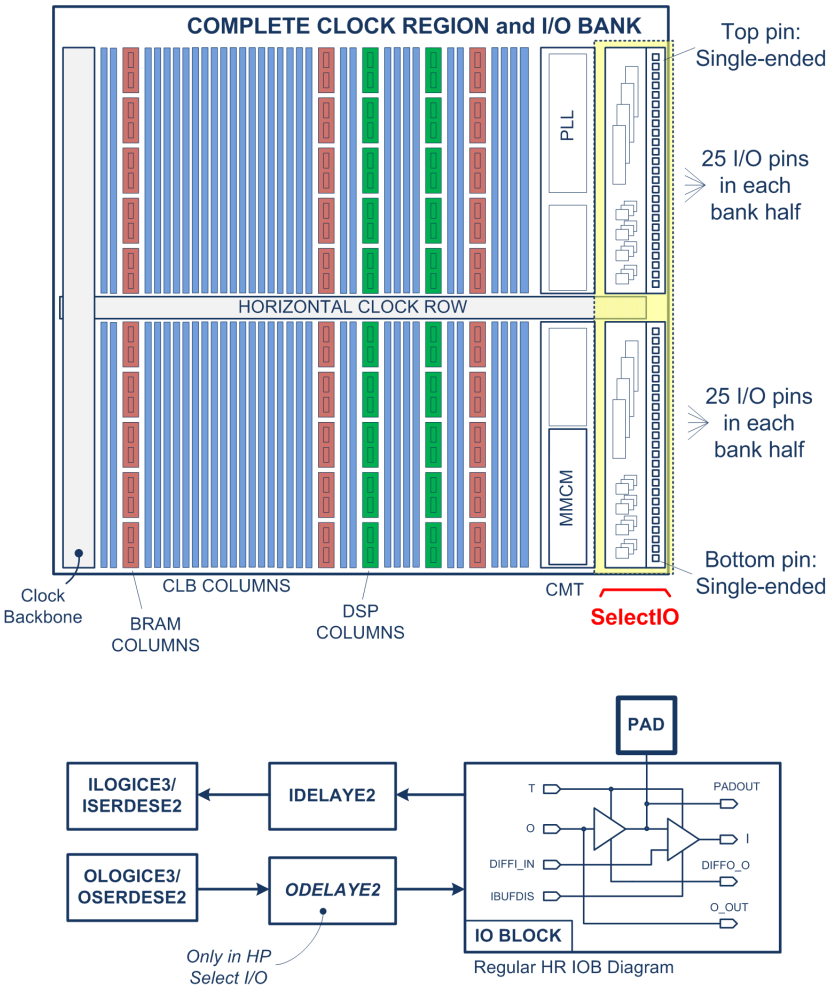

Figure 3.11. Xilinx 7-Series SelectIO. A basic illustration of the Regular HR IOB is shown, as part of the HR Bank I/O Tile.

although the ODELAY block is only available on HP pins. Finally, serializer-deserializer circuits (ISERDES and OSERDES) are available in the I/O path, allowing the internal FPGA logic to run at a slower rate than the I/O block (making it easier to close FPGA timing). The ISERDES is an input serial-to-parallel converter, and the OSERDES is an output parallel-to-serial converter.

3.4.7 Multi-Gigabit Transceivers and PCIe

The XC7Z020 doesn't include any Multi-Gigabit Transceiver (MGT) tiles, but for completeness a brief summary of Zynq-7000 MGT support will be presented here. The three highest-performing Zynq-7000 devices (XC7Z035 to XC7Z100) all contain 16x GTX transceivers capable of running at 12.5 Gbps for a total possible bandwidth of 200 Gbps, while the XC7Z030 contains 4x GTX transceivers (also 12.5 Gbps) for a total bandwidth of 50 Gbps. In the cost-optimised range, MGTs are included in just the XC7Z012S and XC7Z015 devices, and even then only the lower performing GTP transceivers are used — these circuits can run at 6.25 Gbps for a total bandwidth of 25 Gbps (each device contains a 4x GTP, also known as a single quad transceiver).

When MGTs are implemented in a device, they take the place of one normal SelectIO bank, as shown in Figure 3.12. The left-hand side of this diagram shows the Zynq XC7Z020 device in CL485/CLG485 packaging, and the right side shows the XC7Z012S (or XC7Z015) device in the same package. When the GTP tile is used in the latter device, it replaces the SelectIO and CMT technology that make up Bank 33 in the former [81, 56].

As mentioned earlier, a GTP tile contains quad transceivers, each capable of running at 6.25 Gbps. A single transceiver consists of high-speed serial transmitter and receiver sections. The transmitter is fundamentally a parallel-to-serial converter, encoding parallel data received from the programmable logic, and transmitting the corresponding serial bit stream on a high-speed differential link. The data is encoded in 8B/10B or 64b/66b format, and the transmitter can apply pre-emphasis to the bit stream — this boosts the leading edge of the signal, improving signal integrity. The receiver carries out the reverse operation, accepting an incoming differential bit stream and converting it to parallel data for the programmable logic. It must recover the serial clock from the incoming data, perform comma alignment operations, and decode the data.

The transceiver also includes protocol-specific functionality such as out-of-band (OOB) signalling for Serial ATA and beacon signalling for PCIe. Finally, some of the main protocols

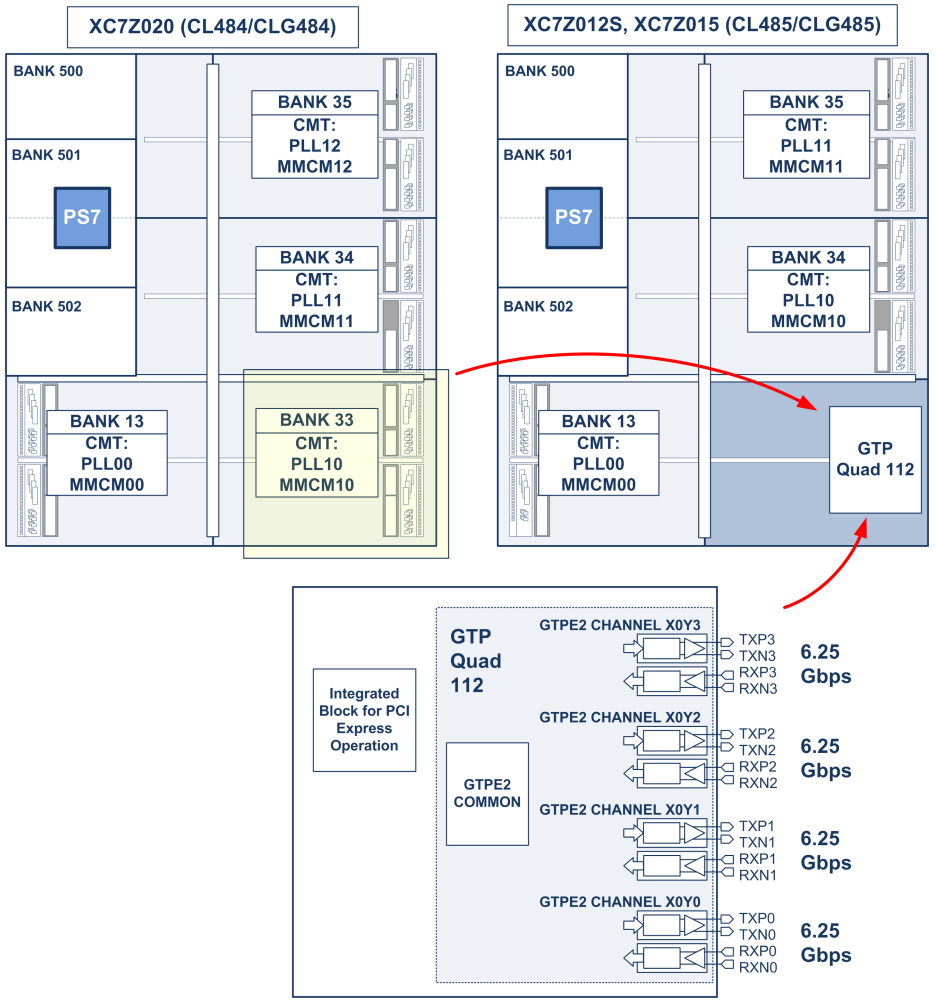

Figure 3.12. Transceiver tiles replace a SelectIO location in supported devices.

supported by the GTP transceiver include PCIe 1.1/2.0, Interlaken, 10 Gb XAUI, JESD204, HDMI and SATA/SAS.

3.4.8 XADC Dual 12-Bit 1MSPS Analog-to-Digital Converter

The FPGA section of all Zynq-7000 devices contains a 7-Series Dual 12-bit 1 Mega-sample per second (MSPS) ADC, more commonly known as the XADC (Figure 3.13). This is a precision analog hardware block providing 17 differential analog inputs (which can also be used in single-ended fashion). One of the analog input pairs, V_P/V_N, is always available on the device, while the other 16 pairs connect to multi-purpose I/Os on the FPGA, meaning that they can be used as normal digital pins if desired. On the XADC, these alternative pins are called auxiliary inputs (V_{AUXP}[15:0], V_{AUXN}[15:0]).

The XADC also contains sensors and alarms to monitor the power-supply voltages and die temperature of the Zynq-7000. On the processing system side, V_{CCPINT}, V_{CCPAUX}, and V_{CCO_DDR} are monitored, while on the FPGA side V_{CCINT}, V_{CCAUX} and $V_{CCOBRAM}$ are monitored.

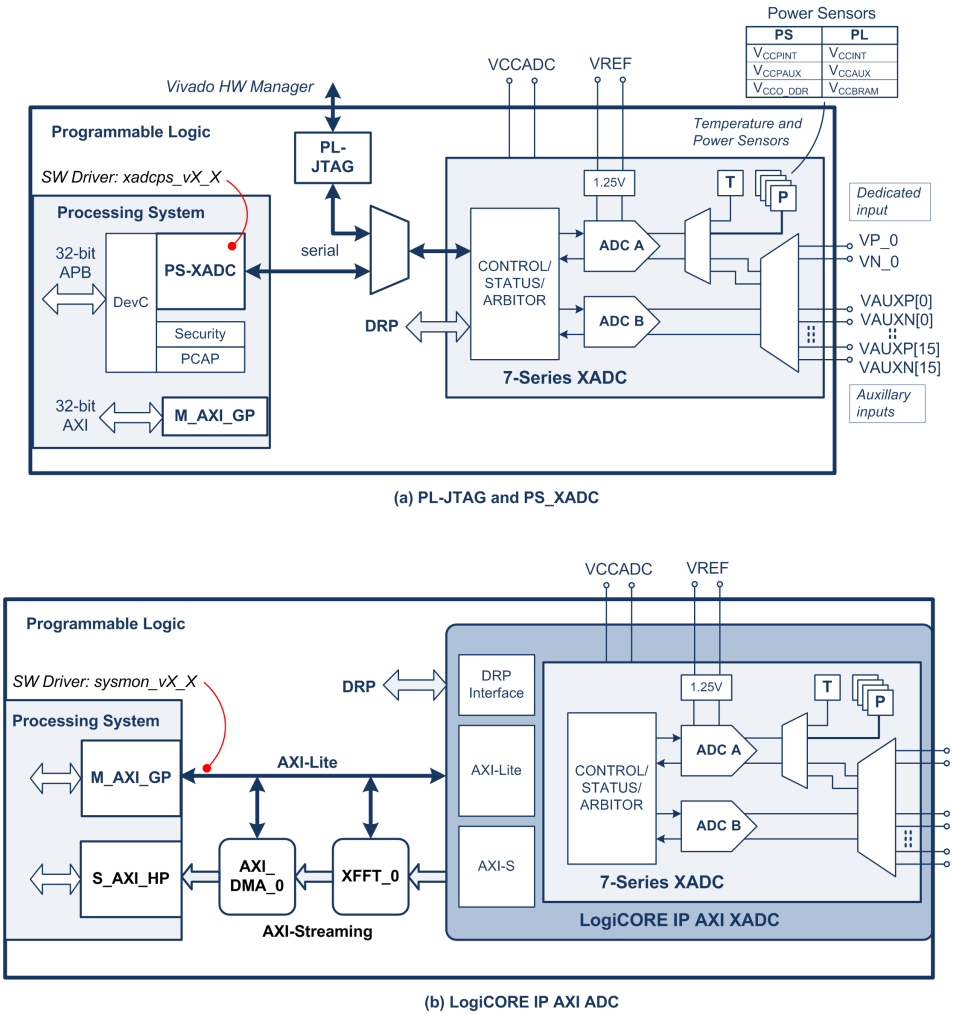

Figure 3.13. Zynq XADC Dual 12-bit 1MSPS ADC. (a) Default PL-JTAG and PS-XADC interfaces. (b) Xilinx LogiCORE AXI XADC IP: AXI-Lite and AXI-S interfaces.

A range of operating modes are supported by the XADC. At its most basic, it can be set to Single Channel Mode, where, quite obviously, it monitors a single channel. The sequence modes are much more useful, where Default, Single Pass, Continuous, and Simultaneous Sampling options are available (see [82] for more details). In addition, the XADC can be configured in continuous sampling mode or event-driven sampling mode.

Three main approaches exist to communicate with the XADC on the Zynq-7000. (As an aside, note that the following discussion is specific to the Zynq-7000, and does not necessarily apply to standard 7-series FPGAs.) In Figure 3.13 (a), two methods are shown. First, the PL-JTAG interface allows the XADC to be accessed using the JTAG port, which is the same port used for device configuration (see Section 3.6.4). This functionality is readily available using the Hardware Manager feature in Vivado. The second approach is through the PS-XADC interface, which is the standard way for the user to access the XADC in the processing system. The XADC effectively appears as a memory-mapped peripheral in this approach, and device software can be written for it using the low-level Xilinx driver, **`xadcps_v*_*`**.

For the two cases just mentioned, the user does not have to do anything during the hardware design phase to enable the XADC — it is available by default. Figure 3.13 (b) shows a third way to use the XADC, whereby the user instantiates the LogiCORE AXI XADC IP using a simple Wizard in IP Integrator. This generates a wrapper around the XADC hard macro, providing access to AXI-Lite and AXI-Streaming (AXI-S) interfaces. The AXI-Lite interface ensures that the XADC appears like a memory-mapped component to the user, just like the PS-XADC interface described earlier. A different low-level Xilinx driver is used in this case, **`sysmon_v*_*`** (i.e. System Monitor).

The AXI-S interface is more interesting, as it allows the XADC to be used in high-throughput streaming applications. For example, Figure 3.13 (b) shows the XADC used in an FFT application, with a DMA block completing the connection to a high performance port on the PS.

A final note on the XADC concerns its power supply requirements. The main core supply is VCCADC, which is 1.8V +/-5%, while the analog supply, VREF, is 1.25V. If high sampling accuracy is needed in an application, VREF should be powered by a precision external reference (1.25V +/-0.2%, 50ppm/C). However, an on-chip reference is also available for VREF, and while the accuracy is unlikely to be as good as when an external reference is used, it is suitable for monitoring the on-die temperature and power-supply parameters.

3.5 PS-PL Interconnect

Having a powerful multi-core processing system and high-speed programmable logic on a single chip is of little use without an effective means to share information between the two. In the Zynq-7000, a wide range of interfaces are provided for this critical functionality, and a broad overview is given in this section. The discussion is not exhaustive, however, and the reader is referred to Chapter 2 and Chapter 5 of the Zynq-7000 TRM [39] for more comprehensive details. In this text, the following topics will be covered (refer also to Figure 3.14):

- Clocking and Reset
- Interrupts
- AXI Interconnect
- DMA Access

- Extended Multiplexed I/O
- Debug and Trace
- Event/Miscellaneous Signals

3.5.1 Clocking and Reset

As mentioned in Section 3.3.3.1, the PS supplies four clocks to the programmable logic, *FCLKCLK[3:0]*, all of which are completely asynchronous to, and independent of, each other. Each clock is generated from an individual PLL and can be configured in the SLCR, although the designer can easily enable and set the desired clocks in IP Integrator in Vivado.

Each of the four PL clocks has an associated reset signal, *FCLKRESETN[3:0]*. These resets are also configurable in the SLCR, but again they are easily enabled in Vivado. The complex topic of reset handling in FPGAs is also made much easier in IP Integrator, as a Processor System Reset IP Module can be automatically added to the user logic during the FPGA design phase. This will be seen in Section 6.7.6.

The other clock-related signals are the PL Clock Throttle input signals, *FCLKCLKTRIGxN*. Clock throttling is a feature used to start and stop the PL clocks under software control, or to run the clocks for a specific number of pulses. Note, though, that the *FCLKCLKTRIGxN* signals are currently not supported in the Zynq-7000. (To elaborate, clock throttling can be implemented using software control, but the *FCLKCLKTRIGxN* signals cannot be used to halt the clock.)

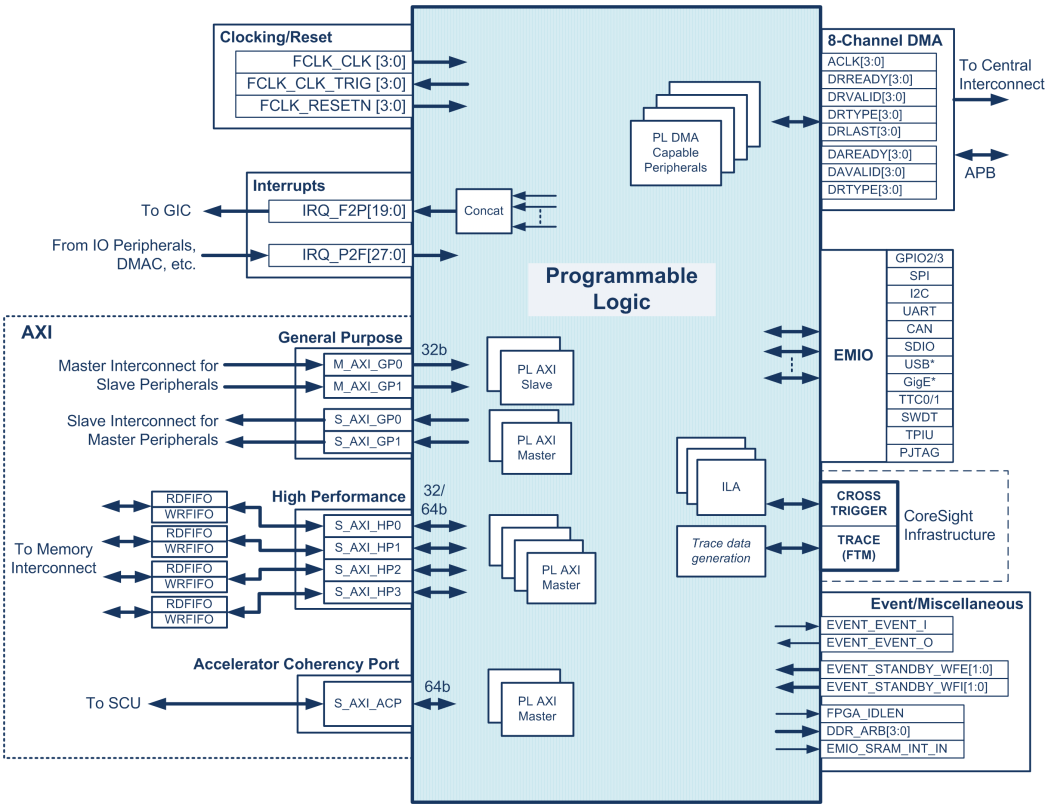

Figure 3.14. PS-PL Interconnections

3.5.2 Interrupts

Two separate sets of interrupt signals are available at the PS-PL interface. One set of twenty signals (*IRQ_F2P[19:0]*) is used to propagate interrupts generated in the programmable logic to the PS interrupt controller, while the other set (*IRQ_P2F[27:0]*) transmits I/O peripheral interrupts from the PS side to the programmable logic. In the projects in this text, we are only concerned with interrupts generated on the PL side — the Concat Utility IP is used in IP Integrator to connect these interrupts to the PS, as will be seen in Chapter 6. The general concepts of interrupt handling in the Zynq-7000 will be covered in Chapter 11, and programmable logic interrupts will be covered in more detail in Chapter 16.

3.5.3 AXI Interconnect

The principle interfaces between the processing system and the programmable logic are based on Arm AMBA AXI3 technology [77]. These interfaces can be grouped into three categories: general purpose (GP) ports; high performance (HP) ports; and an accelerator coherency port (ACP). These are summarised next.

3.5.3.1 AXI General Purpose Ports

Four 32-bit general-purpose ports are available in the PS-PL interface. Two of the ports (*M_AXI_GP[1:0]*) are PS masters, meaning that software running on the PS can control and interact with general purpose slaves in the PL. Each port can handle more than one slave — for example, in the FPGA design in this text, *M_AXI_GP0* is used connects to both an AXI GPIO module and an AXI QSPI IP module (although many more could be added). In this way, the modules become memory-mapped peripherals, just like the PS I/O peripherals discussed earlier. The programmable logic design process is also very simple when using the IP Integrator tool in Vivado, as associated modules like the AXI Interconnect block can be automatically added to the design. For beginners, the simplest way to start developing Zynq-7000 applications is to use the *M_AXI_GP0* port along with Xilinx-supplied slave peripherals. This is the approach used in the hardware design phase in Chapter 6.

The other two general purpose ports, *S_AXI_GP[1:0]*, are PS slaves, allowing AXI masters in the programmable logic to interact with PS I/O peripherals. This functionality is not used in any of the projects related to this text.

3.5.3.2 AXI High-Performance Ports

The *S_AXI_HP[3:0]* ports are high-performance AXI3 interconnects — these allow masters in the programmable logic to interface directly with the PS DDR or OCM memories. In contrast to the master GP ports just discussed, the HP ports are used to transfer large amounts of data between the PS and PL (or vice-versa). Each interface can be configured with 32- or 64-bit data buses, and read/write FIFOs are implemented on the PS side for flow control.

3.5.3.3 Accelerator Coherency Port

The HP ports just discussed come with a performance penalty, as they are not cache-coherent. This means that the cache system might have to be regularly flushed, an operation that can take hundreds of cycles. An alternative method is to use the ACP, which allows PL masters to carry out cache-coherent transactions with the processing system (see also Section 2.6.3). A port like this is especially suited to a programmable logic device like the Zynq-7000, as a wide array of user peripherals can be designed to take advantage of cache

coherency. In general, though, the ACP is recommended for medium-grain applications such as block-level cryptographic acceleration and macro-block-level video processing [39].

3.5.4 DMA Access

Four of the channels in the APU DMA controller are reserved for the programmable logic. This allows PL masters to directly access the Zynq-7000 memory system without processor intervention. The interface uses a request-acknowledge handshaking protocol to execute single or burst AXI transactions.

3.5.5 Extended Multiplexed IO (EMIO)

The EMIO functionality of the Zynq-7000 was introduced earlier in Section 3.3.2. As a reminder, this feature makes up for the limited number of package pins available in the processing system by allowing many PS peripherals to route some or all of their signals to the programmable logic. In the PL, the signals can be routed directly to spare FPGA pins, or used in the digital design itself.

3.5.6 Debug and Trace

Two debug interfaces are available in the PS-PL interface: cross-triggering, and fabric trace. A brief summary of each feature is given here, before the much more comprehensive discussion to come in Section 3.7.

3.5.6.1 Cross-triggering

Cross-triggering is a debug approach where trigger signals generated in the programmable logic design can halt a software program running in SDK (or the Vitis software platform). This technique also works in the opposite direction: the PS can send trigger signals to the PL, allowing the digital design to be debugged. Cross-triggering is covered in more detail in Section 3.7.1.2.

3.5.6.2 Fabric Trace

In an ARM CoreSight trace system, data is collected from various components in the SoC while a program is running. The data is then either transmitted off chip for immediate inspection, or stored in memory for later analysis. In the Zynq-7000, Xilinx has integrated a Fabric Trace Monitor (FTM) in the CoreSight trace system — this allows data in the digital design to be included in trace inspections. CoreSight Trace is covered in Section 3.7.1.3.

3.5.7 Event/Miscellaneous Signals

A number of event and other miscellaneous signals also exist between the PS and PL. For example, an idle signal can be used to indicate that there are no outstanding AXI transactions in the PL, or a wake-up signal can be sent to the processor(s). More details on these extra signals can be found in the Zynq-7000 TRM [39].

3.6 Boot and Configuration

In traditional FPGA architectures — that is, devices that don't include complex microprocessor systems — the configuration process simply consists of downloading a bitstream to the programmable logic. Xilinx FPGAs are SRAM-based (i.e., volatile) meaning that the bitstream must be transferred to the fabric every time the device is powered up.

The boot process for a microprocessor is similar, in that the hardware settings and executable application must also be transferred to the processor on power-up. Of course, the Zynq-7000 contains both PS and PL sections, and so it follows that the start-up process must include: (a) boot and configuration of the processing system; and (b) bitstream download to the programmable logic. (As an aside, the Zynq PS section can be used without downloading a bitstream to the PL, but in the discussion that follows we assume that the PL section is being used.)

The Zynq-7000 boot process can be classified in various ways. First of all, it can be secure or non-secure. Security is a major concern in modern embedded systems, where it is critical that on-chip information is protected, and devices are resistant to attack. Zynq devices promote a 'chain-of-trust' methodology, where security is implemented using AES/RSA encryption keys and a HMAC engine, among other features.

The boot process can also be classified in terms of boot device and boot mode — for example, external flash memory is classed as master-mode, and JTAG cable download is classed as slave mode. The QSPI and other static memory options (NAND/NOR/SD Card) discussed in Section 3.3.1.3 can be used as external boot devices, and the process can be either secure or non-secure.

QSPI and NOR memory can also be used in a mode called 'Execute-in-Place' (XIP), where the boot image is executed directly on the external device, rather than being copied into PS memory. In XIP mode, the boot process is always non-secure.

JTAG boot mode is used during embedded project development — the code is developed using a suitable software tool and then downloaded directly to the target, where it can also be debugged. JTAG mode is always non-secure.

Because the Zynq-7000 uses the application-centric Cortex-A9 processor(s), it is also likely that an operating system such as Linux or Android will be running on the system, adding another layer of complexity. In contrast, a simpler boot sequence is required for bare-metal applications. In the upcoming sections, the boot sequence for an operating system will be discussed first, followed by the bare-metal boot sequence. JTAG boot mode will also be briefly covered.

3.6.1 Booting an Operating System

Figure 3.15 shows the steps required to boot an operating system, such as Linux, from an external boot device. The discussion starts from system power-on/restart.

3.6.1.1 Power-On/Restart Sequence

Device power-on and restart can be categorised in two ways — Power-On Reset (POR), and non-POR restart.

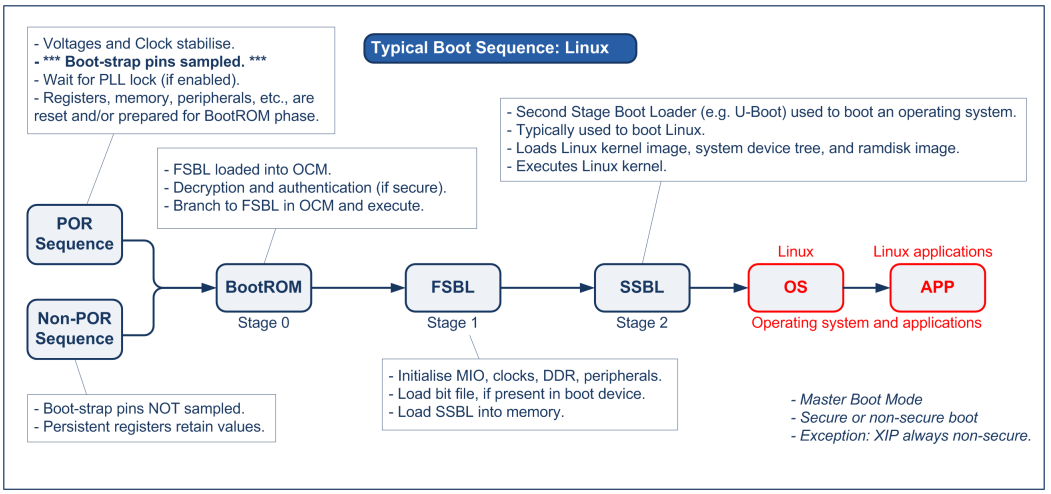

Figure 3.15. Example boot sequence for Linux OS

3.6.1.1.1 Power-On Reset (POR)

When power is first applied to the Zynq-7000, it goes through the Power-On Reset sequence. First, the relevant voltages and clocks in the system are allowed to stabilise, and then the boot-strap pins are sampled to detect options such as boot mode/device, PLL bypass, JTAG configuration, and MIO bank voltage settings. (More details on the boot-strap pins will be given in Section 3.6.4.) The sampled settings are stored in the *BOOT_MODE* register in the SLCR. Registers, memory, and peripherals in the Zynq will also be initialised for the BootROM stage.

3.6.1.1.2 Non-POR Restart

It is sometimes necessary to restart a running embedded system; on the Zynq-7000, this is achieved using the external *PS_SRST_B* signal, or through a software/processor/debug reset. When a restart occurs, the full POR sequence is not repeated — the boot-strap pins will not be re-sampled, and some internal registers will retain their values. For example, the *BOOT_MODE* register in the SLCR will preserve the boot-strap pin settings from the last POR, and the *BOOT_STATUS* register will show the reason for the current reboot. (The latter functionality is demonstrated in the SCU WDT Timer project in Chapter 3.)

Regardless of the type of power-up (POR or restart), the boot sequence moves to the BootROM phase next.

3.6.1.2 Stage 0: BootROM

As the name suggests, the aim in this stage is to execute the BootROM code, which is a fixed program stored in the 128 KB OCM ROM. The BootROM will perform a different routine depending on the boot mode register settings detected during POR. When dealing with a Linux boot from an external flash device, the task of the BootROM code is to prepare the system for a first stage bootloader (FSBL). The external flash memory will be searched to see if it contains an FSBL, and, if one is found, it will be loaded into the Zynq OCM RAM. Security

operations such as RSA decryption will also be carried out if necessary, and ROM code access will be disabled. The processor will then branch to, and start executing, the FSBL code in OCM.

3.6.1.3 Stage 1: First Stage Bootloader

The first task of the FSBL is to initialize the processing system with MIO, DDR, clock and peripheral settings — these are part of the hand-off process at the end of the hardware design phase, and can be viewed in the **`ps7_init`** C or TCL files. (This topic will be discussed again in Chapter 5.) The FSBL then systematically extracts any bitstream or software partitions from the boot device, and transfers them to their correct location on the Zynq, fulfilling any security obligations along the way. The FSBL then loads the second stage bootloader (SSBL) to the correct location in the PS, and hands off execution to the SSBL.

3.6.1.4 Stage 2: Second Stage Bootloader

The SSBL is generally used to launch an operating system, with the open-source software U-Boot (Universal Boot Loader) a popular choice in Zynq-7000 Linux builds. The SSBL first carries out any remaining hardware initialisation, and then loads the Linux kernel, system device tree, and ramdisk images from the boot device. The kernel is then started on the processing system.

3.6.1.5 Linux and Applications

Finally, once the Linux kernel is up and running, the user can launch supported applications, dependent on the Linux build and user access configuration.

3.6.2 Booting a Bare-Metal Application (External Device)

Once the typical Linux boot sequence is understood, the bare-metal application boot process is easy to follow — it is really just a stripped back version of the full boot sequence (see Figure 3.16). Chapter 17 will focus entirely on this type of boot, but for now a quick summary will suffice:

- The POR/restart stage is exactly the same as detailed in Section 3.6.1.1, with the boot-strap pins indicating that the boot image must be loaded from an external device.

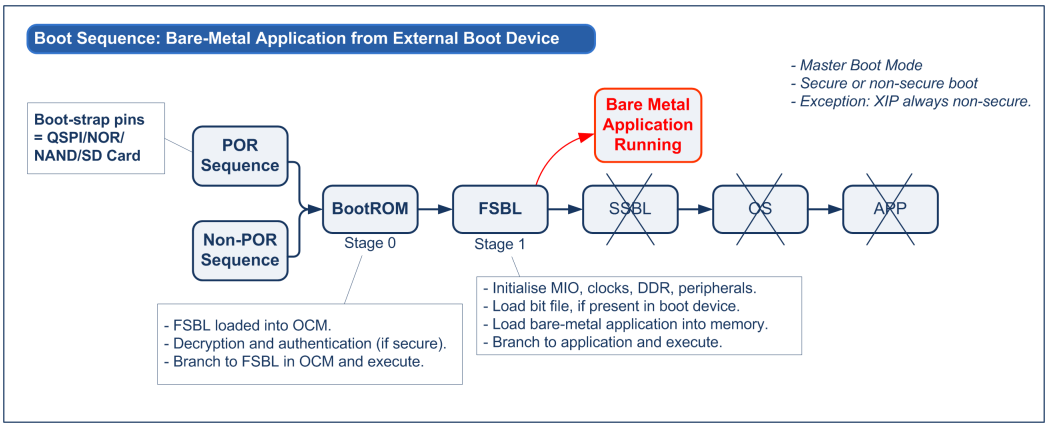

Figure 3.16. Example boot sequence for bare-metal application (external device)

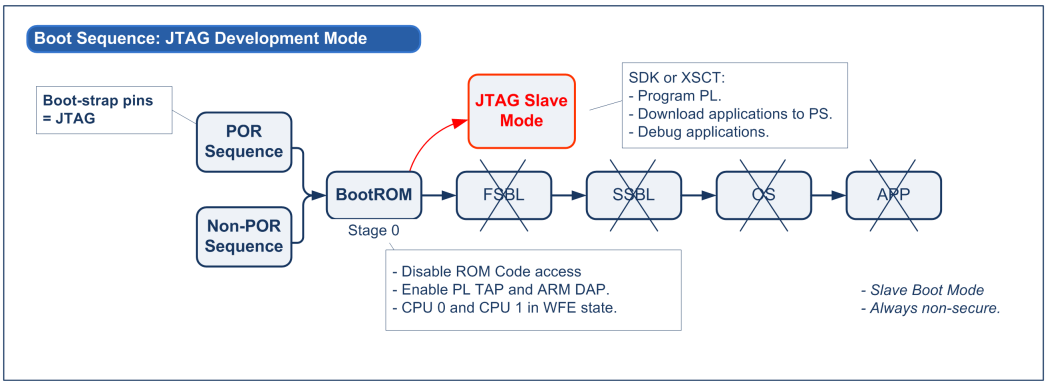

Figure 3.17. Example boot sequence for JTAG development

- When the BootROM phase runs (Stage 0), the BootROM code will load the FSBL from the flash device into OCM RAM, carrying out any security operations as appropriate. It will then hand off execution to the FSBL (Stage 1).
- In the FSBL stage, the PS is first initialized, and the bitstream is then downloaded to the FPGA. Next, rather than loading an SSBL into memory (as happens with a Linux boot), the bare-metal application is loaded into DDR.
- Finally, the PS branches to, and starts running, the bare-metal application.

3.6.3 JTAG Development Mode

Figure 3.17 illustrates the typical boot sequence in JTAG development mode, which again can be viewed as a truncated version of the boot procedures just described above. The POR/restart phase is mostly unchanged, with the exception that JTAG boot-mode will be detected when the boot-strap pins are sampled. In JTAG mode, the BootROM code runs a different routine, briefly summarised as follows:

1. ROM code access is disabled.
2. The JTAG controllers in the PL and PS are enabled.
3. The processors in the MPCore are put in the 'Wait For Event' state.

At this point, a fully-featured development environment on the host PC (such as the Xilinx SDK or Vitis software platform), can interact with the embedded platform. For example, the programmable logic can be configured, and the bare-metal application launched/debugged on the target. A console program such as XSCT can also be used to carry out similar functionality. These topics are covered in more detail in Chapter 5 and Chapter 7.

3.6.4 Configuration and Boot Circuit

Figure 3.18 illustrates the important elements of the Zynq configuration/boot circuit. As discussed in the previous section, sampling the boot-strap pins during POR is a critical part of the boot sequence, and the functionality related to each pin is described below.

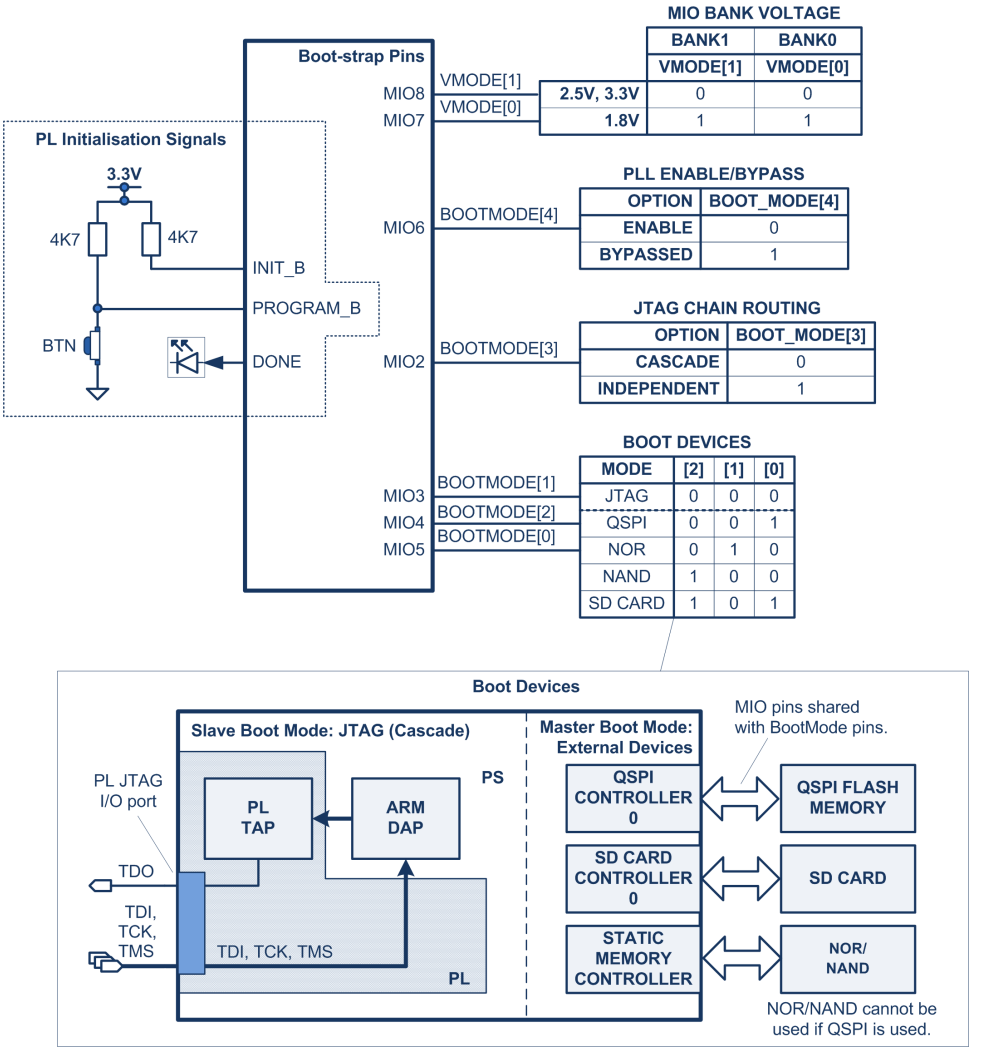

Figure 3.18. Zynq-7000 boot circuit.

MIO2: JTAG Chain Routing

The Zynq-7000 contains two JTAG controllers: the Test Access Port (TAP) in the programmable logic, and the Debug Access Port (DAP) in the ARM processing system. By default (i.e., with MIO2 set to '0'), the JTAG chain is configured in Cascade mode — in this case, both controllers are accessed in sequence using the PL JTAG port. This configuration is illustrated in the bottom left of Figure 3.18. Another option called Independent mode can also be used; this is where the PL TAP is accessed using the PL port (as before), but the ARM DAP is accessed separately via spare MIO or EMIO pins. In this text, it is always assumed that the Cascade configuration is being used.

MIO[5:3]: Boot Devices

The MIO[5:3] pins are used to set the boot mode to JTAG or an external device. (The various options are as shown in the figure.)

MIO6: PLL Enabled/Bypassed

In Section 3.3.3, we saw that the external clock can bypass the internal PLLs and connect directly to the clock generation logic. This functionality is controlled by MIO6 — if the pin is detected as a logic '1' during the POR sequence, the PLLs will be bypassed.

MIO[8:7]: Voltage Bank Mode

The MIO[8:7] pins set the bank voltages for MIO bank 0 (pins 0-15) and MIO bank 1 (pins 16-53). A logic '0' sets the bank voltage to 3.3V (which is also compatible with 2.5V), and a logic '1' sets the bank voltage to 1.8V. Note that if any peripheral interface spans both banks, then each bank must be set to the same voltage.

3.6.4.1 Boot-Strap and External Memory MIO Conflicts

here are some points to be aware of in relation to the QSPI and SMC external memory options. First of all, the controllers share the same pins, meaning that only one of these options can be used (QSPI, SRAM/NOR Flash, or NAND Flash). In addition, the QSPI and SMC also use the MIO[8:2] boot-strap pins, i.e., these MIO pins are dual-purpose. On platform power-up, they are input pins used for boot-mode detection, as already discussed in Section 3.6.1.1.1. Once the POR sequence is complete, the pins are available for use with the desired external memory option.

Note that the SD Card boot option *can* be used along with QSPI/SMC; in this case, the SD Card interface must use the MIO[45:40] pins (plus an extra pin for card-detect). This arrangement is used in the Zybo-Z7-20, as will be seen in the next chapter (Section 4.6).

3.6.4.2 Programmable Logic Initialization Signals

The remaining signals in Figure 3.18 are *PROGRAM_B*, *INIT_B*, and *DONE*, all of which are related to FPGA initialization. A logic-low level on the PROGRAM_B input will clear the programmable logic — a push-button is used in Figure 3.18 for this purpose, but any suitable control logic could be used. The *INIT_B* open-drain I/O goes low when the programmable logic is being programmed, and will stay low if there is a programming error. Finally, the *DONE* output is asserted at the end of the PL programming sequence; it is customary to connect an LED to this pin to indicate programming success. These pins are completely independent of the Zynq-7000 processing system, and have been standard signals in many generations of Xilinx FPGAs.

3.7 Debug System Overview

A brief summary of ARM processor debug techniques was given in the last chapter (Section 2.7), showing the distinction between invasive (e.g., halting-debug) and non-invasive (e.g., trace) debug methodologies. We also saw that the Cortex-A9 provides a range of interfaces to allow these techniques to be implemented as part of a SoC debug solution. For example, halting debug-mode is provided through access to CP14 and the related memory-mapped debug registers. A cross-triggering interface allows processor event signals to be

transmitted to other debug components (and vice-versa), and a program trace macrocell interface provides instruction tracing.

However, implementing an effective debug and trace system in a multi-core SoC is still a daunting prospect, and ARM developed the CoreSight architecture to make the task much easier [72, 70]. Fundamentally, this framework provides a wide range of modular debug components and buses which are integrated in a SoC to achieve a powerful debug solution. A Debug Access Port (DAP) is used to permit access to the CoreSight system — this allows external debuggers to connect to the debug infrastructure and system memory (or other elements) without halting the processor. Also, because CoreSight components can be connected up in a variety of ways, a topology detection framework is defined to allow debuggers to map out the system. This is achieved using one or more ROM tables in the design, which point to other CoreSight components in the system. Components are mapped to a 4KB space in system memory, and a subset of registers is reserved to support topology detection.

The main buses and components in the CoreSight architecture are summarised below.

- **Buses**: There are two main debug buses in a CoreSight system: the Debug APB, which provides access to the memory-mapped CoreSight components, and the AMBA Trace Bus (ATB) [79], which is used to transport trace data in the system. AHB [78] and AXI buses [77] are also used in the CoreSight system (to provide access to SoC memory, for example).
- **Control and access components**: The main control components are the Cross Trigger Interface (CTI) and the Cross Trigger matrix (CTM), both of which are part of the Embedded Cross Trigger (ECT) functionality [73]. The Debug Access Port (DAP) [73, 83] is the primary access component.
- **Trace source components** [73]: An embedded trace macro cell (ETM) is provided for instruction and data tracing, while a program trace macrocell (PTM) is available for instruction tracing only. The Instrumentation Trace Macrocell (ITM) is also available for more general `printf` style debugging, although this has been superseded by the System Trace Macrocell (STM).
- **Trace link components** [73]: These are effectively trace 'support' components, and they are used for the connection, triggering and flow of data between trace source and sink components. For example, a Trace Funnel is used to combine two trace sources into a single stream, while a Trace Replicator is used to duplicate a single stream onto two paths.
- **Trace sink components** [73]: Trace sink components are the endpoints for trace data. For example, a Trace Port Interface Unit (TPIU) is used to transfer data off-chip in real-time to an appropriate test tool. Alternatively, the trace data could be stored on-chip in another type of sink component, the Embedded Trace Buffer (ETB), for later extraction and analysis.

3.7.1 The Zynq-7000 CoreSight System

The Zynq-7000 debug system is strongly based on Arm CoreSight technology. Even basic Halting debug-mode, which doesn't necessarily need to be part of a CoreSight system, can be accessed through the framework. The system also includes embedded cross-trigger

components and a full trace solution. The complete implementation is quite complex, and so the Zynq-7000 debug system will be presented in three parts, as follows:

1. The first system (Section 3.7.1.1) is quite basic, with just Halting-debug, performance monitoring, and TCF profiling available. These features can be accessed in SDK or the Vitis software platform.
2. Cross-triggering features are added to the second system (Section 3.7.1.2) to show the debug interaction between the Cortex-A9 processors and the programmable logic. This functionality can also be accessed in SDK or the Vitis software platform.
3. Finally, the third system (Section 3.7.1.3) will illustrate the full CoreSight Debug and Trace implementation in the Zynq-7000. This is a complex arrangement, and unfortunately SDK or the Vitis software platform cannot be used in this case. Instead, a third-party solution such as the TRACE32® product line offered by Lauterbach Development Tools® must be used to configure, collect and view trace data in a Zynq-7000 system. An example of this will be seen in Section 3.7.1.3.4.

3.7.1.1 Basic System

Figure 3.19 shows the first system, consisting of a Debug Access Port (DAP), the Debug APB bus, two ROM tables, and one of the Cortex-A9 cores. (The second core could also be shown in this basic system, but it is left out for simplicity.) By using the Xilinx SDK tool, this view of the system provides standard halting-debug functionality, along with performance monitoring and simple TCF function profiling. Some system resources can also be accessed — for example, DDR memory can be read without halting the processor (as was required in legacy debug systems). Each component in this basic system is discussed next.

3.7.1.1.1 Debug Access Port

The DAP provides the bridge between external debug tools and the debug system on the Zynq-7000. There are two types of ports in the DAP: the Debug Port (DP); and the Access Port (AP).

Debug Port

The debug port connects to the external host, and it is typically available as a dual-interface Serial Wire JTAG Debug Port (SWJ-DP). This can be used in a low-pin count, serial-wire mode, or in standard JTAG mode. (Note that in the Zynq-7000, only the JTAG-DP option is implemented). Also, as discussed earlier in Section 3.6.4, the FPGA TAP controller is part of the JTAG chain by default, although it is possible to route the DAP signals separately in Independent JTAG mode if desired.

Access Ports and Debug APB

Access ports provide the interface between the debug port and the system debug components. There are two main types: the Memory Access Port (MEM-AP), and the JTAG Access Port (JTAG-AP). The JTAG-AP has limited connectivity in the Zynq-7000 (and is not shown in Figure 3.19), but there are two Memory Access ports. The first of these is the APB-AP port, which connects to the Debug APB bus on the system, allowing a host debugger to access the debug system and all CoreSight components. The second is the AHB-AP port, which is used to permit connection to the main system bus, allowing the debugger to interrogate system memory without having to halt the processor (for example).

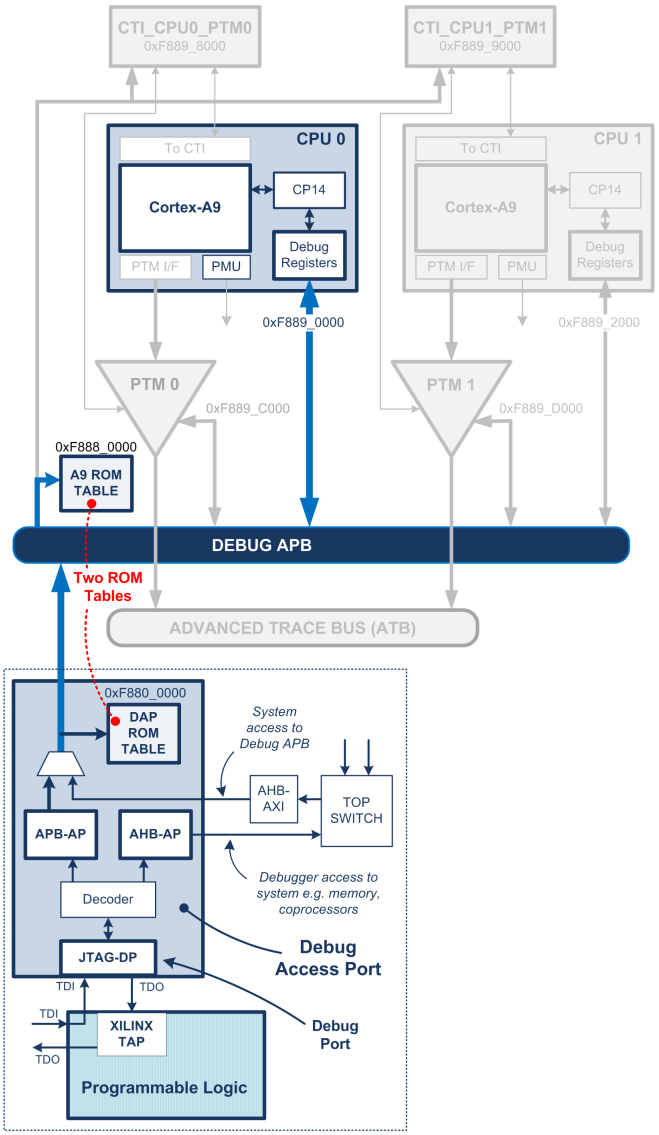

Figure 3.19. Basic halting-debug configuration, which also provides performance monitoring and TCF profiling.

3.7.1.1.2 ROM Tables

ROM tables are used to allow the external debugger to identify the target device (i.e., the Zynq-7000 in this case), and to allow discovery of all debug components in the system. The tables are themselves CoreSight components, occupying a 4KB memory space and containing information which marks them as ROM tables, rather than normal debug components. A series of addresses are contained in the table which point to other CoreSight components in the system, and these entries can also be other ROM tables. For example, in Figure 3.19 the DAP ROM table points to the address of the A9-ROM table. This second table contains address offsets for components related to the two Cortex-A9 cores, such as the PMU, PTM and CTI blocks.

3.7.1.1.3 Debug Options in Basic View

Three main debug options are available in this basic view, all easily accessible using SDK or the Vitis software platform. First of all, halting-debug mode can be used to single-step through the code, allowing details like processor registers, program variables, memory, and disassembly code to be viewed. A simple TCF Profiler is also available which allows the user to see where the program is spending its time, providing the developer with information on how the code might be optimised. The third debug option is provided by switching the development tool to the Performance Monitoring perspective, where the PMU is used to collect information on a range of processor events. Note that while the PMU isn't technically a CoreSight component, it connects to the system as if it was. For example, both PMUs in the Zynq-7000 system are included in the A9 ROM Table, and they each have their own 4KB space, along with associated ID registers.

These debug options are covered in more detail in Section 5.4.1.6 and Section 7.2.6.

3.7.1.2 Basic System + PL Cross-Triggering

The second view of the Zynq debug system is shown in Figure 3.20, and it illustrates the cross-triggering functionality between CPU core 0 and the programmable logic. (As with the first system, the second Cortex-A9 core could also be included, but again it is left out for clarity.) The new features in this view are discussed below.

3.7.1.2.1 Embedded Cross-Trigger (ECT)

The CoreSight Embedded Cross Trigger (ECT) functionality includes two main components: a Cross Trigger Interface (CTI), and a Cross-Trigger Matrix (CTM). The CTI contains eight trigger inputs and eight trigger outputs, allowing various components in the debug system to send trigger events and acknowledgements to each other. The CTM provides a channel interface to connect multiple CTIs together, allowing events to be distributed across the debug system. Usually, each CTI will be related to one or two main debug components — for example, in Figure 3.20 one CTI is associated with Core 0 and PTM0, while another CTI is solely associated with the Fabric Trace Monitor. The CTI components can be programmed via the Debug APB bus, but the CTM does not have a programming interface, as it is just used to link CTIs and CTMs together.

3.7.1.2.2 Fabric Trace Monitor

The Fabric Trace Monitor (FTM) is an application-specific CoreSight component developed by Xilinx. The main function of the FTM is to transfer trace data from the programmable logic to the CoreSight system (which will be covered in the next section), but it is also central to the PL-PS cross-triggering interface. It allows the digital design in the FPGA to send trigger signals to one or both MPCore processors, causing an application to halt just as if a hardware breakpoint had been reached. Cross-triggering also works in the opposite direction: the processor core(s) can send trigger signal(s) to the digital design, allowing the logic to be halted and analysed.

To use the cross triggering functionality in the Xilinx tool-chain, the relevant trigger inputs and outputs must first be enabled during the hardware design phase in Vivado.

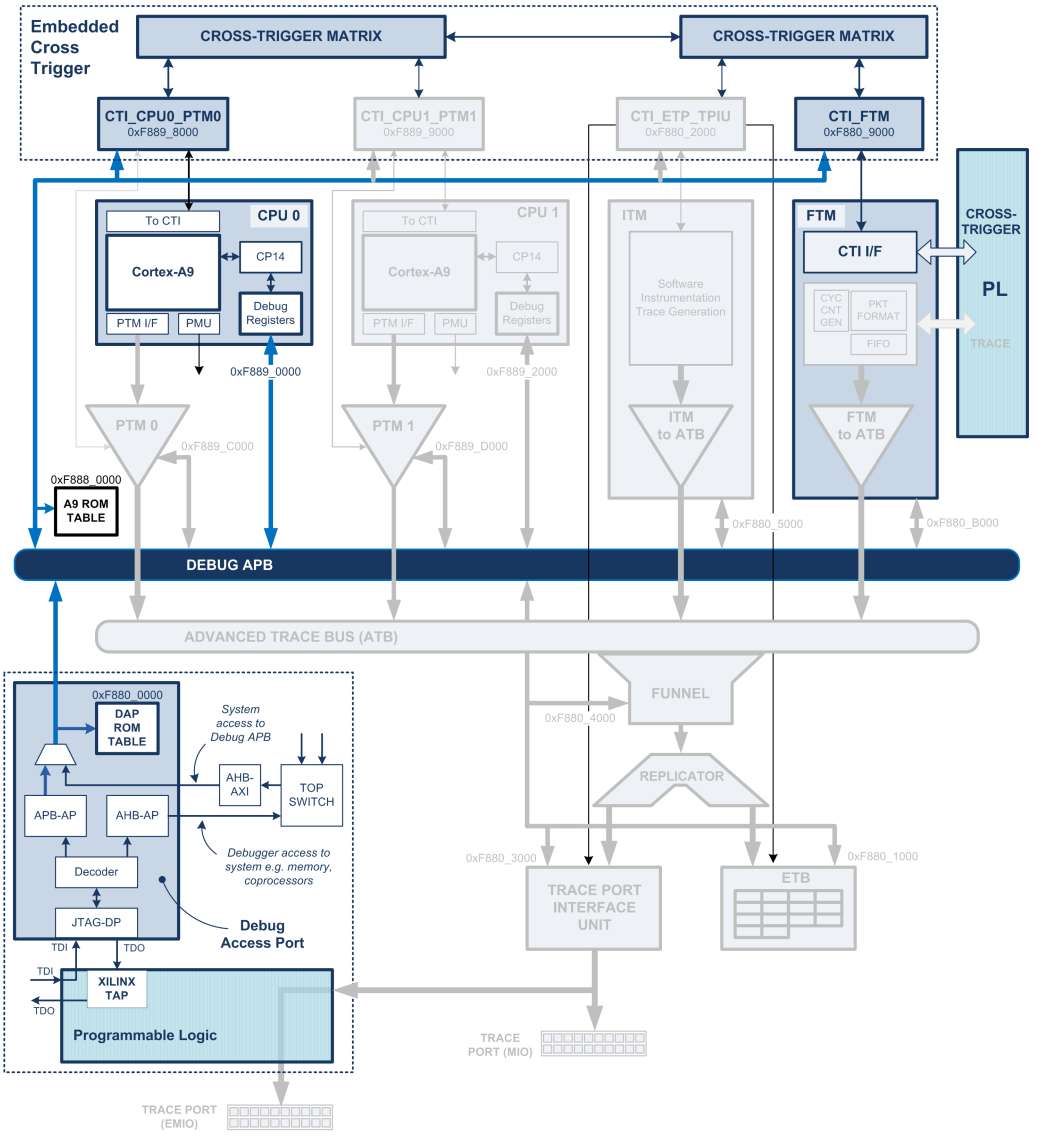

Figure 3.20. Halting-debug system with cross-triggering functionality added.

Integrated Logic Analyser (ILA) blocks are then added to the hardware design to monitor signals of interest. When the design is implemented, the Vivado Hardware Manager is used to configure trigger events, and view signals on a logic analyser-style interface. On the embedded software side, cross-triggering events can be set up using SDK or the Vitis software platform. (A comprehensive overview of this flow is given in [84] for the Xilinx SDK, and [62] for the Vitis software platform.)

3.7.1.3 Full System: CoreSight Trace and Debug

Finally, we come to the very comprehensive CoreSight trace and debug system in the Zynq-7000, as shown in Figure 3.21. A range of trace source, link and sink components have been added to the system, along with the Advanced Trace Bus (ATB). The components in this system are summarised next.

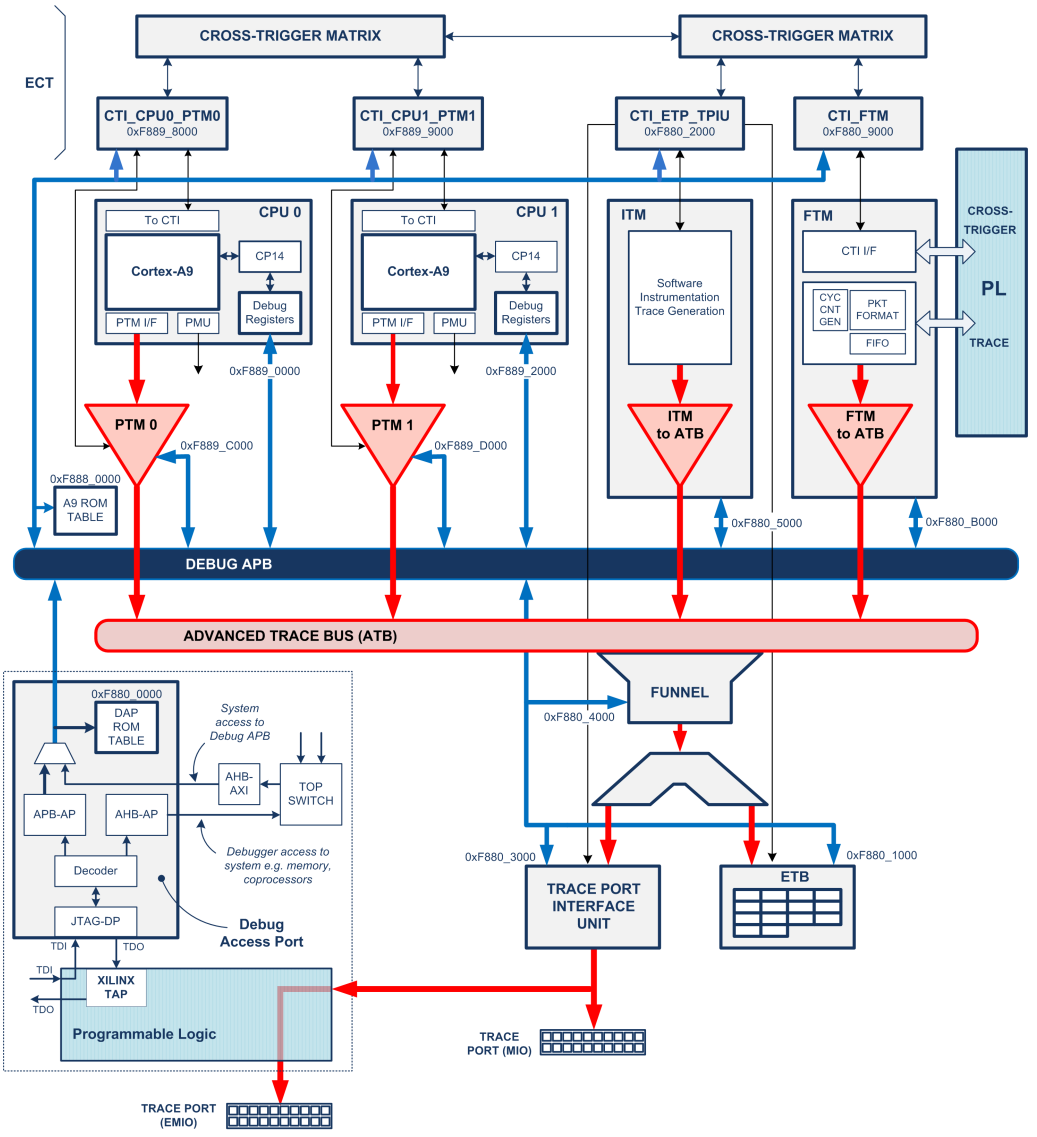

Figure 3.21. Complete Zynq-7000 CoreSight Trace and Debug system.

3.7.1.3.1 Trace Sources

Program Trace Macrocell

As mentioned earlier, Trace Macrocells come in two basic types: an Embedded Trace Macrocell (ETM), and a Program Trace Macrocell (PTM). The main difference between the two is that the ETM is used for tracing instructions and data, while the PTM is used for instructions only. Of these two macrocells, the PTM is implemented in the Cortex-A9, meaning that just an instruction trace can be generated. The PTM operates by tracking waypoints in the code, such as program branches and exceptions, and it also includes filtering and time-stamping

features. In a multi-core system, each core will be linked to its own PTM (i.e., PTMs are not shared).

Instrumentation Trace Macrocell

The Instrumentation Trace Macrocell (ITM) is used to support messaging-debug functionality, for example by outputting `printf`-style messages to a suitable debugger. It can also be used to trace OS and application events, and generate diagnostic system information.

Fabric Trace Monitor

Introduced earlier when discussing the ECT functionality (Section 3.7.1.2), the FTM is a trace source, collecting trace data from the digital design in the PL and passing it to the CoreSight system in the PS. It includes a packet formatter and cycle counter to generate suitable packets for the trace system. A 64-packet FIFO is also included, although trace data can still be lost if an overflow occurs. A comprehensive overview of the FTM is given in the Zynq-7000 TRM [39].

3.7.1.3.2 Trace Links

Trace Funnel

The Trace Funnel is used to combine multiple trace streams into a single trace output. In the Zynq-7000, the funnel takes the trace streams from the PTMs, the ITM, and the FTM, and passes the resulting stream to the Replicator.

Replicator

The Replicator connects a single incoming trace stream to two trace sinks, allowing identical trace data to be sent to each component. In the Zynq-7000, the trace sinks are the Embedded Trace Buffer (ETB) and the Trace Port Interface Unit (TPIU), both of which are discussed next.

3.7.1.3.3 Trace Sinks

Embedded Trace Buffer

The ETB allows trace data to be stored on-chip in real-time and at full speed, where it can be extracted later for analysis. Generally, though, the capacity will be very limited — the ETB in the Zynq-7000 is just 4KB in size, for example. (As an aside, if the ETB is not used for trace functionality, it can be used as normal memory, which might be relevant in memory-limited systems. Of course, the Zynq-7000 will rarely be part of a memory-limited system!)

Trace Port Interface Unit

The TPIU is used to transfer the trace stream to an external Trace Port Analyser (TPA), which could be a logic analyser or a third-party solution such as TRACE32® by Lauterbach Development Tools® (more on this below). In the Zynq-7000, the TPIU stream can be routed to MIO pins, or to the programmable logic via EMIO. Of these options, EMIO has two clear advantages: (1) The port size can be set to 32 bits (compared to 16 bits when MIO is used); and (2) there are likely to be many more I/O pins available on the programmable logic side (remember, the MIO pins are limited in the Zynq-7000). The EMIO approach also means that

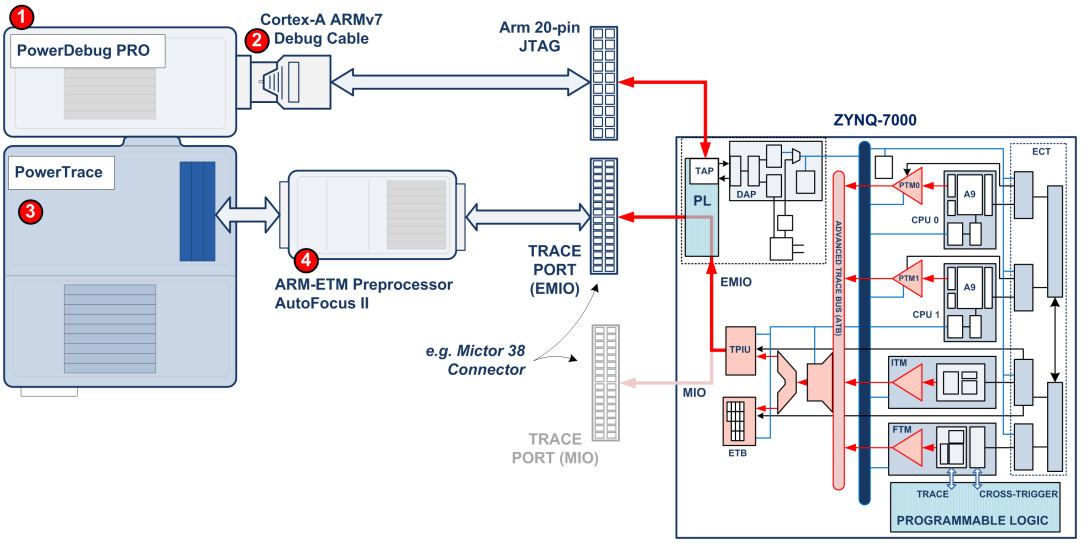

Figure 3.22. A typical configuration for Zynq-7000 CoreSight trace and debug, using the TRACE32® solution provided by Lauterbach Development Tools®.

the designer could implement a novel solution such as routing the trace steam back into DDR memory, and post-process the data in some other way. EMIO, then, is certainly the more favourable option.

3.7.1.3.4 Cortex-A9 Trace Solutions

The CoreSight trace feature is too complex to be supported by SDK or the Vitis software platform, and so the user must instead turn to third-party solutions. One popular option is the TRACE32® product line offered by Lauterbach Development Tools® [85], a company that provides a range of modular and upgradeable test development tools compatible with a wide range of microprocessors. To wrap up this section on the Zynq-7000 CoreSight debug and trace system, a typical TRACE32® configuration will be presented (see Figure 3.22). The main points can be summarised as follows:

1. The PowerDebug PRO is a universal debug controller, meaning that it is a base module for all TRACE32® debuggers/debug cables. The "PRO" version features Gigabit Ethernet and USB3.0 connectivity.
2. The Debugger for Cortex A/R (ARMv7 32-bit) encompasses both the related IDE and the Debug Cable, with the cable providing the interface between the PowerDebug PRO and the Zynq-7000 JTAG port. A 20-pin JTAG connector is typically used on the board side, although a range of adapters for other connection options are also available. Along with the associated IDE, the PowerDebug PRO and Debugger for Cortex A/R effectively replace the Xilinx software development tools in the test solution.
3. On the trace side, the PowerTrace is the primary component — this is a very high-bandwidth buffer that records trace information generated by the target (the Zynq-7000 in this case). Note that the PowerTrace and PowerDebug units are connected together, forming a synchronised system.

4. The final component in the trace path is the AutoFocus II Preprocessor for ARM-ETM — this provides the interface between the PowerTrace and the Zynq-7000. A 38-pin Mictor connector is typically used on the board, although other high-speed adapters are also available. Finally, while the trace data can be transmitted using either MIO or EMIO pins (as discussed earlier), the EMIO path is shown in the figure, as this is the most practical option.

3.8 Memory Map

To conclude this chapter, the Zynq-7000 memory map is shown in Figure 3.23. Only summary details are presented here, though, and as always the TRM [39] should be referred to for complete details (specifically Chapter 4). Some main points as they apply to the projects in this text are as follows:

- The OCM comprises four 64 KB sections which can be individually mapped to either the top or bottom of the memory map.
- The DDR occupies the lower 1 GB of the address space (with the exception that the OCM might be mapped to some or all of the lower 256 KB, as just mentioned). By default, the entry point of the user application is address **`0x0010_0000`**.
- In the FPGA design developed in Chapter 6, the AXI GPIO and QSPI modules are connected to M_AXI_GP0 in the PS-PL interface. The related register base addresses are **`0x4120_0000`** for the GPIO, and **`0x41E0_0000`** for the QSPI.
- The processing system I/O peripherals are mapped from **`0xE000_2000`** to **`0xE02F_0000`**. The UART1 and GPIO peripherals can be found in this area.
- The other processing system peripherals are mapped from **`0xF800_0000`** to **`0xF800_0BFF`**. For example, the SWDT, TTC, and OCM (control registers) can be found in this area.
- The CPU registers are mapped from **`0xF890_0000`** to **`0xF8F0_2FFF`**, and this area includes the Arm MPCore GIC and private timer registers.

Moving on

A comprehensive overview of the Zynq-7000 architecture has been presented in this chapter, and the trilogy of hardware topics will be completed in the next chapter with an analysis of the Digilent Zybo-Z7-20 development board.

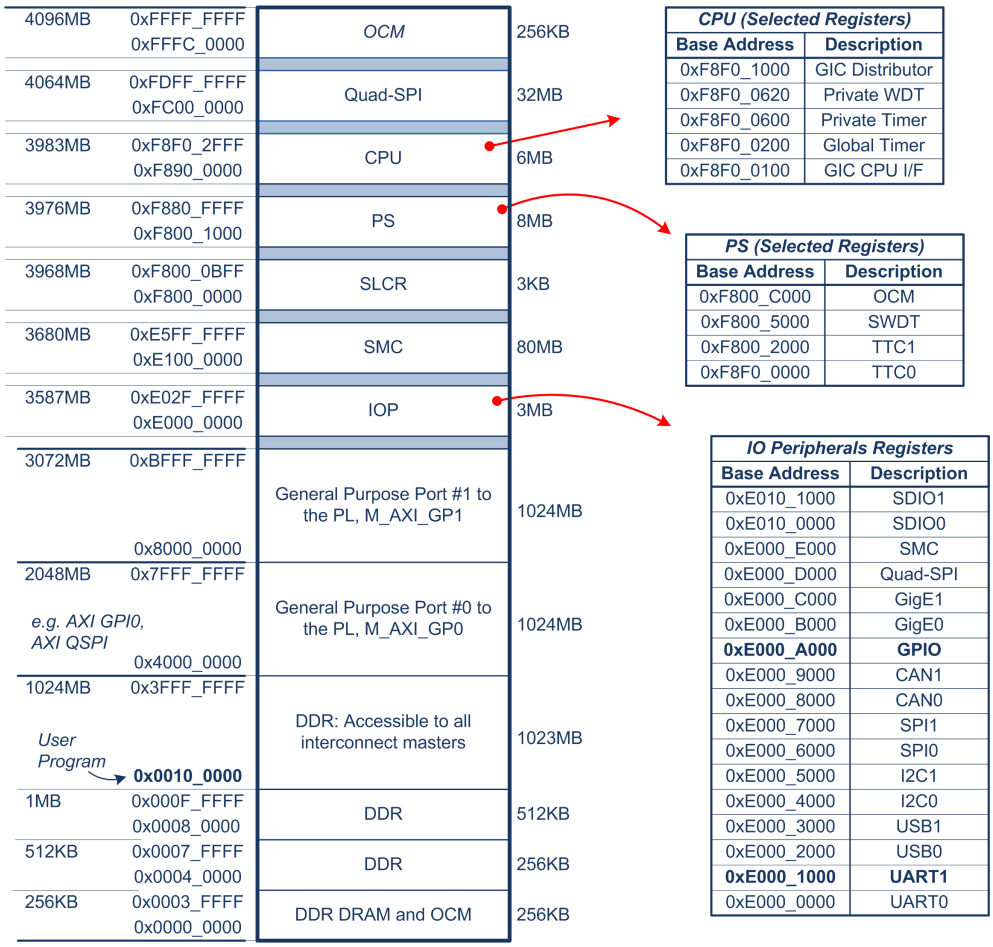

Figure 3.23. Zynq-7000 memory map, showing some locations which are relevant to the projects in this text.

3.9 References

[34] Cortex-A9 MPCore Technical Reference Manual (r3p0), ARM DDI0407G.

[38] CoreLink Level 2 Cache Controller (L2C-310) Technical Reference Manual (r3p3), ARM DDI0246H.

[74] CoreLink DMA Controller DMA-330 Technical Reference Manual (r1p1), ARM DDI042C.

[75] PrimeCell Static Memory Controller (PL350 series) (r2p1), Technical Reference Manual, ARM DDI0380G.

[39] Zynq-7000 SoC Technical Reference Manual, Xilinx UG585 (v1.12.2).

[76] AMBA Network Interconnect (NIC-301) Technical Reference Manual (r2p0), ARM DDI0397F.

[77] AMBA AXI and ACE Protocol Specification: AXI3, AXI4, and AXI4-Lite ACE and ACE-Lite, ARM IHI0022E.

[78] AMBA Specification (Rev 2.0), ARM IHI0011.

[79] AMBA 3 ATB Protocol Specification, IHI0032A.

[80] 7 Series FPGAs Memory Resources User Guide, Xilinx UG473 (v1.14) July 3, 2019.

[81] 7 Series FPGAs GTP Transceivers User Guide, Xilinx UG482 (v1.9) December 19, 2016.

[56] Zynq-7000 SoC Packaging and Pinout Product Specification UG865 (v1.8.1) June 22, 2018.

[82] 7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter User Guide, Xilinx UG480 (v1.10.1) July 23, 2018.

[72] ARM CoreSight Technical Introduction (Whitepaper), ARM-EPM-039795.

[70] ARM CoreSight Architecture Specification, IHI0029D.

[73] CoreSight Components Technical Reference Manual, DDI0314H.

[83] Arm Debug Interface Architecture Specification ADIv5.0 to ADIv5.2, IHI0031E.

[84] Vivado Design Suite Tutorial, Embedded Processor Hardware Design, Xilinx UG940 (v2019.1), June 27, 2019.

[62] Vivado Design Suite Tutorial Embedded Processor Hardware Design, Xilinx UG940 (v2020.2), December 14, 2020.

[85] Lauterbach development Tools®; www.lauterbach.com