XUF210-512-FB236 Datasheet

2020/10/05
Document Number: X007798

XMOS

Table of Contents

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.

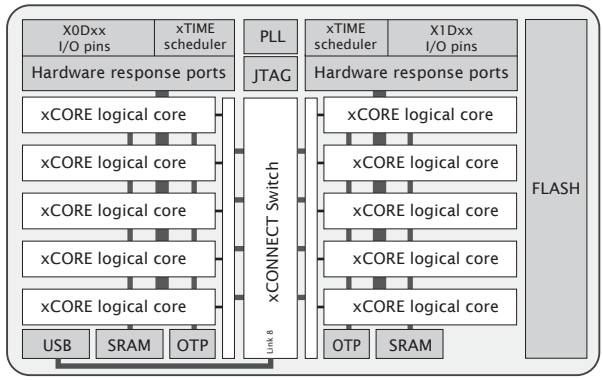


Figure 1: XUF210-512-FB236 block diagram

Key features of the XUF210-512-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
- **Flash** The device has a built-in 2MBflash. Section 8
- **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.ai/software-tools.

2 XUF210-512-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
  - 2MB internal flash for application code and overlays
- **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**
  - -40 °C to 85 °C
- **Speed Grade**
  - 24: 1200 MIPS
  - 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 rx1	4F X1D29 rx_dll		8D X1D41 tx2	OTP_VCC		NC	MODE[0]		4F X0D29	VDDIOR	GND
B	1M X0D36	VDDIOL	VDDIOL	TDO	TMS	TRST_N	4E X1D33 rx3	4E X1D32 rx2	4F X1D28 rx_clk	4E X1D26 tx_clk	8D X1D42 tx1	OTP_VCC	NC	NC	MODE[1]	4E X0D33	4E X0D32	VDDIOR	VDDIOR
C	1N X0D37 $X_0L^4_3$	1O X0D38 $X_0L^3_3$	VDDIOL	TDI	DEBUG_N	RST_N	1C X1D10	1D X1D11	4F X1D30 rx0	4E X1D27 tx_dll	8D X1D43 tx0	8D X1D40 tx3	NC	NC	4F X0D31	4F X0D30	4F X0D28	4E X0D26 $X_0L^5_7$	4E X0D27 $X_0L^4_7$
D		1P X0D39 $X_0L^2_3$	8D X0D40 $X_0L^1_3$														1K X0D34 $X_0L^1_7$	1L X0D35 $X_0L^0_7$	
E	8D X0D43 $X_0L^1_3$	8D X0D42 $X_0L^0_3$	8D X0D41 $X_0L^0_2$														1J X0D25 $X_0L^0_7$	1I X0D24 $X_0L^0_6$	1B X1D01 $X_0L^7_1$
F	1K X1D34 $X_0L^0_2$	1L X1D35 $X_0L^0_3$	1M X1D36 $X_0L^0_4$				NC	VDD	VDD	VDDIOT	VDD	VDD	PLL_AVDD	PLL_AGND			4A X1D08 $X_0L^7_4$	4A X1D09 $X_0L^7_5$	1A X1D00 $X_0L^7_0$
G		32A X1D49 $X_0L^4_1$	32A X1D50 $X_0L^5_1$			VDD		GND		GND		GND		VDD			32A X0D69 $X_0L^5_6$	32A X0D70 $X_0L^6_6$	
H	32A X1D53 $X_0L^6_1$	32A X1D52 $X_0L^7_1$	32A X1D51 $X_0L^7_2$			VDD	GND	GND	GND	GND	GND	GND	GND	VDD			32A X0D68 $X_0L^4_6$	32A X0D67 $X_0L^5_6$	32A X0D66 $X_0L^6_6$
J	32A X1D54 $X_0L^6_1$	32A X1D55 $X_0L^5_1$	32A X1D56 $X_0L^6_2$			VDD		GND		GND		GND		VDD			32A X0D63 $X_0L^3_6$	32A X0D64 $X_0L^4_6$	32A X0D65 $X_0L^5_6$
K		32A X1D58 $X_0L^4_1$	32A X1D57 $X_0L^5_1$			VDD	GND	GND	GND	GND	GND	GND	GND	VDD			32A X0D62 $X_0L^3_6$	32A X0D61 $X_0L^4_6$	
L	32A X1D63 $X_0L^2_1$	32A X1D62 $X_0L^3_1$	32A X1D61 $X_0L^3_2$			VDD		GND		GND		GND		VDD			32A X0D58 $X_0L^0_6$	32A X0D57 $X_0L^1_6$	32A X0D56 $X_0L^2_6$
M	32A X1D64 $X_0L^1_2$	32A X1D65 $X_0L^0_2$	32A X1D66 $X_0L^0_3$			VDD	GND	GND	GND	GND	GND	GND	GND	VDD			32A X0D53 $X_0L^2_5$	32A X0D54 $X_0L^3_5$	32A X0D55 $X_0L^4_5$
N		32A X1D67 $X_0L^1_2$	32A X1D68 $X_0L^2_2$			VDD		GND		GND		GND		VDD			32A X0D51 $X_0L^0_5$	32A X0D52 $X_0L^1_5$	
P	32A X1D70 $X_0L^4_2$	32A X1D69 $X_0L^3_2$	1N X1D37 $X_0L^4_4$			VDD	VDD	VDD	USB_VDD	USB_VDD	VDD	VDD	VDD	NC			4B X1D07 $X_0L^4_4$	32A X0D50 $X_0L^0_5$	32A X0D49 $X_0L^6_5$
R	1O X1D38 $X_0L^3_3$	1P X1D39 $X_0L^2_4$	4D X1D17 $X_0L^5_5$														4A X1D03 $X_0L^0_4$	4B X1D05 $X_0L^2_4$	4B X1D06 $X_0L^3_4$
T		4D X1D16 $X_0L^3_5$	4D X1D18 $X_0L^4_5$														4A X1D02 $X_0L^7_4$	4B X1D04 $X_0L^1_4$	
U	1C X0D10 $X_0L^3_3$	1B X0D01 $X_0L^2_3$	4D X1D19 $X_0L^1_3$	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 $X_0L^1_4$	4D X0D18 $X_0L^2_4$	4D X0D17 $X_0L^3_4$
V	1G X1D22 $X_0L^4_3$	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 $X_0L^4_4$
W	GND	VDDIOL	1H X1D23		4B X0D05	4A X0D02		1F 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 XUF210-512-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 13.3.
- 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 power for PLL	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, active low	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, active low	Input	IOL, PU, ST

I/O pins (128)								
Signal	**Function**						**Type**	**Properties**
X0D00		$1A^{0}$					I/O	IOL, PD
X0D01	$X_0L3^{2}_{out}$	$1B^{0}$					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	$X_0L3^{3}_{out}$	$1C^{0}$					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	$X_0L4^{4}_{in}$		$4D^{0}$	$8B^{2}$	$16A^{10}$		I/O	IOR, PD
X0D17	$X_0L4^{3}_{in}$		$4D^{1}$	$8B^{3}$	$16A^{11}$		I/O	IOR, PD
X0D18	$X_0L4^{2}_{in}$		$4D^{2}$	$8B^{4}$	$16A^{12}$		I/O	IOR, PD
X0D19	$X_0L4^{1}_{in}$		$4D^{3}$	$8B^{5}$	$16A^{13}$		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	$X_0L7^{0}_{in}$	$1I^{0}$					I/O	IOR, PD
X0D25	$X_0L7^{0}_{out}$	$1J^{0}$					I/O	IOR, PD
X0D26	$X_0L7^{3}_{out}$		$4E^{0}$	$8C^{0}$	$16B^{0}$		I/O	IOR, PD
X0D27	$X_0L7^{4}_{out}$		$4E^{1}$	$8C^{1}$	$16B^{1}$		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	$X_0L7^{1}_{out}$	$1K^{0}$					I/O	IOR, PD
X0D35	$X_0L7^{2}_{out}$	$1L^{0}$					I/O	IOR, PD
X0D36		$1M^{0}$		$8D^{0}$	$16B^{8}$		I/O	IOL, PD
X0D37	$X_0L0^{4}_{in}$	$1N^{0}$		$8D^{1}$	$16B^{9}$		I/O	IOL, PD
X0D38	$X_0L0^{3}_{in}$	$1O^{0}$		$8D^{2}$	$16B^{10}$		I/O	IOL, PD
X0D39	$X_0L0^{2}_{in}$	$1P^{0}$		$8D^{3}$	$16B^{11}$		I/O	IOL, PD
X0D40	$X_0L0^{1}_{in}$			$8D^{4}$	$16B^{12}$		I/O	IOL, PD

(continued)

Signal	Function					Type	Properties
X0D41	$X_0L0^0_{in}$		$8D^5$	$16B^{13}$		I/O	IOL, PD
X0D42	$X_0L0^0_{out}$		$8D^6$	$16B^{14}$		I/O	IOL, PD
X0D43	$X_0L0^1_{out}$		$8D^7$	$16B^{15}$		I/O	IOL, PD
X0D49	$X_0L5^4_{in}$				$32A^0$	I/O	IOR, PD
X0D50	$X_0L5^3_{in}$				$32A^1$	I/O	IOR, PD
X0D51	$X_0L5^2_{in}$				$32A^2$	I/O	IOR, PD
X0D52	$X_0L5^1_{in}$				$32A^3$	I/O	IOR, PD
X0D53	$X_0L5^0_{in}$				$32A^4$	I/O	IOR, PD
X0D54	$X_0L5^0_{out}$				$32A^5$	I/O	IOR, PD
X0D55	$X_0L5^1_{out}$				$32A^6$	I/O	IOR, PD
X0D56	$X_0L5^2_{out}$				$32A^7$	I/O	IOR, PD
X0D57	$X_0L5^3_{out}$				$32A^8$	I/O	IOR, PD
X0D58	$X_0L5^4_{out}$				$32A^9$	I/O	IOR, PD
X0D61	$X_0L6^4_{in}$				$32A^{10}$	I/O	IOR, PD
X0D62	$X_0L6^3_{in}$				$32A^{11}$	I/O	IOR, PD
X0D63	$X_0L6^2_{in}$				$32A^{12}$	I/O	IOR, PD
X0D64	$X_0L6^1_{in}$				$32A^{13}$	I/O	IOR, PD
X0D65	$X_0L6^0_{in}$				$32A^{14}$	I/O	IOR, PD
X0D66	$X_0L6^0_{out}$				$32A^{15}$	I/O	IOR, PD
X0D67	$X_0L6^1_{out}$				$32A^{16}$	I/O	IOR, PD
X0D68	$X_0L6^2_{out}$				$32A^{17}$	I/O	IOR, PD
X0D69	$X_0L6^3_{out}$				$32A^{18}$	I/O	IOR, PD
X0D70	$X_0L6^4_{out}$				$32A^{19}$	I/O	IOR, PD
X1D00	$X_0L7^2_{in}$	$1A^0$				I/O	IOR, PD
X1D01	$X_0L7^1_{in}$	$1B^0$				I/O	IOR, PD
X1D02	$X_0L4^0_{in}$		$4A^0$	$8A^0$	$16A^0$ $32A^{20}$	I/O	IOR, PD
X1D03	$X_0L4^0_{out}$		$4A^1$	$8A^1$	$16A^1$ $32A^{21}$	I/O	IOR, PD
X1D04	$X_0L4^1_{out}$		$4B^0$	$8A^2$	$16A^2$ $32A^{22}$	I/O	IOR, PD
X1D05	$X_0L4^2_{out}$		$4B^1$	$8A^3$	$16A^3$ $32A^{23}$	I/O	IOR, PD
X1D06	$X_0L4^3_{out}$		$4B^2$	$8A^4$	$16A^4$ $32A^{24}$	I/O	IOR, PD
X1D07	$X_0L4^4_{out}$		$4B^3$	$8A^5$	$16A^5$ $32A^{25}$	I/O	IOR, PD
X1D08	$X_0L7^4_{in}$		$4A^2$	$8A^6$	$16A^6$ $32A^{26}$	I/O	IOR, PD
X1D09	$X_0L7^3_{in}$		$4A^3$	$8A^7$	$16A^7$ $32A^{27}$	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	$X_0L3^1_{in}$		$4D^0$	$8B^2$	$16A^{10}$	I/O	IOL, PD
X1D17	$X_0L3^0_{in}$		$4D^1$	$8B^3$	$16A^{11}$	I/O	IOL, PD
X1D18	$X_0L3^0_{out}$		$4D^2$	$8B^4$	$16A^{12}$	I/O	IOL, PD
X1D19	$X_0L3^1_{out}$		$4D^3$	$8B^5$	$16A^{13}$	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	$X_0L3^4_{out}$	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	tx_clk (rgmii)		4E[0]	8C[0]	16B[0]		I/O	IOT, PD
X1D27	tx_ctl (rgmii)		4E[1]	8C[1]	16B[1]		I/O	IOT, PD
X1D28	rx_clk (rgmii)		4F[0]	8C[2]	16B[2]		I/O	IOT, PD
X1D29	rx_ctl (rgmii)		4F[1]	8C[3]	16B[3]		I/O	IOT, PD
X1D30	rx0 (rgmii)		4F[2]	8C[4]	16B[4]		I/O	IOT, PD
X1D31	rx1 (rgmii)		4F[3]	8C[5]	16B[5]		I/O	IOT, PD
X1D32	rx2 (rgmii)		4E[2]	8C[6]	16B[6]		I/O	IOT, PD
X1D33	rx3 (rgmii)		4E[3]	8C[7]	16B[7]		I/O	IOT, PD
X1D34	$X_0L0^2_{out}$	1K[0]					I/O	IOL, PD
X1D35	$X_0L0^3_{out}$	1L[0]					I/O	IOL, PD
X1D36	$X_0L0^4_{out}$	1M[0]		8D[0]	16B[8]		I/O	IOL, PD
X1D37	$X_0L3^4_{in}$	1N[0]		8D[1]	16B[9]		I/O	IOL, PD
X1D38	$X_0L3^3_{in}$	1O[0]		8D[2]	16B[10]		I/O	IOL, PD
X1D39	$X_0L3^2_{in}$	1P[0]		8D[3]	16B[11]		I/O	IOL, PD
X1D40	tx3 (rgmii)			8D[4]	16B[12]		I/O	IOT, PD
X1D41	tx2 (rgmii)			8D[5]	16B[13]		I/O	IOT, PD
X1D42	tx1 (rgmii)			8D[6]	16B[14]		I/O	IOT, PD
X1D43	tx0 (rgmii)			8D[7]	16B[15]		I/O	IOT, PD
X1D49	$X_0L1^4_{in}$					32A[0]	I/O	IOL, PD
X1D50	$X_0L1^3_{in}$					32A[1]	I/O	IOL, PD
X1D51	$X_0L1^2_{in}$					32A[2]	I/O	IOL, PD
X1D52	$X_0L1^1_{in}$					32A[3]	I/O	IOL, PD
X1D53	$X_0L1^0_{in}$					32A[4]	I/O	IOL, PD
X1D54	$X_0L1^0_{out}$					32A[5]	I/O	IOL, PD
X1D55	$X_0L1^1_{out}$					32A[6]	I/O	IOL, PD
X1D56	$X_0L1^2_{out}$					32A[7]	I/O	IOL, PD
X1D57	$X_0L1^3_{out}$					32A[8]	I/O	IOL, PD
X1D58	$X_0L1^4_{out}$					32A[9]	I/O	IOL, PD
X1D61	$X_0L2^4_{in}$					32A[10]	I/O	IOL, PD
X1D62	$X_0L2^3_{in}$					32A[11]	I/O	IOL, PD
X1D63	$X_0L2^2_{in}$					32A[12]	I/O	IOL, PD
X1D64	$X_0L2^1_{in}$					32A[13]	I/O	IOL, PD
X1D65	$X_0L2^0_{in}$					32A[14]	I/O	IOL, PD
X1D66	$X_0L2^0_{out}$					32A[15]	I/O	IOL, PD
X1D67	$X_0L2^1_{out}$					32A[16]	I/O	IOL, PD
X1D68	$X_0L2^2_{out}$					32A[17]	I/O	IOL, PD
X1D69	$X_0L2^3_{out}$					32A[18]	I/O	IOL, PD

(continued)

Signal	Function		Type	Properties
X1D70	$X_0L2^4_{out}$	$32A^{19}$	I/O	IOL, PD

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

usb pins (5)			
Signal	**Function**	**Type**	**Properties**
USB_DM	USB Data-	I/O	
USB_DP	USB Data+	I/O	
USB_ID	USB Identification	I/O	
USB_RTUNE	USB resistor	I/O	
USB_VBUS	USB Power Detect Pin	I/O	

5 Example Application Diagram

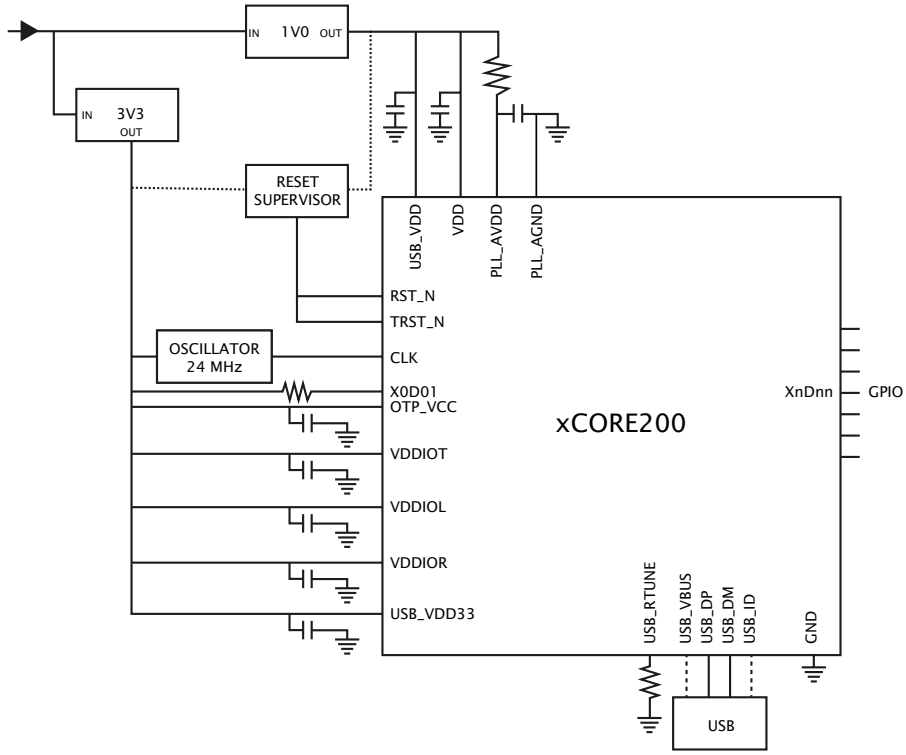


Figure 2: Simplified Reference Schematic

- 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 XUF210-512-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 XUF210-512-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. xCORE200 IO pins can 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.

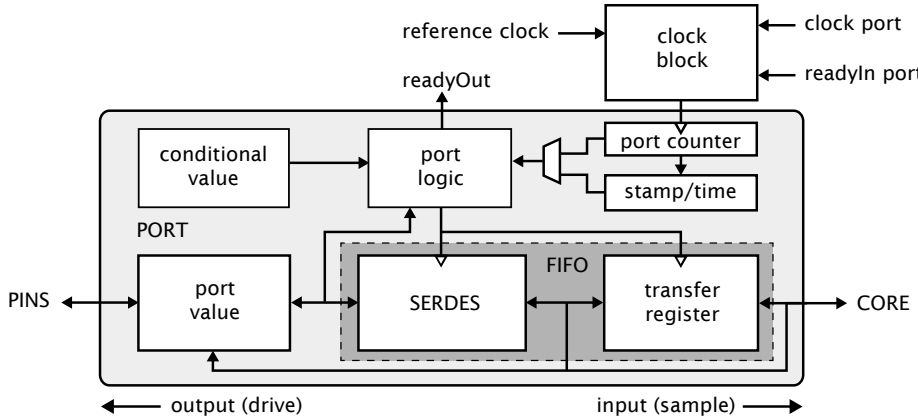


Figure 4: Port block diagram

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. xCORE200 clock blocks optionally divide the clock input from a 1-bit port.

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.

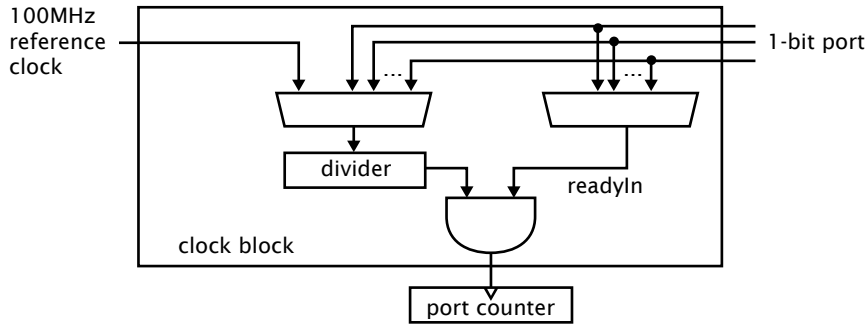


Figure 5: Clock block diagram

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.

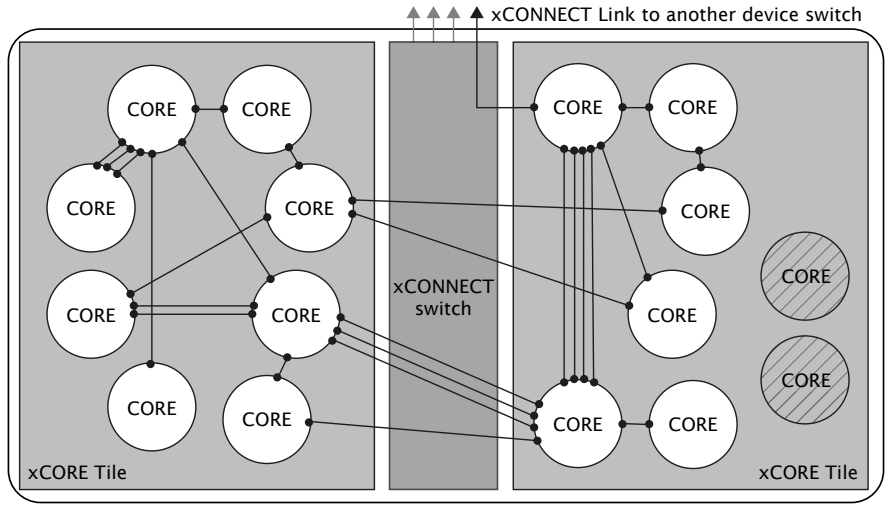


Figure 6: Switch, links and channel ends

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.

Information on the supported routing topologies that can be used to connect multiple devices together can be found in the XS1-UF 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:

Figure 7: PLL multiplier values and MODE pins

Oscillator Frequency	MODE 1	MODE 0	Tile Boot Frequency	PLL Ratio	PLL settings 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 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. The processor must be held in reset until VDDIOL is in spec for at least

1 ms. 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 device boots from a QSPI flash (IS25LP016D) that is embedded in the device. The QSPI flash is connected to the ports on Tile 0 as shown in Figure 8. An external 1K resistor must connect X0D01 to VDDIOL. X0D10 should ideally not be connected. If X0D10 is connected, then a 150 ohm series resistor close to the device is recommended. X0D04..X0D07 should be not connected.

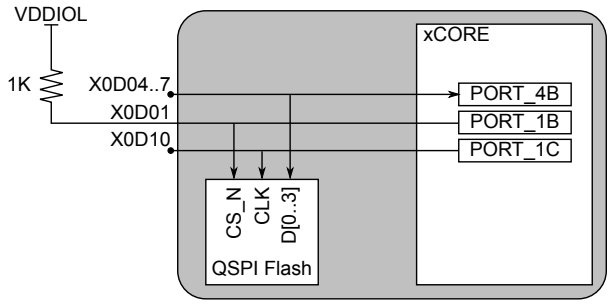


Figure 8: QSPI port connectivity

The xCORE Tile boot procedure is illustrated in Figure 9. If bit 5 of the security register (*see* §9.1) is set, the device boots from OTP. Otherwise, the device boots from the internal flash.

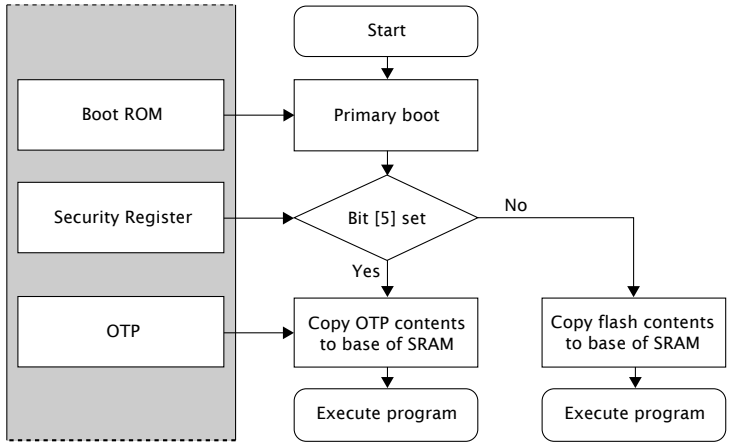


Figure 9: Boot procedure

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 Security register

The security register enables security features on the xCORE tile. The features shown in Figure 10 provide a strong level of protection and are sufficient for providing strong IP security.

Feature	Bit	Description
Disable JTAG	0	The JTAG interface is disabled, making it impossible for the tile state or memory content to be accessed via the JTAG interface.
Disable Link access	1	Other tiles are forbidden access to the processor state via the system switch. Disabling both JTAG and Link access transforms an xCORE Tile into a "secure island" with other tiles free for non-secure user application code.
Secure Boot	5	The xCORE Tile is forced to boot from address 0 of the OTP, allowing the xCORE Tile boot ROM to be bypassed (*see* §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 accessable security register available to end-users.
	31..22	General purpose user programmable JTAG UserID code extension.

Figure 10: Security register features

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 256KB SRAM bank for both instructions and data. All internal memory is 32 bits wide, and instructions are either 16-bit or 32-bit. Byte (8-bit), half-word (16-bit) or word (32-bit) accesses are supported and are executed within one tile clock cycle. There is no dedicated external memory interface, although data memory can be expanded through appropriate use of the ports.

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 11. 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 xCORE200.

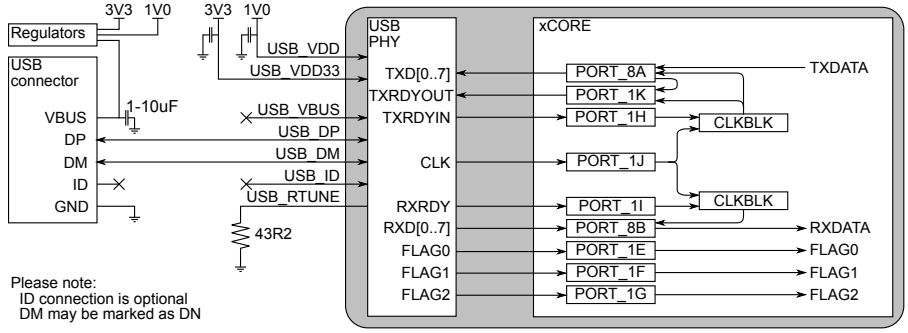


Figure 11: Bus powered *USB-device*

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 12.

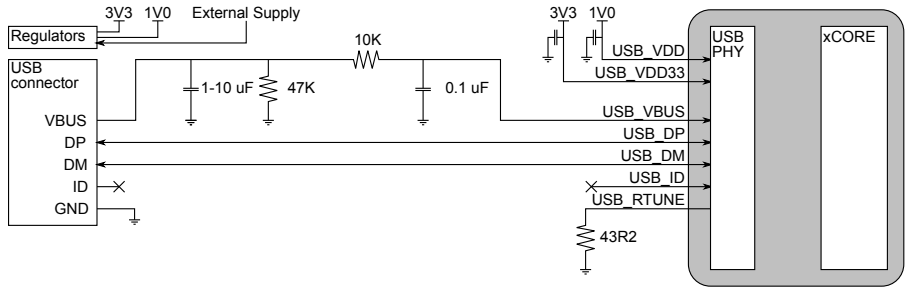


Figure 12: Self powered *USB-device*

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 12 ensures that the transient does not damage the device. The 10k series resistor and 0.1uF capacitor ensure than 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.

The JTAG chain structure is illustrated in Figure 13. 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.

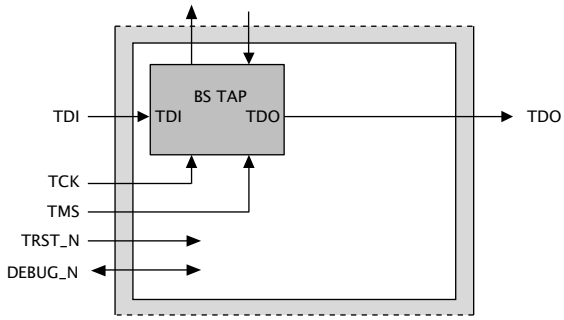


Figure 13: JTAG chain structure

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 14.

Figure 14: 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	1	0	1	0	1	1	0	0	0	1	1	0	0	1	1
0				0				0				0				5				6				3				3			

The JTAG usercode register can be read by using the USERCODE instruction. Its contents are specified in Figure 15. 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 15: 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	1	0	1	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.

VDDIO/OTP_VCC and VDD can ramp up independently. In order to reduce stresses on the device, it is preferable to make them ramp up in a short time frame of each other, no more than 50 ms apart. RST_N and TRST_N should be kept low until all power supplies are stable and within tolerances of their final voltage. If your design is powered by VBUS, then RST_N should go high within 10 ms of attaching to VBUS in order to ensure that USB timings are met. RST_N should be at least 1 ms after VDDIO good to enable the built-in flash to settle. Power sequencing is summarised in Figure 16

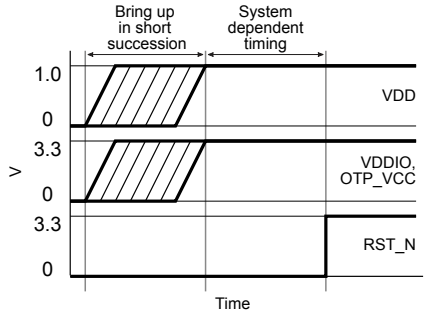


Figure 16: Sequencing of power supplies and RST_N

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 and 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.

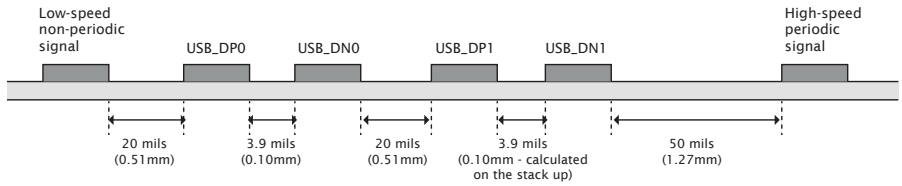


Figure 17: USB trace separation showing a low speed signal, two differential pairs and a high-speed clock

12.2.1 General routing and placement guidelines

The following guidelines will help to avoid signal quality and EMI problems on high speed USB designs. They relate to a four-layer (Signal, GND, Power, Signal) PCB.

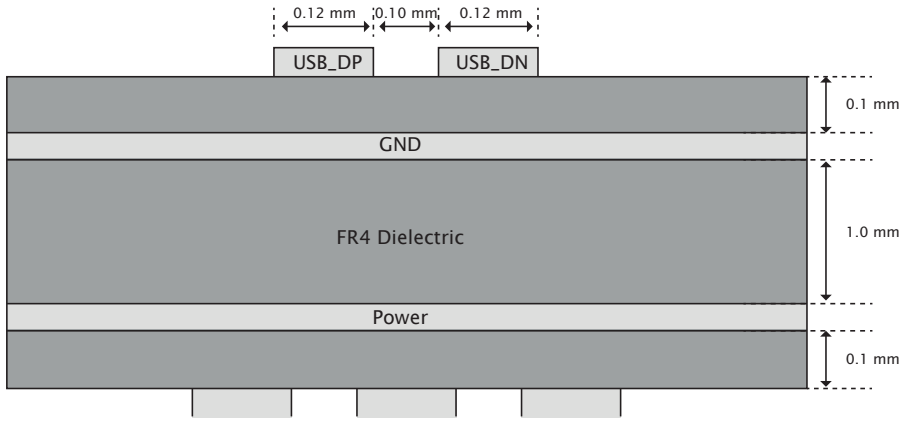


Figure 18: Example USB board stack

For best results, most of the routing should be done on the top layer (assuming the USB connector and XUF210-512-FB236 are on the top layer) closest to GND. Reference planes should be below the transmission lines in order to maintain control of the trace impedance.

We recommend that the high-speed clock and high-speed USB differential pairs are routed first before any other routing. When routing high speed USB signals, the following guidelines should be followed:

- High speed differential pairs should be routed together.
- High-speed USB signal pair traces should be trace-length matched. Maximum trace-length mismatch should be no greater than 4mm.
- Ensure that high speed signals (clocks, USB differential pairs) are routed as far away from off-board connectors as possible.
- High-speed clock and periodic signal traces that run parallel should be at least 1.27mm away from USB_DP/USB_DN (see Figure 17).
- Low-speed and non-periodic signal traces that run parallel should be at least 0.5mm away from USB_DP/USB_DN (see Figure 17).
- Route high speed USB signals on the top of the PCB wherever possible.
- Route high speed USB traces over continuous power planes, with no breaks. If a trade-off must be made, changing signal layers is preferable to crossing plane splits.
- Follow the $20 \times h$ rule; keep traces $20 \times h$ (the height above the power plane) away from the edge of the power plane.
- Use a minimum of vias in high speed USB traces.
- Avoid corners in the trace. Where necessary, rather than turning through a 90 degree angle, use two 45 degree turns or an arc.
- DO NOT route USB traces near clock sources, clocked circuits or magnetic devices.
- Avoid stubs on high speed USB signals.

12.3 Land patterns and solder stencils

The package is a 236 ball Fine Ball Grid Array (FBGA) on a 0.5 mm pitch. We recommend you use HDI or better PCB technology. The missing balls in the outer rows can be used to route the first inner row out over the top layer. The missing balls in the center can be used for ground vias. The missing rows four and five can be used for VDD vias if required.

The land patterns and solder stencils will depend on the PCB manufacturing process. We recommend you design them with using the IPC specifications *"Generic Requirements for Surface Mount Design and Land Pattern Standards"* IPC-7351B. This standard aims to achieve desired targets of heel, toe and side fillets for solder-joints. The mechanical drawings in Section 14 specify the dimensions and tolerances.

12.4 Ground and Thermal Vias

Vias from the ground balls into the ground plane of the PCB are recommended for a low inductance ground connection and good thermal performance. Typical designs could use 16 vias in a 4 x 4 grid, equally spaced amongst the ground balls.

12.5 Moisture Sensitivity

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

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

13 Electrical Characteristics

13.1 Absolute Maximum Ratings

Stresses beyond those listed under Absolute Maximum Ratings may cause permanent damage to the device. Exposure to any Absolute Maximum Rating condition for extended periods may affect device reliability and lifetime.

Symbol	Parameter	MIN	MAX	UNITS	Notes
VDD	Tile DC supply voltage	-0.2	1.1	V	
PLL_AVDD	PLL analog supply	-0.2	1.1	V	
VDDIO	I/O supply voltage	-0.3	3.75	V	
OTP_VCC	OTP supply voltage	-0.3	3.75	V	
Tj	Junction temperature		125	°C	
Tstg	Storage temperature	-65	150	°C	
V(Vin)	Voltage applied to any IO pin	-0.3	3.75	V	
I(XxDxx)	GPIO current	-30	30	mA	
V(X0D03-8)	Voltage applied to flash pins	-0.3	VDDIO+0.5	V	
I(VDDIOL)	Current for VDDIOL domain		588	mA	A, B, C
I(VDDIOR)	Current for VDDIOR domain		686	mA	A, B, C
I(VDDIOT)	Current for VDDIOT domain		98	mA	A, C
USB_VDD	USB tile DC supply voltage	-0.2	1.1	V	
USB_VDD33	USB tile analog supply voltage	-0.3	3.75	V	
USB_VBUS	USB VBUS voltage	-0.3	5.75	V	
USB_DP	USB DP voltage	-0.3	5.5	V	
USB_DM	USB DM voltage	-0.3	5.5	V	
USB_ID	USB ID voltage	-0.3	2.75	V	

Figure 19: Absolute maximum ratings

A Exceeding these current limits will result in premature aging and reduced lifetime.

B This current consumption must be evenly distributed over all VDDIO pins.

C All main power (VDD, VDDIO) and ground (VSS) pins must always be connected.

13.2 Operating Conditions

Symbol	Parameter	MIN	TYP	MAX	UNITS	Notes
VDD	Tile DC supply voltage	0.95	1.00	1.05	V	
VDDIOL	I/O supply voltage	3.135	3.30	3.465	V	
VDDIOR	I/O supply voltage	3.135	3.30	3.465	V	
VDDIOT 3v3	I/O supply voltage	3.135	3.30	3.465	V	
VDDIOT 2v5	I/O supply voltage	2.375	2.50	2.625	V	
USB_VDD	USB tile DC supply voltage	0.95	1.00	1.05	V	
VDD33	Peripheral supply	3.135	3.30	3.465	V	
PLL_AVDD	PLL analog supply	0.95	1.00	1.05	V	
Cl	xCORE Tile I/O load capacitance			25	pF	
Ta	Ambient operating temperature ()	0		70	°C	
	Ambient operating temperature ()	-40		85	°C	
Tj	Junction temperature			125	°C	

Figure 20: Operating conditions

13.3 DC Characteristics, VDDIO=3V3

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	2.20			V	B, C
V(OL)	Output low voltage			0.40	V	B, C
I(PU)	Internal pull-up current (Vin=0V)	-100			µA	D
I(PD)	Internal pull-down current (Vin=3.3V)			100	µA	D
I(LC)	Input leakage current	-10		10	µA	

Figure 21: DC characteristics

A All pins except power supply pins.

B Pins X1D40, X1D41, X1D42, X1D43, X1D26, and X1D27 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. In order to pull the pin to the opposite state, a 4K7 resistor is recommended to overome the internal pull current.

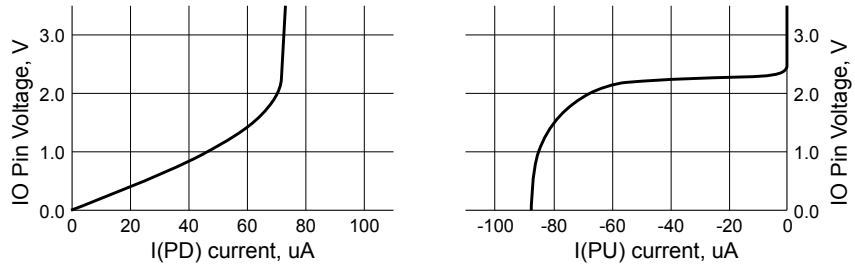


Figure 22: Typical internal pull-down and pull-up currents

13.4 ESD Stress Voltage

Figure 23: 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	

13.5 Reset Timing

Figure 24: 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.

13.6 Power Consumption

Figure 25: xCORE Tile currents

Symbol	Parameter	MIN	TYP	MAX	UNITS	Notes
I(DDCQ)	Quiescent VDD current		45		mA	A, B, C
PD	Tile power dissipation		325		µW/MIPS	A, D, E, F
IDD	Active VDD current		570	700	mA	A, G
I(ADDPLL)	PLL_AVDD current		5	7	mA	H
I(VDD33)	VDD33 current		26.7		mA	I
I(USB_VDD)	USB_VDD current		8.27		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 xCORE-200 Power Consumption document,

13.7 Clock

Figure 26: 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 xCORE-200 Clock Frequency Control document,

13.8 xCORE Tile I/O AC Characteristics

Figure 27: I/O AC characteristics

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

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

Information on interfacing to high-speed synchronous interfaces can be found in the Port I/O Timing document, X5821.

13.9 xConnect Link Performance

Figure 28: Link performance

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

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

B 7.5 ns symbol time.

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

13.10 JTAG Timing

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

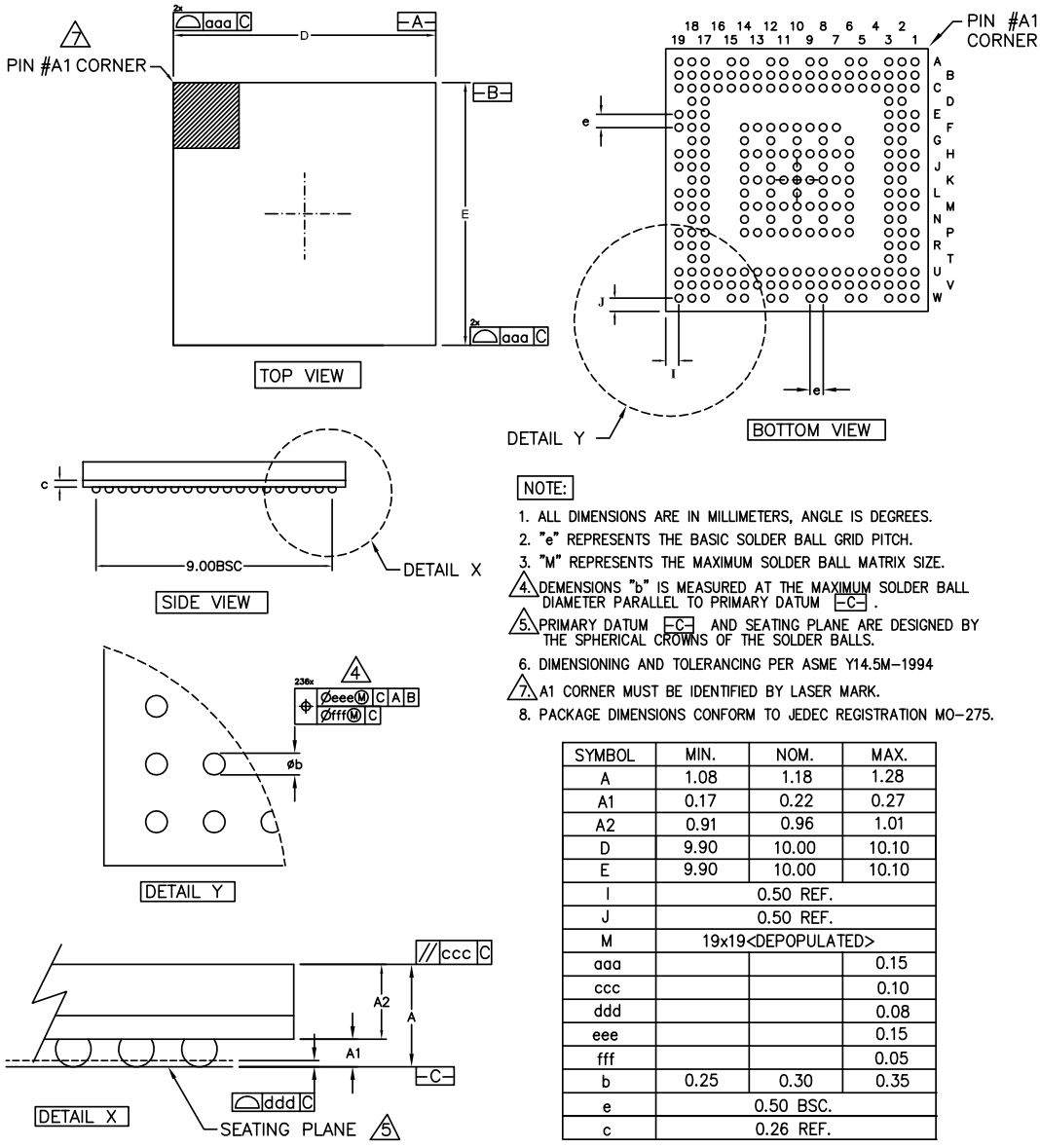
Figure 29: JTAG timing

A Timing applies to TMS and TDI inputs.

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

All JTAG operations are synchronous to TCK apart from the global asynchronous reset TRST_N.

14 Package Information



NOTE:

1. ALL DIMENSIONS ARE IN MILLIMETERS, ANGLE IS DEGREES.
2. "e" REPRESENTS THE BASIC SOLDER BALL GRID PITCH.
3. "M" REPRESENTS THE MAXIMUM SOLDER BALL MATRIX SIZE.
4. DEMENSIONS "b" IS MEASURED AT THE MAXIMUM SOLDER BALL DIAMETER PARALLEL TO PRIMARY DATUM -C- .
5. PRIMARY DATUM -C- AND SEATING PLANE ARE DESIGNED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.
6. DIMENSIONING AND TOLERANCING PER ASME Y14.5M–1994
7. A1 CORNER MUST BE IDENTIFIED BY LASER MARK.
8. PACKAGE DIMENSIONS CONFORM TO JEDEC REGISTRATION MO–275.

SYMBOL	MIN.	NOM.	MAX.
A	1.08	1.18	1.28
A1	0.17	0.22	0.27
A2	0.91	0.96	1.01
D	9.90	10.00	10.10
E	9.90	10.00	10.10
I		0.50 REF.	
J		0.50 REF.	
M		19x19<DEPOPULATED>	
aaa			0.15
ccc			0.10
ddd			0.08
eee			0.15
fff			0.05
b	0.25	0.30	0.35
e		0.50 BSC.	
c		0.26 REF.	

14.1 Part Marking

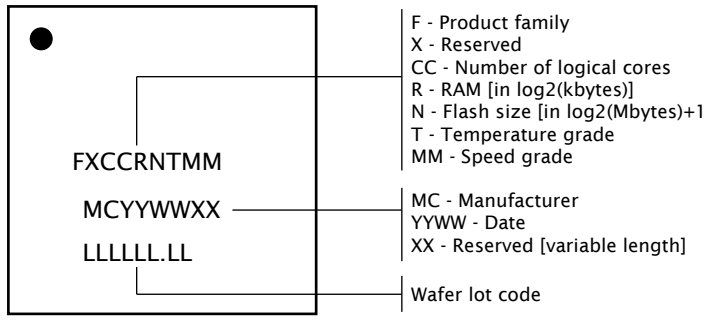


Figure 30: Part marking scheme

15 Ordering Information

Figure 31: Orderable part numbers

Product Code	Marking	Qualification	Speed Grade
XUF210-512-FB236-C20A	`U11092C20`	Commercial	1000 MIPS
XUF210-512-FB236-I20A	`U11092I20`	Industrial	1000 MIPS

Appendices

A Configuration of the XUF210-512-FB236

The device is configured through banks of registers, as shown in Figure 32.

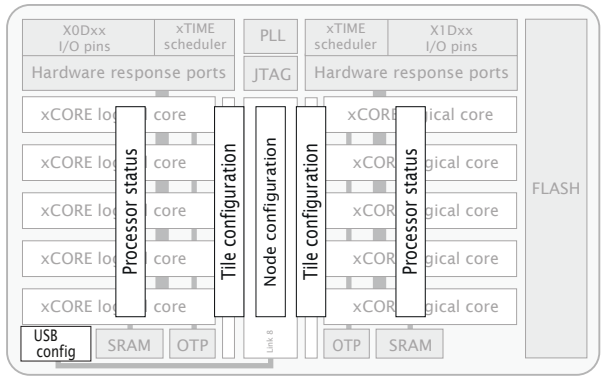


Figure 32: Registers

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 0x0B. Alternatively, the functions `getps(reg)` and `setps(reg,value)` can be used from XC.

A.2 Accessing an xCORE Tile configuration register

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

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

A write message comprises the following:

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

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

A read message comprises the following:

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

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

A.3 Accessing node configuration

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

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

A write message comprises the following:

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

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

A read message comprises the following:

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

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

A.4 Accessing a register of an analogue peripheral

Peripheral registers can be accessed through the interconnect using the functions `write_periph_32(device, peripheral, ...)`, `read_periph_32(device, peripheral, ...)`, `write_periph_8(device, peripheral, ...)`, and `read_periph_8(device, peripheral, ...)`; where device is the name of the analogue device, and peripheral is the number of the peripheral. These functions implement the protocols described below.

A channel-end should be allocated to communicate with the configuration registers. The destination of the channel-end should be set to `0xnnnnpp02` where `nnnn` is the node-identifier and `pp` is the peripheral identifier.

A write message comprises the following:

control-token 36	24-bit response channel-end identifier	8-bit register number	8-bit size	data	control-token 1

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

A read message comprises the following:

control-token 37	24-bit response channel-end identifier	8-bit register number	8-bit size	control-token 1

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

B Processor Status Configuration

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

The identifiers for the registers needs a prefix "`XS1_PS_`" and a postfix "`_NUM`", and are declared in "`xs1.h`"

Number	Perm	Description	Register identifier
0x00	RW	RAM base address	RAM_BASE
0x01	RW	Vector base address	VECTOR_BASE
0x02	RW	xCORE Tile control	XCORE_CTRL0
0x03	RO	xCORE Tile boot status	BOOT_CONFIG
0x05	RW	Security configuration	SECURITY_CONFIG
0x06	RW	Ring Oscillator Control	RING_OSC_CTRL
0x07	RO	Ring Oscillator Value	RING_OSC_DATA0
0x08	RO	Ring Oscillator Value	RING_OSC_DATA1
0x09	RO	Ring Oscillator Value	RING_OSC_DATA2
0x0A	RO	Ring Oscillator Value	RING_OSC_DATA3
0x0C	RO	RAM size	RAM_SIZE
0x10	DRW	Debug SSR	DBG_SSR
0x11	DRW	Debug SPC	DBG_SPC
0x12	DRW	Debug SSP	DBG_SSP
0x13	DRW	DGETREG operand 1	DBG_T_NUM
0x14	DRW	DGETREG operand 2	DBG_T_REG
0x15	DRW	Debug interrupt type	DBG_TYPE
0x16	DRW	Debug interrupt data	DBG_DATA
0x18	DRW	Debug core control	DBG_RUN_CTRL
0x20 .. 0x27	DRW	Debug scratch	DBG_SCRATCH
0x30 .. 0x33	DRW	Instruction breakpoint address	DBG_IBREAK_ADDR
0x40 .. 0x43	DRW	Instruction breakpoint control	DBG_IBREAK_CTRL
0x50 .. 0x53	DRW	Data watchpoint address 1	DBG_DWATCH_ADDR1
0x60 .. 0x63	DRW	Data watchpoint address 2	DBG_DWATCH_ADDR2
0x70 .. 0x73	DRW	Data breakpoint control register	DBG_DWATCH_CTRL

Figure 33: Summary

Figure 34: Summary (continued)

Number	Perm	Description	Register identifier
0x80 .. 0x83	DRW	Resources breakpoint mask	DBG_RWATCH_ADDR1
0x90 .. 0x93	DRW	Resources breakpoint value	DBG_RWATCH_ADDR2
0x9C .. 0x9F	DRW	Resources breakpoint control register	DBG_RWATCH_CTRL

B.1 RAM base address

RAM_BASE 0x00

This register contains the base address of the RAM. It is initialized to 0x00040000.

0x00: RAM base address

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

B.2 Vector base address

VECTOR_BASE 0x01

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

0x01: Vector base address

Bits	Perm	Init	Description	Identifier
31:18	RW		The event and interrupt vectors.	VECTOR_BASE
17:0	RO	-	Reserved	

B.3 xCORE Tile control

XCORE_CTRL0 0x02

Register to control features in the xCORE tile

0x02: xCORE Tile control

Bits	Perm	Init	Description	Identifier
31:26	RO	-	Reserved	
25:18	RW	0	RGMII TX data delay value (in PLL output cycle increments)	XCORE_CTRL0_RGMII_DELAY
17:9	RW	0	RGMII TX clock divider value. TX clk rises when counter (clocked by PLL output) reaches this value and falls when counter reaches (value»1). Value programmed into this field should be actual divide value required minus 1	XCORE_CTRL0_RGMII_DIVIDE
8	RW	0	Enable RGMII interface periph ports	XCORE_CTRL0_RGMII_ENABLE
7:6	RO	-	Reserved	
5	RW	0	Select the dynamic mode (1) for the clock divider when the clock divider is enabled. In dynamic mode the clock divider is only activated when all active threads are paused. In static mode the clock divider is always enabled.	XCORE_CTRL0_CLK_DIVIDER_DYN
4	RW	0	Enable the clock divider. This divides the output of the PLL to facilitate one of the low power modes.	XCORE_CTRL0_CLK_DIVIDER_EN
3	RO	-	Reserved	
2	RW		Select between UTMI (1) and ULPI (0) mode.	XCORE_CTRL0_USB_MODE
1	RW		Enable the ULPI Hardware support module	XCORE_CTRL0_USB_ENABLE
0	RO	-	Reserved	

B.4 xCORE Tile boot status

`BOOT_CONFIG 0x03`

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

0x03: xCORE Tile boot status

Bits	Perm	Init	Description	Identifier
31:24	RO	-	Reserved	
23:16	RO		Processor number.	BOOT_CONFIG_PROCESSOR
15:9	RO	-	Reserved	
8	RO		Overwrite BOOT_MODE.	BOOT_CONFIG_SECURE_BOOT
7:6	RO	-	Reserved	
5	RO		Indicates if core1 has been powered off	BOOT_CONFIG_CORE1_POWER_DOWN_N
4	RO		Cause the ROM to not poll the OTP for correct read levels	BOOT_CONFIG_DISABLE_OTP_POLL
3	RO		Boot ROM boots from RAM	BOOT_CONFIG_BOOT_FROM_RAM
2	RO		Boot ROM boots from JTAG	BOOT_CONFIG_BOOT_FROM_JTAG
1:0	RO		The boot PLL mode pin value.	BOOT_CONFIG_PLL_MODE_PINS

B.5 Security configuration

SECURITY_CONFIG 0x05

Copy of the security register as read from OTP.

0x05: Security configuration

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

B.6 Ring Oscillator Control

RING_OSC_CTRL 0x06

There are four free-running oscillators that clock four counters. The oscillators can be started and stopped using this register. The counters should only be read when the ring oscillator has been stopped for at least 10 core clock cycles (this can be achieved by inserting two nop instructions between the SETPS and GETPS). The counter values can be read using four subsequent registers. The ring oscillators are asynchronous to the xCORE tile clock and can be used as a source of random bits.

0x06: Ring Oscillator Control

Bits	Perm	Init	Description	Identifier
31:2	RO	-	Reserved	
1	RW	0	Core ring oscillator enable.	RING_OSC_CORE_ENABLE
0	RW	0	Peripheral ring oscillator enable.	RING_OSC_PERPH_ENABLE

B.7 Ring Oscillator Value

RING_OSC_DATA0 0x07

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

0x07: Ring Oscillator Value

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

B.8 Ring Oscillator Value

RING_OSC_DATA1 0x08

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

0x08: Ring Oscillator Value

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

B.9 Ring Oscillator Value

RING_OSC_DATA2 0x09

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

0x09: Ring Oscillator Value

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

B.10 Ring Oscillator Value

RING_OSC_DATA3 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	Identifier
31:16	RO	-	Reserved	
15:0	RO	0	Ring oscillator Counter data.	RING_OSC_DATA

B.11 RAM size

RAM_SIZE 0x0C

The size of the RAM in bytes

0x0C: RAM size

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

B.12 Debug SSR

`DBG_SSR 0x10`

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

0x10: Debug SSR

Bits	Perm	Init	Description	Identifier
31:11	RO	-	Reserved	
10	DRW		Address space indentifier	SR_QUEUE
9	DRW		Determines the issue mode (DI bit) upon Kernel Entry after Exception or Interrupt.	SR_KEDI
8	RO		Determines the issue mode (DI bit).	SR_DI
7	DRW		When 1 the thread is in fast mode and will continually issue.	SR_FAST
6	DRW		When 1 the thread is paused waiting for events, a lock or another resource.	SR_WAITING
5	RO	-	Reserved	
4	DRW		1 when in kernel mode.	SR_INK
3	DRW		1 when in an interrupt handler.	SR_ININT
2	DRW		1 when in an event enabling sequence.	SR_INENB
1	DRW		When 1 interrupts are enabled for the thread.	SR_IEBLE
0	DRW		When 1 events are enabled for the thread.	SR_EEBLE

B.13 Debug SPC

`DBG_SPC 0x11`

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

0x11: Debug SPC

Bits	Perm	Init	Description	Identifier
31:0	DRW		Value.	ALL_BITS

B.14 Debug SSP

`DBG_SSP 0x12`

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

0x12: Debug SSP

Bits	Perm	Init	Description	Identifier
31:0	DRW		Value.	ALL_BITS

B.15 DGETREG operand 1

`DBG_T_NUM 0x13`

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

0x13: DGETREG operand 1

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

B.16 DGETREG operand 2 DBG_T_REG 0x14

Register number to be read by DGETREG

0x14: DGETREG operand 2

Bits	Perm	Init	Description	Identifier
31:5	RO	-	Reserved	
4:0	DRW		Register number to be read	DBG_T_REG_REG

B.17 Debug interrupt type DBG_TYPE 0x15

Register that specifies what activated the debug interrupt.

0x15: Debug interrupt type

Bits	Perm	Init	Description	Identifier
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.	DBG_TYPE_HW_NUM
15:8	DRW		Number of thread which caused the debug interrupt (always 0 in the case of =HOST=).	DBG_TYPE_T_NUM
7:3	RO	-	Reserved	
2:0	DRW	0	Indicates the cause of the debug interrupt 1: Host initiated a debug interrupt through JTAG 2: Program executed a DCALL instruction 3: Instruction breakpoint 4: Data watch point 5: Resource watch point	DBG_TYPE_CAUSE

B.18 Debug interrupt data DBG_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.

0x16: Debug interrupt data

Bits	Perm	Init	Description	Identifier
31:0	DRW		Value.	ALL_BITS

B.19 Debug core control

DBG_RUN_CTRL 0x18

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

0x18: Debug core control

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

B.20 Debug scratch

DBG_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	Identifier
31:0	DRW		Value.	ALL_BITS

B.21 Instruction breakpoint address

DBG_IBREAK_ADDR 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	Identifier
31:0	DRW		Value.	ALL_BITS

B.22 Instruction breakpoint control

DBG_IBREAK_CTRL 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	Identifier
31:24	RO	-	Reserved	
23:16	DRW	0	A bit for each thread in the machine allowing the breakpoint to be enabled individually for each thread.	BRK_THREADS
15:2	RO	-	Reserved	
1	DRW	0	When 0 break when PC == IBREAK_ADDR. When 1 = break when PC != IBREAK_ADDR.	IBRK_CONDITION
0	DRW	0	When 1 the instruction breakpoint is enabled.	BRK_ENABLE

B.23 Data watchpoint address 1 DBG_DWATCH_ADDR1 0x50 .. 0x53

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

0x50 .. 0x53: Data watchpoint address 1

Bits	Perm	Init	Description	Identifier
31:0	DRW		Value.	ALL_BITS

B.24 Data watchpoint address 2 DBG_DWATCH_ADDR2 0x60 .. 0x63

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

0x60 .. 0x63: Data watchpoint address 2

Bits	Perm	Init	Description	Identifier
31:0	DRW		Value.	ALL_BITS

B.25 Data breakpoint control register DBG_DWATCH_CTRL 0x70 .. 0x73

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

0x70 .. 0x73: Data breakpoint control register

Bits	Perm	Init	Description	Identifier
31:24	RO	-	Reserved	
23:16	DRW	0	A bit for each thread in the machine allowing the breakpoint to be enabled individually for each thread.	BRK_THREADS
15:3	RO	-	Reserved	
2	DRW	0	When 1 the breakpoints will be be triggered on loads.	BRK_LOAD
1	DRW	0	Determines the break condition: 0 = A AND B, 1 = A OR B.	DBRK_CONDITION
0	DRW	0	When 1 the instruction breakpoint is enabled.	BRK_ENABLE

B.26 Resources breakpoint mask DBG_RWATCH_ADDR1 0x80 .. 0x83

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

0x80 .. 0x83: Resources breakpoint mask

Bits	Perm	Init	Description	Identifier
31:0	DRW		Value.	ALL_BITS

B.27 Resources breakpoint value DBG_RWATCH_ADDR2 0x90 .. 0x93

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

0x90 .. 0x93: Resources breakpoint value

Bits	Perm	Init	Description	Identifier
31:0	DRW		Value.	ALL_BITS

B.28 Resources breakpoint control register DBG_RWATCH_CTRL 0x9C .. 0x9F

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

0x9C .. 0x9F: Resources breakpoint control register

Bits	Perm	Init	Description	Identifier
31:24	RO	-	Reserved	
23:16	DRW	0	A bit for each thread in the machine allowing the breakpoint to be enabled individually for each thread.	BRK_THREADS
15:2	RO	-	Reserved	
1	DRW	0	When 0 break when condition A is met. When 1 = break when condition B is met.	RBRK_CONDITION
0	DRW	0	When 1 the instruction breakpoint is enabled.	BRK_ENABLE

C Tile Configuration

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

The identifiers for the registers needs a prefix "`XS1_PSWITCH_`" and a postfix "`_NUM`", and are declared in "`xs1.h`"

Figure 35: Summary

Number	Perm	Description	Register identifier
0x00	CRO	Device identification	DEVICE_ID0
0x01	CRO	xCORE Tile description 1	DEVICE_ID1
0x02	CRO	xCORE Tile description 2	DEVICE_ID2
0x04	CRW	Control PSwitch permissions to debug registers	DBG_CTRL
0x05	CRW	Cause debug interrupts	DBG_INT
0x06	CRW	xCORE Tile clock divider	PLL_CLK_DIVIDER
0x07	CRO	Security configuration	SECU_CONFIG
0x20 .. 0x27	CRW	Debug scratch	DBG_SCRATCH
0x40	CRO	PC of logical core 0	T0_PC
0x41	CRO	PC of logical core 1	T1_PC
0x42	CRO	PC of logical core 2	T2_PC
0x43	CRO	PC of logical core 3	T3_PC
0x44	CRO	PC of logical core 4	T4_PC
0x45	CRO	PC of logical core 5	T5_PC
0x46	CRO	PC of logical core 6	T6_PC
0x47	CRO	PC of logical core 7	T7_PC
0x60	CRO	SR of logical core 0	T0_SR
0x61	CRO	SR of logical core 1	T1_SR
0x62	CRO	SR of logical core 2	T2_SR
0x63	CRO	SR of logical core 3	T3_SR
0x64	CRO	SR of logical core 4	T4_SR
0x65	CRO	SR of logical core 5	T5_SR
0x66	CRO	SR of logical core 6	T6_SR
0x67	CRO	SR of logical core 7	T7_SR

C.1 Device identification

DEVICE_ID0 0x00

This register identifies the xCORE Tile

0x00: Device identification

Bits	Perm	Init	Description	Identifier
31:24	CRO		Processor ID of this XCore.	DEVICE_ID0_PID
23:16	CRO		Number of the node in which this XCore is located.	DEVICE_ID0_NODE
15:8	CRO		XCore revision.	DEVICE_ID0_REVISION
7:0	CRO		XCore version.	DEVICE_ID0_VERSION

C.2 xCORE Tile description 1 DEVICE_ID1 0x01

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

0x01: xCORE Tile description 1

Bits	Perm	Init	Description	Identifier
31:24	CRO		Number of channel ends.	DEVICE_ID1_NUM_CHANENDS
23:16	CRO		Number of the locks.	DEVICE_ID1_NUM_LOCKS
15:8	CRO		Number of synchronisers.	DEVICE_ID1_NUM_SYNCS
7:0	RO	-	Reserved	

C.3 xCORE Tile description 2 DEVICE_ID2 0x02

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

0x02: xCORE Tile description 2

Bits	Perm	Init	Description	Identifier
31:16	RO	-	Reserved	
15:8	CRO		Number of clock blocks.	DEVICE_ID2_NUM_CLKBLKS
7:0	CRO		Number of timers.	DEVICE_ID2_NUM_TIMERS

C.4 Control PSwitch permissions to debug registers DBG_CTRL 0x04

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

0x04: Control PSwitch permissions to debug registers

Bits	Perm	Init	Description	Identifier
31	CRW	0	When 1 the PSwitch is restricted to RO access to all CRW registers from SSwitch, XCore(PS_DBG_Scratch) and JTAG	DBG_CTRL_PSWITCH_RO
30:1	RO	-	Reserved	
0	CRW	0	When 1 the PSwitch is restricted to RO access to all CRW registers from SSwitch	DBG_CTRL_PSWITCH_RO_EXT

C.5 Cause debug interrupts

DBG_INT 0x05

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

0x05: Cause debug interrupts

Bits	Perm	Init	Description	Identifier
31:2	RO	-	Reserved	
1	CRW	0	1 when the processor is in debug mode.	DBG_INT_IN_DBG
0	CRW	0	Request a debug interrupt on the processor.	DBG_INT_REQ_DBG

C.6 xCORE Tile clock divider

PLL_CLK_DIVIDER 0x06

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

0x06: xCORE Tile clock divider

Bits	Perm	Init	Description	Identifier
31	CRW	0	Clock disable. Writing '1' will remove the clock to the tile.	PLL_CLK_DISABLE
30:16	RO	-	Reserved	
15:0	CRW	0	Clock divider.	PLL_CLK_DIVIDER

C.7 Security configuration

SECU_CONFIG 0x07

Copy of the security register as read from OTP.

0x07: Security configuration

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

C.8 Debug scratch

DBG_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	Identifier
31:0	CRW		Value.	ALL_BITS

C.9 PC of logical core 0

T0_PC 0x40

Value of the PC of logical core 0.

0x40: PC of logical core 0

Bits	Perm	Init	Description	Identifier
31:0	CRO		Value.	ALL_BITS

C.10 PC of logical core 1

T1_PC 0x41

Value of the PC of logical core 1.

0x41: PC of logical core 1

Bits	Perm	Init	Description	Identifier
31:0	CRO		Value.	ALL_BITS

C.11 PC of logical core 2 T2_PC 0x42

Value of the PC of logical core 2.

0x42: PC of logical core 2

Bits	Perm	Init	Description	Identifier
31:0	CRO		Value.	ALL_BITS

C.12 PC of logical core 3 T3_PC 0x43

Value of the PC of logical core 3.

0x43: PC of logical core 3

Bits	Perm	Init	Description	Identifier
31:0	CRO		Value.	ALL_BITS

C.13 PC of logical core 4 T4_PC 0x44

Value of the PC of logical core 4.

0x44: PC of logical core 4

Bits	Perm	Init	Description	Identifier
31:0	CRO		Value.	ALL_BITS

C.14 PC of logical core 5 T5_PC 0x45

Value of the PC of logical core 5.

0x45: PC of logical core 5

Bits	Perm	Init	Description	Identifier
31:0	CRO		Value.	ALL_BITS

C.15 PC of logical core 6 T6_PC 0x46

Value of the PC of logical core 6.

0x46: PC of logical core 6

Bits	Perm	Init	Description	Identifier
31:0	CRO		Value.	ALL_BITS

C.16 PC of logical core 7

T7_PC 0x47

Value of the PC of logical core 7.

0x47: PC of logical core 7

Bits	Perm	Init	Description	Identifier
31:0	CRO		Value.	ALL_BITS

C.17 SR of logical core 0

T0_SR 0x60

Value of the SR of logical core 0

0x60: SR of logical core 0

Bits	Perm	Init	Description	Identifier
31:0	CRO		Value.	ALL_BITS

C.18 SR of logical core 1

T1_SR 0x61

Value of the SR of logical core 1

0x61: SR of logical core 1

Bits	Perm	Init	Description	Identifier
31:0	CRO		Value.	ALL_BITS

C.19 SR of logical core 2

T2_SR 0x62

Value of the SR of logical core 2

0x62: SR of logical core 2

Bits	Perm	Init	Description	Identifier
31:0	CRO		Value.	ALL_BITS

C.20 SR of logical core 3

T3_SR 0x63

Value of the SR of logical core 3

0x63: SR of logical core 3

Bits	Perm	Init	Description	Identifier
31:0	CRO		Value.	ALL_BITS

C.21 SR of logical core 4

T4_SR 0x64

Value of the SR of logical core 4

0x64: SR of logical core 4

Bits	Perm	Init	Description	Identifier
31:0	CRO		Value.	ALL_BITS

C.22 SR of logical core 5

T5_SR 0x65

Value of the SR of logical core 5

0x65: SR of logical core 5

Bits	Perm	Init	Description	Identifier
31:0	CRO		Value.	ALL_BITS

C.23 SR of logical core 6

T6_SR 0x66

Value of the SR of logical core 6

0x66: SR of logical core 6

Bits	Perm	Init	Description	Identifier
31:0	CRO		Value.	ALL_BITS

C.24 SR of logical core 7

T7_SR 0x67

Value of the SR of logical core 7

0x67: SR of logical core 7

Bits	Perm	Init	Description	Identifier
31:0	CRO		Value.	ALL_BITS

D Node Configuration

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

The identifiers for the registers needs a prefix "`XS1_SSWITCH_`" and a postfix "`_NUM`", and are declared in "`xs1.h`"

Figure 36: Summary

Number	Perm	Description	Register identifier
0x00	RO	Device identification	DEVICE_ID0
0x01	RO	System switch description	DEVICE_ID1
0x04	RW	Switch configuration	NODE_CONFIG
0x05	RW	Switch node identifier	NODE_ID
0x06	RW	PLL settings	PLL_CTL
0x07	RW	System switch clock divider	CLK_DIVIDER
0x08	RW	Reference clock	REF_CLK_DIVIDER
0x09	R	System JTAG device ID register	JTAG_DEVICE_ID
0x0A	R	System USERCODE register	JTAG_USERCODE
0x0C	RW	Directions 0-7	DIMENSION_DIRECTION0
0x0D	RW	Directions 8-15	DIMENSION_DIRECTION1
0x10	RW	DEBUG_N configuration, tile 0	XCORE0_GLOBAL_DEBUG_CONFIG
0x11	RW	DEBUG_N configuration, tile 1	XCORE1_GLOBAL_DEBUG_CONFIG
0x1F	RO	Debug source	GLOBAL_DEBUG_SOURCE
0x20 .. 0x28	RW	Link status, direction, and network	SLINK
0x40 .. 0x47	RO	PLink status and network	PLINK
0x80 .. 0x88	RW	Link configuration and initialization	XLINK
0xA0 .. 0xA7	RW	Static link configuration	XSTATIC

D.1 Device identification

DEVICE_ID0 0x00

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

0x00: Device identification

Bits	Perm	Init	Description	Identifier
31:24	RO	-	Reserved	
23:16	RO		Sampled values of BootCtl pins on Power On Reset.	SS_DEVICE_ID0_BOOT_CTRL
15:8	RO		SSwitch revision.	SS_DEVICE_ID0_REVISION
7:0	RO		SSwitch version.	SS_DEVICE_ID0_VERSION

D.2 System switch description

DEVICE_ID1 0x01

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

0x01: System switch description

Bits	Perm	Init	Description	Identifier
31:24	RO	-	Reserved	
23:16	RO		Number of SLinks on the SSwitch.	SS_DEVICE_ID1_NUM_SLINKS
15:8	RO		Number of processors on the SSwitch.	SS_DEVICE_ID1_NUM_PROCESSORS
7:0	RO		Number of processors on the device.	SS_DEVICE_ID1_NUM_PLINKS_PER_PROC

D.3 Switch configuration

NODE_CONFIG 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	Identifier
31	RW	0	0 = SSCTL registers have write access. 1 = SSCTL registers can not be written to.	SS_NODE_CONFIG_DISABLE_SSCTL_UPDATE
30:9	RO	-	Reserved	
8	RW	0	0 = PLL_CTL_REG has write access. 1 = PLL_CTL_REG can not be written to.	SS_NODE_CONFIG_DISABLE_PLL_CTL_REG
7:1	RO	-	Reserved	
0	RW	0	0 = 2-byte headers, 1 = 1-byte headers (reset as 0).	SS_NODE_CONFIG_HEADERS

D.4 Switch node identifier

NODE_ID 0x05

This register contains the node identifier.

0x05: Switch node identifier

Bits	Perm	Init	Description	Identifier
31:16	RO	-	Reserved	
15:0	RW	0	The unique ID of this node.	SS_NODE_ID_ID

D.5 PLL settings

PLL_CTL 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	Identifier
31	RW		If set to 1, the chip will not be reset	SS_PLL_CTL_NRESET
30	RW		If set to 1, the chip will not wait for the PLL to re-lock. Only use this if a gradual change is made to the PLL	SS_PLL_CTL_NLOCK
29	DW		If set to 1, set the PLL to be bypassed	SS_TEST_MODE_PLL_BYPASS
28	DW		If set to 1, set the boot mode to boot from JTAG	SS_TEST_MODE_BOOT_JTAG
27:26	RO	-	Reserved	
25:23	RW		Output divider value range from 0 (8'h0) to 7 (8'h7). OD value.	SS_PLL_CTL_POST_DIVISOR
22:21	RO	-	Reserved	
20:8	RW		Feedback multiplication ratio, range from 0 (8'h0) to 4095 (8'h3FF). F value.	SS_PLL_CTL_FEEDBACK_MUL
7	RO	-	Reserved	
6:0	RW		Oscilator input divider value range from 0 (8'h0) to 63 (8'h3F). R value.	SS_PLL_CTL_INPUT_DIVISOR

D.6 System switch clock divider

CLK_DIVIDER 0x07

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

0x07: System switch clock divider

Bits	Perm	Init	Description	Identifier
31:16	RO	-	Reserved	
15:0	RW	0	SSwitch clock generation	SS_CLK_DIVIDER_CLK_DIV

D.7 Reference clock

REF_CLK_DIVIDER 0x08

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

0x08: Reference clock

Bits	Perm	Init	Description	Identifier
31:16	RO	-	Reserved	
15:0	RW	3	Software ref. clock divider	SS_SSWITCH_REF_CLK_DIV

D.8 System JTAG device ID register

`JTAG_DEVICE_ID 0x09`

0x09: System JTAG device ID register

Bits	Perm	Init	Description	Identifier
31:28	RO			SS_JTAG_DEVICE_ID_VERSION
27:12	RO			SS_JTAG_DEVICE_ID_PART_NUM
11:1	RO			SS_JTAG_DEVICE_ID_MANU_ID
0	RO			SS_JTAG_DEVICE_ID_CONST_VAL

D.9 System USERCODE register

`JTAG_USERCODE 0x0A`

0x0A: System USERCODE register

Bits	Perm	Init	Description	Identifier
31:18	RO		JTAG USERCODE value programmed into OTP SR	SS_JTAG_USERCODE_OTP
17:0	RO		metal fixable ID code	SS_JTAG_USERCODE_MASKID

D.10 Directions 0-7

`DIMENSION_DIRECTION0 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.

0x0C: Directions 0-7

Bits	Perm	Init	Description	Identifier
31:28	RW	0	The direction for packets whose dimension is 7.	DIM7_DIR
27:24	RW	0	The direction for packets whose dimension is 6.	DIM6_DIR
23:20	RW	0	The direction for packets whose dimension is 5.	DIM5_DIR
19:16	RW	0	The direction for packets whose dimension is 4.	DIM4_DIR
15:12	RW	0	The direction for packets whose dimension is 3.	DIM3_DIR
11:8	RW	0	The direction for packets whose dimension is 2.	DIM2_DIR
7:4	RW	0	The direction for packets whose dimension is 1.	DIM1_DIR
3:0	RW	0	The direction for packets whose dimension is 0.	DIM0_DIR

D.11 Directions 8-15

`DIMENSION_DIRECTION1 0x0D`

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

Bits	Perm	Init	Description	Identifier
31:28	RW	0	The direction for packets whose dimension is F.	DIMF_DIR
27:24	RW	0	The direction for packets whose dimension is E.	DIME_DIR
23:20	RW	0	The direction for packets whose dimension is D.	DIMD_DIR
19:16	RW	0	The direction for packets whose dimension is C.	DIMC_DIR
15:12	RW	0	The direction for packets whose dimension is B.	DIMB_DIR
11:8	RW	0	The direction for packets whose dimension is A.	DIMA_DIR
7:4	RW	0	The direction for packets whose dimension is 9.	DIM9_DIR
3:0	RW	0	The direction for packets whose dimension is 8.	DIM8_DIR

0x0D: Directions 8-15

D.12 DEBUG_N configuration, tile 0 XCORE0_GLOBAL_DEBUG_CONFIG 0x10

Configures the behavior of the DEBUG_N pin.

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

0x10: DEBUG_N configuration, tile 0

D.13 DEBUG_N configuration, tile 1 XCORE1_GLOBAL_DEBUG_CONFIG 0x11

Configures the behavior of the DEBUG_N pin.

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

0x11: DEBUG_N configuration, tile 1

D.14 Debug source GLOBAL_DEBUG_SOURCE 0x1F

Contains the source of the most recent debug event.

0x1F: Debug source

Bits	Perm	Init	Description	Identifier
31:5	RO	-	Reserved	
4	RW		If set, external pin, is the source of last GlobalDebug event.	GLOBAL_DEBUG_SOURCE_EXTERNAL_PAD_INDEBUG
3:2	RO	-	Reserved	
1	RW		If set, XCore1 is the source of last GlobalDebug event.	GLOBAL_DEBUG_SOURCE_XCORE1_INDEBUG
0	RW		If set, XCore0 is the source of last GlobalDebug event.	GLOBAL_DEBUG_SOURCE_XCORE0_INDEBUG

D.15 Link status, direction, and network

SLINK 0x20 .. 0x28

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 0..7.

0x20 .. 0x28: Link status, direction, and network

Bits	Perm	Init	Description	Identifier
31:26	RO	-	Reserved	
25:24	RO		Identify the SRC_TARGET type 0 - SLink, 1 - PLink, 2 - SSCTL, 3 - Undefine.	SLINK_SRC_TARGET_TYPE
23:16	RO		When the link is in use, this is the destination link number to which all packets are sent.	SLINK_SRC_TARGET_ID
15:12	RO	-	Reserved	
11:8	RW	0	The direction that this link operates in.	LINK_DIRECTION
7:6	RO	-	Reserved	
5:4	RW	0	Determines the network to which this link belongs, reset as 0.	LINK_NETWORK
3	RO	-	Reserved	
2	RO		1 when the current packet is considered junk and will be thrown away.	LINK_JUNK
1	RO		1 when the dest side of the link is in use.	LINK_DST_INUSE
0	RO		1 when the source side of the link is in use.	LINK_SRC_INUSE

D.16 PLink status and network

PLINK 0x40 .. 0x47

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

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

0x40 .. 0x47: PLink status and network

D.17 Link configuration and initialization

`XLINK 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	Identifier
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.	XLINK_ENABLE
30	RW	0	0: operate in 2 wire mode; 1: operate in 5 wire mode	XLINK_WIDE
29:28	RO	-	Reserved	
27	RO		Rx buffer overflow or illegal token encoding received.	XLINK_RX_ERROR
26	RO	0	This end of the xlink has issued credit to allow the remote end to transmit	RX_CREDIT
25	RO	0	This end of the xlink has credit to allow it to transmit.	TX_CREDIT
24	WO		Clear this end of the xlink's credit and issue a HELLO token.	XLINK_HELLO
23	WO		Reset the receiver. The next symbol that is detected will be the first symbol in a token.	XLINK_RX_RESET
22	RO	-	Reserved	
21:11	RW	0	Specify min. number of idle system clocks between two continuous symbols witin a transmit token -1.	XLINK_INTRA_TOKEN_DELAY
10:0	RW	0	Specify min. number of idle system clocks between two continuous transmit tokens -1.	XLINK_INTER_TOKEN_DELAY

0x80 .. 0x88: Link configuration and initialization

D.18 Static link configuration

`XSTATIC 0xA0 .. 0xA7`

These registers are used for static (ie, non-routed) links. When a link is made static, all traffic is forwarded to the designated channel end and no routing is attempted. The registers control links C, D, A, B, G, H, E, and F in that order.

0xA0 .. 0xA7: Static link configuration

Bits	Perm	Init	Description	Identifier
31	RW	0	Enable static forwarding.	XSTATIC_ENABLE
30:9	RO	-	Reserved	
8	RW	0	The destination processor on this node that packets received in static mode are forwarded to.	XSTATIC_DEST_PROC
7:5	RO	-	Reserved	
4:0	RW	0	The destination channel end on this node that packets received in static mode are forwarded to.	XSTATIC_DEST_CHAN_END

E USB Node Configuration

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

Figure 37: Summary

Number	Perm	Description	Register identifier
0x00	RO	Device identification register	DEV_ID
0x04	RW	Node configuration register	NODE_CFG
0x05	RW	Node identifier	NODE_ID_SCTH
0x51	RW	System clock frequency	SYS_CLK_FREQ
0x80	RW	Link Control and Status	LINK_CTRL

E.1 Device identification register DEV_ID 0x00

This register contains version information, and information on power-on behavior.

0x00: Device identification register

Bits	Perm	Init	Description	Identifier
31:24	RO	0x0F	Chip identifier	GLX_CFG_CHIP_ID
23:16	RO	-	Reserved	
15:8	RO	0x02	Revision number of the USB block	GLX_CFG_REVISION
7:0	RO	0x00	Version number of the USB block	GLX_CFG_VERSION

E.2 Node configuration register NODE_CFG 0x04

This register is used to set the communication model to use (1 or 3 byte headers), and to prevent any further updates.

0x04: Node configuration register

Bits	Perm	Init	Description	Identifier
31	RW	0	Set to 1 to disable further updates to the node configuration and link control and status registers.	GLX_CFG_DISABLE_UPDATES
30:1	RO	-	Reserved	
0	RW	0	Header mode. 0: 3-byte headers; 1: 1-byte headers.	GLX_CFG_HDR_MODE

E.3 Node identifier

NODE_ID_SCTH 0x05

0x05: Node identifier

Bits	Perm	Init	Description	Identifier
31:16	RO	-	Reserved	
15:0	RW	0	16-bit node identifier. This does not need to be set, and is present for compatibility with XS1-switches.	GLX_CFG_NODE_ID_SCTH

E.4 System clock frequency

SYS_CLK_FREQ 0x51

0x51: System clock frequency

Bits	Perm	Init	Description	Identifier
31:7	RO	-	Reserved	
6:0	RW	25	Oscillator clock frequency in MHz rounded up to the nearest integer value. Only values between 5 and 100 MHz are valid - writes outside this range are ignored and will be NACKed. This field must be set on start up of the device and any time that the input oscillator clock frequency is changed. It must contain the system clock frequency in MHz rounded up to the nearest integer value.	GLX_CFG_SYS_CLK_FREQ

E.5 Link Control and Status

LINK_CTRL 0x80

0x80: Link Control and Status

Bits	Perm	Init	Description	Identifier
31:28	RO	-	Reserved	
27	RO		Rx buffer overflow or illegal token encoding received.	XLINK_RX_ERROR
26	RO	0	This end of the xlink has issued credit to allow the remote end to transmit	RX_CREDIT
25	RO	0	This end of the xlink has credit to allow it to transmit.	TX_CREDIT
24	WO		Clear this end of the xlink's credit and issue a HELLO token.	XLINK_HELLO
23	WO		Reset the receiver. The next symbol that is detected will be the first symbol in a token.	XLINK_RX_RESET
22	RO	-	Reserved	
21:11	RW	1	Specify min. number of idle system clocks between two continuous symbols witin a transmit token -1.	XLINK_INTRA_TOKEN_DELAY
10:0	RW	1	Specify min. number of idle system clocks between two continuous transmit tokens -1.	XLINK_INTER_TOKEN_DELAY

F USB PHY Configuration

The USB PHY is connected to the ports shown in section 10.

The *USB PHY* is peripheral 1. The control registers are accessed using 32-bit reads and writes (use `write_periph_32(device, 1, ...)` and `read_periph_32(device, 1, ...)` for reads and writes).

Figure 38: Summary

Number	Perm	Description	Register identifier
0x00	WO	UIFM reset	GLX_PER_UIFM_RESET
0x04	RW	UIFM IFM control	GLX_PER_UIFM_CONTROL
0x08	RW	UIFM Device Address	GLX_PER_UIFM_DEVICE_ADDRESS
0x0C	RW	UIFM functional control	GLX_PER_UIFM_FUNC_CONTROL
0x10	RW	UIFM on-the-go control	GLX_PER_UIFM_OTG_CONTROL
0x14	RO	UIFM on-the-go flags	GLX_PER_UIFM_OTG_FLAGS
0x18	RW	UIFM Serial Control	GLX_PER_UIFM_SERIAL_MODE
0x1C	RW	UIFM signal flags	GLX_PER_UIFM_IFM_FLAGS
0x20	RW	UIFM Sticky flags	GLX_PER_UIFM_FLAGS_STICKY
0x24	RW	UIFM port masks	GLX_PER_UIFM_MASK
0x28	RW	UIFM SOF value	GLX_PER_UIFM_SOFCOUNT
0x2C	RO	UIFM PID	GLX_PER_UIFM_PID
0x30	RO	UIFM Endpoint	GLX_PER_UIFM_ENDPOINT
0x34	RW	UIFM Endpoint match	GLX_PER_UIFM_ENDPOINT_MATCH
0x38	RW	OTG Flags mask	GLX_PER_UIFM_OTG_FLAGS_MASK
0x3C	RW	UIFM power signalling	GLX_PER_UIFM_PWRSIG
0x40	RW	UIFM PHY control	GLX_PER_UIFM_PHY_CONTROL

F.1 UIFM reset

GLX_PER_UIFM_RESET 0x00

A write to this register with any data resets all UIFM state, but does not otherwise affect the phy.

0x00: UIFM reset

Bits	Perm	Init	Description	Identifier
31:0	WO		Value.	ALL_BITS

F.2 UIFM IFM control

GLX_PER_UIFM_CONTROL 0x04

General settings of the UIFM IFM state machine.

0x04: UIFM IFM control

Bits	Perm	Init	Description	Identifier
31:8	RO	-	Reserved	
7	RW	0	Set to 1 to enable XEVACKMODE mode.	UIFM_IFM_CONTROL_XEVACKMODE
6	RW	0	Set to 1 to enable SOFISTOKEN mode.	UIFM_IFM_CONTROL_SOFISTOKEN
5	RW	0	Set to 1 to enable UIFM power signalling mode.	UIFM_IFM_CONTROL_PWRSIGMODE
4	RW	0	Set to 1 to enable IF timing mode.	UIFM_IFM_CONTROL_IFTIMINGMODE
3	RO	-	Reserved	
2	RW	0	Set to 1 to enable UIFM linestate decoder.	UIFM_IFM_CONTROL_DECODELINESTATE
1	RW	0	Set to 1 to enable UIFM CHECKTOKENS mode.	UIFM_IFM_CONTROL_CHECKTOKENS
0	RW	0	Set to 1 to enable UIFM DOTOKENS mode.	UIFM_IFM_CONTROL_DOTOKENS

F.3 UIFM Device Address

GLX_PER_UIFM_DEVICE_ADDRESS 0x08

The device address whose packets should be received. 0 until enumeration, it should be set to the assigned value after enumeration.

0x08: UIFM Device Address

Bits	Perm	Init	Description	Identifier
31:7	RO	-	Reserved	
6:0	RW	0	The enumerated USB device address must be stored here. Only packets to this address are passed on.	UIFM_DEVICE_ADDRESS_ADDRESS

F.4 UIFM functional control

GLX_PER_UIFM_FUNC_CONTROL 0x0C

0x0C: UIFM functional control

Bits	Perm	Init	Description	Identifier
31:5	RO	-	Reserved	
4:2	RW	1	Set to 0 to disable UIFM to UTMI+ OPMODE mode.	UIFM_FUNC_CONTROL_OPMODE
1	RW	1	Set to 1 to switch UIFM to UTMI+ TERMSELECT mode.	UIFM_FUNC_CONTROL_TERMSELECT
0	RW	1	Set to 1 to switch UIFM to UTMI+ XCVRSELECT mode.	UIFM_FUNC_CONTROL_XCVRSELECT

F.5 UIFM on-the-go control

GLX_PER_UIFM_OTG_CONTROL 0x10

This register is used to negotiate an on-the-go connection.

Bits	Perm	Init	Description	Identifier
31:8	RO	-	Reserved	
7	RW	0	Set to 1 to switch UIFM to EXTVBUSIND mode.	UIFM_OTG_CONTROL_EXTVBUSIND
6	RW	0	Set to 1 to switch UIFM to DRVVBUSEXT mode.	UIFM_OTG_CONTROL_DRVVBUSEXT
5	RO	-	Reserved	
4	RW	0	Set to 1 to switch UIFM to UTMI+ CHRGVBUS mode.	UIFM_OTG_CONTROL_CHRGVBUS
3	RW	0	Set to 1 to switch UIFM to UTMI+ DISCHRGVBUS mode.	UIFM_OTG_CONTROL_DISCHRGVBUS
2	RW	0	Set to 1 to switch UIFM to UTMI+ DMPULLDOWN mode.	UIFM_OTG_CONTROL_DMPULLDOWN
1	RW	0	Set to 1 to switch UIFM to UTMI+ DPPULLDOWN mode.	UIFM_OTG_CONTROL_DPPULLDOWN
0	RW	0	Set to 1 to switch UIFM to IDPULLUP mode.	UIFM_OTG_CONTROL_IDPULLUP

0x10: UIFM on-the-go control

F.6 UIFM on-the-go flags

GLX_PER_UIFM_OTG_FLAGS 0x14

Status flags used for on-the-go negotiation

Bits	Perm	Init	Description	Identifier
31:6	RO	-	Reserved	
5	RO	0	Value of UTMI+ Bvalid flag.	UIFM_OTG_FLAGS_SESSVLDB
4	RO	0	Value of UTMI+ IDGND flag.	UIFM_OTG_FLAGS_NIDGND
3	RO	0	Value of UTMI+ HOSTDIS flag.	UIFM_OTG_FLAGS_HOSTDIS
2	RO	0	Value of UTMI+ VBUSVLD flag.	UIFM_OTG_FLAGS_VBUSVLD
1	RO	0	Value of UTMI+ SESSVLD flag.	UIFM_OTG_FLAGS_SESSVLD
0	RO	0	Value of UTMI+ SESSEND flag.	UIFM_OTG_FLAGS_SESSEND

0x14: UIFM on-the-go flags

F.7 UIFM Serial Control

`GLX_PER_UIFM_SERIAL_MODE 0x18`

0x18: UIFM Serial Control

Bits	Perm	Init	Description	Identifier
31:7	RO	-	Reserved	
6	RO	0	1 if UIFM is in UTMI+ RXRCV mode.	UIFM_SERIAL_MODE_RXRCV
5	RO	0	1 if UIFM is in UTMI+ RXDM mode.	UIFM_SERIAL_MODE_RXDM
4	RO	0	1 if UIFM is in UTMI+ RXDP mode.	UIFM_SERIAL_MODE_RXDP
3	RW	0	Set to 1 to switch UIFM to UTMI+ TXSE0 mode.	UIFM_SERIAL_MODE_TXSE0
2	RW	0	Set to 1 to switch UIFM to UTMI+ TXDATA mode.	UIFM_SERIAL_MODE_TXDAT
1	RW	1	Set to 0 to switch UIFM to UTMI+ TXENABLE mode.	UIFM_SERIAL_MODE_TXENN
0	RW	0	Set to 1 to switch UIFM to UTMI+ FSLSSERIAL mode.	UIFM_SERIAL_MODE_FSLSMODE

F.8 UIFM signal flags

`GLX_PER_UIFM_IFM_FLAGS 0x1C`

Set of flags that monitor line and error states. These flags normally clear on the next packet, but they may be made sticky by using PER_UIFM_FLAGS_STICKY, in which they must be cleared explicitly.

0x1C: UIFM signal flags

Bits	Perm	Init	Description	Identifier
31:7	RO	-	Reserved	
6	RW	0	Set to 1 when the UIFM decodes a token successfully (e.g. it passes CRC5, PID check and has matching device address).	UIFM_IFM_FLAGS_NEWTOKEN
5	RW	0	Set to 1 when linestate indicates an SE0 symbol.	UIFM_IFM_FLAGS_SE0
4	RW	0	Set to 1 when linestate indicates a K symbol.	UIFM_IFM_FLAGS_K
3	RW	0	Set to 1 when linestate indicates a J symbol.	UIFM_IFM_FLAGS_J
2	RW	0	Set to 1 if an incoming datapacket fails the CRC16 check.	UIFM_IFM_FLAGS_CRC16FAIL
1	RW	0	Set to the value of the UTMI_RXACTIVE input signal.	UIFM_IFM_FLAGS_RXACTIVE
0	RW	0	Set to the value of the UTMI_RXERROR input signal	UIFM_IFM_FLAGS_RXERROR

F.9 UIFM Sticky flags

`GLX_PER_UIFM_FLAGS_STICKY 0x20`

These bits define the sticky-ness of the bits in the UIFM IFM FLAGS register. A 1 means that bit will be sticky (hold its value until a 1 is written to that bitfield), or normal, in which case signal updates to the UIFM IFM FLAGS bits may be over-written by subsequent changes in those signals.

0x20: UIFM Sticky flags

Bits	Perm	Init	Description	Identifier
31:7	RO	-	Reserved	
6:0	RW	0	Stickyness for each flag.	UIFM_FLAGS_STICKY_STICKY

F.10 UIFM port masks

GLX_PER_UIFM_MASK 0x24

Set of masks that identify how port 1N, port 1O and port 1P are affected by changes to the flags in FLAGS

0x24: UIFM port masks

Bits	Perm	Init	Description	Identifier
31:24	RW	0	Bit mask that determines which flags in UIFM_IFM_FLAG[6:0] contribute to port 1?. If any flag listed in this bitmask is high, port 1? will be high.	UIFM_FLAGS_MASK_MASK3
23:16	RW	0	Bit mask that determines which flags in UIFM_IFM_FLAG[6:0] contribute to port 1P. If any flag listed in this bitmask is high, port 1P will be high.	UIFM_FLAGS_MASK_MASK2
15:8	RW	0	Bit mask that determines which flags in UIFM_IFM_FLAG[6:0] contribute to port 1O. If any flag listed in this bitmask is high, port 1O will be high.	UIFM_FLAGS_MASK_MASK1
7:0	RW	0	Bit mask that determines which flags in UIFM_IFM_FLAG[6:0] contribute to port 1N. If any flag listed in this bitmask is high, port 1N will be high.	UIFM_FLAGS_MASK_MASK0

F.11 UIFM SOF value

GLX_PER_UIFM_SOFCOUNT 0x28

USB Start-Of-Frame counter

0x28: UIFM SOF value

Bits	Perm	Init	Description	Identifier
31:11	RO	-	Reserved	
10:8	RW	0	Most significant 3 bits of SOF counter	UIFM_SOFCOUNT_COUNT2
7:0	RW	0	Least significant 8 bits of SOF counter	UIFM_SOFCOUNT_COUNT1

F.12 UIFM PID

GLX_PER_UIFM_PID 0x2C

The last USB packet identifier received

0x2C: UIFM PID

Bits	Perm	Init	Description	Identifier
31:4	RO	-	Reserved	
3:0	RO	0	Value of the last received PID.	UIFM_PID_PID

F.13 UIFM Endpoint

`GLX_PER_UIFM_ENDPOINT 0x30`

The last endpoint seen

0x30: UIFM Endpoint

Bits	Perm	Init	Description	Identifier
31:5	RO	-	Reserved	
4	RO	0	1 if endpoint contains a valid value.	UIFM_ENDPOINT_MATCH
3:0	RO	0	A copy of the last received endpoint.	UIFM_ENDPOINT_ENDPOINT

F.14 UIFM Endpoint match

`GLX_PER_UIFM_ENDPOINT_MATCH 0x34`

This register can be used to mark UIFM endpoints as special.

0x34: UIFM Endpoint match

Bits	Perm	Init	Description	Identifier
31:16	RO	-	Reserved	
15:0	RW	0	This register contains a bit for each endpoint. If its bit is set, the endpoint will be supplied on the RX port when ORed with 0x10.	UIFM_ENDPOINT_MATCH_MATCH

F.15 OTG Flags mask

`GLX_PER_UIFM_OTG_FLAGS_MASK 0x38`

0x38: OTG Flags mask

Bits	Perm	Init	Description	Identifier
31:0	RW	0	Data	OTG_FLAGS_MASK_DEFINED

F.16 UIFM power signalling

`GLX_PER_UIFM_PWRSIG 0x3C`

0x3C: UIFM power signalling

Bits	Perm	Init	Description	Identifier
31:9	RO	-	Reserved	
8	RW	0	Valid	UIFM_PWRSIG_VALID
7:0	RW	0	Data	UIFM_PWRSIG_DATA

F.17 UIFM PHY control

GLX_PER_UIFM_PHY_CONTROL 0x40

0x40: UIFM PHY control

Bits	Perm	Init	Description	Identifier
31:19	RO	-	Reserved	
18	RW	0	Set to 1 to disable pulldowns on ports 8A and 8B.	UIFM_PHY_CONTROL_PULLDOWN_DISABLE
17:14	RO	-	Reserved	
13	RW	0	After an auto-resume, this bit is set to indicate that the resume signalling was for reset (se0). Set to 0 to clear.	UIFM_PHY_CONTROL_RESUMESE0
12	RW	0	After an auto-resume, this bit is set to indicate that the resume signalling was for resume (K). Set to 0 to clear.	UIFM_PHY_CONTROL_RESUMEK
11:8	RW	0	Log-2 number of clocks before any linestate change is propagated.	UIFM_PHY_CONTROL_SE0FILTVAL
7	RW	0	Set to 1 to use the suspend controller handle to resume from suspend. Otherwise, the program has to poll the linestate_filt field in phy_teststatus.	UIFM_PHY_CONTROL_AUTORESUME
6:4	RW	0	Control the the conf1,2,3 input pins of the PHY.	UIFM_PHY_CONTROL_PHYCONF
3:0	RO	-	Reserved	

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 39 shows a decision diagram which explains what type of xSYS connectivity you need. The three subsections below explain the options in detail.

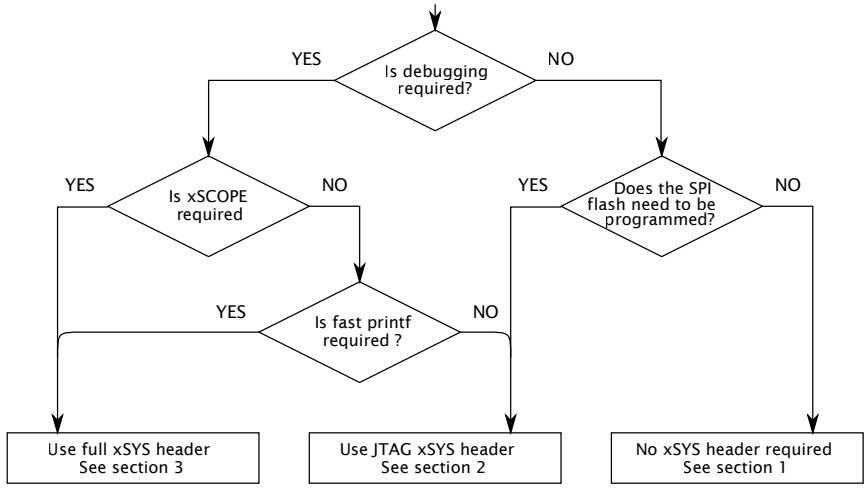


Figure 39: Decision diagram for the xSYS header

G.1 No xSYS header

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

G.2 JTAG-only xSYS header

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

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

- TDI to pin 5 of the xSYS header
- TMS to pin 7 of the xSYS header
- TCK to pin 9 of the xSYS header
- DEBUG_N to pin 11 of the xSYS header
- TDO to pin 13 of the xSYS header

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

G.3 Full xSYS header

For a full xSYS header you will need to connect the pins as discussed in Section G.2, and then connect a 2-wire xCONNECT Link to the xSYS header. The links can be found in the Signal description table (Section 4): they are labelled XL0, XL1, etc in the function column. The 2-wire link comprises two inputs and outputs, labelled $^{1}_{out}$, $^{0}_{out}$, $^{0}_{in}$, and $^{1}_{in}$. For example, if you choose to use XL0 for xSCOPE I/O, you need to connect up $XL0^{1}_{out}$, $XL0^{0}_{out}$, $XL0^{0}_{in}$, $XL0^{1}_{in}$ as follows:

- $XL0^{1}_{out}$ (X0D43) to pin 6 of the xSYS header with a 33R series resistor close to the device.
- $XL0^{0}_{out}$ (X0D42) to pin 10 of the xSYS header with a 33R series resistor close to the device.
- $XL0^{0}_{in}$ (X0D41) to pin 14 of the xSYS header.
- $XL0^{1}_{in}$ (X0D40) to pin 18 of the xSYS header.

H Schematics Design Check List

☑ This section is a checklist for use by schematics designers using the XUF210-512-FB236. Each of the following sections contains items to check for each design.

H.1 Power supplies

☐ The VDD (core) supply ramps monotonically (rises constantly) from 0V to its final value (0.95V - 1.05V) within 10ms (Section 12).

☐ The VDD (core) supply is capable of supplying 700 mA (Section 12 and Figure 21).

☐ PLL_AVDD is filtered with a low pass filter, for example an RC filter, see Section 12

H.2 Power supply decoupling

☐ The design has multiple decoupling capacitors per supply, for example at least four0402 or 0603 size surface mount capacitors of 100nF in value, per supply (Section 12).

☐ A bulk decoupling capacitor of at least 10uF is placed on each supply (Section 12).

H.3 Power on reset

☐ The RST_N and TRST_N pins are asserted (low) until all supplies are good. There is enough time between VDDIO power good and RST_N to allow any boot flash to settle. RST_N is fast enough to meet USB timings.

H.4 Clock

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

☐ Pins MODE0 and MODE1 are set to the correct value for the chosen oscillator frequency. The MODE settings are shown in the Oscillator section, Section 7. If you have a choice between two values, choose the value with the highest multiplier ratio since that will boot faster.

H.5 Boot

☐ X0D01 has a 1K pull-up to VDDIOL (Section 8).

- ☐ The device is kept in reset for at least 1 ms after VDDIOL has reached its minimum level (Section 8).

H.6 JTAG, XScope, and debugging

- ☐ You have decided as to whether you need an XSYS header or not (Section G)
- ☐ If you have not included an XSYS header, you have devised a method to program the SPI-flash or OTP (Section G).

H.7 GPIO

- ☐ You have not mapped both inputs and outputs to the same multi-bit port.
- ☐ Pins X0D04, X0D05, X0D06, and X0D07 are output only and are, during and after reset, pulled low or not connected (Section 8)

H.8 Multi device designs

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

- ☐ One device is connected to a QSPI or SPI flash for booting.
- ☐ Devices that boot from link have, for example, X0D06 pulled high and have link XL0 connected to a device to boot from (Section 8).

I PCB Layout Design Check List

☑ This section is a checklist for use by PCB designers using the XS2-UF10B-512-FB236. Each of the following sections contains items to check for each design.

I.1 Ground Plane

☐ Each ground ball has a via to minimize impedance and conduct heat away from the device. (Section 12.4)

☐ Other than ground vias, there are no (or only a few) vias underneath or closely around the device. This create a good, solid, ground plane.

I.2 Power supply decoupling

☐ The decoupling capacitors are all placed close to a supply pin (Section 12).

☐ The decoupling capacitors are spaced around the device (Section 12).

☐ The ground side of each decoupling capacitor has a direct path back to the center ground of the device.

I.3 PLL_AVDD

☐ The PLL_AVDD filter (especially the capacitor) is placed close to the PLL_AVDD pin (Section 12).

J Associated Design Documentation

Document Title	Information	Document
Estimating Power Consumption For XS1-UF Devices	Power consumption	
XMOS Programming Guide	Timers, ports, clocks, cores and channels	Link
xTIMEcomposer User Guide	Compilers, assembler and linker/mapper Timing analyzer, xScope, debugger Flash and OTP programming utilities	Link

K Related Documentation

Document Title	Information	Document
xCORE200: the XMOS XS2 Architecture	ISA manual	Link
I/O timings for xCORE200	Port timings	Link
xCONNECT Architecture	Link, switch and system information	Link
XS1-UF Link Performance and Design Guidelines	Link timings	
xCORE-200 Clock Frequency Control	Advanced clock control	Link

L Revision History

Date	Description
2015-03-20	Preliminary release
2015-04-14	Added RST to pins to be pulled hard, and removed reference to TCK from Errata Removed TRST_N references in packages that have no TRST_N New diagram for boot from embedded flash showing ports Pull up requirements for shared clock and external resistor for QSPI
2015-05-06	Removed references to DEBUG_N
2015-07-09	Updated electrical characteristics - Section 13
2015-08-19	Added I(USB_VDD) - Section 13 Added USB layout guidelines - Section 12
2015-08-27	Updated part marking - Section 15
2016-04-20	Typical internal pull-up and pull down current diagrams added - Section 13
2017-02-02	Updated USB VBUS wiring description with bus-powered usb-device instructions - Section 10
2017-09-19	Added Absolute Maximum Ratings - Section 13.1 Reference document links updated - Section J
2018-03-23	Incorrect IDCODE return value updated - Section 11 Incorrect VBUS signal name updated to GND in USB diagrams - Section 10
2020-10-05	Released documentation for A revision that uses different flash - Section 8