

# XS1-L8A-64-TQ128 Datasheet

---

## Table of Contents

1	xCORE Multicore Microcontrollers	2
2	XS1-L8A-64-TQ128 Features	4
3	Pin Configuration	5
4	Signal Description	6
5	Product Overview	8
6	PLL	10
7	Boot Procedure	10
8	Memory	14
9	JTAG	14
10	Board Integration	15
11	DC and Switching Characteristics	20
12	Package Information	24
13	Ordering Information	25
Appendices		26
A	Configuration of the XS1	26
B	Processor Status Configuration	28
C	Tile Configuration	37
D	Node Configuration	45
E	XMOS USB Interface	52
F	Device Errata	52
G	JTAG, xSCOPE and Debugging	53
H	Schematics Design Check List	55
I	PCB Layout Design Check List	58
J	Associated Design Documentation	59
K	Related Documentation	59
L	Revision History	60

---

### TO OUR VALUED CUSTOMERS

It is our intention to provide you with accurate and comprehensive documentation for the hardware and software components used in this product. To subscribe to receive updates, visit <http://www.xmos.com/>.

XMOS Ltd. is the owner or licensee of the information in this document and is providing it to you "AS IS" with no warranty of any kind, express or implied and shall have no liability in relation to its use. XMOS Ltd. makes no representation that the information, or any particular implementation thereof, is or will be free from any claims of infringement and again, shall have no liability in relation to any such claims.

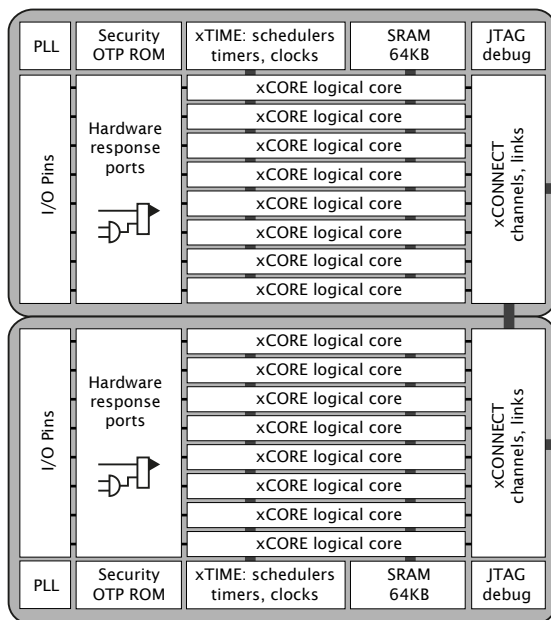
XMOS and the XMOS logo are registered trademarks of XMOS Ltd in the United Kingdom and other countries, and may not be used without written permission. Company and product names mentioned in this document are the trademarks or registered trademarks of their respective owners.

# 1 xCORE Multicore Microcontrollers

The XS1-L Series is a comprehensive range of 32-bit multicore microcontrollers that brings the low latency and timing determinism of the xCORE architecture to mainstream embedded applications. Unlike conventional microcontrollers, xCORE multicore microcontrollers execute multiple real-time tasks simultaneously. Devices consist of one or more xCORE tiles, each containing between four and eight independent xCORE logical processors. Each logical core can execute computational code, advanced DSP code, control software (including logic decisions and executing a state machine) or software that handles I/O.

Because xCORE multicore microcontrollers are completely deterministic, you can write software to implement functions that traditionally require dedicated hardware. You can simulate your program like hardware, and perform static timing analysis using the xTIMEcomposer development tools.

The devices include scheduling hardware that performs functions similar to those of an RTOS; and hardware that connects the cores directly to I/O ports, ensuring not only fast processing but extremely low latency. The use of interrupts is eliminated, ensuring deterministic operation.



**Figure 1:**  
XS1-L Series:  
4-16 core  
devices

XS1-L devices are available in a range of resource densities, package, performance and temperature grades depending on your needs. XS1-L devices range from 4-16 logical cores divided between one or two xCORE tiles, providing 400-1000 MIPS, up to 84 GPIO, and 64Kbytes or 128Kbytes of SRAM.

## 1.1 xSOFTip

xCORE devices are backed with tested and proven IP blocks from the xSOFTip library, which allow you to quickly add interface and processor functionality such as Ethernet, PWM, graphics driver, and audio EQ to your xCORE device.

xSOFTip blocks are written in high level languages and use xCORE resources to implement given function. This means xSOFTip is software and brings the associated benefits of easy maintenance and fast compilation time, while being accessible to anyone with embedded C skills.

The graphical xSOFTip Explorer tool lets you browse available xSOFTip blocks from our library, understand the resource usage, configure the blocks to your specification, and estimates the right device for your design. It is included in xTIMEcomposer Studio or available as a standalone tool from [xmos.com/downloads](http://xmos.com/downloads).

## 1.2 xTIMEcomposer Studio

Designing with XS1-L devices is simple thanks to the xTIMEcomposer Studio development environment, which includes a highly efficient compiler, debugger and device programming tools. Because xCORE devices operate deterministically, they can be simulated like hardware within the development tools: uniquely in the embedded world, xTIMEcomposer Studio therefore includes a static timing analyzer, cycle-accurate simulator, and high-speed in-circuit instrumentation.

xTIMEcomposer can also be used to load the executable file onto the device and debug it over JTAG, programmed it into flash memory on the board, or write it to OTP memory on the device. The tools can also encrypt the flash image and write the decryption key securely to OTP memory.

xTIMEcomposer can be driven from either a graphical development environment that will be familiar to any C programmer, or the command line. They are supported on Windows, Linux and MacOS X and available at no cost from [xmos.com/downloads](http://xmos.com/downloads).

Information on using the tools is provided in a separate user guide, [X3766](#).

## 2 XS1-L8A-64-TQ128 Features

### ► Eight-Core Multicore Microcontroller with Advanced Multi-Core RISC Architecture

- Up to 500 MIPS shared between up to 8 real-time logical cores
- Each logical core has:
  - Guaranteed throughput of between  $\frac{1}{4}$  and  $\frac{1}{8}$  of tile MIPS
  - 16x32bit dedicated registers
- 159 high-density 16/32-bit instructions
  - All have single clock-cycle execution (except for divide)
  - 32x32→64-bit MAC instructions for DSP, arithmetic and user-definable cryptographic functions

### ► Programmable I/O

- 64 general-purpose I/O pins, configurable as input or output
  - Up to 16 x 1bit port, 6 x 4bit port, 4 x 8bit port, 2 x 16bit port, 1 x 32bit port
  - 4 xCONNECT links
- Port sampling rates of up to 60 MHz with respect to an external clock
- 32 channel ends for communication with other cores, on or off-chip

### ► Memory

- 64KB internal single-cycle SRAM for code and data storage
- 8KB internal OTP for application boot code

### ► Hardware resources

- 6 clock blocks
- 10 timers
- 4 locks

### ► JTAG Module for On-Chip Debug

### ► Security Features

- Programming lock disables debug and prevents read-back of memory contents
- AES bootloader ensures secrecy of IP held on external flash memory

### ► Ambient Temperature Range

- Commercial qualification: 0°C to 70°C
- Industrial qualification: -40°C to 85°C

### ► Speed Grade

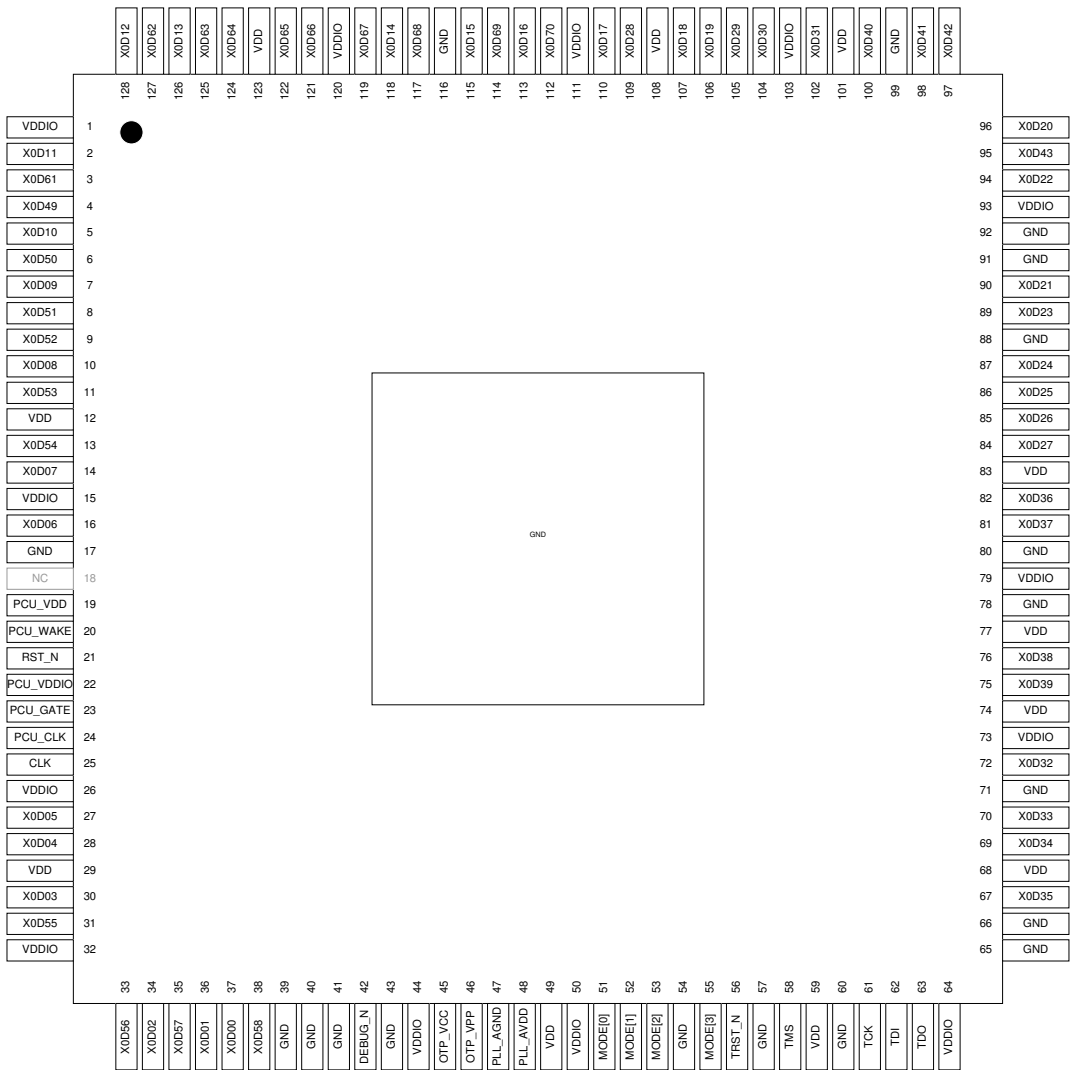
- 5: 500 MIPS
- 4: 400 MIPS

### ► Power Consumption

- Active Mode
  - 200 mA at 500 MHz (typical)
  - 160 mA at 400 MHz (typical)
- Standby Mode
  - 14 mA
- Sleep Mode
  - Programmable PCU module puts device into sleep mode
  - Wakeup on external signal or timeout

### ► 128-pin TQFP package 0.4 mm pitch

### 3 Pin Configuration



### 4 Signal Description

Module	Signal	Function	Type	Active	Properties
PU=Pull Up, PD=Pull Down, ST=Schmitt Trigger Input, OT=Output Tristate, S=Switchable R <sub>S</sub> =Required for SPI boot (§7), R <sub>U</sub> =Required for USB-enabled devices (§E)					
Power	GND	Digital ground	GND	—	
	OTP_VCC	OTP power supply	PWR	—	
	OTP_VPP	OTP programming voltage	PWR	—	
	PCU_CLK	Clock input	Input	—	PD, ST
	PCU_GATE	Power control gate control	Input	—	OT
	PCU_VDD	PCU tile power	PWR	—	
	PCU_VDDIO	PCU I/O supply	PWR	—	
	PCU_WAKE	Wakeup reset	Input	—	
	PLL_AGND	Analog ground for PLL	GND	—	
	PLL_AVDD	Analog PLL power	PWR	—	
	RST_N	Global reset input	Input	Low	
VDD	Digital tile power	PWR	—		
VDDIO	Digital I/O power	PWR	—		
Clocks	CLK	PLL reference clock	Input	—	PD, ST
	MODE[3:0]	Boot mode select	Input	—	PU, ST
JTAG	DEBUG_N	Multi-chip debug	I/O	Low	PU
	TCK	Test clock	Input	—	PU, ST
	TDI	Test data input	Input	—	PU, ST
	TDO	Test data output	Output	—	PD, OT
	TMS	Test mode select	Input	—	PU, ST
	TRST_N	Test reset input	Input	Low	PU, ST
I/O	X0D00	P1A <sup>0</sup>	I/O	—	PD <sub>S</sub> , R <sub>S</sub>
	X0D01	XLA <sub>out</sub> <sup>4</sup> P1B <sup>0</sup>	I/O	—	PD <sub>S</sub> , R <sub>S</sub>
	X0D02	XLA <sub>out</sub> <sup>3</sup> P4A <sup>0</sup> P8A <sup>0</sup> P16A <sup>0</sup> P32A <sup>20</sup>	I/O	—	PD <sub>S</sub> , R <sub>U</sub>
	X0D03	XLA <sub>out</sub> <sup>2</sup> P4A <sup>1</sup> P8A <sup>1</sup> P16A <sup>1</sup> P32A <sup>21</sup>	I/O	—	PD <sub>S</sub> , R <sub>U</sub>
	X0D04	XLA <sub>out</sub> <sup>1</sup> P4B <sup>0</sup> P8A <sup>2</sup> P16A <sup>2</sup> P32A <sup>22</sup>	I/O	—	PD <sub>S</sub> , R <sub>U</sub>
	X0D05	XLA <sub>out</sub> <sup>0</sup> P4B <sup>1</sup> P8A <sup>3</sup> P16A <sup>3</sup> P32A <sup>23</sup>	I/O	—	PD <sub>S</sub> , R <sub>U</sub>
	X0D06	XLA <sub>in</sub> <sup>0</sup> P4B <sup>2</sup> P8A <sup>4</sup> P16A <sup>4</sup> P32A <sup>24</sup>	I/O	—	PD <sub>S</sub> , R <sub>U</sub>
	X0D07	XLA <sub>in</sub> <sup>1</sup> P4B <sup>3</sup> P8A <sup>5</sup> P16A <sup>5</sup> P32A <sup>25</sup>	I/O	—	PD <sub>S</sub> , R <sub>U</sub>
	X0D08	XLA <sub>in</sub> <sup>2</sup> P4A <sup>2</sup> P8A <sup>6</sup> P16A <sup>6</sup> P32A <sup>26</sup>	I/O	—	PD <sub>S</sub> , R <sub>U</sub>
	X0D09	XLA <sub>in</sub> <sup>3</sup> P4A <sup>3</sup> P8A <sup>7</sup> P16A <sup>7</sup> P32A <sup>27</sup>	I/O	—	PD <sub>S</sub> , R <sub>U</sub>
	X0D10	XLA <sub>in</sub> <sup>4</sup> P1C <sup>0</sup>	I/O	—	PD <sub>S</sub> , R <sub>S</sub>
	X0D11	P1D <sup>0</sup>	I/O	—	PD <sub>S</sub> , R <sub>S</sub>
	X0D12	P1E <sup>0</sup>	I/O	—	PD <sub>S</sub> , R <sub>U</sub>
	X0D13	XLB <sub>out</sub> <sup>4</sup> P1F <sup>0</sup>	I/O	—	PD <sub>S</sub> , R <sub>U</sub>
	X0D14	XLB <sub>out</sub> <sup>3</sup> P4C <sup>0</sup> P8B <sup>0</sup> P16A <sup>8</sup> P32A <sup>28</sup>	I/O	—	PD <sub>S</sub> , R <sub>U</sub>
	X0D15	XLB <sub>out</sub> <sup>2</sup> P4C <sup>1</sup> P8B <sup>1</sup> P16A <sup>9</sup> P32A <sup>29</sup>	I/O	—	PD <sub>S</sub> , R <sub>U</sub>
	X0D16	XLB <sub>out</sub> <sup>1</sup> P4D <sup>0</sup> P8B <sup>2</sup> P16A <sup>10</sup>	I/O	—	PD <sub>S</sub> , R <sub>U</sub>
	X0D17	XLB <sub>out</sub> <sup>0</sup> P4D <sup>1</sup> P8B <sup>3</sup> P16A <sup>11</sup>	I/O	—	PD <sub>S</sub> , R <sub>U</sub>
	X0D18	XLB <sub>in</sub> <sup>0</sup> P4D <sup>2</sup> P8B <sup>4</sup> P16A <sup>12</sup>	I/O	—	PD <sub>S</sub> , R <sub>U</sub>
	X0D19	XLB <sub>in</sub> <sup>1</sup> P4D <sup>3</sup> P8B <sup>5</sup> P16A <sup>13</sup>	I/O	—	PD <sub>S</sub> , R <sub>U</sub>
	X0D20	XLB <sub>in</sub> <sup>2</sup> P4C <sup>2</sup> P8B <sup>6</sup> P16A <sup>14</sup> P32A <sup>30</sup>	I/O	—	PD <sub>S</sub> , R <sub>U</sub>
	X0D21	XLB <sub>in</sub> <sup>3</sup> P4C <sup>3</sup> P8B <sup>7</sup> P16A <sup>15</sup> P32A <sup>31</sup>	I/O	—	PD <sub>S</sub> , R <sub>U</sub>
	X0D22	XLB <sub>in</sub> <sup>4</sup> P1G <sup>0</sup>	I/O	—	PD <sub>S</sub> , R <sub>U</sub>
X0D23	P1H <sup>0</sup>	I/O	—	PD <sub>S</sub> , R <sub>U</sub>	

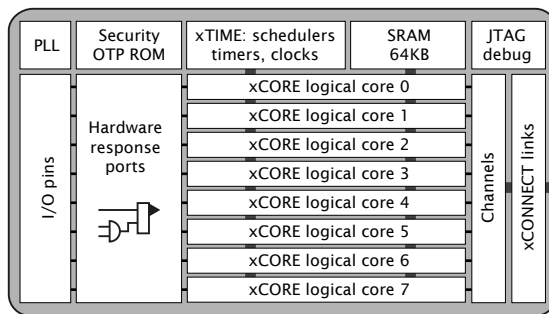
(continued)

Module	Name	Function	Type	Active	Properties	
I/O	X0D24	P1J <sup>0</sup>	I/O	—	PD <sub>S</sub>	
	X0D25	P1J <sup>0</sup>	I/O	—	PD <sub>S</sub>	
	X0D26	P4E <sup>0</sup> P8C <sup>0</sup> P16B <sup>0</sup>	I/O	—	PD <sub>S</sub> , RU	
	X0D27	P4E <sup>1</sup> P8C <sup>1</sup> P16B <sup>1</sup>	I/O	—	PD <sub>S</sub> , RU	
	X0D28	P4F <sup>0</sup> P8C <sup>2</sup> P16B <sup>2</sup>	I/O	—	PD <sub>S</sub> , RU	
	X0D29	P4F <sup>1</sup> P8C <sup>3</sup> P16B <sup>3</sup>	I/O	—	PD <sub>S</sub> , RU	
	X0D30	P4F <sup>2</sup> P8C <sup>4</sup> P16B <sup>4</sup>	I/O	—	PD <sub>S</sub> , RU	
	X0D31	P4F <sup>3</sup> P8C <sup>5</sup> P16B <sup>5</sup>	I/O	—	PD <sub>S</sub> , RU	
	X0D32	P4E <sup>2</sup> P8C <sup>6</sup> P16B <sup>6</sup>	I/O	—	PD <sub>S</sub> , RU	
	X0D33	P4E <sup>3</sup> P8C <sup>7</sup> P16B <sup>7</sup>	I/O	—	PD <sub>S</sub> , RU	
	X0D34	P1K <sup>0</sup>	I/O	—	PD <sub>S</sub>	
	X0D35	P1L <sup>0</sup>	I/O	—	PD <sub>S</sub>	
	X0D36	P1M <sup>0</sup> P8D <sup>0</sup> P16B <sup>8</sup>	I/O	—	PD <sub>S</sub>	
	X0D37	P1N <sup>0</sup> P8D <sup>1</sup> P16B <sup>9</sup>	I/O	—	PD <sub>S</sub> , RU	
	X0D38	P1O <sup>0</sup> P8D <sup>2</sup> P16B <sup>10</sup>	I/O	—	PD <sub>S</sub> , RU	
	X0D39	P1P <sup>0</sup> P8D <sup>3</sup> P16B <sup>11</sup>	I/O	—	PD <sub>S</sub> , RU	
	X0D40	P8D <sup>4</sup> P16B <sup>12</sup>	I/O	—	PD <sub>S</sub> , RU	
	X0D41	P8D <sup>5</sup> P16B <sup>13</sup>	I/O	—	PD <sub>S</sub> , RU	
	X0D42	P8D <sup>6</sup> P16B <sup>14</sup>	I/O	—	PD <sub>S</sub> , RU	
	X0D43	P8D <sup>7</sup> P16B <sup>15</sup>	I/O	—	PU <sub>S</sub> , RU	
	X0D49	XLC <sup>4</sup> <sub>out</sub>	P32A <sup>0</sup>	I/O	—	PD <sub>S</sub>
	X0D50	XLC <sup>3</sup> <sub>out</sub>	P32A <sup>1</sup>	I/O	—	PD <sub>S</sub>
	X0D51	XLC <sup>2</sup> <sub>out</sub>	P32A <sup>2</sup>	I/O	—	PD <sub>S</sub>
	X0D52	XLC <sup>1</sup> <sub>out</sub>	P32A <sup>3</sup>	I/O	—	PD <sub>S</sub>
	X0D53	XLC <sup>0</sup> <sub>out</sub>	P32A <sup>4</sup>	I/O	—	PD <sub>S</sub>
	X0D54	XLC <sup>0</sup> <sub>in</sub>	P32A <sup>5</sup>	I/O	—	PD <sub>S</sub>
	X0D55	XLC <sup>1</sup> <sub>in</sub>	P32A <sup>6</sup>	I/O	—	PD <sub>S</sub>
	X0D56	XLC <sup>2</sup> <sub>in</sub>	P32A <sup>7</sup>	I/O	—	PD <sub>S</sub>
	X0D57	XLC <sup>3</sup> <sub>in</sub>	P32A <sup>8</sup>	I/O	—	PD <sub>S</sub>
	X0D58	XLC <sup>4</sup> <sub>in</sub>	P32A <sup>9</sup>	I/O	—	PD <sub>S</sub>
	X0D61	XLD <sup>4</sup> <sub>out</sub>	P32A <sup>10</sup>	I/O	—	PD <sub>S</sub>
	X0D62	XLD <sup>3</sup> <sub>out</sub>	P32A <sup>11</sup>	I/O	—	PD <sub>S</sub>
	X0D63	XLD <sup>2</sup> <sub>out</sub>	P32A <sup>12</sup>	I/O	—	PD <sub>S</sub>
	X0D64	XLD <sup>1</sup> <sub>out</sub>	P32A <sup>13</sup>	I/O	—	PD <sub>S</sub>
	X0D65	XLD <sup>0</sup> <sub>out</sub>	P32A <sup>14</sup>	I/O	—	PD <sub>S</sub>
	X0D66	XLD <sup>0</sup> <sub>in</sub>	P32A <sup>15</sup>	I/O	—	PD <sub>S</sub>
	X0D67	XLD <sup>1</sup> <sub>in</sub>	P32A <sup>16</sup>	I/O	—	PD <sub>S</sub>
	X0D68	XLD <sup>2</sup> <sub>in</sub>	P32A <sup>17</sup>	I/O	—	PD <sub>S</sub>
	X0D69	XLD <sup>3</sup> <sub>in</sub>	P32A <sup>18</sup>	I/O	—	PD <sub>S</sub>
	X0D70	XLD <sup>4</sup> <sub>in</sub>	P32A <sup>19</sup>	I/O	—	PD <sub>S</sub>



## 5 Product Overview

The XMOS XS1-L8A-64-TQ128 is a powerful device that provides a simple design process and highly-flexible solution to many applications. The device consists of a single xCORE Tile, which comprises a flexible multicore microcontroller with tightly integrated I/O and on-chip memory. The processor runs multiple tasks simultaneously using logical cores, each of which is guaranteed a slice of processing power and can execute computational code, control software and I/O interfaces. Logical cores use channels to exchange data within a tile or across tiles. Multiple devices can be deployed and connected using an integrated switching network, enabling more resources to be added to a design. The I/O pins are driven using intelligent ports that can serialize data, interpret strobe signals and wait for scheduled times or events, making the device ideal for real-time control applications.



**Figure 2:**  
Block  
Diagram

The device can be configured using a set of software components that are rapidly customized and composed. XMOS provides source code libraries for many standard components. The device can be programmed using high-level languages such as C/C++ and XMOS-originated extensions to C, called XC, that simplify the control over concurrency, I/O and time.

The XMOS toolchain includes compilers, a simulator, debugger and static timing analyzer. The combination of real-time software, a compiler and timing analyzer enables the programmer to close timings on components of the design without a detailed understanding of the hardware characteristics.

### 5.1 Logical cores, Synchronizers and Locks

The tile has up to 8 active logical cores, which issue instructions down a shared four-stage pipeline. Instructions from the active cores are issued round-robin. If up to 4 logical cores are active, each core is allocated a quarter of the processing cycles. If more than four logical cores are active, each core is allocated at least  $1/n$  cycles (for  $n$  cores). Figure 3 shows the guaranteed core performance depending on the number of cores used.

There is no way that the performance of a logical core can be reduced below these predicted levels. Because cores may be delayed on I/O, however, their unused processing cycles can be taken by other cores. This means that for more than

**Figure 3:**  
Logical core performance

Speed Grade, MIPS, and frequency	Minimum MIPS per core (for <i>n</i> cores)							
	1	2	3	4	5	6	7	8
4: 400 MIPS, 400 MHz	100	100	100	100	80	67	57	50
5: 500 MIPS, 500 MHz	125	125	125	125	100	83	71	63

four logical cores, the performance of each core is often higher than the predicted minimum.

Synchronizers are provided for fast synchronization in a group of logical cores. In a single instruction a logical core can block until all other logical cores in a group have reached the synchronizer. Locks are provided for fast mutual exclusion. A logical core can acquire or release a lock in a single instruction.

### 5.2 Channel Ends, Links and Switch

Logical cores communicate using point-to-point connections formed between two channel ends. Between tiles, channel communications are implemented over xConnect Links and routed through switches. The links operate in either 2 wires per direction or 5 wires per direction mode, depending on the amount of bandwidth required. Circuit switched, streaming and packet switched data can both be supported efficiently. Streams provide the fastest possible data rates between xCORE Tiles (up to 250 MBit/s), but each stream requires a single link to be reserved between switches on two tiles. All packet communications can be multiplexed onto a single link. A total of four 5bit links are available between both cores.

Information on the supported routing topologies that can be used to connect multiple devices together can be found in the XS1-L Link Performance and Design Guide, [X2999](#).

### 5.3 Ports and Clock Blocks

Ports provide an interface between the logical cores and I/O pins. The XS1-L8A-64-TQ128 includes a combination of 1bit, 4bit and 8bit ports. In addition the wider ports are partially or fully bonded out making the connected pins available for I/O or xCONNECT links. All pins of a port provide either output or input. Signals in different directions cannot be mapped onto the same port.

The operation of each port can be synchronized to a clock block. A clock block can be connected to an external clock input, or it can be run from the divided reference clock. A clock block can also output its signal to a pin. On reset, each port is connected to clock block 0, which runs from the xCORE Tile reference clock.

The ports and links are multiplexed, allowing the pins to be configured for use by ports of different widths or links. If an xConnect Link is enabled, the pins of the underlying ports are disabled. If a port is enabled, it overrules ports with higher widths that share the same pins. The pins on the wider port that are not shared

remain available for use when the narrower port is enabled. Ports always operate at their specified width, even if they share pins with another port.

### 5.4 Timers

Timers are 32-bit counters that are relative to the xCORE Tile reference clock. A timer is defined to tick every 10 ns. This value is derived from the reference clock, which is configured to tick at 100 MHz by default.

## 6 PLL

The PLL creates a high-speed clock that is used for the switch, tile, and reference clock. The PLL multiplication value is selected through the two MODE pins, and can be changed by software to speed up the tile or use less power. The MODE pins are set as shown in Figure 4:

**Figure 4:**  
PLL multiplier values and MODE pins

Oscillator Frequency	MODE		Tile Frequency	PLL Ratio	PLL settings		
	1	0			OD	F	R
5-13 MHz	0	0	130-399.75 MHz	30.75	1	122	0
13-20 MHz	1	1	260-400.00 MHz	20	2	119	0
20-48 MHz	1	0	167-400.00 MHz	8.33	2	49	0
48-100 MHz	0	1	196-400.00 MHz	4	2	23	0

Figure 4 also lists the values of *OD*, *F* and *R*, which are the registers that define the ratio of the tile frequency to the oscillator frequency:

$$F_{core} = F_{osc} \times \frac{F + 1}{2} \times \frac{1}{R + 1} \times \frac{1}{OD + 1}$$

*OD*, *F* and *R* must be chosen so that  $0 \leq R \leq 63$ ,  $0 \leq F \leq 4095$ ,  $0 \leq OD \leq 7$ , and  $260MHz \leq F_{osc} \times \frac{F+1}{2} \times \frac{1}{R+1} \leq 1.3GHz$ . The *OD*, *F*, and *R* values can be modified by writing to the digital node PLL configuration register.

The MODE pins must be held at a static value until the third rising edge of the system clock following the deassertion of the system reset.

For 500 MHz parts, once booted, the PLL must be reprogrammed to provide this tile frequency. The XMOS tools perform this operation by default.

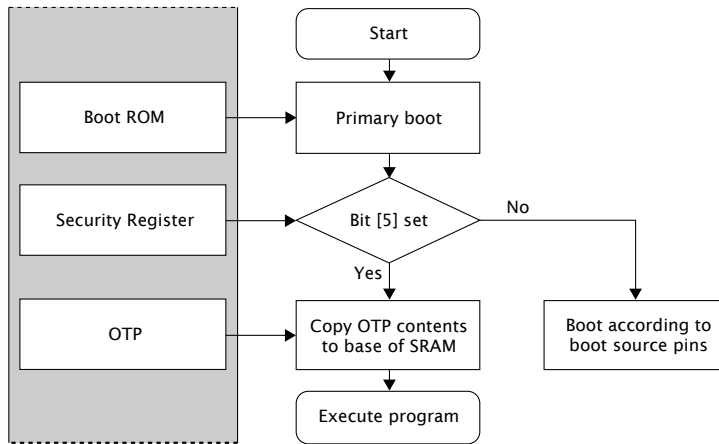
Further details on configuring the clock can be found in the XS1-L Clock Frequency Control document, [X1433](#).

## 7 Boot Procedure

The device is kept in reset by driving RST\_N low. When in reset, all GPIO pins are high impedance. When the device is taken out of reset by releasing RST\_N the processor starts its internal reset process. After 15-150 μs (depending on the input

clock), all GPIO pins have their internal pull-resistor enabled, and the processor boots at a clock speed that depends on MODE0 and MODE1.

The xCORE Tile boot procedure is illustrated in Figure 5. In normal usage, MODE[3:2] controls the boot source according to the table in Figure 6. If bit 5 of the security register (see §8.1) is set, the device boots from OTP.



**Figure 5:**  
Boot procedure

MODE[3]	MODE[2]	Boot Source
0	0	None: Device waits to be booted via JTAG
0	1	Reserved
1	0	xConnect Link B
1	1	SPI

**Figure 6:**  
Boot source pins

The boot image has the following format:

- ▶ A 32-bit program size  $s$  in words.
- ▶ Program consisting of  $s \times 4$  bytes.
- ▶ A 32-bit CRC, or the value 0x0D15AB1E to indicate that no CRC check should be performed.

The program size and CRC are stored least significant byte first. The program is loaded into the lowest memory address of RAM, and the program is started from that address. The CRC is calculated over the byte stream represented by the program size and the program itself. The polynomial used is 0xEDB88320 (IEEE 802.3); the CRC register is initialized with 0xFFFFFFFF and the residue is inverted to produce the CRC.

### 7.1 Boot from SPI

If set to boot from SPI, the processor enables the four pins specified in Figure 7, and drives the SPI clock at 2.5 MHz (assuming a 400 MHz core clock). A READ

command is issued with a 24-bit address 0x000000. The clock polarity and phase are 0 / 0.

**Figure 7:**  
SPI pins

Pin	Signal	Description
X0D00	MISO	Master In Slave Out (Data)
X0D01	SS	Slave Select
X0D10	SCLK	Clock
X0D11	MOSI	Master Out Slave In (Data)

The xCORE Tile expects each byte to be transferred with the *least-significant bit first*. Programmers who write bytes into an SPI interface using the most significant bit first may have to reverse the bits in each byte of the image stored in the SPI device.

If a large boot image is to be read in, it is faster to first load a small boot-loader that reads the large image using a faster SPI clock, for example 50 MHz or as fast as the flash device supports.

The pins used for SPI boot are hardcoded in the boot ROM and cannot be changed. If required, an SPI boot program can be burned into OTP that uses different pins.

## 7.2 Boot from xConnect Link

If set to boot from an xConnect Link, the processor enables Link B around 200 ns after the boot process starts. Enabling the Link switches off the pull-down on resistors X0D16..X0D19, drives X0D16 and X0D17 low (the initial state for the Link), and monitors pins X0D18 and X0D19 for boot-traffic. X0D18 and X0D19 must be low at this stage. If the internal pull-down is too weak to drain any residual charge, external pull-downs of 10K may be required on those pins.

The boot-rom on the core will then:

1. Allocate channel-end 0.
2. Input a word on channel-end 0. It will use this word as a channel to acknowledge the boot. Provide the null-channel-end 0x0000FF02 if no acknowledgment is required.
3. Input the boot image specified above, including the CRC.
4. Input an END control token.
5. Output an END control token to the channel-end received in step 2.
6. Free channel-end 0.
7. Jump to the loaded code.

### 7.3 Boot from OTP

If an xCORE tile is set to use secure boot (see Figure 5), the boot image is read from address 0 of the OTP memory in the tile's security module.

This feature can be used to implement a secure bootloader which loads an encrypted image from external flash, decrypts and CRC checks it with the processor, and discontinues the boot process if the decryption or CRC check fails. XMOS provides a default secure bootloader that can be written to the OTP along with secret decryption keys.

Each tile has its own individual OTP memory, and hence some tiles can be booted from OTP while others are booted from SPI or the channel interface. This enables systems to be partially programmed, dedicating one or more tiles to perform a particular function, leaving the other tiles user-programmable.

### 7.4 Security register

The security register enables security features on the xCORE tile. The features shown in Figure 8 provide a strong level of protection and are sufficient for providing strong IP security.

Feature	Bit	Description
Disable JTAG	0	The JTAG interface is disabled, making it impossible for the tile state or memory content to be accessed via the JTAG interface.
Disable Link access	1	Other tiles are forbidden access to the processor state via the system switch. Disabling both JTAG and Link access transforms an xCORE Tile into a "secure island" with other tiles free for non-secure user application code.
Secure Boot	5	The xCORE Tile is forced to boot from address 0 of the OTP, allowing the xCORE Tile boot ROM to be bypassed ( <i>see §7</i> ).
Redundant rows	7	Enables redundant rows in OTP.
Sector Lock 0	8	Disable programming of OTP sector 0.
Sector Lock 1	9	Disable programming of OTP sector 1.
Sector Lock 2	10	Disable programming of OTP sector 2.
Sector Lock 3	11	Disable programming of OTP sector 3.
OTP Master Lock	12	Disable OTP programming completely: disables updates to all sectors and security register.
Disable JTAG-OTP	13	Disable all (read & write) access from the JTAG interface to this OTP.
Disable Global Debug	14	Disables access to the DEBUG_N pin.
	21..15	General purpose software accessible security register available to end-users.
	31..22	General purpose user programmable JTAG UserID code extension.

**Figure 8:**  
Security register features

## 8 Memory

### 8.1 OTP

The xCORE Tile integrates 8 KB one-time programmable (OTP) memory along with a security register that configures system wide security features. The OTP holds data in four sectors each containing 512 rows of 32 bits which can be used to implement secure bootloaders and store encryption keys. Data for the security register is loaded from the OTP on power up. All additional data in OTP is copied from the OTP to SRAM and executed first on the processor.

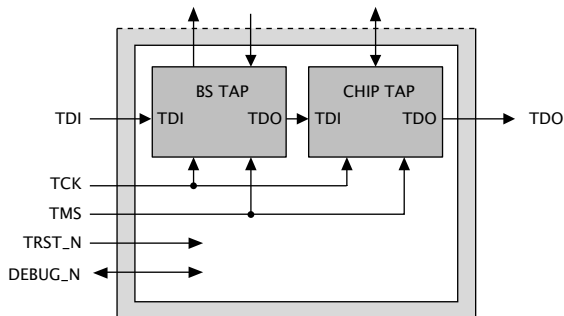
The OTP memory is programmed using three special I/O ports: the OTP address port is a 16-bit port with resource ID 0x100200, the OTP data is written via a 32-bit port with resource ID 0x200100, and the OTP control is on a 16-bit port with ID 0x100300. Programming is performed through `libotp` and `xburn`.

### 8.2 SRAM

The xCORE Tile integrates a single 64 KB SRAM bank for both instructions and data. All internal memory is 32 bits wide, and instructions are either 16-bit or 32-bit. Byte (8-bit), half-word (16-bit) or word (32-bit) accesses are supported and are executed within one tile clock cycle. There is no dedicated external memory interface, although data memory can be expanded through appropriate use of the ports.

## 9 JTAG

The JTAG module can be used for loading programs, boundary scan testing, in-circuit source-level debugging and programming the OTP memory.



**Figure 9:**  
JTAG chain structure

The JTAG chain structure is illustrated in Figure 9. Directly after reset, two TAP controllers are present in the JTAG chain: the boundary scan TAP and the chip TAP. The boundary scan TAP is a standard 1149.1 compliant TAP that can be used for boundary scan of the I/O pins. The chip TAP provides access into the xCORE Tile, switch and OTP for loading code and debugging.





- ▶ OTP\_VCC pins for the OTP
- ▶ OTP\_VPP pins for faster programming the OTP (optional)

Several pins of each type are provided to minimize the effect of inductance within the package, all of which must be connected. The power supplies must be brought up monotonically and input voltages must not exceed specification at any time.

The VDD supply must ramp from 0 V to its final value within 10 ms to ensure correct startup.

The VDDIO supply must ramp to its final value before VDD reaches 0.4 V.

The PLL\_AVDD supply should be separated from the other noisier supplies on the board. The PLL requires a very clean power supply, and a low pass filter (for example, a 4.7  $\Omega$  resistor and 100 nF multi-layer ceramic capacitor) is recommended on this pin.

The PCU\_VDD supply must be connected to the VDD supply.

The PCU\_VDDIO supply must be connected to the VDDIO supply.

The OTP\_VCC supply should be connected to the VDDIO supply.

The OTP\_VPP supply can be optionally provided for faster OTP programming times, otherwise an internal charge pump is used.

The following ground pins are provided:

- ▶ PLL\_AGND for PLL\_AVDD
- ▶ GND for all other supplies

All ground pins must be connected directly to the board ground.

The VDD and VDDIO supplies should be decoupled close to the chip by several 100 nF low inductance multi-layer ceramic capacitors between the supplies and GND (for example, 4x100nF 0402 low inductance MLCCs per supply rail). The ground side of the decoupling capacitors should have as short a path back to the GND pins as possible. A bulk decoupling capacitor of at least 10  $\mu$ F should be placed on each of these supplies.

RST\_N is an active-low asynchronous-assertion global reset signal. Following a reset, the PLL re-establishes lock after which the device boots up according to the boot mode (*see* §7). RST\_N and must be asserted low during and after power up for 100 ns.

## 10.1 Land patterns and solder stencils

The land pattern recommendations in this document are based on a RoHS compliant process and derived, where possible, from the nominal *Generic Requirements for Surface Mount Design and Land Pattern Standards IPC-7351B* specifications. This

standard aims to achieve desired targets of heel, toe and side fillets for solder-joints.

Solder paste and ground via recommendations are based on our engineering and development kit board production. They have been found to work and optimized as appropriate to achieve a high yield. The size, type and number of vias used in the center pad affects how much solder wicks down the vias during reflow. This in turn, along with solder paster coverage, affects the final assembled package height. These factors should be taken into account during design and manufacturing of the PCB.

The following land patterns and solder paste contains recommendations. Final land pattern and solder paste decisions are the responsibility of the customer. These should be tuned during manufacture to suit the manufacturing process.

The package is a 128 pin Thin Quad Flat Pack package with exposed heat slug on a 0.4mm pitch.

An example land pattern is shown in Figure 12.

Pad widths and spacings are such that solder mask can still be applied between the pads using standard design rules. This is recommended to reduce solder shorts.

The center pad solder paste level needs to be controlled so the device sits the correct height from the circuit board. For the 128 pin TQFP package, a 3x3 array of squares for solder paste is recommended as shown in Figure 13. This gives a paste level of 56%.

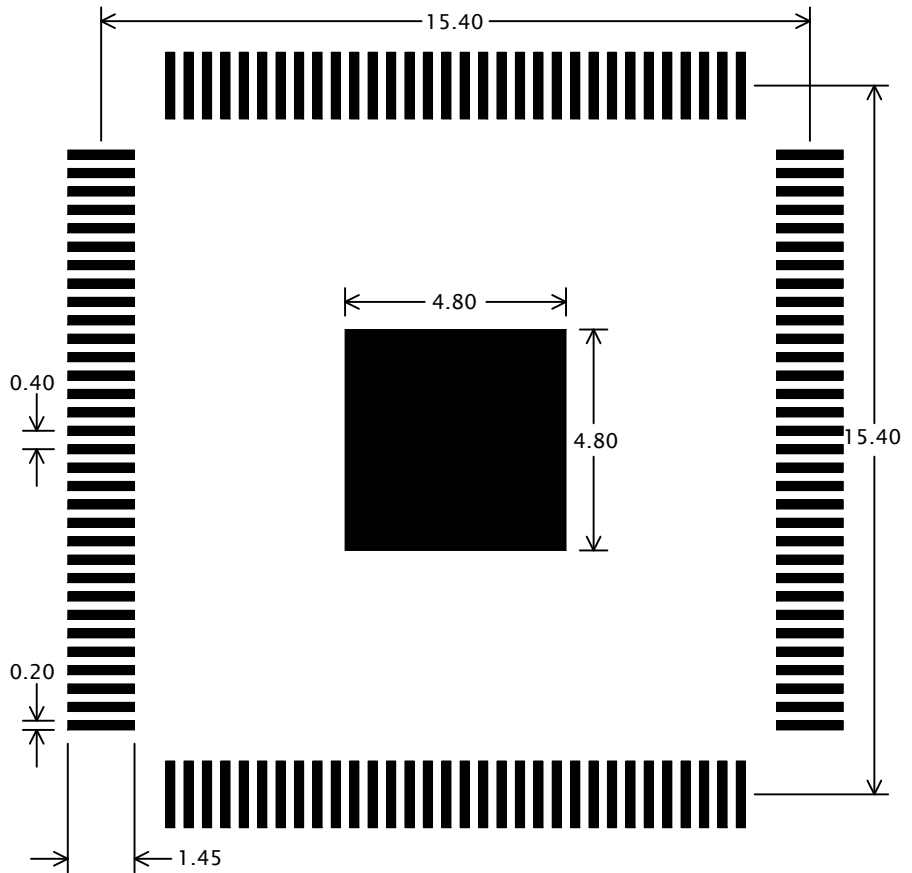
## 10.2 Ground and Thermal Vias

Vias under the heat slug into the ground plane of the PCB are recommended for a low inductance ground connection and good thermal performance. A 3 x 3 grid of vias, with a 0.6mm diameter annular ring and a 0.3mm drill, equally spaced across the heat slug, would be suitable.

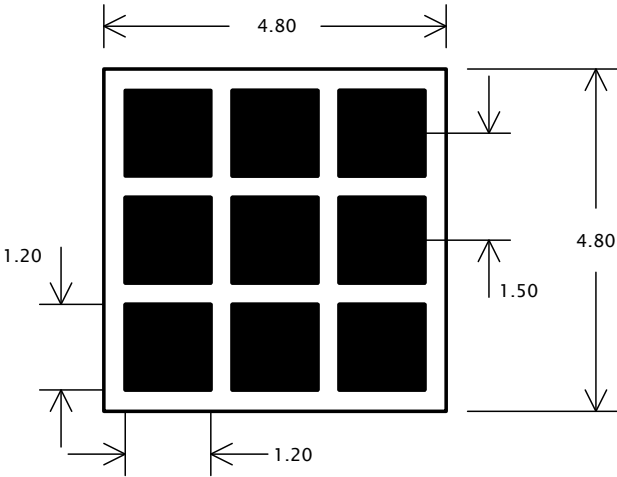
## 10.3 Moisture Sensitivity

XMOS devices are, like all semiconductor devices, susceptible to moisture absorption. When removed from the sealed packaging, the devices slowly absorb moisture from the surrounding environment. If the level of moisture present in the device is too high during reflow, damage can occur due to the increased internal vapour pressure of moisture. Example damage can include bond wire damage, die lifting, internal or external package cracks and/or delamination.

All XMOS devices are Moisture Sensitivity Level (MSL) 3 - devices have a shelf life of 168 hours between removal from the packaging and reflow, provided they are stored below 30C and 60% RH. If devices have exceeded these values or an included moisture indicator card shows excessive levels of moisture, then the parts should be baked as appropriate before use. This is based on information from *Joint IPC/JEDEC Standard For Moisture/Reflow Sensitivity Classification For Nonhermetic Solid State Surface-Mount Devices J-STD-020* Revision D.



**Figure 12:**  
Example land pattern



**Figure 13:**  
Solder stencil  
for centre  
pad

## 11 DC and Switching Characteristics

### 11.1 Operating Conditions

Symbol	Parameter	MIN	TYP	MAX	UNITS	Notes
VDD	Tile DC supply voltage	0.95	1.00	1.05	V	
VDDIO	I/O supply voltage	3.00	3.30	3.60	V	
PLL_AVDD	PLL analog supply	0.95	1.00	1.05	V	
PCU_VDD	PCU tile DC supply voltage	0.95	1.00	1.05	V	
PCU_VDDIO	PCU I/O DC supply voltage	3.00	3.30	3.60	V	
OTP_VCC	OTP supply voltage	3.00	3.30	3.60	V	
OTP_VPP	OTP external programming voltage (optional program only)	6.18	6.50	6.83	V	
Cl	xCORE Tile I/O load capacitance			25	pF	
Ta	Ambient operating temperature (Commercial)	0		70	°C	
	Ambient operating temperature (Industrial)	-40		85	°C	
Tj	Junction temperature			125	°C	
Tstg	Storage temperature	-65		150	°C	

**Figure 14:**  
Operating conditions

### 11.2 DC Characteristics

Symbol	Parameter	MIN	TYP	MAX	UNITS	Notes
V(IH)	Input high voltage	2.00		3.60	V	A
V(IL)	Input low voltage	-0.30		0.70	V	A
V(OH)	Output high voltage	2.70			V	B, C
V(OL)	Output low voltage			0.60	V	B, C
R(PU)	Pull-up resistance		35K		Ω	D
R(PD)	Pull-down resistance		35K		Ω	D

**Figure 15:**  
DC characteristics

A All pins except power supply pins.

B Ports 1A, 1D, 1E, 1H, 1I, 1J, 1K and 1L are nominal 8 mA drivers, the remainder of the general-purpose I/Os are 4 mA.

C Measured with 4 mA drivers sourcing 4 mA, 8 mA drivers sourcing 8 mA.

D Used to guarantee logic state for an I/O when high impedance. The internal pull-ups/pull-downs should not be used to pull external circuitry.

### 11.3 ESD Stress Voltage

Symbol	Parameter	MIN	TYP	MAX	UNITS	Notes
HBM	Human body model	-2.00		2.00	KV	
MM	Machine model	-200		200	V	

**Figure 16:**  
ESD stress voltage

### 11.4 Reset Timing

**Figure 17:**  
Reset timing

Symbol	Parameters	MIN	TYP	MAX	UNITS	Notes
T(RST)	Reset pulse width	5			us	
T(INIT)	Initialization time			150	µs	A

A Shows the time taken to start booting after RST\_N has gone high.

### 11.5 Power Consumption

**Figure 18:**  
xCORE Tile currents

Symbol	Parameter	MIN	TYP	MAX	UNITS	Notes
I(DDCQ)	Quiescent VDD current		14		mA	A, B, C
PD	Tile power dissipation		450		µW/MIPS	A, D, E, F
IDD	Active VDD current (Speed Grade 4)		160	300	mA	A, G
	Active VDD current (Speed Grade 5)		200	375	mA	A, H
I(ADDPLL)	PLL_AVDD current			7	mA	I

A Use for budgetary purposes only.

B Assumes typical tile and I/O voltages with no switching activity.

C Includes PLL current.

D Assumes typical tile and I/O voltages with nominal switching activity.

E Assumes 1 MHz = 1 MIPS.

F PD(TYP) value is the usage power consumption under typical operating conditions.

G Measurement conditions: VDD = 1.0 V, VDDIO = 3.3 V, 25 °C, 400 MHz, average device resource usage.

H Measurement conditions: VDD = 1.0 V, VDDIO = 3.3 V, 25 °C, 500 MHz, average device resource usage.

I PLL\_AVDD = 1.0 V



The tile power consumption of the device is highly application dependent and should be used for budgetary purposes only.

More detailed power analysis can be found in the XS1-L Power Consumption document, [X2999](#).

### 11.6 Clock

Symbol	Parameter	MIN	TYP	MAX	UNITS	Notes
f	Frequency	4.22	20	100	MHz	
SR	Slew rate	0.10			V/ns	
TJ(LT)	Long term jitter (pk-pk)			2	%	A
f(MAX)	Processor clock frequency (Speed Grade 4)			400	MHz	B
	Processor clock frequency (Speed Grade 5)			500	MHz	B

**Figure 19:**  
Clock

A Percentage of CLK period.  
 B Assumes typical tile and I/O voltages with nominal activity.

Further details can be found in the XS1-L Clock Frequency Control document, [X1433](#).

The OTP may be programmed using its internal charge pump or by supplying a 6.5V VPP programming voltage on the OTP\_VPP pin. Unless a programming cycle is underway the OTP\_VPP pins should be left undriven.

### 11.7 xCORE Tile I/O AC Characteristics

Symbol	Parameter	MIN	TYP	MAX	UNITS	Notes
T(XOVALID)	Input data valid window	8			ns	
T(XOINVALID)	Output data invalid window	9			ns	
T(XIFMAX)	Rate at which data can be sampled with respect to an external clock			60	MHz	

**Figure 20:**  
I/O AC characteristics

The input valid window parameter relates to the capability of the device to capture data input to the chip with respect to an external clock source. It is calculated as the sum of the input setup time and input hold time with respect to the external clock as measured at the pins. The output invalid window specifies the time for which an output is invalid with respect to the external clock. Note that these parameters are specified as a window rather than absolute numbers since the device provides functionality to delay the incoming clock with respect to the incoming data.

Information on interfacing to high-speed synchronous interfaces can be found in the XS1 Port I/O Timing document, [X5821](#).

### 11.8 xConnect Link Performance

Symbol	Parameter	MIN	TYP	MAX	UNITS	Notes
B(2blinkP)	2wire link bandwidth (packetized)			87	MBit/s	A, B
B(5blinkP)	5wire link bandwidth (packetized)			217	MBit/s	A, B
B(2blinkS)	2wire link bandwidth (streaming)			100	MBit/s	B
B(5blinkS)	5wire link bandwidth (streaming)			250	MBit/s	B

**Figure 21:**  
Link performance

A Assumes 32-byte packet in 3-byte header mode. Actual performance depends on size of the header and payload.

B 7.5 ns symbol time.

The asynchronous nature of links means that the relative phasing of CLK clocks is not important in a multi-clock system, providing each meets the required stability criteria.

### 11.9 JTAG Timing

Symbol	Parameter	MIN	TYP	MAX	UNITS	Notes
f(TCK_D)	TCK frequency (debug)			18	MHz	
f(TCK_B)	TCK frequency (boundary scan)			10	MHz	
T(SETUP)	TDO to TCK setup time	5			ns	A
T(HOLD)	TDO to TCK hold time	5			ns	A
T(DELAY)	TCK to output delay			15	ns	B

**Figure 22:**  
JTAG timing

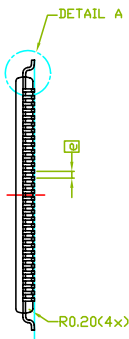
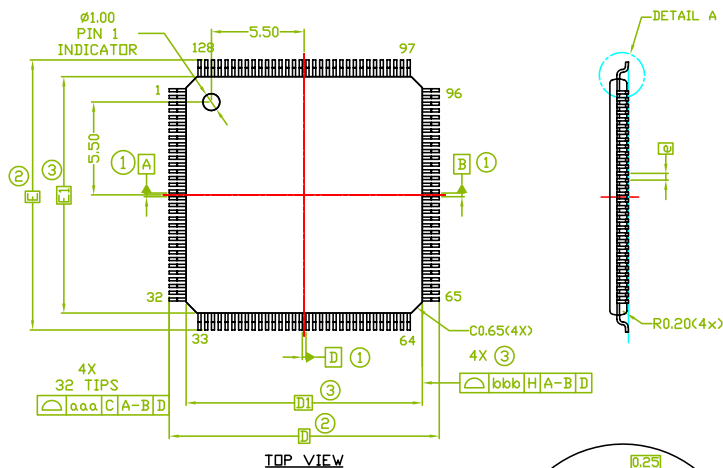
A Timing applies to TMS and TDI inputs.

B Timing applies to TDO output from negative edge of TCK.

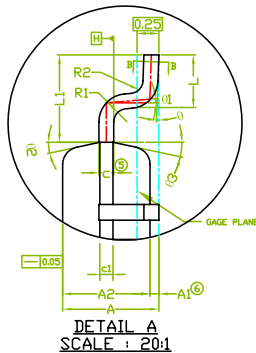
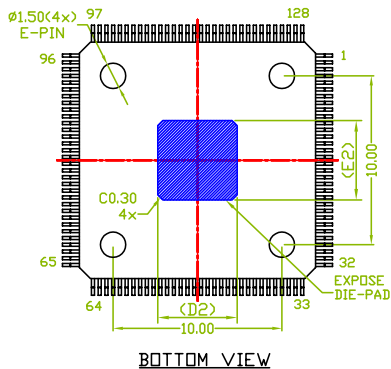
All JTAG operations are synchronous to TCK apart from the global asynchronous reset TRST\_N.



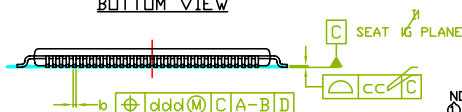
### 12 Package Information



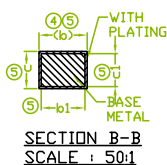
SYMBOL	Min.	Nom.	Max.
A	-	-	1.20
A1	0.05	-	0.15
A2	0.95	1.00	1.05
b	0.13	0.18	0.23
b1	0.13	0.18	0.19
D	16.00 BSC		
D1	14.00 BSC		
e	0.40 BSC		
E	16.00 BSC		
E1	14.00 BSC		
θ	0°	3.5°	7°
θ1	0°	-	-
θ2	11°	12°	13°
θ3	11°	12°	13°
c	0.09	-	0.20
e1	0.09	-	0.18
L	0.45	0.60	0.75
L1	1.00 REF		
R1	0.08	-	-
R2	0.08	-	0.20



REF	TOLERANCES OF FORM AND POSITION
aaa	0.20
bbb	0.20
ccc	0.08
ddd	0.07



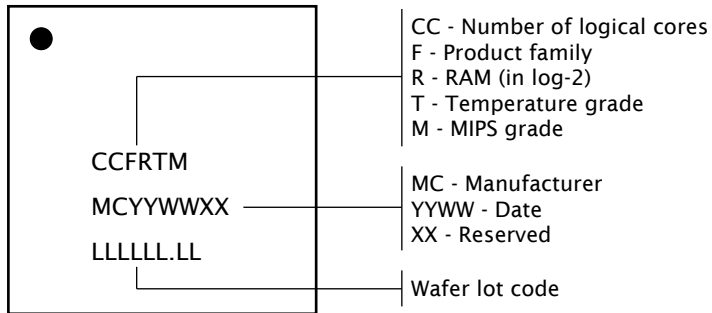
LF Ref#	Symbol	Min	Nom	Max
L-17-09011	D2	4.60	4.70	4.80
	E2	4.60	4.70	4.80



NOTE :

- DATUM A-B AND D TO DETERMINE AT DATUM PLANE H.
- TO BE DETERMINED AT SEATING DATUM PLAN C.
- DIMENSION D1 AND E1 DO NOT INCLUDE MOLD PROTRUSIONS. ALLOWABLE PROTRUSION IS 0.25 mm PER SIDE. S1 AND E1 ARE MAXIMUM PLASTIC BODY SIZE DIMENSION INCLUDING MOLD MISMATCH.
- DIMENSION b DO NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED THE MAXIMUM b DIMENSION BY MORE THAN 0.08 mm. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT. MINIMUM SPACE BETWEEN PROTRUSION AND ADJACENT LEAD IS 0.07 mm FOR 0.4mm AND 0.5 mm PITCH PACKAGE.
- THESE DIMENSIONS APPLY TO THE FLAT SECTION OF THE BETWEEN 0.10 mm AND 0.25 mm FROM THE LEAD TIP.
- A1 IS THE DEFINED AS THE DISTANCE FROM THE SEATING PLAN TO THE LOWEST POINT ON THE PACKAGE BODY.
- PACKAGE LEAD COUNT IS NON-JEDEC STANDARD.

### 12.1 Part Marking



**Figure 23:**  
Part marking  
scheme

## 13 Ordering Information

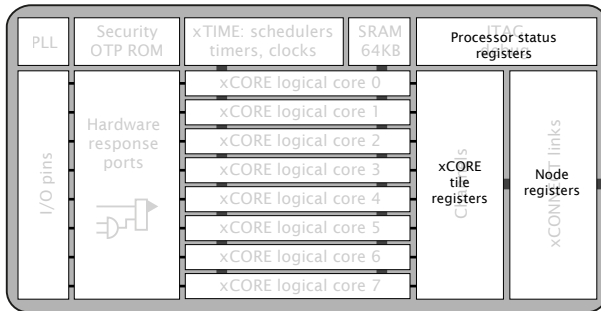
Product Code	Marking	Qualification	Speed Grade
XS1-L8A-64-TQ128-C4	8L6C4	Commercial	400 MIPS
XS1-L8A-64-TQ128-C5	8L6C5	Commercial	500 MIPS
XS1-L8A-64-TQ128-I4	8L6I4	Industrial	400 MIPS
XS1-L8A-64-TQ128-I5	8L6I5	Industrial	500 MIPS

**Figure 24:**  
Orderable  
part numbers

## Appendices

### A Configuration of the XS1

The device is configured through three banks of registers, as shown in Figure 25.



**Figure 25:**  
Registers

The following communication sequences specify how to access those registers. Any messages transmitted contain the most significant 24 bits of the channel-end to which a response is to be sent. This comprises the node-identifier and the channel number within the node. If no response is required on a write operation, supply 24-bits with the last 8-bits set, which suppresses the reply message. Any multi-byte data is sent most significant byte first.

#### A.1 Accessing a processor status register

The processor status registers are accessed directly from the processor instruction set. The instructions GETPS and SETPS read and write a word. The register number should be translated into a processor-status resource identifier by shifting the register number left 8 places, and ORing it with 0x0C. Alternatively, the functions `getps(reg)` and `setps(reg,value)` can be used from XC.

#### A.2 Accessing an xCORE Tile configuration register

xCORE Tile configuration registers can be accessed through the interconnect using the functions `write_tile_config_reg(tileref, ...)` and `read_tile_config_reg(tile ← ref, ...)`, where `tileref` is the name of the xCORE Tile, e.g. `tile[1]`. These functions implement the protocols described below.

Instead of using the functions above, a channel-end can be allocated to communicate with the xCORE tile configuration registers. The destination of the channel-end should be set to `0xnnnnC20C` where `nnnnn` is the tile-identifier.

A write message comprises the following:

control-token 192	24-bit response channel-end identifier	16-bit register number	32-bit data	control-token 1
----------------------	-------------------------------------------	---------------------------	----------------	--------------------

The response to a write message comprises either control tokens 3 and 1 (for success), or control tokens 4 and 1 (for failure).

A read message comprises the following:

control-token 193	24-bit response channel-end identifier	16-bit register number	control-token 1
----------------------	-------------------------------------------	---------------------------	--------------------

The response to the read message comprises either control token 3, 32-bit of data, and control-token 1 (for success), or control tokens 4 and 1 (for failure).

### A.3 Accessing node configuration

Node configuration registers can be accessed through the interconnect using the functions `write_node_config_reg(device, ...)` and `read_node_config_reg(device, ↪ ...)`, where `device` is the name of the node. These functions implement the protocols described below.

Instead of using the functions above, a channel-end can be allocated to communicate with the node configuration registers. The destination of the channel-end should be set to `0xnnnnC30C` where `nnnn` is the node-identifier.

A write message comprises the following:

control-token 192	24-bit response channel-end identifier	16-bit register number	32-bit data	control-token 1
----------------------	-------------------------------------------	---------------------------	----------------	--------------------

The response to a write message comprises either control tokens 3 and 1 (for success), or control tokens 4 and 1 (for failure).

A read message comprises the following:

control-token 193	24-bit response channel-end identifier	16-bit register number	control-token 1
----------------------	-------------------------------------------	---------------------------	--------------------

The response to a read message comprises either control token 3, 32-bit of data, and control-token 1 (for success), or control tokens 4 and 1 (for failure).

## B Processor Status Configuration

The processor status control registers can be accessed directly by the processor using processor status reads and writes (use `getps(reg)` and `setps(reg,value)` for reads and writes).

Number	Perm	Description
0x00	RW	RAM base address
0x01	RW	Vector base address
0x02	RW	xCORE Tile control
0x03	RO	xCORE Tile boot status
0x05	RO	Security configuration
0x06	RW	Ring Oscillator Control
0x07	RO	Ring Oscillator Value
0x08	RO	Ring Oscillator Value
0x09	RO	Ring Oscillator Value
0x0A	RO	Ring Oscillator Value
0x10	DRW	Debug SSR
0x11	DRW	Debug SPC
0x12	DRW	Debug SSP
0x13	DRW	DGETREG operand 1
0x14	DRW	DGETREG operand 2
0x15	DRW	Debug interrupt type
0x16	DRW	Debug interrupt data
0x18	DRW	Debug core control
0x20 .. 0x27	DRW	Debug scratch
0x30 .. 0x33	DRW	Instruction breakpoint address
0x40 .. 0x43	DRW	Instruction breakpoint control
0x50 .. 0x53	DRW	Data watchpoint address 1
0x60 .. 0x63	DRW	Data watchpoint address 2
0x70 .. 0x73	DRW	Data breakpoint control register
0x80 .. 0x83	DRW	Resources breakpoint mask
0x90 .. 0x93	DRW	Resources breakpoint value
0x9C .. 0x9F	DRW	Resources breakpoint control register

**Figure 26:**  
Summary

**B.1 RAM base address: 0x00**

This register contains the base address of the RAM. It is initialized to 0x00010000.

<b>0x00:</b> RAM base address	Bits	Perm	Init	Description
	31:2	RW		Most significant 16 bits of all addresses.
	1:0	RO	-	Reserved

**B.2 Vector base address: 0x01**

Base address of event vectors in each resource. On an interrupt or event, the 16 most significant bits of the destination address are provided by this register; the least significant 16 bits come from the event vector.

<b>0x01:</b> Vector base address	Bits	Perm	Init	Description
	31:16	RW		The most significant bits for all event and interrupt vectors.
	15:0	RO	-	Reserved

**B.3 xCORE Tile control: 0x02**

Register to control features in the xCORE tile

<b>0x02:</b> xCORE Tile control	Bits	Perm	Init	Description
	31:6	RO	-	Reserved
	5	RW	0	Set to 1 to select the dynamic mode for the clock divider when the clock divider is enabled. In dynamic mode the clock divider is only activated when all active logical cores are paused. In static mode the clock divider is always enabled.
	4	RW	0	Set to 1 to enable the clock divider. This slows down the xCORE tile clock in order to use less power.
	3:0	RO	-	Reserved

**B.4 xCORE Tile boot status: 0x03**

This read-only register describes the boot status of the xCORE tile.

**0x03:**  
xCORE Tile  
boot status

Bits	Perm	Init	Description
31:24	RO	-	Reserved
23:16	RO		xCORE tile number on the switch.
15:9	RO	-	Reserved
8	RO		Set to 1 if boot from OTP is enabled.
7:0	RO		The boot mode pins MODE0, MODE1, ..., specifying the boot frequency, boot source, etc.

### B.5 Security configuration: 0x05

Copy of the security register as read from OTP.

**0x05:**  
Security  
configuration

Bits	Perm	Init	Description
31:0	RO		Value.

### B.6 Ring Oscillator Control: 0x06

There are four free-running oscillators that clock four counters. The oscillators can be started and stopped using this register. The counters should only be read when the ring oscillator is stopped. The counter values can be read using four subsequent registers. The ring oscillators are asynchronous to the xCORE tile clock and can be used as a source of random bits.

**0x06:**  
Ring  
Oscillator  
Control

Bits	Perm	Init	Description
31:2	RO	-	Reserved
1	RW	0	Set to 1 to enable the xCORE tile ring oscillators
0	RW	0	Set to 1 to enable the peripheral ring oscillators

### B.7 Ring Oscillator Value: 0x07

This register contains the current count of the xCORE Tile Cell ring oscillator. This value is not reset on a system reset.

**0x07:**  
Ring  
Oscillator  
Value

Bits	Perm	Init	Description
31:16	RO	-	Reserved
15:0	RO	-	Ring oscillator counter data.

**B.8 Ring Oscillator Value: 0x08**

This register contains the current count of the xCORE Tile Wire ring oscillator. This value is not reset on a system reset.

<b>0x08:</b> Ring Oscillator Value	Bits	Perm	Init	Description
	31:16	RO	-	Reserved
	15:0	RO	-	Ring oscillator counter data.

**B.9 Ring Oscillator Value: 0x09**

This register contains the current count of the Peripheral Cell ring oscillator. This value is not reset on a system reset.

<b>0x09:</b> Ring Oscillator Value	Bits	Perm	Init	Description
	31:16	RO	-	Reserved
	15:0	RO	-	Ring oscillator counter data.

**B.10 Ring Oscillator Value: 0x0A**

This register contains the current count of the Peripheral Wire ring oscillator. This value is not reset on a system reset.

<b>0x0A:</b> Ring Oscillator Value	Bits	Perm	Init	Description
	31:16	RO	-	Reserved
	15:0	RO	-	Ring oscillator counter data.

**B.11 Debug SSR: 0x10**

This register contains the value of the SSR register when the debugger was called.

<b>0x10:</b> Debug SSR	Bits	Perm	Init	Description
	31:0	RO	-	Reserved

**B.12 Debug SPC: 0x11**

This register contains the value of the SPC register when the debugger was called.



<b>0x11:</b> Debug SPC	Bits	Perm	Init	Description
	31:0	DRW		Value.

**B.13 Debug SSP: 0x12**

This register contains the value of the SSP register when the debugger was called.

<b>0x12:</b> Debug SSP	Bits	Perm	Init	Description
	31:0	DRW		Value.

**B.14 DGETREG operand 1: 0x13**

The resource ID of the logical core whose state is to be read.

<b>0x13:</b> DGETREG operand 1	Bits	Perm	Init	Description
	31:8	RO	-	Reserved
	7:0	DRW		Thread number to be read

**B.15 DGETREG operand 2: 0x14**

Register number to be read by DGETREG

<b>0x14:</b> DGETREG operand 2	Bits	Perm	Init	Description
	31:5	RO	-	Reserved
	4:0	DRW		Register number to be read

**B.16 Debug interrupt type: 0x15**

Register that specifies what activated the debug interrupt.

**0x15:**  
Debug  
interrupt type

Bits	Perm	Init	Description
31:18	RO	-	Reserved
17:16	DRW		If the debug interrupt was caused by a hardware breakpoint or hardware watchpoint, this field contains the number of the breakpoint or watchpoint. If multiple breakpoints or watchpoints trigger at once, the lowest number is taken.
15:8	DRW		If the debug interrupt was caused by a logical core, this field contains the number of that core. Otherwise this field is 0.
7:3	RO	-	Reserved
2:0	DRW	0	Indicates the cause of the debug interrupt 1: Host initiated a debug interrupt through JTAG 2: Program executed a DCALL instruction 3: Instruction breakpoint 4: Data watch point 5: Resource watch point

**B.17 Debug interrupt data: 0x16**

On a data watchpoint, this register contains the effective address of the memory operation that triggered the debugger. On a resource watchpoint, it contains the resource identifier.

**0x16:**  
Debug  
interrupt data

Bits	Perm	Init	Description
31:0	DRW		Value.

**B.18 Debug core control: 0x18**

This register enables the debugger to temporarily disable logical cores. When returning from the debug interrupts, the cores set in this register will not execute. This enables single stepping to be implemented.

**0x18:**  
Debug core  
control

Bits	Perm	Init	Description
31:8	RO	-	Reserved
7:0	DRW		1-hot vector defining which logical cores are stopped when not in debug mode. Every bit which is set prevents the respective logical core from running.

**B.19 Debug scratch: 0x20 .. 0x27**

A set of registers used by the debug ROM to communicate with an external debugger, for example over JTAG. This is the same set of registers as the [Debug Scratch registers in the xCORE tile configuration](#).

**0x20 .. 0x27:**  
Debug  
scratch

Bits	Perm	Init	Description
31:0	DRW		Value.

**B.20 Instruction breakpoint address: 0x30 .. 0x33**

This register contains the address of the instruction breakpoint. If the PC matches this address, then a debug interrupt will be taken. There are four instruction breakpoints that are controlled individually.

**0x30 .. 0x33:**  
Instruction  
breakpoint  
address

Bits	Perm	Init	Description
31:0	DRW		Value.

**B.21 Instruction breakpoint control: 0x40 .. 0x43**

This register controls which logical cores may take an instruction breakpoint, and under which condition.

Bits	Perm	Init	Description
31:24	RO	-	Reserved
23:16	DRW	0	A bit for each logical core in the tile allowing the breakpoint to be enabled individually for each logical core.
15:2	RO	-	Reserved
1	DRW	0	Set to 1 to cause an instruction breakpoint if the PC is not equal to the breakpoint address. By default, the breakpoint is triggered when the PC is equal to the breakpoint address.
0	DRW	0	When 1 the instruction breakpoint is enabled.

**0x40 .. 0x43:**  
Instruction  
breakpoint  
control

**B.22 Data watchpoint address 1: 0x50 .. 0x53**

This set of registers contains the first address for the four data watchpoints.

**0x50 .. 0x53:**  
Data  
watchpoint  
address 1

Bits	Perm	Init	Description
31:0	DRW		Value.

**B.23 Data watchpoint address 2: 0x60 .. 0x63**

This set of registers contains the second address for the four data watchpoints.

**0x60 .. 0x63:**  
Data  
watchpoint  
address 2

Bits	Perm	Init	Description
31:0	DRW		Value.

**B.24 Data breakpoint control register: 0x70 .. 0x73**

This set of registers controls each of the four data watchpoints.

**0x70 .. 0x73:**  
Data  
breakpoint  
control  
register

Bits	Perm	Init	Description
31:24	RO	-	Reserved
23:16	DRW	0	A bit for each logical core in the tile allowing the breakpoint to be enabled individually for each logical core.
15:3	RO	-	Reserved
2	DRW	0	Set to 1 to enable breakpoints to be triggered on loads. Breakpoints always trigger on stores.
1	DRW	0	By default, data watchpoints trigger if memory in the range [Address1..Address2] is accessed (the range is inclusive of Address1 and Address2). If set to 1, data watchpoints trigger if memory outside the range (Address2..Address1) is accessed (the range is exclusive of Address2 and Address1).
0	DRW	0	When 1 the instruction breakpoint is enabled.

**B.25 Resources breakpoint mask: 0x80 .. 0x83**

This set of registers contains the mask for the four resource watchpoints.

**0x80 .. 0x83:**  
Resources  
breakpoint  
mask

Bits	Perm	Init	Description
31:0	DRW		Value.

### B.26 Resources breakpoint value: 0x90 .. 0x93

This set of registers contains the value for the four resource watchpoints.

**0x90 .. 0x93:**  
Resources  
breakpoint  
value

Bits	Perm	Init	Description
31:0	DRW		Value.

### B.27 Resources breakpoint control register: 0x9C .. 0x9F

This set of registers controls each of the four resource watchpoints.

**0x9C .. 0x9F:**  
Resources  
breakpoint  
control  
register

Bits	Perm	Init	Description
31:24	RO	-	Reserved
23:16	DRW	0	A bit for each logical core in the tile allowing the breakpoint to be enabled individually for each logical core.
15:2	RO	-	Reserved
1	DRW	0	By default, resource watchpoints trigger when the resource id masked with the set <a href="#">Mask</a> equals the <a href="#">Value</a> . If set to 1, resource watchpoints trigger when the resource id masked with the set <a href="#">Mask</a> is not equal to the <a href="#">Value</a> .
0	DRW	0	When 1 the instruction breakpoint is enabled.

## C Tile Configuration

The xCORE Tile control registers can be accessed using configuration reads and writes (use `write_tile_config_reg(tileref, ...)` and `read_tile_config_reg(tileref, ...)` for reads and writes).

Number	Perm	Description
0x00	RO	Device identification
0x01	RO	xCORE Tile description 1
0x02	RO	xCORE Tile description 2
0x04	CRW	Control PSwitch permissions to debug registers
0x05	CRW	Cause debug interrupts
0x06	RW	xCORE Tile clock divider
0x07	RO	Security configuration
0x10 .. 0x13	RO	PLink status
0x20 .. 0x27	CRW	Debug scratch
0x40	RO	PC of logical core 0
0x41	RO	PC of logical core 1
0x42	RO	PC of logical core 2
0x43	RO	PC of logical core 3
0x44	RO	PC of logical core 4
0x45	RO	PC of logical core 5
0x46	RO	PC of logical core 6
0x47	RO	PC of logical core 7
0x60	RO	SR of logical core 0
0x61	RO	SR of logical core 1
0x62	RO	SR of logical core 2
0x63	RO	SR of logical core 3
0x64	RO	SR of logical core 4
0x65	RO	SR of logical core 5
0x66	RO	SR of logical core 6
0x67	RO	SR of logical core 7
0x80 .. 0x9F	RO	Chanend status

**Figure 27:**  
Summary

### C.1 Device identification: 0x00

**0x00:**  
Device  
identification

Bits	Perm	Init	Description
31:24	RO		Processor ID of this xCORE tile.
23:16	RO		Number of the node in which this xCORE tile is located.
15:8	RO		xCORE tile revision.
7:0	RO		xCORE tile version.

### C.2 xCORE Tile description 1: 0x01

This register describes the number of logical cores, synchronisers, locks and channel ends available on this xCORE tile.

**0x01:**  
xCORE Tile  
description 1

Bits	Perm	Init	Description
31:24	RO		Number of channel ends.
23:16	RO		Number of locks.
15:8	RO		Number of synchronisers.
7:0	RO	-	Reserved

### C.3 xCORE Tile description 2: 0x02

This register describes the number of timers and clock blocks available on this xCORE tile.

**0x02:**  
xCORE Tile  
description 2

Bits	Perm	Init	Description
31:16	RO	-	Reserved
15:8	RO		Number of clock blocks.
7:0	RO		Number of timers.

### C.4 Control PSwitch permissions to debug registers: 0x04

This register can be used to control whether the debug registers (marked with permission CRW) are accessible through the tile configuration registers. When this bit is set, write -access to those registers is disabled, preventing debugging of the xCORE tile over the interconnect.

**0x04:**  
Control  
PSwitch  
permissions  
to debug  
registers

Bits	Perm	Init	Description
31:1	RO	-	Reserved
0	CRW		Set to 1 to restrict PSwitch access to all CRW marked registers to become read-only rather than read-write.

### C.5 Cause debug interrupts: 0x05

This register can be used to raise a debug interrupt in this xCORE tile.

**0x05:**  
Cause debug  
interrupts

Bits	Perm	Init	Description
31:2	RO	-	Reserved
1	RO	0	Set to 1 when the processor is in debug mode.
0	CRW	0	Set to 1 to request a debug interrupt on the processor.

### C.6 xCORE Tile clock divider: 0x06

This register contains the value used to divide the PLL clock to create the xCORE tile clock. The divider is enabled under control of the [tile control register](#)

**0x06:**  
xCORE Tile  
clock divider

Bits	Perm	Init	Description
31:8	RO	-	Reserved
7:0	RW		Value of the clock divider minus one.

### C.7 Security configuration: 0x07

Copy of the security register as read from OTP.

**0x07:**  
Security  
configuration

Bits	Perm	Init	Description
31:0	RO		Value.

### C.8 PLink status: 0x10 .. 0x13

Status of each of the four processor links; connecting the xCORE tile to the switch.



Bits	Perm	Init	Description
31:26	RO	-	Reserved
25:24	RO		00 - ChannelEnd, 01 - ERROR, 10 - PSCTL, 11 - Idle.
23:16	RO		Based on SRC_TARGET_TYPE value, it represents channelEnd ID or Idle status.
15:6	RO	-	Reserved
5:4	RO		Two-bit network identifier
3	RO	-	Reserved
2	RO		1 when the current packet is considered junk and will be thrown away.
1	RO	0	Set to 1 if the switch is routing data into the link, and if a route exists from another link.
0	RO	0	Set to 1 if the link is routing data into the switch, and if a route is created to another link on the switch.

**0x10 .. 0x13:**  
PLink status

### C.9 Debug scratch: 0x20 .. 0x27

A set of registers used by the debug ROM to communicate with an external debugger, for example over the switch. This is the same set of registers as the [Debug Scratch registers in the processor status](#).

**0x20 .. 0x27:**  
Debug  
scratch

Bits	Perm	Init	Description
31:0	CRW		Value.

### C.10 PC of logical core 0: 0x40

Value of the PC of logical core 0.

**0x40:**  
PC of logical  
core 0

Bits	Perm	Init	Description
31:0	RO		Value.

**C.11 PC of logical core 1: 0x41**

**0x41:**  
PC of logical  
core 1

Bits	Perm	Init	Description
31:0	RO		Value.

**C.12 PC of logical core 2: 0x42**

**0x42:**  
PC of logical  
core 2

Bits	Perm	Init	Description
31:0	RO		Value.

**C.13 PC of logical core 3: 0x43**

**0x43:**  
PC of logical  
core 3

Bits	Perm	Init	Description
31:0	RO		Value.

**C.14 PC of logical core 4: 0x44**

**0x44:**  
PC of logical  
core 4

Bits	Perm	Init	Description
31:0	RO		Value.

**C.15 PC of logical core 5: 0x45**

**0x45:**  
PC of logical  
core 5

Bits	Perm	Init	Description
31:0	RO		Value.

**C.16 PC of logical core 6: 0x46**

**0x46:**  
PC of logical core 6

Bits	Perm	Init	Description
31:0	RO		Value.

**C.17 PC of logical core 7: 0x47**

**0x47:**  
PC of logical core 7

Bits	Perm	Init	Description
31:0	RO		Value.

**C.18 SR of logical core 0: 0x60**

Value of the SR of logical core 0

**0x60:**  
SR of logical core 0

Bits	Perm	Init	Description
31:0	RO		Value.

**C.19 SR of logical core 1: 0x61**

**0x61:**  
SR of logical core 1

Bits	Perm	Init	Description
31:0	RO		Value.

**C.20 SR of logical core 2: 0x62**

**0x62:**  
SR of logical core 2

Bits	Perm	Init	Description
31:0	RO		Value.

**C.21 SR of logical core 3: 0x63**

**0x63:**  
SR of logical core 3

Bits	Perm	Init	Description
31:0	RO		Value.

**C.22 SR of logical core 4: 0x64**

**0x64:**  
SR of logical core 4

Bits	Perm	Init	Description
31:0	RO		Value.

**C.23 SR of logical core 5: 0x65**

**0x65:**  
SR of logical core 5

Bits	Perm	Init	Description
31:0	RO		Value.

**C.24 SR of logical core 6: 0x66**

**0x66:**  
SR of logical core 6

Bits	Perm	Init	Description
31:0	RO		Value.

**C.25 SR of logical core 7: 0x67**

**0x67:**  
SR of logical core 7

Bits	Perm	Init	Description
31:0	RO		Value.

**C.26 Chanend status: 0x80 .. 0x9F**

These registers record the status of each channel-end on the tile.

**0x80 .. 0x9F:**  
Chanend  
status

Bits	Perm	Init	Description
31:26	RO	-	Reserved
25:24	RO		00 - ChannelEnd, 01 - ERROR, 10 - PSCTL, 11 - Idle.
23:16	RO		Based on SRC_TARGET_TYPE value, it represents channelEnd ID or Idle status.
15:6	RO	-	Reserved
5:4	RO		Two-bit network identifier
3	RO	-	Reserved
2	RO		1 when the current packet is considered junk and will be thrown away.
1	RO	0	Set to 1 if the switch is routing data into the link, and if a route exists from another link.
0	RO	0	Set to 1 if the link is routing data into the switch, and if a route is created to another link on the switch.

## D Node Configuration

The digital node control registers can be accessed using configuration reads and writes (use `write_node_config_reg(device, ...)` and `read_node_config_reg(device, ...)` for reads and writes).

Number	Perm	Description
0x00	RO	Device identification
0x01	RO	System switch description
0x04	RW	Switch configuration
0x05	RW	Switch node identifier
0x06	RW	PLL settings
0x07	RW	System switch clock divider
0x08	RW	Reference clock
0x0C	RW	Directions 0-7
0x0D	RW	Directions 8-15
0x10	RW	DEBUG_N configuration
0x1F	RO	Debug source
0x20 .. 0x27	RW	Link status, direction, and network
0x40 .. 0x43	RW	PLink status and network
0x80 .. 0x87	RW	Link configuration and initialization
0xA0 .. 0xA7	RW	Static link configuration

**Figure 28:**  
Summary

### D.1 Device identification: 0x00

This register contains version and revision identifiers and the mode-pins as sampled at boot-time.

Bits	Perm	Init	Description
31:24	RO	0x00	Chip identifier.
23:16	RO		Sampled values of pins MODE0, MODE1, ... on reset.
15:8	RO		SSwitch revision.
7:0	RO		SSwitch version.

**0x00:**  
Device  
identification

### D.2 System switch description: 0x01

This register specifies the number of processors and links that are connected to this switch.

**0x01:**  
System  
switch  
description

Bits	Perm	Init	Description
31:24	RO	-	Reserved
23:16	RO		Number of links on the switch.
15:8	RO		Number of cores that are connected to this switch.
7:0	RO		Number of links per processor.

### D.3 Switch configuration: 0x04

This register enables the setting of two security modes (that disable updates to the PLL or any other registers) and the header-mode.

**0x04:**  
Switch  
configuration

Bits	Perm	Init	Description
31	RO	0	Set to 1 to disable any write access to the configuration registers in this switch.
30:9	RO	-	Reserved
8	RO	0	Set to 1 to disable updates to the PLL configuration register.
7:1	RO	-	Reserved
0	RO	0	Header mode. Set to 1 to enable 1-byte headers. This must be performed on all nodes in the system.

### D.4 Switch node identifier: 0x05

This register contains the node identifier.

**0x05:**  
Switch node  
identifier

Bits	Perm	Init	Description
31:16	RO	-	Reserved
15:0	RW	0	The unique 16-bit ID of this node. This ID is matched most-significant-bit first with incoming messages for routing purposes.

### D.5 PLL settings: 0x06

An on-chip PLL multiplies the input clock up to a higher frequency clock, used to clock the I/O, processor, and switch, see [Oscillator](#). Note: a write to this register will cause the tile to be reset.

**0x06:**  
PLL settings

Bits	Perm	Init	Description
31:26	RO	-	Reserved
25:23	RW		OD: Output divider value The initial value depends on pins MODE0 and MODE1.
22:21	RO	-	Reserved
20:8	RW		F: Feedback multiplication ratio The initial value depends on pins MODE0 and MODE1.
7	RO	-	Reserved
6:0	RW		R: Oscillator input divider value The initial value depends on pins MODE0 and MODE1.

### D.6 System switch clock divider: 0x07

Sets the ratio of the PLL clock and the switch clock.

**0x07:**  
System  
switch clock  
divider

Bits	Perm	Init	Description
31:16	RO	-	Reserved
15:0	RW	0	Switch clock divider. The PLL clock will be divided by this value plus one to derive the switch clock.

### D.7 Reference clock: 0x08

Sets the ratio of the PLL clock and the reference clock used by the node.

**0x08:**  
Reference  
clock

Bits	Perm	Init	Description
31:16	RO	-	Reserved
15:0	RW	3	Architecture reference clock divider. The PLL clock will be divided by this value plus one to derive the 100 MHz reference clock.

### D.8 Directions 0-7: 0x0C

This register contains eight directions, for packets with a mismatch in bits 7..0 of the node-identifier. The direction in which a packet will be routed is governed by the most significant mismatching bit.



**0x0C:**  
Directions  
0-7

Bits	Perm	Init	Description
31:28	RW	0	The direction for packets whose first mismatching bit is 7.
27:24	RW	0	The direction for packets whose first mismatching bit is 6.
23:20	RW	0	The direction for packets whose first mismatching bit is 5.
19:16	RW	0	The direction for packets whose first mismatching bit is 4.
15:12	RW	0	The direction for packets whose first mismatching bit is 3.
11:8	RW	0	The direction for packets whose first mismatching bit is 2.
7:4	RW	0	The direction for packets whose first mismatching bit is 1.
3:0	RW	0	The direction for packets whose first mismatching bit is 0.

### D.9 Directions 8-15: 0x0D

This register contains eight directions, for packets with a mismatch in bits 15..8 of the node-identifier. The direction in which a packet will be routed is governed by the most significant mismatching bit.

**0x0D:**  
Directions  
8-15

Bits	Perm	Init	Description
31:28	RW	0	The direction for packets whose first mismatching bit is 15.
27:24	RW	0	The direction for packets whose first mismatching bit is 14.
23:20	RW	0	The direction for packets whose first mismatching bit is 13.
19:16	RW	0	The direction for packets whose first mismatching bit is 12.
15:12	RW	0	The direction for packets whose first mismatching bit is 11.
11:8	RW	0	The direction for packets whose first mismatching bit is 10.
7:4	RW	0	The direction for packets whose first mismatching bit is 9.
3:0	RW	0	The direction for packets whose first mismatching bit is 8.

### D.10 DEBUG\_N configuration: 0x10

Configures the behavior of the DEBUG\_N pin.

**0x10:**  
DEBUG\_N  
configuration

Bits	Perm	Init	Description
31:2	RO	-	Reserved
1	RW	0	Set to 1 to enable signals on DEBUG_N to generate DCALL on the core.
0	RW	0	When set to 1, the DEBUG_N wire will be pulled down when the node enters debug mode.

### D.11 Debug source: 0x1F

Contains the source of the most recent debug event.

**0x1F:**  
Debug source

Bits	Perm	Init	Description
31:5	RO	-	Reserved
4	RW		If set, the external DEBUG_N pin is the source of the most recent debug interrupt.
3:1	RO	-	Reserved
0	RW		If set, the xCORE Tile is the source of the most recent debug interrupt.

### D.12 Link status, direction, and network: 0x20 .. 0x27

These registers contain status information for low level debugging (read-only), the network number that each link belongs to, and the direction that each link is part of.

**0x20 .. 0x27:**  
Link status,  
direction, and  
network

Bits	Perm	Init	Description
31:26	RO	-	Reserved
25:24	RO		If this link is currently routing data into the switch, this field specifies the type of link that the data is routed to: 0: external link 1: plink 2: internal control link
23:16	RO	0	If the link is routing data into the switch, this field specifies the destination link number to which all tokens are sent.
15:12	RO	-	Reserved
11:8	RW	0	The direction that this this link is associated with; set for routing.
7:6	RO	-	Reserved
5:4	RW	0	Determines the network to which this link belongs, set for quality of service.
3	RO	-	Reserved
2	RO	0	Set to 1 if the current packet is junk and being thrown away. A packet is considered junk if, for example, it is not routable.
1	RO	0	Set to 1 if the switch is routing data into the link, and if a route exists from another link.
0	RO	0	Set to 1 if the link is routing data into the switch, and if a route is created to another link on the switch.

### D.13 PLink status and network: 0x40 .. 0x43

These registers contain status information and the network number that each processor-link belongs to.

Bits	Perm	Init	Description
31:26	RO	-	Reserved
25:24	RO		If this link is currently routing data into the switch, this field specifies the type of link that the data is routed to: 0: external link 1: plink 2: internal control link
23:16	RO	0	If the link is routing data into the switch, this field specifies the destination link number to which all tokens are sent.
15:6	RO	-	Reserved
5:4	RW	0	Determines the network to which this link belongs, set for quality of service.
3	RO	-	Reserved
2	RO	0	Set to 1 if the current packet is junk and being thrown away. A packet is considered junk if, for example, it is not routable.
1	RO	0	Set to 1 if the switch is routing data into the link, and if a route exists from another link.
0	RO	0	Set to 1 if the link is routing data into the switch, and if a route is created to another link on the switch.

**0x40 .. 0x43:**  
PLink status  
and network

### D.14 Link configuration and initialization: 0x80 .. 0x87

These registers contain configuration and debugging information specific to external links. The link speed and width can be set, the link can be initialized, and the link status can be monitored.

Bits	Perm	Init	Description
31	RW	0	Write '1' to this bit to enable the link, write '0' to disable it. This bit controls the muxing of ports with overlapping links.
30	RW	0	Set to 0 to operate in 2 wire mode or 1 to operate in 5 wire mode
29:28	RO	-	Reserved
27	RO	0	Set to 1 on error: an RX buffer overflow or illegal token encoding has been received. This bit clears on reading.
26	RO	0	1 if this end of the link has issued credit to allow the remote end to transmit.
25	RO	0	1 if this end of the link has credits to allow it to transmit.
24	WO	0	Set to 1 to initialize a half-duplex link. This clears this end of the link's credit and issues a HELLO token; the other side of the link will reply with credits. This bit is self-clearing.
23	WO	0	Set to 1 to reset the receiver. The next symbol that is detected will be assumed to be the first symbol in a token. This bit is self-clearing.
22	RO	-	Reserved
21:11	RW	0	The number of system clocks between two subsequent transitions within a token
10:0	RW	0	The number of system clocks between two subsequent transmit tokens.

**0x80 .. 0x87:**  
Link configuration and initialization

### D.15 Static link configuration: 0xA0 .. 0xA7

These registers are used for static (ie, non-routed) links. When a link is made static, all traffic is forwarded to the designated channel end and no routing is attempted.

Bits	Perm	Init	Description
31	RW	0	Enable static forwarding.
30:5	RO	-	Reserved
4:0	RW	0	The destination channel end on this node that packets received in static mode are forwarded to.

**0xA0 .. 0xA7:**  
Static link configuration

## E XMOS USB Interface

XMOS provides a low-level USB interface for connecting the device to a USB transceiver using the UTMI+ Low Pin Interface (ULPI). The ULPI signals must be connected to the pins named in Figure 29. Note also that some ports on the same tile are used internally and are not available for use when the USB driver is active (they are available otherwise).

Pin	Signal
XnD02	Unavailable when USB active
XnD03	
XnD04	
XnD05	
XnD06	
XnD07	
XnD08	
XnD09	

Pin	Signal
XnD12	ULPI_STP
XnD13	ULPI_NXT
XnD14	ULPI_DATA[0]
XnD15	ULPI_DATA[1]
XnD16	ULPI_DATA[2]
XnD17	ULPI_DATA[3]
XnD18	ULPI_DATA[4]
XnD19	ULPI_DATA[5]
XnD20	ULPI_DATA[6]
XnD21	ULPI_DATA[7]
XnD22	ULPI_DIR
XnD23	ULPI_CLK

Pin	Signal
XnD26	Unavailable when USB active
XnD27	
XnD28	
XnD29	
XnD30	
XnD31	
XnD32	
XnD33	

XnD37	Unavailable when USB active
XnD38	
XnD39	
XnD40	
XnD41	
XnD42	
XnD43	

**Figure 29:**  
ULPI signals provided by the XMOS USB driver

## F Device Errata

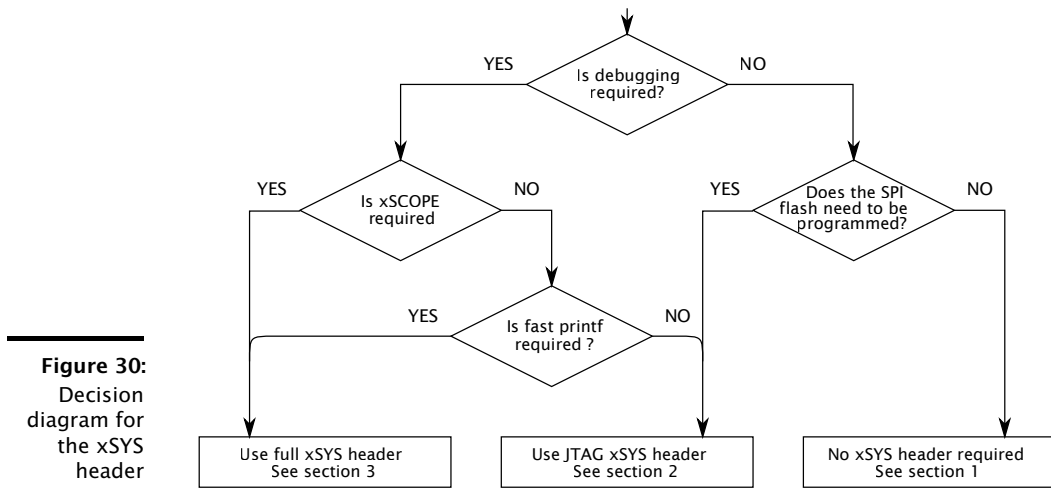
This section describes minor operational differences from the data sheet and recommended workarounds. As device and documentation issues become known, this section will be updated the document revised.

To guarantee a logic low is seen on the pins RST\_N, DEBUG\_N, MODE[3:0], TRST\_N, TMS, TCK and TDI, the driving circuit should present an impedance of less than 100Ω to ground. Usually this is not a problem for CMOS drivers driving single inputs. If one or more of these inputs are placed in parallel, however, additional logic buffers may be required to guarantee correct operation.

For static inputs tied high or low, the relevant input pin should be tied directly to GND or VDDIO.

## G JTAG, xSCOPE and Debugging

If you intend to design a board that can be used with the XMOS toolchain and xTAG debugger, you will need an xSYS header on your board. Figure 30 shows a decision diagram which explains what type of xSYS connectivity you need. The three subsections below explain the options in detail.



**Figure 30:**  
Decision diagram for the xSYS header

### G.1 No xSYS header

The use of an xSYS header is optional, and may not be required for volume production designs. However, the XMOS toolchain expects the xSYS header; if you do not have an xSYS header then you must provide your own method for writing to flash/OTP and for debugging.

### G.2 JTAG-only xSYS header

The xSYS header connects to an xTAG debugger, which has a 20-pin 0.1" female IDC header. The design will hence need a male IDC header. We advise to use a boxed header to guard against incorrect plug-ins. If you use a 90 degree angled header, make sure that pins 2, 4, 6, ..., 20 are along the edge of the PCB.

Connect pins 4, 8, 12, 16, 20 of the xSYS header to ground, and then connect:

- ▶ TDI to pin 5 of the xSYS header
- ▶ TMS to pin 7 of the xSYS header
- ▶ TCK to pin 9 of the xSYS header
- ▶ DEBUG\_N to pin 11 of the xSYS header

- ▶ TDO to pin 13 of the xSYS header
- ▶ RST\_N and TRST\_N to pin 15 of the xSYS header
- ▶ If MODE2 is configured high, connect MODE2 to pin 3 of the xSYS header. Do not connect to VDDIO.
- ▶ If MODE3 is configured high, connect MODE3 to pin 3 of the xSYS header. Do not connect to VDDIO.

The RST\_N net should be open-drain, active-low, and have a pull-up to VDDIO.

### G.3 Full xSYS header

For a full xSYS header you will need to connect the pins as discussed in Section G.2, and then connect a 2-wire xCONNECT Link to the xSYS header. The links can be found in the Signal description table (Section 4): they are labelled XLA, XLB, etc in the function column. The 2-wire link comprises two inputs and outputs, labelled  ${}^1_{out}$ ,  ${}^0_{out}$ ,  ${}^0_{in}$ , and  ${}^1_{in}$ . For example, if you choose to use XLB of tile 0 for xSCOPE I/O, you need to connect up  ${}^1_{out}$ ,  ${}^0_{out}$ ,  ${}^0_{in}$ ,  ${}^1_{in}$  as follows:

- ▶  ${}^1_{out}$  (X0D16) to pin 6 of the xSYS header with a 33R series resistor close to the device.
- ▶  ${}^0_{out}$  (X0D17) to pin 10 of the xSYS header with a 33R series resistor close to the device.
- ▶  ${}^0_{in}$  (X0D18) to pin 14 of the xSYS header.
- ▶  ${}^1_{in}$  (X0D19) to pin 18 of the xSYS header.

## H Schematics Design Check List

- This section is a checklist for use by schematics designers using the XS1-L8A-64-TQ128. Each of the following sections contains items to check for each design.

### H.1 Power supplies

- VDDIO supply is within specification (3.0V - 3.6V) before the VDD (core) supply is turned on. Specifically, the VDDIO supply is within specification before VDD (core) reaches 0.4V (Section 10).
- The VDD (core) supply ramps monotonically (rises constantly) from 0V to its final value (0.95V - 1.05V) within 10ms (Section 10).
- The VDD (core) supply is capable of supplying 300mA (Section 10).
- PLL\_AVDD is filtered with a low pass filter, for example An RC filter, see Section 10

### H.2 Power supply decoupling

- The design has multiple decoupling capacitors per supply, for example at least four 0402 or 0603 size surface mount capacitors of 100nF in value, per supply (Section 10).
- A bulk decoupling capacitor of at least 10uF is placed on each supply (Section 10).

### H.3 Power on reset

- The RST\_N and TRST\_Npins are asserted (low) during or after power up. The device is not used until these resets have taken place. As the errata in the datasheets show, the internal pull-ups on these two pins can occasionally provide stronger than normal pull-up currents. For this reason, an RC type reset circuit is discouraged as behavior would be unpredictable. A voltage supervisor type reset device is recommended to guarantee a good reset. This also has the benefit of resetting the system should the relevant supply go out of specification.

### H.4 Clock

- The CLK input pin is supplied with a clock with monotonic rising edges and low jitter.



- Pins MODE0 and MODE1 are set to the correct value for the chosen oscillator frequency. The MODE settings are shown in the Oscillator section, Section 6. If you have a choice between two values, choose the value with the highest multiplier ratio since that will boot faster.

## H.5 USB ULPI Mode

This section can be skipped if you do not have an external USB PHY.

- If using ULPI, the ULPI signals are connected to specific ports as shown in Section E.
- If using ULPI, the ports that are used internally are not connected, see Section E. (Note that this limitation only applies when the ULPI is enabled, they can still be used before or after the ULPI is being used.)

## H.6 Boot

- The device is connected to a SPI flash for booting, connected to X0D0, X0D01, X0D10, and X0D11 (Section 7). If not, you must boot the device through OTP or JTAG.
- The device that is connected to flash has both MODE2 and MODE3 NC (Section 7).
- The SPI flash that you have chosen is supported by **xflash**, or you have created a specification file for it.

## H.7 JTAG, XScope, and debugging

- You have decided as to whether you need an XSYS header or not (Section G).
- If you included an XSYS header, you connected pin 3 to any MODE2/MODE3 pin that would otherwise be NC (Section G).
- If you have not included an XSYS header, you have devised a method to program the SPI-flash or OTP (Section G).

## H.8 Multi device designs

Skip this section if your design only includes a single XMOS device.

- One device is connected to a SPI flash for booting.

- Devices that boot from link have MODE2 grounded and MODE3 NC. These device must have link XLB connected to a device to boot from (see [7](#)).
- If you included an XSYS header, you have included buffers for RST\_N, TRST\_N, TMS, TCK, MODE2, and MODE3 (Section [F](#)).

## I PCB Layout Design Check List

- This section is a checklist for use by PCB designers using the XS1-L8A-64-TQ128. Each of the following sections contains items to check for each design.

### I.1 Ground Plane

- Multiple vias (eg, 9) have been used to connect the center pad to the PCB ground plane. These minimize impedance and conduct heat away from the device. (Section 10.2).
- Other than ground vias, there are no (or only a few) vias underneath or closely around the device. This create a good, solid, ground plane. Examples of a good and bad designs are shown in Figure XXX.

### I.2 Power supply decoupling

- The decoupling capacitors are all placed close to a supply pin (Section 10).
- The decoupling capacitors are spaced around the device (Section 10).
- The ground side of each decoupling capacitor has a direct path back to the center ground the device. Examples of good and bad decoupling designs are shown in Figure XXX.

### I.3 PLL\_AVDD

- The PLL\_AVDD filter (especially the capacitor) is placed close to the PLL\_AVDD pin (Section 10).

## J Associated Design Documentation

Document Title	Information	Document Number
XS1-L Hardware Design Checklist	Board design checklist	<a href="#">X6277</a>
Estimating Power Consumption For XS1-L Devices	Power consumption	<a href="#">X4271</a>
Programming XC on XMOS Devices	Timers, ports, clocks, cores and channels	<a href="#">X9577</a>
xTIMEcomposer User Guide	Compilers, assembler and linker/mapper Timing analyzer, xScope, debugger Flash and OTP programming utilities	<a href="#">X3766</a>

## K Related Documentation

Document Title	Information	Document Number
The XMOS XS1 Architecture	ISA manual	<a href="#">X7879</a>
XS1 Port I/O Timing	Port timings	<a href="#">X5821</a>
XS1-L System Specification	Link, switch and system information	<a href="#">X1151</a>
XS1-L Link Performance and Design Guidelines	Link timings	<a href="#">X2999</a>
XS1-L Clock Frequency Control	Advanced clock control	<a href="#">X1433</a>
XS1-L Active Power Conservation	Low-power mode during idle	<a href="#">X7411</a>

## L Revision History

Date	Description
2013-01-30	New datasheet - revised part numbering
2013-02-26	New multicore microcontroller introduction Moved configuration sections to appendices
2013-07-19	Updated Features list with available ports and links - Section <a href="#">2</a> Simplified link bits in Signal Description - Section <a href="#">4</a> New JTAG, xSCOPE and Debugging appendix - Section <a href="#">G</a> New Schematics Design Check List - Section <a href="#">H</a> New PCB Layout Design Check List - Section <a href="#">I</a>



Copyright © 2013, All Rights Reserved.

---

Xmos Ltd. is the owner or licensee of this design, code, or Information (collectively, the "Information") and is providing it to you "AS IS" with no warranty of any kind, express or implied and shall have no liability in relation to its use. Xmos Ltd. makes no representation that the Information, or any particular implementation thereof, is or will be free from any claims of infringement and again, shall have no liability in relation to any such claims.