



Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!



Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832

Email & Skype: info@chipsmall.com Web: www.chipsmall.com

Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China



XU210-512-FB236 Datasheet

Table of Contents

1	xCORE Multicore Microcontrollers	2
2	XU210-512-FB236 Features	4
3	Pin Configuration	5
4	Signal Description	6
5	Example Application Diagram	11
6	Product Overview	12
7	PLL	15
8	Boot Procedure	16
9	Memory	19
10	USB PHY	20
11	JTAG	22
12	Board Integration	23
13	DC and Switching Characteristics	27
14	Package Information	31
15	Ordering Information	32
	Appendices	33
A	Configuration of the XU210-512-FB236	33
B	Processor Status Configuration	36
C	Tile Configuration	47
D	Node Configuration	55
E	USB Node Configuration	63
F	USB PHY Configuration	65
G	JTAG, xSCOPE and Debugging	72
H	Schematics Design Check List	74
I	PCB Layout Design Check List	76
J	Associated Design Documentation	77
K	Related Documentation	77
L	Revision History	78

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 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.

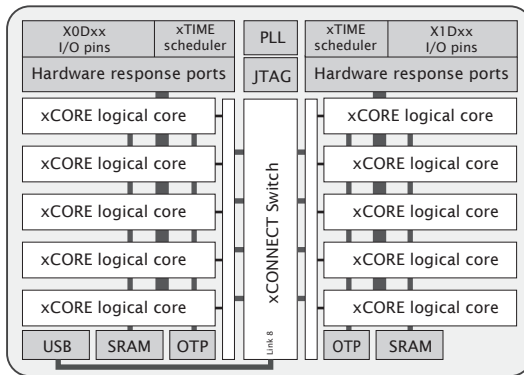


Figure 1:
XU210-512-
FB236 block
diagram

Key features of the XU210-512-FB236 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](#)
- ▶ **USB** The USB PHY provides High-Speed and Full-Speed, device, host, and on-the-go functionality. Data is communicated through ports on the digital node. A library is provided to implement USB device functionality. Section [10](#)
- ▶ **JTAG** The JTAG module can be used for loading programs, boundary scan testing, in-circuit source-level debugging and programming the OTP memory. Section [11](#)

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](#).

2 XU210-512-FB236 Features

▶ **Multicore Microcontroller with Advanced Multi-Core RISC Architecture**

- 10 real-time logical cores on 2 xCORE tiles
- Cores share up to 1000 MIPS
 - Up to 2000 MIPS in dual issue mode
- Each logical core has:
 - Guaranteed throughput of between 1/5 and 1/5 of tile MIPS
 - 16x32bit dedicated registers
- 167 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

▶ **USB PHY, fully compliant with USB 2.0 specification**

▶ **Programmable I/O**

- 128 general-purpose I/O pins, configurable as input or output
 - Up to 32 x 1bit port, 12 x 4bit port, 8 x 8bit port, 4 x 16bit port, 2 x 32bit port
 - 8 xCONNECT links
- Port sampling rates of up to 60 MHz with respect to an external clock
- 64 channel ends (32 per tile) for communication with other cores, on or off-chip

▶ **Memory**

- 512KB internal single-cycle SRAM (max 256KB per tile) for code and data storage
- 16KB internal OTP (max 8KB per tile) for application boot code

▶ **Hardware resources**

- 12 clock blocks (6 per tile)
- 20 timers (10 per tile)
- 8 locks (4 per tile)

▶ **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**

- 20: 1000 MIPS

▶ **Power Consumption**

- 570 mA (typical)

▶ **236-pin FBGA package 0.5 mm pitch**

3 Pin Configuration

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	GND	VDDIOL	VDDIOL		TCK	CLK		4F X1D31	4F X1D29		8D X1D41	OTP VCC ⁻		NC	MODE[0]		4F X0D29	VDDIOR	GND
B	1B X0D36	VDDIOL	VDDIOL	TDO	TMS	TRST _N	4E X1D33	4E X1D32	4F X1D28	4E X1D26	8D X1D42	OTP VCC ⁻	NC	NC	MODE[1]	4E X0D33	4E X0D32	VDDIOR	VDDIOR
C	1N X0D37	1O X0D38	VDDIOL	TDI	DEBUG N	RST _N	1C X1D10	1D X1D11	4F X1D30	4E X1D27	8D X1D43	8D X1D40	NC	NC	4F X0D31	4F X0D30	4F X0D28	4E X0D26	4E X0D27
D		1P X0D39	8D X0D40														1K X0D34	1L X0D35	
E	8D X0D43	8D X0D42	8D X0D41														1J X0D25	1I X0D24	1B X1D01
F	1K X1D34	1M X1D35	1M X1D36				NC	VDD	VDD	VDDIOT	VDD	VDD	PLL AVDD	PLL AGND			4A X1D08	4F X1D09	1A X1D00
G		32A X1D49	32A X1D50			VDD		GND		GND		GND		VDD			32A X0D69	32A X0D70	
H	32A X1D53	32A X1D52	32A X1D51			VDD	GND	GND	GND	GND	GND	GND	GND	VDD			32A X0D68	32A X0D67	32A X0D66
J	32A X1D54	32A X1D55	32A X1D56			VDD		GND		GND		GND		VDD			32A X0D63	32A X0D64	32A X0D65
K		32A X1D58	32A X1D57			VDD	GND	GND	GND	GND	GND	GND	GND	VDD			32A X0D62	32A X0D61	
L	32A X1D63	32A X1D62	32A X1D61			VDD		GND		GND		GND		VDD			32A X0D58	32A X0D57	32A X0D56
M	32A X1D64	32A X1D65	32A X1D66			VDD	GND	GND	GND	GND	GND	GND	GND	VDD			32A X0D53	32A X0D54	32A X0D55
N		32A X1D67	32A X1D68			VDD		GND		GND		GND		VDD			32A X0D51	32A X0D52	
P	32A X1D70	32A X1D69	1N X1D37			VDD	VDD	VDD	USB VDD ⁻	USB VDD ⁻	VDD	VDD	VDD	NC			4B X1D07	32A X0D50	32A X0D49
R	1D X1D38	1P X1D39	4D X1D17														4A X1D03	4B X1D05	4B X1D06
T		4D X1D16	4D X1D18														4B X1D02	4B X1D04	
U	1C X0D10	1B X0D01	4D X1D19	1A X0D00	1D X0D11	4B X0D07	1E X1D12	USB VDD33	USB VBUS	USB ID ⁻	USB VSSAC	NC	1I X1D24	1G X0D22	1F X0D13	1H X0D23	4D X0D19	4D X0D18	4D X0D17
V	1G X1D22	VDDIOL	VDDIOL	4B X0D04	4B X0D06	4A X0D03	4A X0D08	4A X0D09	USB DM	USB DP ⁻	4C X1D21	4C X1D14	1J X1D25	4C X0D21	4C X0D14	1E X0D12	VDDIOR	VDDIOR	4D X0D16
W	GND	VDDIOL	X1D23		4B X0D05	4A X0D02		1P X1D13	USB RTUNE		4C X1D20	4C X1D15		4C X0D20	4C X0D15		VDDIOR	VDDIOR	GND

4 Signal Description

This section lists the signals and I/O pins available on the XU210-512-FB236. The device provides a combination of 1bit, 4bit, 8bit and 16bit ports, as well as wider ports that are fully or partially (gray) bonded out. All pins of a port provide either output or input, but signals in different directions cannot be mapped onto the same port.

Pins may have one or more of the following properties:

- ▶ PD/PU: The IO pin has a weak pull-down or pull-up resistor. The resistor is enabled during and after reset. Enabling a link or port that uses the pin disables the resistor. Thereafter, the resistor can be enabled or disabled under software control. The resistor is designed to ensure defined logic input state for unconnected pins. It should not be used to pull external circuitry. Note that the resistors are highly non-linear and only a maximum pull current is specified in Section 13.2.
- ▶ ST: The IO pin has a Schmitt Trigger on its input.
- ▶ IOL/IOT/IOR: The IO pin is powered from VDDIOL, VDDIOT, and VDDIOR respectively

Power pins (11)			
Signal	Function	Type	Properties
GND	Digital ground	GND	
OTP_VCC	OTP power supply	PWR	
PLL_AGND	Analog ground for PLL	PWR	
PLL_AVDD	Analog PLL power	PWR	
USB_VDD	Digital tile power	PWR	
USB_VDD33	USB Analog power	PWR	
USB_VSSAC	USB analog ground	GND	
VDD	Digital tile power	PWR	
VDDIOL	Digital I/O power (left)	PWR	
VDDIOR	Digital I/O power (right)	PWR	
VDDIOT	Digital I/O power (top)	PWR	

JTAG pins (6)			
Signal	Function	Type	Properties
RST_N	Global reset input	Input	IOL, PU, ST
TCK	Test clock	Input	IOL, PD, ST
TDI	Test data input	Input	IOL, PU
TDO	Test data output	Output	IOL, PD
TMS	Test mode select	Input	IOL, PU
TRST_N	Test reset input	Input	IOL, PU, ST

I/O pins (128)					
Signal	Function	Type	Properties		
X0D00	1A ⁰	I/O	IOL, PD		
X0D01	X ₀ L3 _{out} ² 1B ⁰	I/O	IOL, PD		
X0D02	4A ⁰ 8A ⁰ 16A ⁰ 32A ²⁰	I/O	IOL, PD		
X0D03	4A ¹ 8A ¹ 16A ¹ 32A ²¹	I/O	IOL, PD		
X0D04	4B ⁰ 8A ² 16A ² 32A ²²	I/O	IOL, PD		
X0D05	4B ¹ 8A ³ 16A ³ 32A ²³	I/O	IOL, PD		
X0D06	4B ² 8A ⁴ 16A ⁴ 32A ²⁴	I/O	IOL, PD		
X0D07	4B ³ 8A ⁵ 16A ⁵ 32A ²⁵	I/O	IOL, PD		
X0D08	4A ² 8A ⁶ 16A ⁶ 32A ²⁶	I/O	IOL, PD		
X0D09	4A ³ 8A ⁷ 16A ⁷ 32A ²⁷	I/O	IOL, PD		
X0D10	X ₀ L3 _{out} ³ 1C ⁰	I/O	IOL, PD		
X0D11	1D ⁰	I/O	IOL, PD		
X0D12	1E ⁰	I/O	IOR, PD		
X0D13	1F ⁰	I/O	IOR, PD		
X0D14	4C ⁰ 8B ⁰ 16A ⁸ 32A ²⁸	I/O	IOR, PD		
X0D15	4C ¹ 8B ¹ 16A ⁹ 32A ²⁹	I/O	IOR, PD		
X0D16	X ₀ L4 _{in} ⁴ 4D ⁰ 8B ² 16A ¹⁰	I/O	IOR, PD		
X0D17	X ₀ L4 _{in} ³ 4D ¹ 8B ³ 16A ¹¹	I/O	IOR, PD		
X0D18	X ₀ L4 _{in} ² 4D ² 8B ⁴ 16A ¹²	I/O	IOR, PD		
X0D19	X ₀ L4 _{in} ¹ 4D ³ 8B ⁵ 16A ¹³	I/O	IOR, PD		
X0D20	4C ² 8B ⁶ 16A ¹⁴ 32A ³⁰	I/O	IOR, PD		
X0D21	4C ³ 8B ⁷ 16A ¹⁵ 32A ³¹	I/O	IOR, PD		
X0D22	1G ⁰	I/O	IOR, PD		
X0D23	1H ⁰	I/O	IOR, PD		
X0D24	X ₀ L7 _{in} ⁰ 1I ⁰	I/O	IOR, PD		
X0D25	X ₀ L7 _{out} ⁰ 1J ⁰	I/O	IOR, PD		
X0D26	X ₀ L7 _{out} ³ 4E ⁰ 8C ⁰ 16B ⁰	I/O	IOR, PD		
X0D27	X ₀ L7 _{out} ⁴ 4E ¹ 8C ¹ 16B ¹	I/O	IOR, PD		
X0D28	4F ⁰ 8C ² 16B ²	I/O	IOR, PD		
X0D29	4F ¹ 8C ³ 16B ³	I/O	IOR, PD		
X0D30	4F ² 8C ⁴ 16B ⁴	I/O	IOR, PD		
X0D31	4F ³ 8C ⁵ 16B ⁵	I/O	IOR, PD		
X0D32	4E ² 8C ⁶ 16B ⁶	I/O	IOR, PD		
X0D33	4E ³ 8C ⁷ 16B ⁷	I/O	IOR, PD		
X0D34	X ₀ L7 _{out} ¹ 1K ⁰	I/O	IOR, PD		
X0D35	X ₀ L7 _{out} ² 1L ⁰	I/O	IOR, PD		
X0D36	1M ⁰ 8D ⁰ 16B ⁸	I/O	IOL, PD		
X0D37	X ₀ L0 _{in} ⁴ 1N ⁰ 8D ¹ 16B ⁹	I/O	IOL, PD		
X0D38	X ₀ L0 _{in} ³ 1O ⁰ 8D ² 16B ¹⁰	I/O	IOL, PD		
X0D39	X ₀ L0 _{in} ² 1P ⁰ 8D ³ 16B ¹¹	I/O	IOL, PD		
X0D40	X ₀ L0 _{in} ¹ 8D ⁴ 16B ¹²	I/O	IOL, PD		

(continued)

Signal	Function	Type	Properties
X0D41	$X_0L0_{in}^0$ 8D ⁵ 16B ¹³	I/O	IOL, PD
X0D42	$X_0L0_{out}^0$ 8D ⁶ 16B ¹⁴	I/O	IOL, PD
X0D43	$X_0L0_{out}^0$ 8D ⁷ 16B ¹⁵	I/O	IOL, PD
X0D49	$X_0L5_{in}^4$ 32A ⁰	I/O	IOR, PD
X0D50	$X_0L5_{in}^3$ 32A ¹	I/O	IOR, PD
X0D51	$X_0L5_{in}^2$ 32A ²	I/O	IOR, PD
X0D52	$X_0L5_{in}^1$ 32A ³	I/O	IOR, PD
X0D53	$X_0L5_{in}^0$ 32A ⁴	I/O	IOR, PD
X0D54	$X_0L5_{out}^0$ 32A ⁵	I/O	IOR, PD
X0D55	$X_0L5_{out}^1$ 32A ⁶	I/O	IOR, PD
X0D56	$X_0L5_{out}^2$ 32A ⁷	I/O	IOR, PD
X0D57	$X_0L5_{out}^3$ 32A ⁸	I/O	IOR, PD
X0D58	$X_0L5_{out}^4$ 32A ⁹	I/O	IOR, PD
X0D61	$X_0L6_{in}^4$ 32A ¹⁰	I/O	IOR, PD
X0D62	$X_0L6_{in}^3$ 32A ¹¹	I/O	IOR, PD
X0D63	$X_0L6_{in}^2$ 32A ¹²	I/O	IOR, PD
X0D64	$X_0L6_{in}^1$ 32A ¹³	I/O	IOR, PD
X0D65	$X_0L6_{in}^0$ 32A ¹⁴	I/O	IOR, PD
X0D66	$X_0L6_{out}^0$ 32A ¹⁵	I/O	IOR, PD
X0D67	$X_0L6_{out}^1$ 32A ¹⁶	I/O	IOR, PD
X0D68	$X_0L6_{out}^2$ 32A ¹⁷	I/O	IOR, PD
X0D69	$X_0L6_{out}^3$ 32A ¹⁸	I/O	IOR, PD
X0D70	$X_0L6_{out}^4$ 32A ¹⁹	I/O	IOR, PD
X1D00	$X_0L7_{in}^2$ 1A ⁰	I/O	IOR, PD
X1D01	$X_0L7_{in}^1$ 1B ⁰	I/O	IOR, PD
X1D02	$X_0L4_{in}^0$ 4A ⁰ 8A ⁰ 16A ⁰ 32A ²⁰	I/O	IOR, PD
X1D03	$X_0L4_{out}^0$ 4A ¹ 8A ¹ 16A ¹ 32A ²¹	I/O	IOR, PD
X1D04	$X_0L4_{out}^1$ 4B ⁰ 8A ² 16A ² 32A ²²	I/O	IOR, PD
X1D05	$X_0L4_{out}^2$ 4B ¹ 8A ³ 16A ³ 32A ²³	I/O	IOR, PD
X1D06	$X_0L4_{out}^3$ 4B ² 8A ⁴ 16A ⁴ 32A ²⁴	I/O	IOR, PD
X1D07	$X_0L4_{out}^4$ 4B ³ 8A ⁵ 16A ⁵ 32A ²⁵	I/O	IOR, PD
X1D08	$X_0L7_{in}^4$ 4A ² 8A ⁶ 16A ⁶ 32A ²⁶	I/O	IOR, PD
X1D09	$X_0L7_{in}^3$ 4A ³ 8A ⁷ 16A ⁷ 32A ²⁷	I/O	IOR, PD
X1D10	1C ⁰	I/O	IOT, PD
X1D11	1D ⁰	I/O	IOT, PD
X1D12	1E ⁰	I/O	IOL, PD
X1D13	1F ⁰	I/O	IOL, PD
X1D14	4C ⁰ 8B ⁰ 16A ⁸ 32A ²⁸	I/O	IOR, PD
X1D15	4C ¹ 8B ¹ 16A ⁹ 32A ²⁹	I/O	IOR, PD
X1D16	$X_0L3_{in}^1$ 4D ⁰ 8B ² 16A ¹⁰	I/O	IOL, PD
X1D17	$X_0L3_{in}^0$ 4D ¹ 8B ³ 16A ¹¹	I/O	IOL, PD
X1D18	$X_0L3_{out}^0$ 4D ² 8B ⁴ 16A ¹²	I/O	IOL, PD
X1D19	$X_0L3_{out}^1$ 4D ³ 8B ⁵ 16A ¹³	I/O	IOL, PD

(continued)

Signal	Function	Type	Properties
X1D20	4C ² 8B ⁶ 16A ¹⁴ 32A ³⁰	I/O	IOR, PD
X1D21	4C ³ 8B ⁷ 16A ¹⁵ 32A ³¹	I/O	IOR, PD
X1D22	X ₀ L3 _{out} ⁴ 1G ⁰	I/O	IOL, PD
X1D23	1H ⁰	I/O	IOL, PD
X1D24	1I ⁰	I/O	IOR, PD
X1D25	1J ⁰	I/O	IOR, 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 ₀ L0 _{out} ² 1K ⁰	I/O	IOL, PD
X1D35	X ₀ L0 _{out} ³ 1L ⁰	I/O	IOL, PD
X1D36	X ₀ L0 _{out} ⁴ 1M ⁰ 8D ⁰ 16B ⁸	I/O	IOL, PD
X1D37	X ₀ L3 _{in} ⁴ 1N ⁰ 8D ¹ 16B ⁹	I/O	IOL, PD
X1D38	X ₀ L3 _{in} ³ 1O ⁰ 8D ² 16B ¹⁰	I/O	IOL, PD
X1D39	X ₀ L3 _{in} ² 1P ⁰ 8D ³ 16B ¹¹	I/O	IOL, 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 ₀ L1 _{in} ⁴ 32A ⁰	I/O	IOL, PD
X1D50	X ₀ L1 _{in} ³ 32A ¹	I/O	IOL, PD
X1D51	X ₀ L1 _{in} ² 32A ²	I/O	IOL, PD
X1D52	X ₀ L1 _{in} ¹ 32A ³	I/O	IOL, PD
X1D53	X ₀ L1 _{in} ⁰ 32A ⁴	I/O	IOL, PD
X1D54	X ₀ L1 _{out} ⁰ 32A ⁵	I/O	IOL, PD
X1D55	X ₀ L1 _{out} ¹ 32A ⁶	I/O	IOL, PD
X1D56	X ₀ L1 _{out} ² 32A ⁷	I/O	IOL, PD
X1D57	X ₀ L1 _{out} ³ 32A ⁸	I/O	IOL, PD
X1D58	X ₀ L1 _{out} ⁴ 32A ⁹	I/O	IOL, PD
X1D61	X ₀ L2 _{in} ⁴ 32A ¹⁰	I/O	IOL, PD
X1D62	X ₀ L2 _{in} ³ 32A ¹¹	I/O	IOL, PD
X1D63	X ₀ L2 _{in} ² 32A ¹²	I/O	IOL, PD
X1D64	X ₀ L2 _{in} ¹ 32A ¹³	I/O	IOL, PD
X1D65	X ₀ L2 _{in} ⁰ 32A ¹⁴	I/O	IOL, PD
X1D66	X ₀ L2 _{out} ⁰ 32A ¹⁵	I/O	IOL, PD
X1D67	X ₀ L2 _{out} ¹ 32A ¹⁶	I/O	IOL, PD
X1D68	X ₀ L2 _{out} ² 32A ¹⁷	I/O	IOL, PD
X1D69	X ₀ L2 _{out} ³ 32A ¹⁸	I/O	IOL, PD

(continued)

Signal	Function	Type	Properties
X1D70	X ₀ L ₂ ⁴ _{out} 32A ¹⁹	I/O	IOL, PD

System pins (3)			
Signal	Function	Type	Properties
CLK	PLL reference clock	Input	IOL, PD, ST
DEBUG_N	Multi-chip debug	I/O	IOL, PU
MODE[1:0]	Boot mode select	Input	PU

usb pins (5)			
Signal	Function	Type	Properties
USB_DM	USB Serial Data Inverted	I/O	
USB_DP	USB Serial Data	I/O	
USB_ID	USB Device ID (OTG) - Reserved	I/O	
USB_RTUNE	USB resistor	I/O	
USB_VBUS	USB Power Detect Pin	I/O	

5 Example Application Diagram

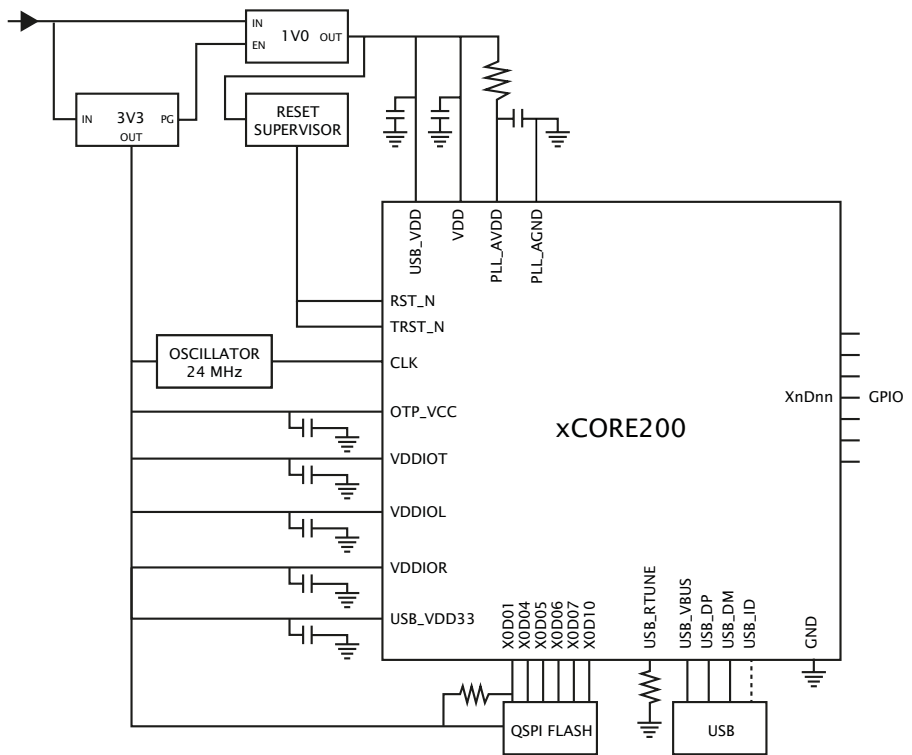


Figure 2:
Simplified
Reference
Schematic

- ▶ see Section 10 for details on the USB PHY
- ▶ see Section 12 for details on the power supplies and PCB design

6 Product Overview

The XU210-512-FB236 is a powerful device that consists of two 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 up to 5 active logical cores, which issue instructions down a shared five-stage pipeline. Instructions from the active cores are issued round-robin. Each core is allocated a fifth of the processing cycles. Figure 3 shows the guaranteed core performance.

Figure 3:
Logical core
performance

Speed grade	MIPS	Frequency	MIPS per logical core
10	1000 MIPS	500 MHz	100

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).

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 XU210-512-FB236, and the software running on it. A combination of 1bit, 4bit, 8bit, 16bit and 32bit ports are available. All pins of a port provide either output or input. Signals in different directions cannot be mapped onto the same port.

The port logic can drive its pins high or low, or it can sample the value on its pins, optionally waiting for a particular condition. Ports are accessed using dedicated instructions that are executed in a single processor cycle. xCORE-200 IO pins can

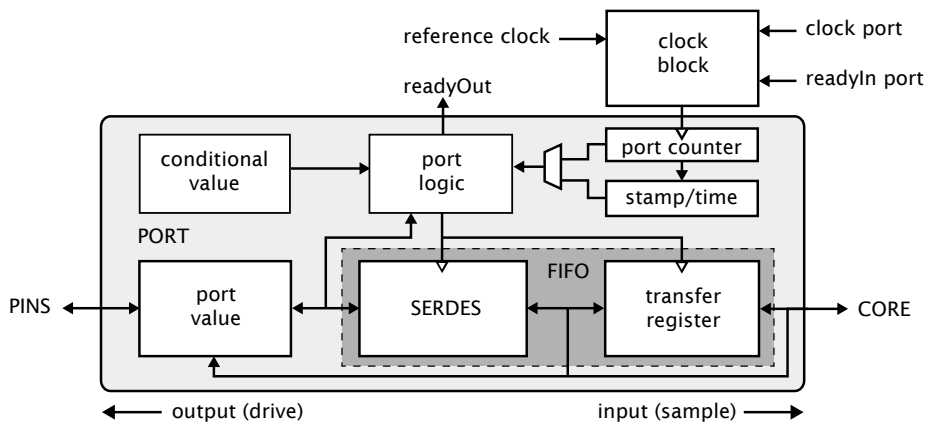


Figure 4:
Port block
diagram

be used as *open collector* outputs, where signals are driven low if a zero is output, but left high impedance if a one is output. This option is set on a per-port basis.

Data is transferred between the pins and core using a FIFO that comprises a SERDES and transfer register, providing options for serialization and buffered data.

Each port has a 16-bit counter that can be used to control the time at which data is transferred between the port value and transfer register. The counter values can be obtained at any time to find out when data was obtained, or used to delay I/O until some time in the future. The port counter value is automatically saved as a timestamp, that can be used to provide precise control of response times.

The ports and xCONNECT links are multiplexed onto the physical pins. 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.

6.4 Clock blocks

xCORE devices include a set of programmable clocks called clock blocks that can be used to govern the rate at which ports execute. Each xCORE tile has six clock blocks: the first clock block provides the tile reference clock and runs at a default frequency of 100MHz; the remaining clock blocks can be set to run at different frequencies.

A clock block can use a 1-bit port as its clock source allowing external application clocks to be used to drive the input and output interfaces. xCORE-200 clock blocks optionally divide the clock input from a 1-bit port.

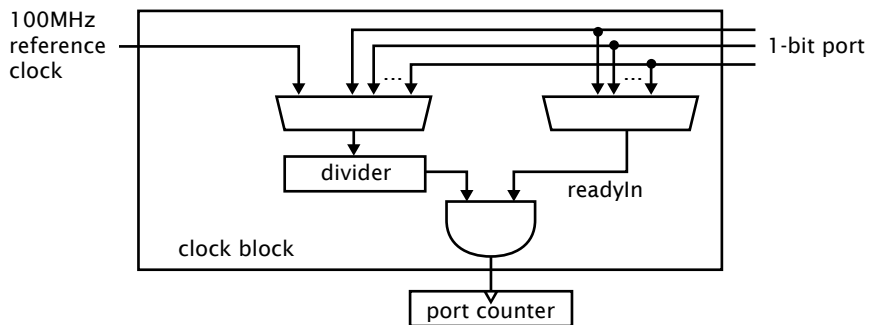


Figure 5:
Clock block
diagram

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.

6.5 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.

6.6 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.

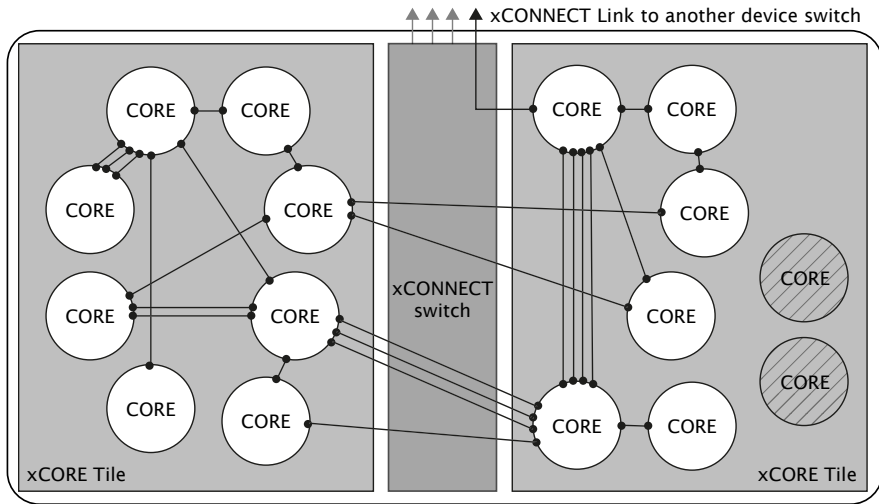


Figure 6:
Switch, links
and channel
ends

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

7 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 7:

Oscillator Frequency	MODE		Tile Boot Frequency	PLL Ratio	PLL settings		
	1	0			OD	F	R
3.25-10 MHz	0	0	130-400 MHz	40	1	159	0
9-25 MHz	1	1	144-400 MHz	16	1	63	0
25-50 MHz	1	0	167-400 MHz	8	1	31	0
50-100 MHz	0	1	196-400 MHz	4	1	15	0

Figure 7:
PLL multiplier
values and
MODE pins

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 the USB PHY is used, then either a 24 MHz or 12 MHz oscillator must be used.

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 μs (depending on the input clock) the processor boots.

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.

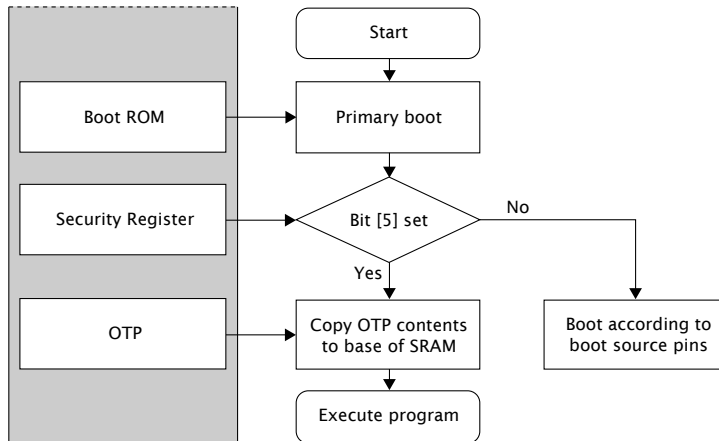


Figure 8:
Boot procedure

X0D06	X0D05	X0D04	Tile 0 boot	Tile 1 boot	Enabled links
0	0	0	QSPI master	Channel end 0	None
0	0	1	SPI master	Channel end 0	None
0	1	0	SPI slave	Channel end 0	None
0	1	1	SPI slave	SPI slave	None
1	0	0	Channel end 0	Channel end 0	XL0 (2w)
1	0	1	Channel end 0	Channel end 0	XL4-XL7 (5w)
1	1	0	Channel end 0	Channel end 0	XL1, XL2, XL5, and XL6 (5w)
1	1	1	Channel end 0	Channel end 0	XL0-XL3 (5w)

Figure 9:
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.

8.1 Boot from QSPI master

If set to boot from QSPI master, the processor enables the six pins specified in Figure 10, and drives the SPI clock at 50 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 10:
QSPI pins

Pin	Signal	Description
X0D01	SS	Slave Select
X0D04..X0D07	SPIO	Data
X0D10	SCLK	Clock

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

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

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.

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

Figure 12:
SPI slave pins

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 μ s 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.

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`.

9.2 SRAM

Each xCORE Tile integrates a single 256KBSRAM 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

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</i> §8).
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 13:
Security register features

interface, although data memory can be expanded through appropriate use of the ports.

10 USB PHY

The USB PHY provides High-Speed and Full-Speed, device, host, and on-the-go functionality. The PHY is configured through a set of peripheral registers (Appendix F), and data is communicated through ports on the digital node. A library, XUD, is provided to implement *USB-device* functionality.

The USB PHY is connected to the ports on Tile 0 and Tile 1 as shown in Figure 14. When the USB PHY is enabled on Tile 0, the ports shown can on Tile 0 only be used with the USB PHY. When the USB PHY is enabled on Tile 1, then the ports shown can on Tile 1 only be used with the USB PHY. All other IO pins and ports are unaffected. The USB PHY should not be enabled on both tiles. Two clock blocks can be used to clock the USB ports. One clock block for the TXDATA path, and one clock block for the RXDATA path. Details on how to connect those ports are documented in an application note on USB for xCORE-200.

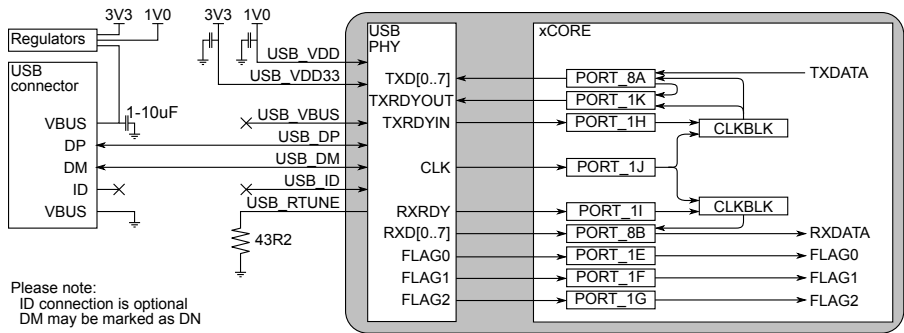


Figure 14:
Bus powered
USB-device

An external resistor of 43.2 ohm (1% tolerance) should connect USB_RTUNE to ground, as close as possible to the device.

10.1 USB VBUS

USB_VBUS need not be connected if the device is wholly powered by USB, and the device is used to implement a *USB-device*.

If you use the USB PHY to design a self-powered *USB-device*, then the device must be able detect the presence of VBus on the USB connector (so the device can disconnect its pull-up resistors from D+/D- to ensure the device does not have any voltage on the D+/D- pins when VBus is not present, “USB Back Voltage Test”). This requires USB_VBUS to be connected to the VBUS pin of the USB connector as is shown in Figure 15.

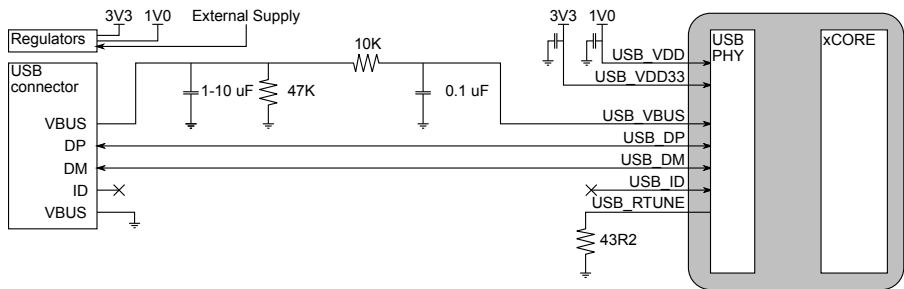


Figure 15:
Self powered
USB-device

When connecting a USB cable to the device it is possible an overvoltage transient will be present on VBus due to the inductance of the USB cable combined with the required input capacitor on VBus. The circuit in Figure 15 ensures that the transient does not damage the device. The 10k series resistor and 0.1uF capacitor ensure that any input transient is filtered and does not reach the device. The 47k resistor to ground is a bleeder resistor to discharge the input capacitor when VBus is not present. The 1-10uF input capacitor is required as part of the USB specification. A typical value would be 2.2uF to ensure the 1uF minimum requirement is met even under voltage bias conditions.

In any case, extra components (such as a ferrite bead and diodes) may be required for EMC compliance and ESD protection. Different wiring is required for USB-host and USB-OTG.

10.2 Logical Core Requirements

The XMOS XUD software component runs in a single logical core with endpoint and application cores communicating with it via a combination of channel communication and shared memory variables.

Each IN (host requests data from device) or OUT (data transferred from host to device) endpoint requires one logical core.

11 JTAG

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

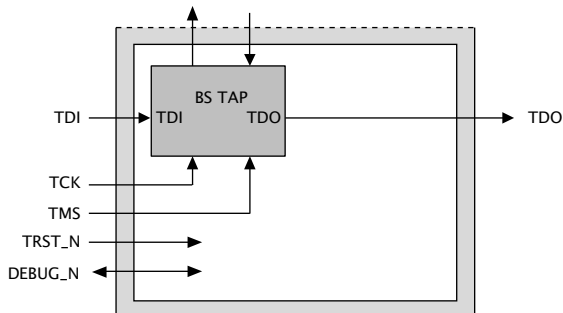


Figure 16:
JTAG chain
structure

The JTAG chain structure is illustrated in Figure 16. It comprises a single 1149.1 compliant TAP that can be used for boundary scan of the I/O pins. It has a 4-bit IR and 32-bit DR. It also provides access to a chip TAP that in turn can access the xCORE Tile for loading code and debugging.

The TRST_N pin must be asserted low during and after power up for 100 ns. If JTAG is not required, the TRST_N pin can be tied to ground to hold the JTAG module in reset.

The DEBUG_N pin is used to synchronize the debugging of multiple xCORE Tiles. This pin can operate in both output and input mode. In output mode and when configured to do so, DEBUG_N is driven low by the device when the processor hits a debug break point. Prior to this point the pin will be tri-stated. In input mode and when configured to do so, driving this pin low will put the xCORE Tile into debug mode. Software can set the behavior of the xCORE Tile based on this pin. This pin should have an external pull up of 4K7-47K Ω or left not connected in single core applications.

The JTAG device identification register can be read by using the IDCODE instruction. Its contents are specified in Figure 17.

Figure 17:
IDCODE
return value

Bit31		Device Identification Register																				Bit0											
Version				Part Number												Manufacturer Identity				1													
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0	1	1	0	0	1	1	1
0				0				0				0				6				6				3				3					

The JTAG usercode register can be read by using the USERCODE instruction. Its contents are specified in Figure 18. The OTP User ID field is read from bits [22:31] of the security register on xCORE Tile 0, see §9.1 (all zero on unprogrammed devices).

Figure 18:
USERCODE
return value

Bit31		Usercode Register																				Bit0																	
OTP User ID								Unused				Silicon Revision																											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0								0				0				2				8								0				0				0			

12 Board Integration

The device has the following power supply pins:

- ▶ VDD pins for the xCORE Tile, including a USB_VDD pin that powers the USB PHY
- ▶ VDDIO pins for the I/O lines. Separate I/O supplies are provided for the left, top, and right side of the package; different I/O voltages may be supplied on those. The signal description (Section 4) specifies which I/O is powered from which power-supply
- ▶ PLL_AVDD pins for the PLL
- ▶ OTP_VCC pins for the OTP
- ▶ A USB_VDD33 pin for the analogue supply to the USB-PHY

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 and OTP_VCC 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 Ω resistor and 100 nF multi-layer ceramic capacitor) is recommended on this pin.

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, 100nF 0402 for every other supply pin). 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 uF 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* §8). RST_N must be asserted low during and after power up for 100 ns.

12.1 USB connections

USB_VBUS should be connected to the VBUS pin of the USB connector. A 2.2 uF capacitor to ground is required on the VBUS pin. A ferrite bead may be used to reduce HF noise.

For self-powered systems, a bleeder resistor may be required to stop VBUS from floating when no USB cable is attached.

USB_DP and USB_DN should be connected to the USB connector. USB_ID does not need to be connected.

12.2 USB signal routing and placement

The USB_DP and USB_DN lines are the positive and negative data polarities of a high speed USB signal respectively. Their high-speed differential nature implies that they must be coupled and properly isolated. The board design must ensure that the board traces for USB_DP and USB_DN are tightly matched. In addition, according to the USB 2.0 specification, the USB_DP and USB_DN differential impedance must be 90 Ω .

Figure 19:
USB trace separation showing a low speed signal, two differential pairs and a high-speed clock

