XMOS glossary of terms

This document contains a list of the most common terms used when referring to XMOS products or applications written for XMOS multicore microcontrollers.

1 Board utilities

In xTIMEcomposer, the term *Board Utilities* refers to a collection of tools (including xRUN, xFLASH and xBURN) that are used to configure a device to boot from a host development PC, onboard FLASH memory or on-chip OTP memory.

2 Blocking

In the xCORE architecture, *Blocking* refers to events or operations that cause a logical core to pause. For example, a port input operation that causes core execution to be paused until a specific time has elapsed, is considered to be blocking. Similarly, channel communications can be blocking to ensure that the cores on each channel end are synchronized at the time of passing the message.

3 Buffered port

A *Buffered Port* holds data in a FIFO until the core is ready to input or output the data, instead of passing the data directly between pins and core. Buffered ports allow the core to execute other instructions during this time, and perform I/O operations on multiple ports in parallel.

4 Channels

In the xCORE architecture, *Channels* provide high-speed logical bidirectional communication and synchronization between channel ends within logical cores. The programming model is the same regardless of whether the channel ends reside on the same tile, a different tile on the same chip or a different chip connected via an xCONNECT Link. Channels are allocated and connected at runtime using instructions generated by the compiler.

5 Channel End

Each channel has two Channel Ends. Channel ends are a shared resource within each tile.

6 Clock

A *Clock* is a digital signal used to coordinate two or more actions on an xCORE tile.

Publication Date: 2014/10/1
Document Number: XM003706A

The xCORE architecture has three types of clock:

- System clock which controls the frequency of the xCORE tile
- Reference clock which controls the frequency of the logical cores
- External clocks which provide an input signal to control the frequency of logical cores

7 Clock block

Each xCORE tile has six *Clock Blocks* that include a programmable 8-bit prescaler. They can be clocked from the tile reference clock or through an input port. A clock block can be associated with one or more xCORE ports so that the internal timing of input/output program execution can be decoupled from synchronous I/O interface operations and I/O can be made on multiple ports at the same time.

8 Combinable task

A *combinable task* shares the processing time with other combinable tasks on a core, in between reacting to events. Tasks marked as combinable are compiled to run on a single processing core, saving xCORE resource use.

9 Composability

Composability refers to the principle of developing individual components that can be composed together to meet a particular defined software system. Each composable component can be verified and deployed independently as well as being integrated into a larger system, leading to greater reuse and easier update to software components.

For example, the XMOS XUD component is a composable component that provides a USB interface, but it can be combined with audio interfaces such as I2S or MIDI to create a USB Audio solution.

10 Communicate

In XMOS applications, tasks *communicate* via explicit transactions between them. Any task can communicate with any other, no matter which tiles and cores the tasks are running on.

The compiler implements the transactions in the most efficient way possible using the underlying communication hardware of the device.

11 Configuration

In the xTIMEcomposer tools, *Configuration* refers to setting values for the properties of an xSOFTip block. For example, the I2S Master Audio Driver has four

properties: DAC Channels, ADC Channels, MCLK Frequency and Initial Audio Sample Frequency. Each of these can be configured in xSOFTip Explorer.

12 Conditional Input

In the xCORE architecture, a *Conditional Input* causes data to be sampled by the port when the specified time or valued sample satisfies a condition, such as "is equal to" or "is not equal to". The input blocks until this time, taking the most recent data sampled.

13 Core

In the xCORE architecture, a *Core* behaves as a 32bit logical processor which takes a share of the main xCORE tile compute. Each core has its own set of registers and operates independently from other cores. It is scheduled on a round-robin basis by the xTIME scheduler, which guarantees a minimum performance.

Cores are triggered by events and run to completion unless paused, without affecting other logical cores on the tile. All cores can execute computation and handle real-time I/O operations.

Logical cores that are idle waiting for an event to occur, do not take up compute resource.

14 Core cycle

The *Core Cycle* specifies the logical core cycle time.

The core cycle time is derived from a set of dividers used on the system clock, which is itself generated from a low frequency external clock and the internal phase locked loop (PLL). The system clock divider can be by-passed so that the system clock frequency can be used as the core clock cycle.

15 Determinism

Determinism describes the ability of a system to deliver desired outcomes in a manner that is predictable in time. For example, a completely deterministic system will always deliver an output, in response to an input, at exactly the same instance.

Applications run on xCORE devices are deterministic due to the unique xCORE architecture. Tools such as the XTA use the determinism of the xCORE architecture to provide worst-case times for blocks of code, and guarantee that interfaces can meet timing requirements.

16 Distributable function

Tasks that contain state and provide services to other tasks, but do not need to react to any external events on their own, can be marked as *Distributable Functions* so that they share the logical cores of the tasks they communicate with.

17 DSP

Digital Signal Processing (DSP) is the mathematical manipulation of a series of data samples to modify or improve it in some way. For example, in audio applications signals are constantly converted from analog to digital, manipulated digitally using DSP, and then converted back to analog form. Electronic products often include a specialized microprocessor called a Digital Signal Processor to handle the digital signal processing requirements of the Application.

The xCORE architecture includes DSP operations that, except for the divide instruction, all run in a single cycle on the logical cores. These include a complex 32bit x 32bit multiply accumulate into 64bit precision, a long add, and 32bit CRC.

18 Ethernet AVB

Audio Video Bridging (AVB) is a set of standards under the IEEE 802.1 Ethernet standard. It defines a high quality streaming AV experience for real-time digital media networks.

The AVB standard has been developed under guidance of the IEEE and the AVnu Alliance (`http://www.avnu.org/`), of which XMOS is a member.

19 Event

In the xCORE architecture *Events* are used to trigger a logical core to start processing.

Event handlers exist in the current context so core execution can be controlled by the xTIME scheduler without the use of an 'if-then-else' structure or polling loops and no context switch is necessary to pull data from memory or caches. This means that event response times are virtually immediate and permit very fast state machines to be programmed in software.

In an application, tasks can react to events using the `select` construct, which pauses the tasks and waits for an event to occur. A `select` can wait for several events and handle the event that occurs first.

Events can be generated by channels, ports and timers.

20 Firmware

The function of a multicore microcontroller is defined by the application it runs. Since the application typically consists of hardware and software tasks, the code modules are collectively called the *Firmware*.

21 Formal Timing Analysis

Formal Timing Analysis describes the use of XMOS Timing Analyzer (XTA) static timing analysis to provide 100% formal coverage of an entire block of functional code.

Following XTA analysis, pragmas can be added into the application code to confirm all timing requirements are met at build time. The tools will provide warnings if the application timing requirements are not met.

22 Hardware Real-Time Operating System

A *Hardware Real-Time Operating System* such as xCORE, delivers the functions required for real-time systems in hardware instead of a separate RTOS. The dedicated Hardware-Response ports and xTIME scheduler make sure that the architecture is deterministic and none of the xCORE processor MIPS are wasted on scheduling or task switching.

Applications built on a Hardware RTOS based architecture are guaranteed to run reliably.

23 Hardware-Response

In the xCORE architecture, *Hardware-Response* ports synchronize the events that occur at the I/O pins with a processing core. Input and output operations are handled by a single instruction reducing latency and providing levels of performance usually associated with hardware.

24 Interface

An *Interface* is an xC construct that provides a structured and flexible method of inter-task communication.

The interface defines the kind of transaction between the tasks, and the data that is passed between them.

25 Jitter

Jitter is a measure of the difference between the time of a transmitted signal and time the logical core receives the signal.

xCORE devices show minimal jitter due to the very low latency of the architecture and the use of a high quality application clock to drive I/O asynchronously.

26 JTAG

Joint Test Action Group (JTAG) is an industry standard interface that provides a method for testing silicon implementations, including boundary scanning and onboard source level debugging options.

XMOS development boards use the xTAG debug adapter to provide access to the JTAG interface via the xSYS connector.

27 Latency

Latency is a measure of the amount of time taken to perform an action after an event or incoming data has occurred to initiate that action.

The XMOS architecture is extremely efficient with very low latency. Most of the instructions, including input and output instructions, operate in a single core cycle so data is passed to a logical core at the same time as the instruction, providing the core is ready to execute the instruction. This removes any latency associated with a real-time operating system (RTOS) which has to service an interrupt, change context to retrieve data and then trigger the processor. For multi-core, multi-tile or multiple device systems, latency remains very low as data is passed via communication channels managed in hardware by the xCONNECT switch. The same single-cycle instructions are used to pass data along the channels.

28 libflash

libflash is a library supported by xCORE that includes functions for reading and writing data to flash devices.

xCORE devices support a set of SPI devices natively. Additional SPI devices can be added to the list by writing a configuration file using the libflash library.

The lib flash library also provides functions to manipulate program image for boot images written by xFLASH.

29 Library

A software *Library* contains a set of functions that implement a defined functionality on any multicore microcontroller based on the xCORE architecture. For example the CAN library can be used in an application to implement the CAN 2.0A/B protocol. The CAN library includes functions that handle data rates, as well as functions to support standard and extended frame formats. Similarly the libflash library includes functions for reading and writing data between an xCORE and flash devices.

A library can include additional complete libraries that implement a specific feature. For example a USB Audio library may include I2S and USB libraries as well as functions for managing data buffering.

Some library functions implement components that relate closely to a hardware component on a traditional microcontroller, such as an I2C interface.

30 Message

Data transmitted across a channel is called a *Message*. Since different types of data can be transmitted, a protocol exists that allows different size messages to be exchanged between channel ends.

31 MIPS

Million Instructions Per Second (MIPS) is the count of how many instructions executed per second.

The xCORE tile runs from single cycle on-chip memory, so it can sustain an issue rate of one instruction per clock and hence f (MHz) = MIPS

32 Multicore

In the xCORE architecture *Multicore* is an abbreviation for multiple logical cores and refers to a silicon device with more than one processing core.

33 Multicore Microcontroller

XMOS *Multicore Microcontrollers* are programmable devices that consist of one or more xCORE tiles, each running up to eight 32bit logical processing cores. Each logical core can process I/O, DSP and application code concurrently, and shares an integrated SRAM memory. xTIME schedulers make sure that each core runs to completion without deadlock, providing a completely deterministic software programmable device that can be used to implement tasks traditionally implemented in hardware.

One or more logical cores may be dedicated to communication with an integrated processor, such as an ARM CORTEX-M3, as part of the xCORE Extended Architecture (xCORE-XA).

34 Multiplex

Multiplexing refers to the sharing of resources.

The most common XMOS example is the multiplexing of pins with various size ports.

35 Operating System

An *Operating System* (OS) is the software that manages the sharing of the resources of a computer and provides programmers with an interface used to access those resources.

The tiles in xCORE devices do not use an operating system. All functions required for core scheduling, event handling, channel communication and timer allocation are implemented in hardware by the xTIME scheduler.

Removing the operating system reduces the latency and jitter in xCORE devices, makes xCORE devices easier to program, and reduces memory requirements.

36 OTP

On-tile *one-time programmable memory* (OTP) ensures that systems can be made secure, with secure boot code and encryption keys safely held.

External flash memory can be used to add as much program code as the system needs.

Each xCORE tile includes 8KB of OTP.

37 Parallelism

Parallelism is an approach to computing in which many instructions are carried out simultaneously. Developers can divide large problems into smaller ones, which can be carried out concurrently.

xCORE multicore microcontrollers are specifically designed to support parallel applications with the ability for logical cores running on a single xCORE tile to communicate with processing cores on the same tile or other xCORE tiles with extremely low latency.

38 Pin

The *Pin* is the physical node on the chip that connects to outside world. Modern surface-mount IC packages typically use a pad or a ball instead of a pin. Pins connect to various internal parts of the device including power supply, clocks, control, test access and ports and are often multiplexed.

39 Port

In the xCORE architecture, a *Port* is a software construct that connects a processor to one or more physical pins, defining the interface between a processor and its environment.

The port logic samples the value on its pins, optionally waiting for a particular condition, or drives its pins high or low. Data can be held in a double-buffered FIFO until the port is ready to input to the processor core. Ports can also serialize and deserialize data so reducing the number of instructions required to input or output data.

Ports are clocked by the master reference clock block on the tile. Alternatively they can be clocked by a separate clock block or input signal, allowing multiple ports to be driven asynchronously to the main application clock.

Each xCORE tile exposes a combination of 1,4, 8, 16 and 32bit ports, depending on the package and resources used internally. The total number of possible GPIO far exceeds the number of pins actually bonded out, so ports and xCONNECT Links are multiplexed. There is a defined precedence when overlapping ports and links exist.

40 Port counter

Each port has a *Port Counter* (also known as a *Port Timer*) that can be used to control the time at which data is transferred between the port value and transfer register. Port counters are 16-bit and tick once for each cycle of the clock input, incrementing on the falling edge of the signal. The counter values can be obtained at any time to find out when data was obtained, or used to delay I/O until some time in the future.

41 Priority Inversion

Priority Inversion occurs when a low priority task blocks a high priority task by holding a shared resource, for example when communicating via a channel that connects two tasks.

The use of events in the xCORE architecture and the select construct reduces some of the problems of priority inversion, however, if there are multiple event handlers in the `select` high priority tasks may be blocked.

42 Real-time Operating System

A *real-time operating system* (RTOS) is an OS designed to run applications that need to meet precise timing requirements and provide a high degree of reliability. To meet precise real-time requirements, the RTOS must guarantee to deliver a known maximum time for each of the critical tasks. Some of these tasks include operating system calls to hardware, or device drivers, as well as interrupt handling. An RTOS that can absolutely guarantee a maximum time for tasks is termed *hard real-time*. Most commercial RTOS however can only guarantee a maximum, most of the time and are referred to as *soft real-time*. As a result considerable design margins need to be added to try and ensure reliable operation.

An RTOS is often used in control applications, or for communications systems, in test and measurement, or in safety critical applications such as medical, automotive or avionics systems.

43 Reference clock

Each xCORE tile has a *Reference Clock* block, which runs at a frequency of 100MHz. Clock blocks may use the reference clock as their clock source and divide the frequency using an 8-bit divider.

The reference clock is derived from a set of dividers used on the system clock, which is itself generated from a low frequency external clock and the internal phase locked loop (PLL). The system clock divider can be by-passed so that the system clock frequency can be used as the reference clock cycle.

44 Resource

A *Resource* is an element of an xCORE tile that can be dynamically allocated or shared. Resources include: logical core, port, timer, channel end, synchronizer and clock block.

45 Select

Select is a programming construct in the XMOS multicore extensions for C, that instructs the logical core to wait for an event before it starts processing. `select` removes the requirement for interrupts to handle events, providing a way to write very efficient state machines with low latency and deterministic I/O.

46 Serialization

Disassembling parallel data into serial data is called *Serialization* and is a common requirement within hardware interfaces.

Streamed ports in the xCORE architecture can serialize and de-serialize data using single instructions making them very efficient.

47 Simulator

The XMOS *Simulator* (xSIM) provides cycle-based simulation allowing developers to build an accurate software model of an application without hardware.

xTIMEcomposer can output data sampled by xSIM to a file for off-line analysis, while xTIMEcomposer Studio provides different graphical views of the data sampled by the simulator.

xSIM can also connect a port on one xCORE tile to a port on another tile, allowing both sides of an interface to be modelled.

48 sliceKIT

sliceKIT is a modular development system for xCORE devices. Each kit consists of a core board with a 16-core multicore microcontroller (xCORE General Purpose, xCORE-USB or xCORE-ANALOG) and one or more I/O sliceCARDS that plug into PCIe style connectors on the core board.

Multiple core boards can be connected together if more xCORE resources or I/O connectors are required.

sliceKIT is supported by xSOFTip libraries, demos and peripherals making it the most flexible development solution available.

49 Strobing

The xCORE architecture provides support for *Strobing*, where data is accompanied by a separate data valid signal. Up to two pins can be used to strobe data (readyIn and readyOut signals) providing four strobe modes: implicit strobing, slave strobing, master strobing and bidirectional strobing.

50 System Clock

The *System Clock* controls the speed at which the xCORE device runs. It is specified in the XN file.

Each logical core is guaranteed *N* system clock cycles where N is the number of active logical cores on the tile. If less than four cores are active on a tile, N equals four.

51 Task

A *Task* consists of a sequence of instructions implementing some function. This function may drive an I/O interface, run some DSP or some application code. A task runs on the logical cores of the xCORE device. Some tasks run on their own logical core and in some cases multiple tasks can be run on the same logical core.

52 Tile

Each xCORE multicore microcontroller includes one or more xCORE *Tiles* which contain the logical processing cores and other xCORE Resources. Tiles also include the xCONNECT switch for communication with other tiles on the same chip or other chips.

Management of the tile resources is controlled by the xTIME scheduler and Hardware Response ports.

53 Timed ports

A *Timed Port* uses a port counter to determine when to perform an input or output. The port waits until the counter has reached the specified value.

54 Timer

Each xCORE tile includes a set of eight programmable 32bit *Timers* that can be used to generate events which trigger program execution. Timers increment at a rate of 100MHz providing a fixed time base regardless of tile frequency. Timer values can be input at any time or delayed until a time in the future.

55 Timestamp

Each port can be *Timestamped* so that the value of the port counter at which data is transferred in or out of the transfer register is captured in the timestamp register. This value can be accessed by the application allowing input and output operations to be made a precise times in the future.

56 Tool chain

See xTIMEcomposer.

57 Transaction

A *Transaction* defines the type of communication between two tasks that exchange data over an interface. Interface functions take the transaction type as an argument to define what data is sent when the transaction between the tasks occurs.

58 Workspace

A *workspace* is a container for several projects in xTIMEcomposer Studio.

59 xBURN

xBURN is a board utility that creates OTP images, and programs images into the OTP memory of xCORE devices.

60 xC

xC provides extensions to the C language that simplify the control over concurrency, I/O and time. These extensions map onto xCORE hardware resources such as cores, channels and ports, avoiding the need to make extensive use of library calls.

xC also includes extensions for secure memory management.

61 XCC

The *XMOS Compiler Collection* (XCC) generates a single binary file that includes instruction and data segments for all devices.

62 xCONNECT

The xCORE architecture includes a high-speed network called *xCONNECT*, which ensures that all logical cores can communicate. The xCONNECT network is also extended to dedicated I/O pins allowing multiple xCORE devices to be connected together so that larger real-time systems can be created. The xCORE programming API provides channels that are used to logically connect real-time tasks. Channels

can be easily configured in a number of different ways to ensure that the data communication is synchronized between tasks.

63 xCONNECT Link

xCONNECT Link is an umbrella term covering all types of inter-core communication on xCORE devices. Links are represented by channels in xC which are physically implemented using channel ends connected to a logical core, and managed by the xCONNECT switch.

64 xCORE

An *xCORE* is a single instance of an XMOS multicore microcontroller, consisting of one or more tiles which each have a collection of resources that include cores, channels and ports. The XS1 family of multicore microcontrollers includes devices with 4, 6, 8, 10, 12, 16, 32 cores.

65 xCORE-Analog

The *xCORE-Analog* family is a comprehensive range of 32bit multicore microcontroller that combines the low latency and timing determinism of the xCORE architecture with the simplicity and ease of use of integrated analog peripherals.

The integrated analog peripherals provide low power sleep modes, simplified board design and precise control of analog inputs. The integrated 12bit 1MSPS analog-to-digital converter offers up to eight multiplexed input channels. The sampling of the ADC is controlled directly by the xCORE Hardware-Response ports, allowing accurate control of the ADC from software. Together with the deterministic execution of the xCORE, this enables the sampling of the ADC to be precisely controlled by application functions.

66 xCORE General Purpose

xCORE General Purpose is a comprehensive range of 32bit multicore microcontroller that brings the low latency and timing determinism of the xCORE architecture to mainstream embedded applications.

Software-programmable in an environment that will be familiar to any C programmer, these devices allow you to blend control code, DSP and software-defined interfaces.

xCORE-General Purpose devices contain between 4-16 logical cores with 400-1000 MIPS and 28-128 user defined GPIO.

67 xCORE Tile

An *xCORE tile* contains up to eight 32bit logical processing cores. Each core can use the xCONNECT switch to communicate with other cores on the same tile, with

cores on the same device, or cores on a different device using xCONNECT Links. Each tile has integrated SRAM memory that is shared by the logical cores.

68 xCORE-USB

The *xCORE-USB* family of multicore microcontrollers combines the flexibility, low latency and determinacy of xCORE, with an integrated High Speed USB 2.0 PHY supporting 480Mbps data rates.

Available in variants with 6, 8, 10, 12 and 16 logical cores, the family addresses a range of demanding applications including high performance peripherals, audiophile consumer audio, sound-bars, multi-channel USB audio interfaces, DJ products, USB speakers, and protocol conversion plus bridging.

Additional features include a multichannel 12bit 1MSPS analog-to-digital converter, standby and deep sleep modes for energy-sensitive applications, power-on-reset, watchdog timer, brownout detection and integrated oscillator circuits.

69 xCORE-XA

The *xCORE eXtended Architecture* (xCORE-XA) family of multicore microcontroller combines the flexibility, low latency and determinacy of xCORE, with an ultra-low power ARM® Cortex® M3 processor, effectively functioning as an extra core within the xCORE architecture.

Available in variants with 7 logical cores and the ARM core, the xCORE-XA allows embedded system designers to use high-level software to configure a device with the exact set of interfaces and peripherals needed for their design, while at the same time including existing ARM binary code or standard libraries, and taking advantage of ultra-low energy peripherals.

70 xFLASH

The *xFLASH* board utility reads/writes data to a SPI device connected to the xCORE chip.

71 xGDB

The *XMOS Debugger* (xGDB) extends the industry-standard GDB debugger to support multicore development. The debugger interacts with all xCORE tiles on the target platform, presenting a collection of logical cores to the programmer that can be viewed together. This allows XMOS devices to be debugged in the same way as conventional processors.

72 xKITS

xKITS is an umbrella term for all XMOS development kits including starter kits, sliceKIT and development modules.

73 XN

An *XN* file that describes the target platform in XML. The platform may include networks of xCORE devices, SPI FLASH memory, an oscillator and JTAG scan chain. XN files allow xTIMEcomposer to fully automate system boot and configuration.

74 xRUN

xRUN loads a compiled binary onto an XMOS development board.

75 xSIM

The xCORE simulator *xSIM* provides a near cycle-accurate model of systems built from one or more xCORE devices. Using the simulator, you can view a processor's instruction trace, visualize machine state and configure loopbacks to model the behavior of components connected to XMOS ports and links.

76 XS1 Architecture

The *XS1 Architecture* consists of one or more xCORE devices, each comprising a multicore microcontroller architected for real-time performance with tightly integrated I/O and on-chip memory.

The architecture is deterministic, with each logical core guaranteed a slice of the processing. The cores can execute computations, handle real-time I/O operations and respond to multiple events. The I/O pins can be sampled or driven using a single instruction, and data rates can be controlled using timers or clocks. A high performance switch enables communication between tiles and makes it easy to construct systems from multiple xCORE devices. Communication between cores on a processor incurs no latency, and between processors the latency can be determined for a known communication pattern.

77 xSCOPE

xSCOPE is a software library that provides real-time, in-circuit instrumentation of user specified data probes, with minimal effect on your design or device operation. Once connected, xSCOPE behaves exactly like a traditional oscilloscope.

Each probe outputs the user data, the formatting, a core ID (in case multiple cores are running the same code) and a 10ns time stamp to a file for offline analysis or to a real-time view in xTIMEcomposer Studio.

78 xSOFTip

xSOFTip is a collection of libraries of fully implemented tasks that support a broad range of interfaces and processing functions. Each component is delivered complete with documentation covering the usage, API and an application example.

79 xSOFTip Explorer

xSOFTip Explorer is a graphical tool for browsing the xSOFTip libraries. xSOTip Explorer lets you browse IP blocks by type and category, and includes a drag-and-drop interface that lets you configure the blocks from the xSOFTip libraries. xSOFTip Explorer shows the total of the xCORE resources used by each component and lists possible devices to support your selected IP blocks. xSOFTip Explorer is integrated into xTIMEcomposer Studio so you can add your blocks directly into your project, and also available as a standalone tool for exploring application solutions.

80 XTA

The *XMOS Timing Analyzer* (XTA) tool uses the deterministic features of the xCORE architecture to analyze application binaries and report the worst and best case timing paths through the code. It offers 100% code coverage without the need for a test bench, eliminating the need to over specify processing power. Knowing exactly what the worst case is means that deadlines can be guaranteed to be met.

The XTA can be used within xTIMEcomposer providing a graphical view of timing paths between two points in the software, or it can script driven and integrated into individual tool chains so that code can be verified on each build.

81 xTAG

The *xTAG* debug adapter provides a high speed, low latency bridge between USB on the host and fast JTAG connection (up to 10Mbps) on the target. Unlike other adapters, the debug stub is executed on the adapter itself. This ensures lightning quick debug functions, such as single stepping, by massively reducing the communications overhead.

In addition to the standard debug capability, optional UART and xCONNECT ports are supported, providing communication with the xCORE during run time. These ports can be used to print to the console and are the key to the very high performance xSCOPE real time instrumentation, which supports application data capture at up to 1MSPS. The debug functions are completely soft because the xTAG only contains a USB bootloader, allowing debug capability to be changed or improved on the fly via firmware.

82 xTIMEcomposer

xTIMEcomposer is the suite of tools used to compile and debug code for a multicore microcontroller built on the xCORE architecture.

The tools are based on a standard embedded software flow that supports C/C++/xC. The xTIMEcomposer compilers are based on the LLVM compiler (http://llvm.org/) and GDB, the GNU debugger (http://www.gnu.org/software/gdb/). Multiple debuggers provide support for xCORE tiles/processors as well as integrated third party processors such as an ARM CORTEX-M3.

The tools include the XTA which can be used to validate that hard real-time constraints are met on your target device and the xSCOPE library which allows developers to instrument their applications and read output data offline or in realtime in an xTIMEcomposer Studio view. Utilities for deploying compiled binaries onto your development board are also included.

xTIMEcomposer can be driven from xTIMEcomposer Studio, a graphical development environment based on the Eclipse IDE (http://www.eclipse.org/), or the command line. xTIMEcomposer is available on Windows, Mac and Linux platforms.

83 xTIMEcomposer Studio

xTIMEcomposer Studio is the graphical development environment for the xTIMEcomposer tools suite, based on the industry standard Eclipse platform IDE. xTIMEcomposer Studio includes enhanced and additional views to make it easier for users to write multicore applications for xCORE devices, all in a familiar and flexible development environment.

84 xTIME Scheduler

The *xTIME Scheduler* handles the events generated by channels and timers, ensuring that all events are serviced in 20ns and synchronized with tasks running on separate logical core. Events that occur at the I/O pins are handled by the Hardware-Response I/O ports and fed directly to the appropriate logical core.

85 XUD

The *XMOS USB Device* (XUD) is a task that runs on a single xCORE device and implements USB Device functionality.