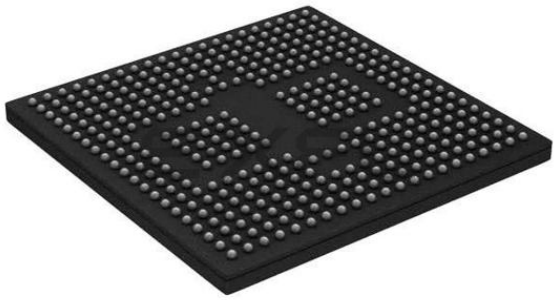

**Welcome to E-XFL.COM**

## 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 24-Core |
| Speed | 4000MIPS |
| Connectivity | - |
| Peripherals | - |
| Number of I/O | 176 |
| Program Memory Size | - |
| Program Memory Type | ROMless |
| EEPROM Size | - |
| RAM Size | 512K x 8 |
| Voltage - Supply (Vcc/Vdd) | 0.95V ~ 3.6V |
| Data Converters | - |
| Oscillator Type | External |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 374-LFBGA |
| Supplier Device Package | 374-FBGA (18x18) |
| Purchase URL | https://www.e-xfl.com/product-detail/xmos/xl232-512-fb374-i40 |

# 1 xCORE Multicore Microcontrollers

The xCORE200 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.

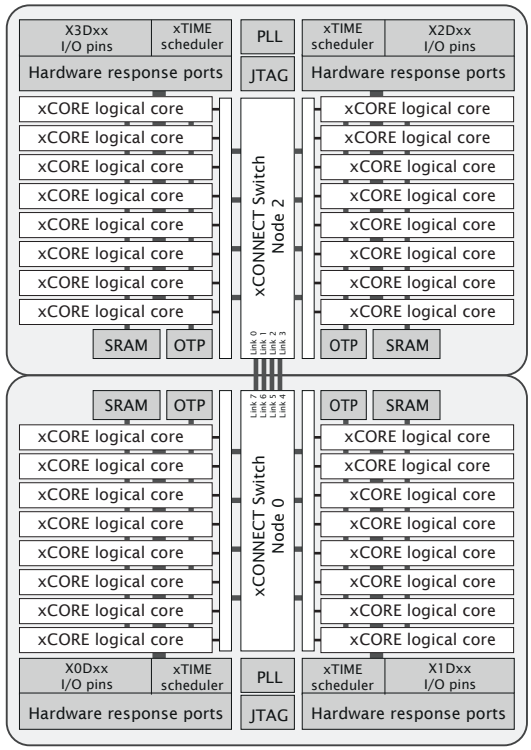


**Figure 1:** XL232-512-FB374 block diagram

Key features of the XL232-512-FB374 include:

- **Tiles**: Devices consist of one or more xCORE tiles. Each tile contains between five 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 6.1
- **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 6.2

- **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 6.5
- **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 6.6
- **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 6.3
- **Clock blocks** xCORE devices include a set of programmable clock blocks that can be used to govern the rate at which ports execute. Section 6.4
- **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 9
- **PLL** The PLL is used to create a high-speed processor clock given a low speed external oscillator. Section 7
- **JTAG** The JTAG module can be used for loading programs, boundary scan testing, in-circuit source-level debugging and programming the OTP memory. Section 10

## 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.

| I/O pins (176) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Signal** | **Function** | | | | | | **Type** | **Properties** |
| X0D00 | | $1A^{0}$ | | | | | I/O | IO, PD |
| X0D01 | $X_0L3^{2}_{out}$ | $1B^{0}$ | | | | | I/O | IO, PD |
| X0D02 | | | $4A^{0}$ | $8A^{0}$ | $16A^{0}$ | $32A^{20}$ | I/O | IO, PD |
| X0D03 | | | $4A^{1}$ | $8A^{1}$ | $16A^{1}$ | $32A^{21}$ | I/O | IO, PD |
| X0D04 | | | $4B^{0}$ | $8A^{2}$ | $16A^{2}$ | $32A^{22}$ | I/O | IO, PD |
| X0D05 | | | $4B^{1}$ | $8A^{3}$ | $16A^{3}$ | $32A^{23}$ | I/O | IO, PD |
| X0D06 | | | $4B^{2}$ | $8A^{4}$ | $16A^{4}$ | $32A^{24}$ | I/O | IO, PD |
| X0D07 | | | $4B^{3}$ | $8A^{5}$ | $16A^{5}$ | $32A^{25}$ | I/O | IO, PD |
| X0D08 | | | $4A^{2}$ | $8A^{6}$ | $16A^{6}$ | $32A^{26}$ | I/O | IO, PD |
| X0D09 | | | $4A^{3}$ | $8A^{7}$ | $16A^{7}$ | $32A^{27}$ | I/O | IO, PD |
| X0D10 | $X_0L3^{3}_{out}$ | $1C^{0}$ | | | | | I/O | IO, PD |
| X0D11 | | $1D^{0}$ | | | | | I/O | IO, PD |
| X0D12 | | $1E^{0}$ | | | | | I/O | IO, PD |
| X0D13 | | $1F^{0}$ | | | | | I/O | IO, PD |
| X0D14 | | | $4C^{0}$ | $8B^{0}$ | $16A^{8}$ | $32A^{28}$ | I/O | IO, PD |
| X0D15 | | | $4C^{1}$ | $8B^{1}$ | $16A^{9}$ | $32A^{29}$ | I/O | IO, PD |
| X0D20 | | | $4C^{2}$ | $8B^{6}$ | $16A^{14}$ | $32A^{30}$ | I/O | IO, PD |
| X0D21 | | | $4C^{3}$ | $8B^{7}$ | $16A^{15}$ | $32A^{31}$ | I/O | IO, PD |
| X0D22 | | $1G^{0}$ | | | | | I/O | IO, PD |
| X0D23 | | $1H^{0}$ | | | | | I/O | IO, PD |
| X0D28 | | | $4F^{0}$ | $8C^{2}$ | $16B^{2}$ | | I/O | IO, PD |
| X0D29 | | | $4F^{1}$ | $8C^{3}$ | $16B^{3}$ | | I/O | IO, PD |
| X0D30 | | | $4F^{2}$ | $8C^{4}$ | $16B^{4}$ | | I/O | IO, PD |
| X0D31 | | | $4F^{3}$ | $8C^{5}$ | $16B^{5}$ | | I/O | IO, PD |
| X0D32 | | | $4E^{2}$ | $8C^{6}$ | $16B^{6}$ | | I/O | IO, PD |
| X0D33 | | | $4E^{3}$ | $8C^{7}$ | $16B^{7}$ | | I/O | IO, PD |
| X0D36 | | $1M^{0}$ | | $8D^{0}$ | $16B^{8}$ | | I/O | IO, PD |
| X0D37 | $X_0L0^{4}_{in}$ | $1N^{0}$ | | $8D^{1}$ | $16B^{9}$ | | I/O | IO, PD |
| X0D38 | $X_0L0^{3}_{in}$ | $1O^{0}$ | | $8D^{2}$ | $16B^{10}$ | | I/O | IO, PD |
| X0D39 | $X_0L0^{2}_{in}$ | $1P^{0}$ | | $8D^{3}$ | $16B^{11}$ | | I/O | IO, PD |
| X0D40 | $X_0L0^{1}_{in}$ | | | $8D^{4}$ | $16B^{12}$ | | I/O | IO, PD |
| X0D41 | $X_0L0^{0}_{in}$ | | | $8D^{5}$ | $16B^{13}$ | | I/O | IO, PD |
| X0D42 | $X_0L0^{0}_{out}$ | | | $8D^{6}$ | $16B^{14}$ | | I/O | IO, PD |
| X0D43 | $X_0L0^{1}_{out}$ | | | $8D^{7}$ | $16B^{15}$ | | I/O | IO, PD |
| X1D10 | | $1C^{0}$ | | | | | I/O | IOT, PD |
| X1D11 | | $1D^{0}$ | | | | | I/O | IOT, PD |
| X1D12 | | $1E^{0}$ | | | | | I/O | IO, PD |
| X1D13 | | $1F^{0}$ | | | | | I/O | IO, PD |
| X1D14 | | | $4C^{0}$ | $8B^{0}$ | $16A^{8}$ | $32A^{28}$ | I/O | IO, PD |
| X1D15 | | | $4C^{1}$ | $8B^{1}$ | $16A^{9}$ | $32A^{29}$ | I/O | IO, PD |
| X1D16 | $X_0L3^{1}_{in}$ | | $4D^{0}$ | $8B^{2}$ | $16A^{10}$ | | I/O | IO, PD |

(continued)

| Signal | Function | | | | | Type | Properties |
|---|---|---|---|---|---|---|---|
| X1D17 | $X_0L3^0_{in}$ | | 4D[1] | 8B[3] | 16A[11] | I/O | IO, PD |
| X1D18 | $X_0L3^0_{out}$ | | 4D[2] | 8B[4] | 16A[12] | I/O | IO, PD |
| X1D19 | $X_0L3^1_{out}$ | | 4D[3] | 8B[5] | 16A[13] | I/O | IO, PD |
| X1D20 | | | 4C[2] | 8B[6] | 16A[14] | 32A[30] | I/O | IO, PD |
| X1D21 | | | 4C[3] | 8B[7] | 16A[15] | 32A[31] | I/O | IO, PD |
| X1D22 | $X_0L3^4_{out}$ | 1G[0] | | | | | I/O | IO, PD |
| X1D23 | | 1H[0] | | | | | I/O | IO, PD |
| X1D24 | | 1I[0] | | | | | I/O | IO, PD |
| X1D25 | | 1J[0] | | | | | I/O | IO, PD |
| X1D26 | | | 4E[0] | 8C[0] | 16B[0] | | I/O | IOT, PD |
| X1D27 | | | 4E[1] | 8C[1] | 16B[1] | | I/O | IOT, PD |
| X1D28 | | | 4F[0] | 8C[2] | 16B[2] | | I/O | IOT, PD |
| X1D29 | | | 4F[1] | 8C[3] | 16B[3] | | I/O | IOT, PD |
| X1D30 | | | 4F[2] | 8C[4] | 16B[4] | | I/O | IOT, PD |
| X1D31 | | | 4F[3] | 8C[5] | 16B[5] | | I/O | IOT, PD |
| X1D32 | | | 4E[2] | 8C[6] | 16B[6] | | I/O | IOT, PD |
| X1D33 | | | 4E[3] | 8C[7] | 16B[7] | | I/O | IOT, PD |
| X1D34 | $X_0L0^2_{out}$ | 1K[0] | | | | | I/O | IO, PD |
| X1D35 | $X_0L0^3_{out}$ | 1L[0] | | | | | I/O | IO, PD |
| X1D36 | $X_0L0^4_{out}$ | 1M[0] | | 8D[0] | 16B[8] | | I/O | IO, PD |
| X1D37 | $X_0L3^4_{in}$ | 1N[0] | | 8D[1] | 16B[9] | | I/O | IO, PD |
| X1D38 | $X_0L3^3_{in}$ | 1O[0] | | 8D[2] | 16B[10] | | I/O | IO, PD |
| X1D39 | $X_0L3^2_{in}$ | 1P[0] | | 8D[3] | 16B[11] | | I/O | IO, PD |
| X1D40 | | | | 8D[4] | 16B[12] | | I/O | IOT, PD |
| X1D41 | | | | 8D[5] | 16B[13] | | I/O | IOT, PD |
| X1D42 | | | | 8D[6] | 16B[14] | | I/O | IOT, PD |
| X1D43 | | | | 8D[7] | 16B[15] | | I/O | IOT, PD |
| X1D49 | $X_0L1^4_{in}$ | | | | | 32A[0] | I/O | IO, PD |
| X1D50 | $X_0L1^3_{in}$ | | | | | 32A[1] | I/O | IO, PD |
| X1D51 | $X_0L1^2_{in}$ | | | | | 32A[2] | I/O | IO, PD |
| X1D52 | $X_0L1^1_{in}$ | | | | | 32A[3] | I/O | IO, PD |
| X1D53 | $X_0L1^0_{in}$ | | | | | 32A[4] | I/O | IO, PD |
| X1D54 | $X_0L1^0_{out}$ | | | | | 32A[5] | I/O | IO, PD |
| X1D55 | $X_0L1^1_{out}$ | | | | | 32A[6] | I/O | IO, PD |
| X1D56 | $X_0L1^2_{out}$ | | | | | 32A[7] | I/O | IO, PD |
| X1D57 | $X_0L1^3_{out}$ | | | | | 32A[8] | I/O | IO, PD |
| X1D58 | $X_0L1^4_{out}$ | | | | | 32A[9] | I/O | IO, PD |
| X1D61 | $X_0L2^4_{in}$ | | | | | 32A[10] | I/O | IO, PD |
| X1D62 | $X_0L2^3_{in}$ | | | | | 32A[11] | I/O | IO, PD |
| X1D63 | $X_0L2^2_{in}$ | | | | | 32A[12] | I/O | IO, PD |
| X1D64 | $X_0L2^1_{in}$ | | | | | 32A[13] | I/O | IO, PD |
| X1D65 | $X_0L2^0_{in}$ | | | | | 32A[14] | I/O | IO, PD |
| X1D66 | $X_0L2^0_{out}$ | | | | | 32A[15] | I/O | IO, PD |

(continued)

# 5 Example Application Diagram

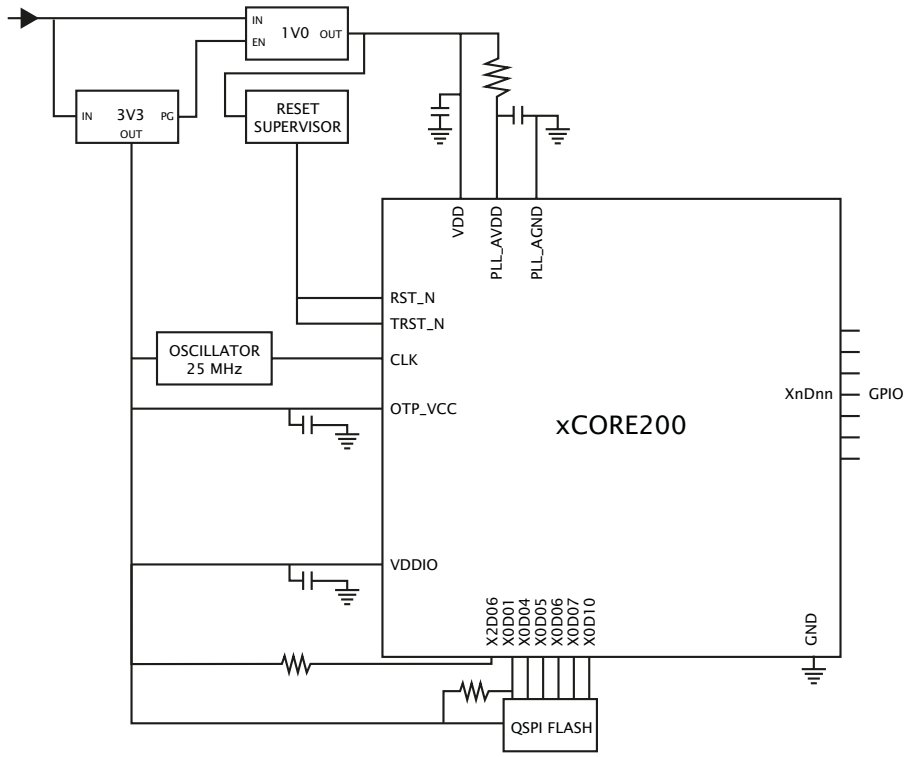


**Figure 2:** Simplified Reference Schematic

- see Section 11 for details on the power supplies and PCB design

# 6 Product Overview

The XL232-512-FB374 is a powerful device that consists of four xCORE Tiles, each comprising a flexible logical processing cores with tightly integrated I/O and on-chip memory.

## 6.1 Logical cores

Each tile has 8 active logical cores, which issue instructions down a shared five-stage pipeline. Instructions from the active cores are issued round-robin. If up to five logical cores are active, each core is allocated a fifth of the processing cycles. If more than five 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.

**Figure 3:** Logical core performance

| Speed grade | MIPS | Frequency | Minimum MIPS per core (for *n* cores) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 20 | 2000 MIPS | 500 MHz | 100 | 100 | 100 | 100 | 100 | 83 | 71 | 63 |

There is no way that the performance of a logical core can be reduced below these predicted levels (unless *priority threads* are used: in this case the guaranteed minimum performance is computed based on the number of priority threads as defined in the architecture manual). 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 five logical cores, the performance of each core is often higher than the predicted minimum but cannot be guaranteed.

The logical cores are triggered by events instead of interrupts and run to completion. A logical core can be paused to wait for an event.

## 6.2 xTIME scheduler

The xTIME scheduler handles the events generated by xCORE Tile resources, such as channel ends, timers and I/O pins. It ensures that all events are serviced and synchronized, without the need for an RTOS. Events that occur at the I/O pins are handled by the Hardware-Response ports and fed directly to the appropriate xCORE Tile. An xCORE Tile can also choose to wait for a specified time to elapse, or for data to become available on a channel.

Tasks do not need to be prioritised as each of them runs on their own logical xCORE. It is possible to share a set of low priority tasks on a single core using cooperative multitasking.

## 6.3 Hardware Response Ports

Hardware Response ports connect an xCORE tile to one or more physical pins and as such define the interface between hardware attached to the XL232-512-FB374, and the software running on it. A combination of 1bit, 4bit, 8bit, 16bit and 32bit

Figure 7 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 during and after deassertion of the system reset.

If a different tile frequency is required (eg, 500 MHz), then the PLL must be reprogrammed after boot to provide the required tile frequency. The XMOS tools perform this operation by default. Further details on configuring the clock can be found in the xCORE-200 Clock Frequency Control document.

# 8 Boot Procedure

The device is kept in reset by driving RST_N low. When in reset, all GPIO pins have a pull-down enabled. When the device is taken out of reset by releasing RST_N the processor starts its internal reset process. After 15-150 $\mu s$ (depending on the input clock) the processor boots.

Pin X2D06 must be pulled high with an external pull-up whilst the chip comes out of reset, to ensure that tile 2 will boot from link. X2D04, X2D05, and X2D07 should be kept low whilst the chip comes out of reset.

The xCORE Tile boot procedure is illustrated in Figure 8. If bit 5 of the security register (*see* §9.1) is set, the device boots from OTP. To get a high value, a 3K3 pull-up resistor should be strapped onto the pin. To assure a low value, a pull-down resistor is required if other external devices are connected to this port.

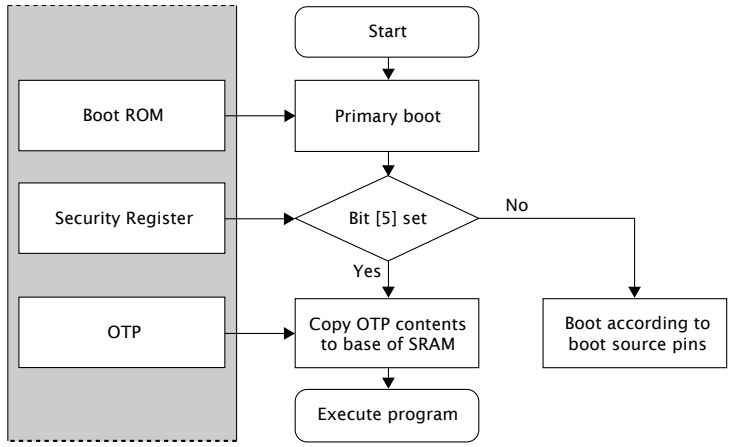


**Figure 8:** Boot procedure

## 8.2 Boot from SPI master

If set to boot from SPI master, the processor enables the four pins specified in Figure 11, 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 11:** SPI master 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.

## 8.3 Boot from SPI slave

If set to boot from SPI slave, the processor enables the three pins specified in Figure 12 and expects a boot image to be clocked in. The supported clock polarity and phase are 0/0 and 1/1.

**Figure 12:** SPI slave pins

| Pin | Signal | Description |
|---|---|---|
| X0D00 | 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*. 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.

## 8.4 Boot from xConnect Link

If set to boot from an xConnect Link, the processor enables its link(s) around 2 us after the boot process starts. Enabling the Link switches off the pull-down resistors on the link, drives all the TX wires low (the initial state for the Link), and monitors the RX pins for boot-traffic; they 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.

### 8.5 Boot from OTP

If an xCORE tile is set to use secure boot (see Figure 8), 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.

### 8.6 Security register

The security register enables security features on the xCORE tile. The features shown in Figure 13 provide a strong level of protection and are sufficient for providing strong IP security.

## 9 Memory

### 9.1 OTP

Each 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.

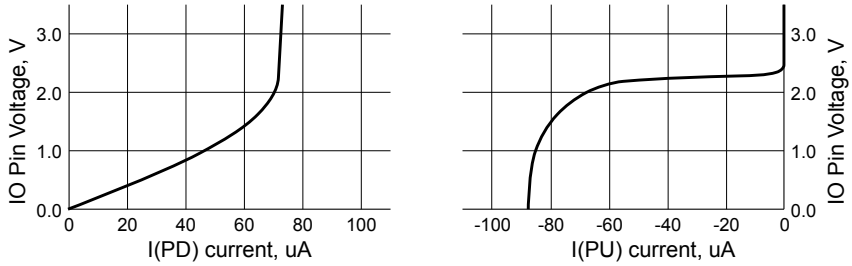


**Figure 19:** Typical internal pull-down and pull-up currents

## 12.3 ESD Stress Voltage

**Figure 20:** ESD stress voltage

| Symbol | Parameter | MIN | TYP | MAX | UNITS | Notes |
|---|---|---|---|---|---|---|
| HBM | Human body model | -2.00 | | 2.00 | KV | |
| CDM | Charged Device Model | -500 | | 500 | V | |

## 12.4 Reset Timing

**Figure 21:** Reset timing

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

A Shows the time taken to start booting after RST_N has gone high.

## 12.5 Power Consumption

**Figure 22:** xCORE Tile currents

| Symbol | Parameter | MIN | TYP | MAX | UNITS | Notes |
|---|---|---|---|---|---|---|
| I(DDCQ) | Quiescent VDD current | | 90 | | mA | A, B, C |
| PD | Tile power dissipation | | 325 | | µW/MIPS | A, D, E, F |
| IDD | Active VDD current | | 1140 | 1400 | mA | A, G |
| I(ADDPLL) | PLL_AVDD current | | 5 | 7 | mA | H |
| I(VDD33) | VDD33 current | | 53.4 | | mA | I |
| I(USB_VDD) | USB_VDD current | | 16.6 | | mA | J |

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, 500 MHz, average device resource usage.

H PLL_AVDD = 1.0 V

I HS mode transmitting while driving all 0's data (constant JKJK on DP/DM). Loading of 10 pF. Transfers do not include any interpacket delay.

J HS receive mode; no traffic.

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,

## 12.6 Clock

**Figure 23:** Clock

| Symbol | Parameter | MIN | TYP | MAX | UNITS | Notes |
|---|---|---|---|---|---|---|
| f | Frequency | 3.25 | 24 | 100 | MHz | |
| SR | Slew rate | 0.10 | | | V/ns | |
| TJ(LT) | Long term jitter (pk-pk) | | | 2 | % | A |
| f(MAX) | Processor clock frequency | | | 500 | MHz | B |

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,

**0x09:** Ring Oscillator Value

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31:16 | RO | - | Reserved |
| 15:0 | RO | 0 | 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.

**0x0A:** Ring Oscillator Value

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

### B.11 RAM size: 0x0C

The size of the RAM in bytes

**0x0C:** RAM size

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31:2 | RO | | Most significant 16 bits of all addresses. |
| 1:0 | RO | - | Reserved |

### B.12 Debug SSR: 0x10

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

0x13: DGETREG operand 1

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31:8 | RO | - | Reserved |
| 7:0 | DRW | | Thread number to be read |

### B.16 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.17 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 | | Number of the hardware breakpoint/watchpoint which caused the interrupt (always 0 for =HOST= and =DCALL=). If multiple breakpoints/watchpoints trigger at once, the lowest number is taken. |
| 15:8 | DRW | | Number of thread which caused the debug interrupt (always 0 in the case of =HOST=). |
| 7:3 | RO | - | Reserved |
| 2:0 | DRW | 0 | Indicates the cause of the debug interrupt<br>1: Host initiated a debug interrupt through JTAG<br>2: Program executed a DCALL instruction<br>3: Instruction breakpoint<br>4: Data watch point<br>5: Resource watch point |

### B.18 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 countains the resource identifier.

0x07: Security configuration

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31 | CRO | | Disables write permission on this register |
| 30:15 | RO | - | Reserved |
| 14 | CRO | | Disable access to XCore's global debug |
| 13 | RO | - | Reserved |
| 12 | CRO | | lock all OTP sectors |
| 11:8 | CRO | | lock bit for each OTP sector |
| 7 | CRO | | Enable OTP reduanacy |
| 6 | RO | - | Reserved |
| 5 | CRO | | Override boot mode and read boot image from OTP |
| 4 | CRO | | Disable JTAG access to the PLL/BOOT configuration registers |
| 3:1 | RO | - | Reserved |
| 0 | CRO | | Disable access to XCore's JTAG debug TAP |

## C.8 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.9 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 | CRO | | Value. |

## C.10 PC of logical core 1: 0x41

Value of the PC of logical core 1.

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

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31:0 | CRO | | Value. |

## C.11 PC of logical core 2: 0x42

Value of the PC of logical core 2.

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

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31:0 | CRO | | Value. |

## C.12 PC of logical core 3: 0x43

Value of the PC of logical core 3.

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

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31:0 | CRO | | Value. |

## C.13 PC of logical core 4: 0x44

Value of the PC of logical core 4.

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

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31:0 | CRO | | Value. |

## C.14 PC of logical core 5: 0x45

Value of the PC of logical core 5.

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

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31:0 | CRO | | Value. |

### C.15 PC of logical core 6: 0x46

Value of the PC of logical core 6.

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

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31:0 | CRO | | Value. |

### C.16 PC of logical core 7: 0x47

Value of the PC of logical core 7.

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

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31:0 | CRO | | Value. |

### C.17 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 | CRO | | Value. |

### C.18 SR of logical core 1: 0x61

Value of the SR of logical core 1

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

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31:0 | CRO | | Value. |

### C.19 SR of logical core 2: 0x62

Value of the SR of logical core 2

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

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31:0 | CRO | | Value. |

## C.20 SR of logical core 3: 0x63

Value of the SR of logical core 3

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

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31:0 | CRO | | Value. |

## C.21 SR of logical core 4: 0x64

Value of the SR of logical core 4

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

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31:0 | CRO | | Value. |

## C.22 SR of logical core 5: 0x65

Value of the SR of logical core 5

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

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31:0 | CRO | | Value. |

## C.23 SR of logical core 6: 0x66

Value of the SR of logical core 6

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

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31:0 | CRO | | Value. |

**0x0D:** Directions 8-15

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31:28 | RW | 0 | The direction for packets whose dimension is F. |
| 27:24 | RW | 0 | The direction for packets whose dimension is E. |
| 23:20 | RW | 0 | The direction for packets whose dimension is D. |
| 19:16 | RW | 0 | The direction for packets whose dimension is C. |
| 15:12 | RW | 0 | The direction for packets whose dimension is B. |
| 11:8 | RW | 0 | The direction for packets whose dimension is A. |
| 7:4 | RW | 0 | The direction for packets whose dimension is 9. |
| 3:0 | RW | 0 | The direction for packets whose dimension is 8. |

### D.12 DEBUG_N configuration, tile 0: 0x10

Configures the behavior of the DEBUG_N pin.

**0x10:** DEBUG_N configuration, tile 0

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31:2 | RO | - | Reserved |
| 1 | RW | 0 | Set 1 to enable GlobalDebug to generate debug request to XCore. |
| 0 | RW | 0 | Set 1 to enable inDebug bit to drive GlobalDebug. |

### D.13 DEBUG_N configuration, tile 1: 0x11

Configures the behavior of the DEBUG_N pin.

**0x11:** DEBUG_N configuration, tile 1

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31:2 | RO | - | Reserved |
| 1 | RW | 0 | Set 1 to enable GlobalDebug to generate debug request to XCore. |
| 0 | RW | 0 | Set 1 to enable inDebug bit to drive GlobalDebug. |

### D.14 Debug source: 0x1F

Contains the source of the most recent debug event.

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31:26 | RO | - | Reserved |
| 25:24 | RO | | Identify the SRC_TARGET type 0 - SLink, 1 - PLink, 2 - SSCTL, 3 - Undefine. |
| 23:16 | RO | | When the link is in use, this is the destination link number to which all packets are sent. |
| 15:6 | RO | - | Reserved |
| 5:4 | RW | 0 | Determines the network to which this link belongs, reset as 0. |
| 3 | RO | - | Reserved |
| 2 | RO | | 1 when the current packet is considered junk and will be thrown away. |
| 1 | RO | | 1 when the dest side of the link is in use. |
| 0 | RO | | 1 when the source side of the link is in use. |

**0x40 .. 0x47:** PLink status and network

## D.17 Link configuration and initialization: 0x80 .. 0x88

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. The registers control links 0..7.

| Bits | Perm | Init | Description |
|---|---|---|---|
| 31 | RW | | Write to this bit with '1' will enable the XLink, writing '0' will disable it. This bit controls the muxing of ports with overlapping xlinks. |
| 30 | RW | 0 | 0: operate in 2 wire mode; 1: operate in 5 wire mode |
| 29:28 | RO | - | Reserved |
| 27 | RO | | Rx buffer overflow or illegal token encoding received. |
| 26 | RO | 0 | This end of the xlink has issued credit to allow the remote end to transmit |
| 25 | RO | 0 | This end of the xlink has credit to allow it to transmit. |
| 24 | WO | | Clear this end of the xlink's credit and issue a HELLO token. |
| 23 | WO | | Reset the receiver. The next symbol that is detected will be the first symbol in a token. |
| 22 | RO | - | Reserved |
| 21:11 | RW | 0 | Specify min. number of idle system clocks between two continuous symbols witin a transmit token -1. |
| 10:0 | RW | 0 | Specify min. number of idle system clocks between two continuous transmit tokens -1. |

**0x80 .. 0x88:** Link configuration and initialization

# H Associated Design Documentation

| Document Title | Information | Document Number |
|---|---|---|
| Estimating Power Consumption For XS1-L Devices | Power consumption | X4271 |
| Programming XC on XMOS Devices | Timers, ports, clocks, cores and channels | X9577 |
| xTIMEcomposer User Guide | Compilers, assembler and linker/mapper<br>Timing analyzer, xScope, debugger<br>Flash and OTP programming utilities | X3766 |

# I Related Documentation

| Document Title | Information | Document Number |
|---|---|---|
| The XMOS XS1 Architecture | ISA manual | X7879 |
| XS1 Port I/O Timing | Port timings | X5821 |
| xCONNECT Architecture | Link, switch and system information | X4249 |
| XS1-L Link Performance and Design Guidelines | Link timings | X2999 |
| XS1-L Clock Frequency Control | Advanced clock control | X1433 |
| XS1-L Active Power Conservation | Low-power mode during idle | X7411 |