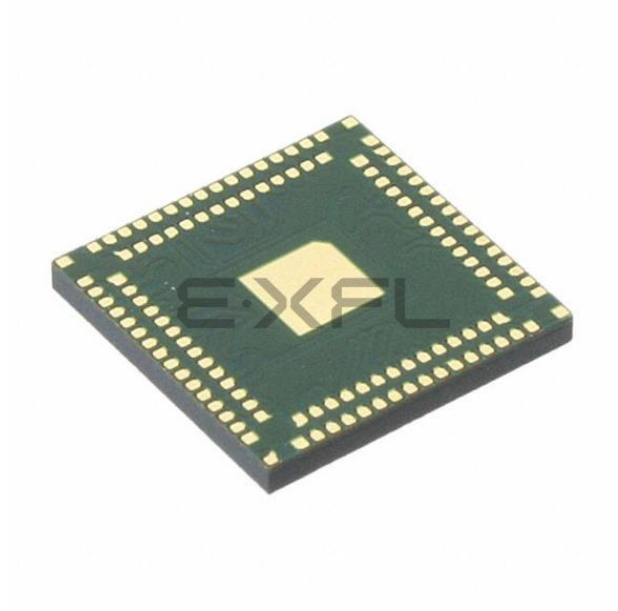


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	XCore
Core Size	32-Bit 10-Core
Speed	800MIPS
Connectivity	Configurable
Peripherals	-
Number of I/O	84
Program Memory Size	128KB (32K x 32)
Program Memory Type	SRAM
EEPROM Size	-
RAM Size	-
Voltage - Supply (Vcc/Vdd)	0.95V ~ 3.6V
Data Converters	-
Oscillator Type	External
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	124-TFQFN Dual Rows, Exposed Pad
Supplier Device Package	124-QFN DualRow (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/xmos/xs1-l10a-128-qf124-c8

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 and communicate between tasks using a high speed network. Because xCORE multicore microcontrollers are completely deterministic, you can write software to implement functions that traditionally require dedicated hardware.

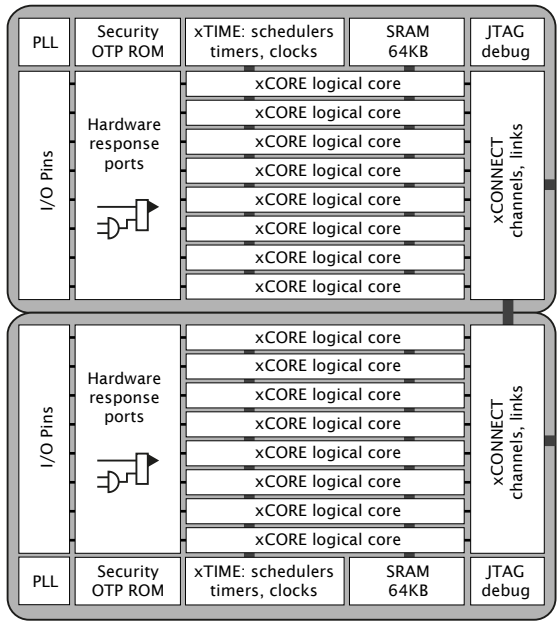


Figure 1: XS1-L Series: 4-16 core devices

Key features of the XS1-L10A-128-QF124 include:

- **Tiles**: Devices consist of one or more xCORE tiles. Each tile contains between four and eight 32-bit xCOREs with highly integrated I/O and on-chip memory.
- **Logical cores** Each logical core can execute tasks such as computational code, DSP code, control software (including logic decisions and executing a state machine) or software that handles I/O. Section 5.1
- **xTIME scheduler** The xTIME scheduler performs functions similar to an RTOS, in hardware. It services and synchronizes events in a core, so there is no requirement for interrupt handler routines. The xTIME scheduler triggers cores on events generated by hardware resources such as the I/O pins, communication channels and timers. Once triggered, a core runs independently and concurrently to other cores, until it pauses to wait for more events. Section 5.2

- **Channels and channel ends** Tasks running on logical cores communicate using channels formed between two channel ends. Data can be passed synchronously or asynchronously between the channel ends assigned to the communicating tasks. Section 5.5
- **xCONNECT Switch and Links** Between tiles, channel communications are implemented over a high performance network of xCONNECT Links and routed through a hardware xCONNECT Switch. Section 5.6
- **Ports** The I/O pins are connected to the processing cores by Hardware Response ports. The port logic can drive its pins high and low, or it can sample the value on its pins optionally waiting for a particular condition. Section 5.3
- **Clock blocks** xCORE devices include a set of programmable clock blocks that can be used to govern the rate at which ports execute. Section 5.4
- **Memory** Each xCORE Tile integrates a bank of SRAM for instructions and data, and a block of one-time programmable (OTP) memory that can be configured for system wide security features. Section 8
- **PLL** The PLL is used to create a high-speed processor clock given a low speed external oscillator. Section 6
- **JTAG** The JTAG module can be used for loading programs, boundary scan testing, in-circuit source-level debugging and programming the OTP memory. Section 9

1.1 Software

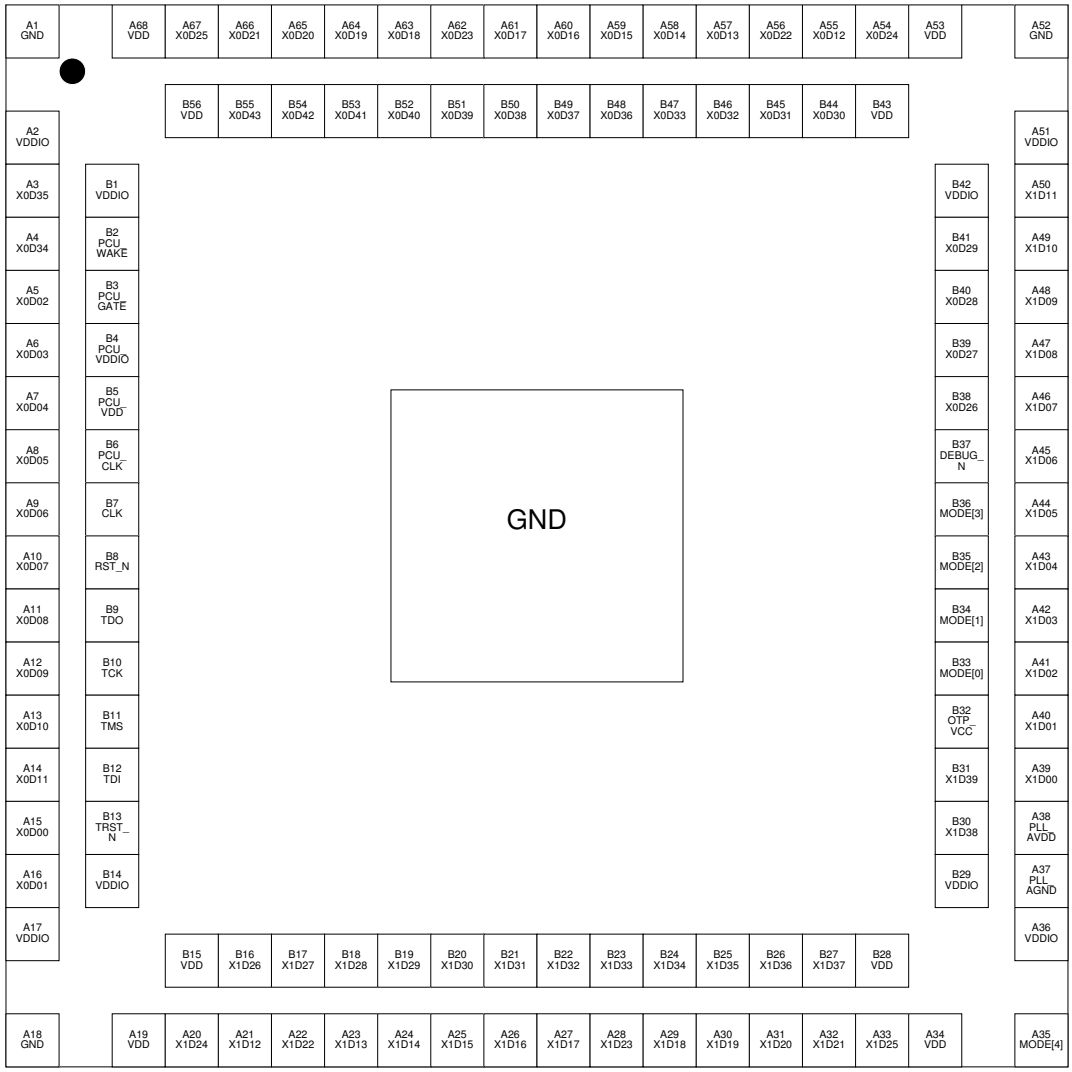
Devices are programmed using C, C++ or xC (C with multicore extensions). XMOS provides tested and proven software libraries, which allow you to quickly add interface and processor functionality such as USB, Ethernet, PWM, graphics driver, and audio EQ to your applications.

1.2 xTIMEcomposer Studio

The xTIMEcomposer Studio development environment provides all the tools you need to write and debug your programs, profile your application, and write images into flash memory or OTP memory on the device. Because xCORE devices operate deterministically, they can be simulated like hardware within xTIMEcomposer: uniquely in the embedded world, xTIMEcomposer Studio therefore includes a static timing analyzer, cycle-accurate simulator, and high-speed in-circuit instrumentation.

xTIMEcomposer can be driven from either a graphical development environment, or the command line. The tools are supported on Windows, Linux and MacOS X and available at no cost from xmos.com/downloads. Information on using the tools is provided in the xTIMEcomposer User Guide, X3766.

3 Pin Configuration



Signal	Function					Type	Properties
X0D01	XLA^4_{out}	$1B^0$				I/O	PD_S, R_S
X0D02	XLA^3_{out}		$4A^0$	$8A^0$	$16A^0$ $32A^{20}$	I/O	PD_S, R_U
X0D03	XLA^2_{out}		$4A^1$	$8A^1$	$16A^1$ $32A^{21}$	I/O	PD_S, R_U
X0D04	XLA^1_{out}		$4B^0$	$8A^2$	$16A^2$ $32A^{22}$	I/O	PD_S, R_U
X0D05	XLA^0_{out}		$4B^1$	$8A^3$	$16A^3$ $32A^{23}$	I/O	PD_S, R_U
X0D06	XLA^0_{in}		$4B^2$	$8A^4$	$16A^4$ $32A^{24}$	I/O	PD_S, R_U
X0D07	XLA^1_{in}		$4B^3$	$8A^5$	$16A^5$ $32A^{25}$	I/O	PD_S, R_U
X0D08	XLA^2_{in}		$4A^2$	$8A^6$	$16A^6$ $32A^{26}$	I/O	PD_S, R_U
X0D09	XLA^3_{in}		$4A^3$	$8A^7$	$16A^7$ $32A^{27}$	I/O	PD_S, R_U
X0D10	XLA^4_{in}	$1C^0$				I/O	PD_S, R_S
X0D11		$1D^0$				I/O	PD_S, R_S
X0D12		$1E^0$				I/O	PD_S, R_U
X0D13	XLB^4_{out}	$1F^0$				I/O	PD_S, R_U
X0D14	XLB^3_{out}		$4C^0$	$8B^0$	$16A^8$ $32A^{28}$	I/O	PD_S, R_U
X0D15	XLB^2_{out}		$4C^1$	$8B^1$	$16A^9$ $32A^{29}$	I/O	PD_S, R_U
X0D16	XLB^1_{out}		$4D^0$	$8B^2$	$16A^{10}$	I/O	PD_S, R_U
X0D17	XLB^0_{out}		$4D^1$	$8B^3$	$16A^{11}$	I/O	PD_S, R_U
X0D18	XLB^0_{in}		$4D^2$	$8B^4$	$16A^{12}$	I/O	PD_S, R_U
X0D19	XLB^1_{in}		$4D^3$	$8B^5$	$16A^{13}$	I/O	PD_S, R_U
X0D20	XLB^2_{in}		$4C^2$	$8B^6$	$16A^{14}$ $32A^{30}$	I/O	PD_S, R_U
X0D21	XLB^3_{in}		$4C^3$	$8B^7$	$16A^{15}$ $32A^{31}$	I/O	PD_S, R_U
X0D22	XLB^4_{in}	$1G^0$				I/O	PD_S, R_U
X0D23		$1H^0$				I/O	PD_S, R_U
X0D24		$1I^0$				I/O	PD_S
X0D25		$1J^0$				I/O	PD_S
X0D26			$4E^0$	$8C^0$	$16B^0$	I/O	PD_S, R_U
X0D27			$4E^1$	$8C^1$	$16B^1$	I/O	PD_S, R_U
X0D28			$4F^0$	$8C^2$	$16B^2$	I/O	PD_S, R_U
X0D29			$4F^1$	$8C^3$	$16B^3$	I/O	PD_S, R_U
X0D30			$4F^2$	$8C^4$	$16B^4$	I/O	PD_S, R_U
X0D31			$4F^3$	$8C^5$	$16B^5$	I/O	PD_S, R_U
X0D32			$4E^2$	$8C^6$	$16B^6$	I/O	PD_S, R_U
X0D33			$4E^3$	$8C^7$	$16B^7$	I/O	PD_S, R_U
X0D34		$1K^0$				I/O	PD_S
X0D35		$1L^0$				I/O	PD_S
X0D36		$1M^0$		$8D^0$	$16B^8$	I/O	PD_S
X0D37		$1N^0$		$8D^1$	$16B^9$	I/O	PD_S, R_U
X0D38		$1O^0$		$8D^2$	$16B^{10}$	I/O	PD_S, R_U
X0D39		$1P^0$		$8D^3$	$16B^{11}$	I/O	PD_S, R_U
X0D40				$8D^4$	$16B^{12}$	I/O	PD_S, R_U
X0D41				$8D^5$	$16B^{13}$	I/O	PD_S, R_U
X0D42				$8D^6$	$16B^{14}$	I/O	PD_S, R_U
X0D43				$8D^7$	$16B^{15}$	I/O	PU_S, R_U

(continued)

pins (5)			
Signal	**Function**	**Type**	**Properties**
PCU_CLK	Clock input		
PCU_GATE	Power control gate control		
PCU_VDD	PCU tile power		
PCU_VDDIO	PCU I/O supply		
PCU_WAKE	Wakeup reset		

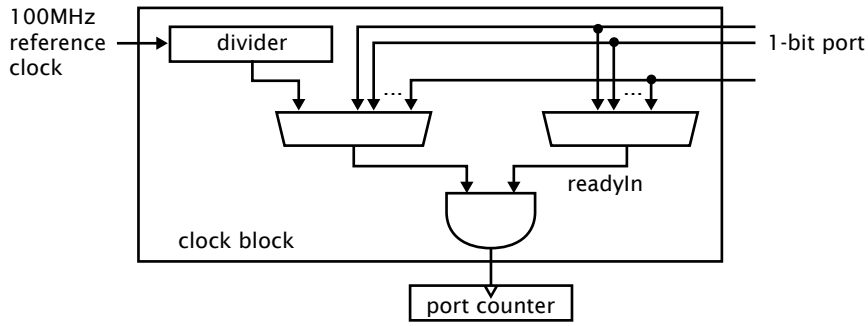


Figure 4: Clock block diagram

In many cases I/O signals are accompanied by strobing signals. The xCORE ports can input and interpret strobe (known as readyIn and readyOut) signals generated by external sources, and ports can generate strobe signals to accompany output data.

On reset, each port is connected to clock block 0, which runs from the xCORE Tile reference clock.

5.5 Channels and Channel Ends

Logical cores communicate using point-to-point connections, formed between two channel ends. A channel-end is a resource on an xCORE tile, that is allocated by the program. Each channel-end has a unique system-wide identifier that comprises a unique number and their tile identifier. Data is transmitted to a channel-end by an output-instruction; and the other side executes an input-instruction. Data can be passed synchronously or asynchronously between the channel ends.

5.6 xCONNECT Switch and Links

XMOS devices provide a scalable architecture, where multiple xCORE devices can be connected together to form one system. Each xCORE device has an xCONNECT interconnect that provides a communication infrastructure for all tasks that run on the various xCORE tiles on the system.

The interconnect relies on a collection of switches and XMOS links. Each xCORE device has an on-chip switch that can set up circuits or route data. The switches are connected by xConnect Links. An XMOS link provides a physical connection between two switches. The switch has a routing algorithm that supports many different topologies, including lines, meshes, trees, and hypercubes.

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.

11 DC and Switching Characteristics

11.1 Operating Conditions

Figure 18: 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	
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	

11.2 DC Characteristics

Figure 19: 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	2.00			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

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.

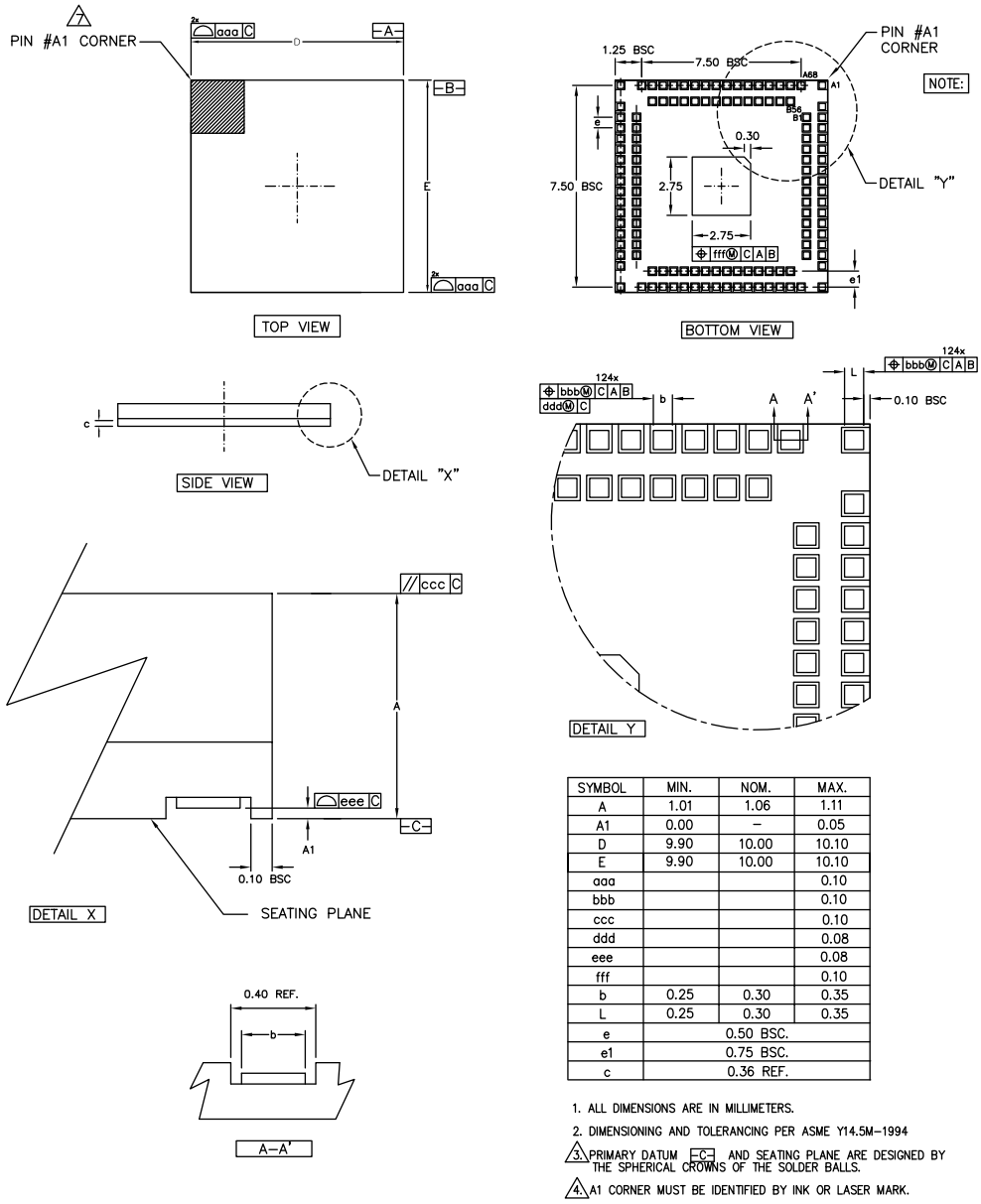
D Used to guarantee logic state for an I/O when high impedance. The internal pull-ups/pull-downs should not be used to pull external circuitry.

11.3 ESD Stress Voltage

Figure 20: 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	

12 Package Information



SYMBOL	MIN.	NOM.	MAX.
A	1.01	1.06	1.11
A1	0.00	–	0.05
D	9.90	10.00	10.10
E	9.90	10.00	10.10
aaa			0.10
bbb			0.10
ccc			0.10
ddd			0.08
eee			0.08
fff			0.10
b	0.25	0.30	0.35
L	0.25	0.30	0.35
e	0.50 BSC.		
e1	0.75 BSC.		
c	0.36 REF.		

1. ALL DIMENSIONS ARE IN MILLIMETERS.
2. DIMENSIONING AND TOLERANCING PER ASME Y14.5M–1994
3. PRIMARY DATUM -C- AND SEATING PLANE ARE DESIGNED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.
4. A1 CORNER MUST BE IDENTIFIED BY INK OR LASER MARK.

12.1 Part Marking

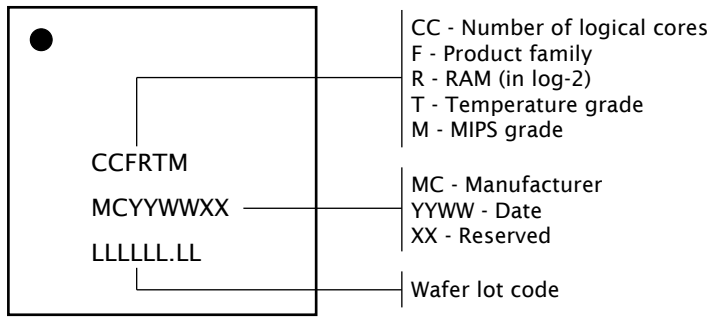


Figure 27: Part marking scheme

13 Ordering Information

Product Code	Marking	Qualification	Speed Grade
XS1-L10A-128-QF124-C8	10L7C8	Commercial	800 MIPS
XS1-L10A-128-QF124-C10	10L7C10	Commercial	1000 MIPS
XS1-L10A-128-QF124-I8	10L7I8	Industrial	800 MIPS
XS1-L10A-128-QF124-I10	10L7I10	Industrial	1000 MIPS

Figure 28: Orderable part numbers

Appendices

A Configuration of the XS1

The device is configured through three banks of registers, as shown in Figure 29.

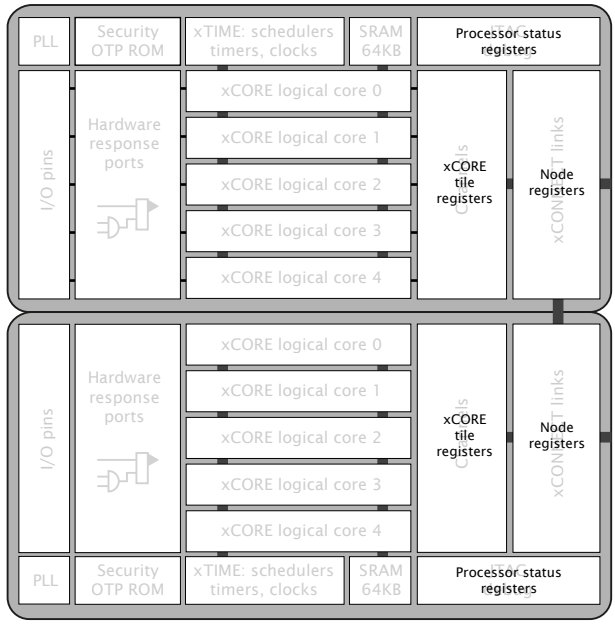


Figure 29: Registers

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`

0x11: 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.

0x12: 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.

0x13: 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

0x14: 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.

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.

0x40 .. 0x43: Instruction breakpoint control

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.

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

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

0x10 .. 0x13: PLink 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.

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.16 SR of logical core 1: 0x61

0x61: SR of logical core 1

Bits	Perm	Init	Description
31:0	RO		Value.

C.17 SR of logical core 2: 0x62

0x62: SR of logical core 2

Bits	Perm	Init	Description
31:0	RO		Value.

C.18 SR of logical core 3: 0x63

0x63: SR of logical core 3

Bits	Perm	Init	Description
31:0	RO		Value.

C.19 SR of logical core 4: 0x64

0x64: SR of logical core 4

Bits	Perm	Init	Description
31:0	RO		Value.

C.20 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.

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: Oscilator 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 goverened by the most significant mismatching bit.

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. The registers control links C, D, A, B, G, H, E, and F in that order.

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: plink 1: external link 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.

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 33. 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
X*n*D02	Unavailable when USB active
X*n*D03	
X*n*D04	
X*n*D05	
X*n*D06	
X*n*D07	
X*n*D08	
X*n*D09	

Pin	Signal
X*n*D12	ULPI_STP
X*n*D13	ULPI_NXT
X*n*D14	ULPI_DATA[0]
X*n*D15	ULPI_DATA[1]
X*n*D16	ULPI_DATA[2]
X*n*D17	ULPI_DATA[3]
X*n*D18	ULPI_DATA[4]
X*n*D19	ULPI_DATA[5]
X*n*D20	ULPI_DATA[6]
X*n*D21	ULPI_DATA[7]
X*n*D22	ULPI_DIR
X*n*D23	ULPI_CLK

Pin	Signal
X*n*D26	Unavailable when USB active
X*n*D27	
X*n*D28	
X*n*D29	
X*n*D30	
X*n*D31	
X*n*D32	
X*n*D33	

X*n*D37	Unavailable when USB active
X*n*D38	
X*n*D39	
X*n*D40	
X*n*D41	
X*n*D42	
X*n*D43	

Figure 33: 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[4:0], TRST_N, TMS, TCK and TDI, the driving circuit should present an impedance of less than 100 Ω 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 34 shows a decision diagram which explains what type of xSYS connectivity you need. The three subsections below explain the options in detail.

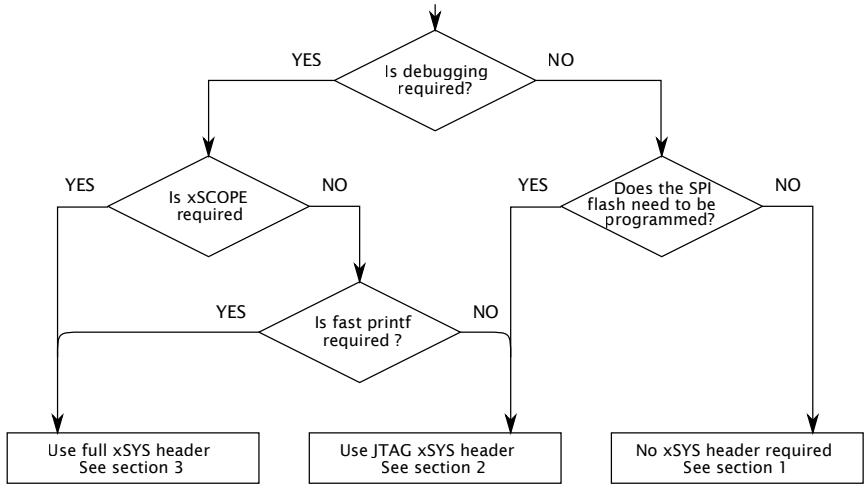


Figure 34: Decision diagram for the xSYS header

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