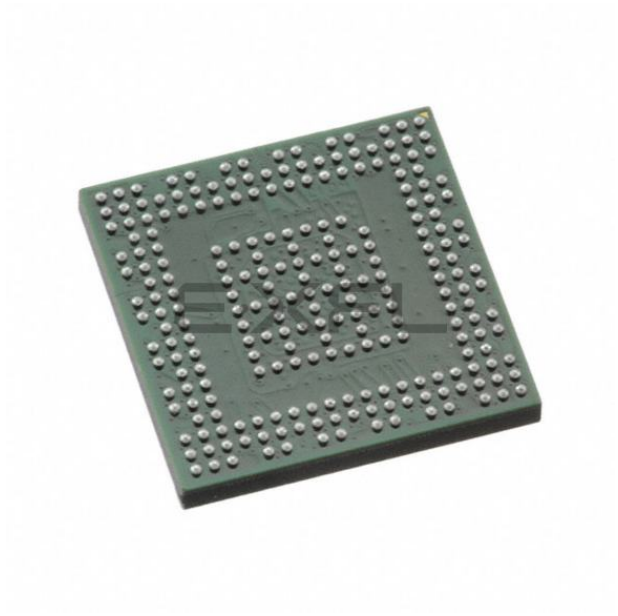

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Discontinued at Digi-Key
Core Processor	XCore
Core Size	32-Bit 16-Core
Speed	2000MIPS
Connectivity	RGMII, USB
Peripherals	-
Number of I/O	73
Program Memory Size	2MB (2M x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	256K x 8
Voltage - Supply (Vcc/Vdd)	0.95V ~ 3.6V
Data Converters	-
Oscillator Type	External
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	236-LFBGA
Supplier Device Package	236-FBGA (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/xmos/xef216-256-fb236-c20

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.

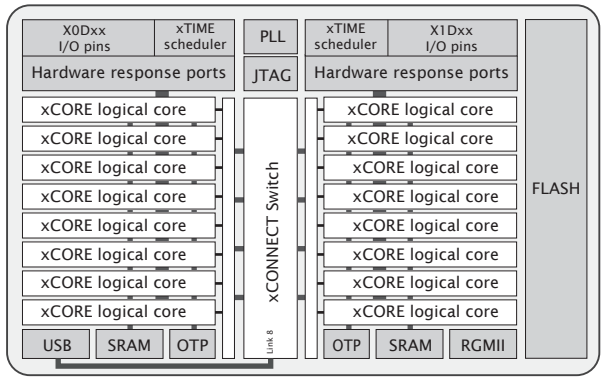

Figure 1: XEF216-512-FB236 block diagram

Key features of the XEF216-512-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

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)

6 Product Overview

The XEF216-512-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 8 active logical cores, which issue instructions down a shared five-stage pipeline. Instructions from the active cores are issued round-robin. If up to five logical cores are active, each core is allocated a fifth of the processing cycles. If more than five logical cores are active, each core is allocated at least $1/n$ cycles (for n cores). Figure 3 shows the guaranteed core performance depending on the number of cores used.

Figure 3: Logical core performance

Speed grade	MIPS	Frequency	Minimum MIPS per core (for *n* cores)							
			1	2	3	4	5	6	7	8
10	1000 MIPS	500 MHz	100	100	100	100	100	83	71	63

There is no way that the performance of a logical core can be reduced below these predicted levels (unless *priority threads* are used: in this case the guaranteed minimum performance is computed based on the number of priority threads as defined in the architecture manual). Because cores may be delayed on I/O, however, their unused processing cycles can be taken by other cores. This means that for more than five logical cores, the performance of each core is often higher than the predicted minimum but cannot be guaranteed.

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

6.2 xTIME scheduler

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

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

6.3 Hardware Response Ports

Hardware Response ports connect an xCORE tile to one or more physical pins and as such define the interface between hardware attached to the XEF216-512-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.

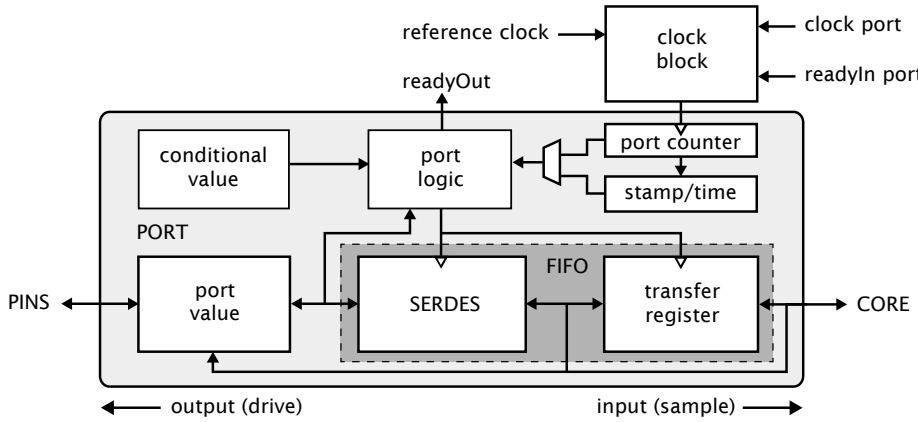

Figure 4: Port block diagram

The port logic can drive its pins high or low, or it can sample the value on its pins, optionally waiting for a particular condition. Ports are accessed using dedicated instructions that are executed in a single processor cycle. xCORE-200 IO pins can be used as *open collector* outputs, where signals are driven low if a zero is output, but left high impedance if a one is output. This option is set on a per-port basis.

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

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

The ports and xCONNECT links are multiplexed onto the physical pins. If an xConnect Link is enabled, the pins of the underlying ports are disabled. If a port is enabled, it overrules ports with higher widths that share the same pins. The pins on the wider port that are not shared remain available for use when the narrower port is enabled. Ports always operate at their specified width, even if they share pins with another port.

6.4 Clock blocks

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

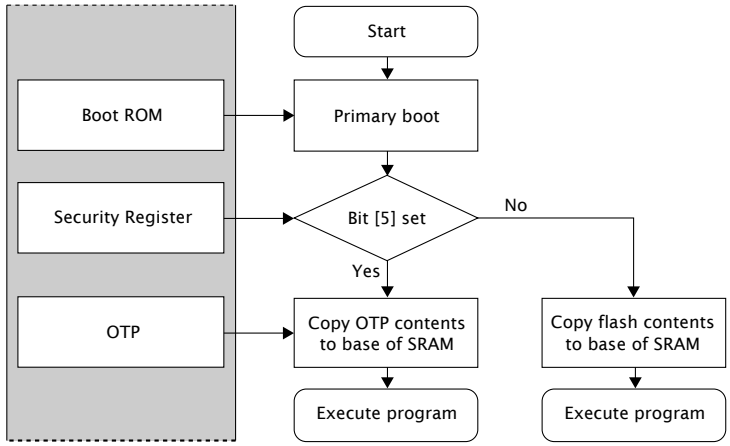

Figure 9: Boot procedure

- 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.

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.

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

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),

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 RGMII

The device has a series of pins that are dedicated to communicate with an RGMII PHY, as per the RGMII v1.3 spec. This can be used to communicate with GBit Ethernet PHYs. The pins and functions are listed in Figure 13. When RGMII mode is enabled (using processor status register 2) these pins can no longer be used as GPIO pins, and will instead be driven directly from an RGMII block that provides DDR to SDR conversion, which in turn is interfaced to a set of ports on Tile 1.

Figure 13: RGMII block pin functions

Pin	RGMII Function	
X1D40	TX3	Transmit bit 3
X1D41	TX2	Transmit bit 2
X1D42	TX1	Transmit bit 1
X1D43	TX0	Transmit bit 0
X1D26	TX_CLK	Receive clock (125 MHz)
X1D27	TX_CTL	Transmit data valid/error
X1D28	RX_CLK	Receive clock (125 MHz)
X1D29	RX_CTL	Receive data valid/error
X1D30	RX0	Receive bit 0
X1D31	RX1	Receive bit 1
X1D32	RX2	Receive bit 2
X1D33	RX3	Receive bit 3

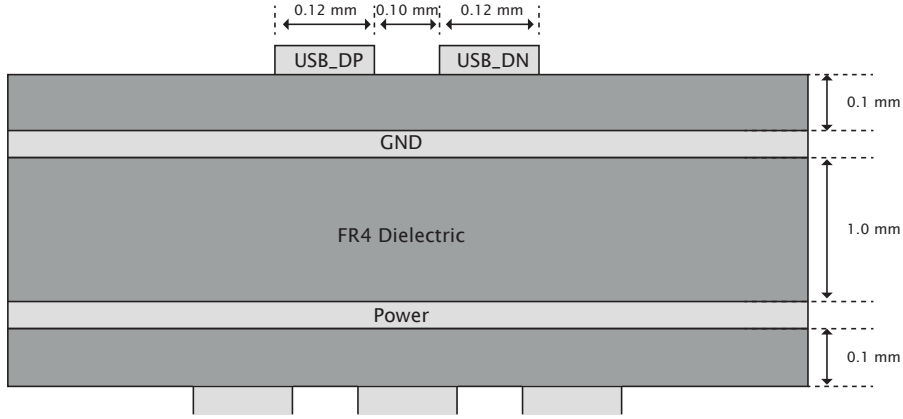

Figure 19: Example USB board stack

For best results, most of the routing should be done on the top layer (assuming the USB connector and XS2-UEF16A-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 18).
- Low-speed and non-periodic signal traces that run parallel should be at least 0.5mm away from USB_DP/USB_DN (see Figure 18).
- 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.

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

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

A read message comprises the following:

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

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

A.3 Accessing node configuration

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

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

A write message comprises the following:

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

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

A read message comprises the following:

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

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

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).

0x12: Debug SSP

Bits	Perm	Init	Description
31:0	DRW		Value.

B.15 DGETREG operand 1: 0x13

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

0x13: DGETREG operand 1

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

B.16 DGETREG operand 2: 0x14

Register number to be read by DGETREG

0x14: DGETREG operand 2

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

B.17 Debug interrupt type: 0x15

Register that specifies what activated the debug interrupt.

0x15: Debug interrupt type

Bits	Perm	Init	Description
31:18	RO	-	Reserved
17:16	DRW		Number of the hardware breakpoint/watchpoint which caused the interrupt (always 0 for =HOST= and =DCALL=). If multiple breakpoints/watchpoints trigger at once, the lowest number is taken.
15:8	DRW		Number of thread which caused the debug interrupt (always 0 in the case of =HOST=).
7:3	RO	-	Reserved
2:0	DRW	0	Indicates the cause of the debug interrupt 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

0x00: Device identification

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

C.2 xCORE Tile description 1: 0x01

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

0x01: xCORE Tile description 1

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

C.3 xCORE Tile description 2: 0x02

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

0x02: xCORE Tile description 2

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

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

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

0x07: Security configuration

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

C.8 Debug scratch: 0x20 .. 0x27

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

0x20 .. 0x27: Debug scratch

Bits	Perm	Init	Description
31:0	CRW		Value.

C.9 PC of logical core 0: 0x40

Value of the PC of logical core 0.

0x40: PC of logical core 0

Bits	Perm	Init	Description
31:0	CRO		Value.

C.10 PC of logical core 1: 0x41

Value of the PC of logical core 1.

C.15 PC of logical core 6: 0x46

Value of the PC of logical core 6.

0x46: PC of logical core 6

Bits	Perm	Init	Description
31:0	CRO		Value.

C.16 PC of logical core 7: 0x47

Value of the PC of logical core 7.

0x47: PC of logical core 7

Bits	Perm	Init	Description
31:0	CRO		Value.

C.17 SR of logical core 0: 0x60

Value of the SR of logical core 0

0x60: SR of logical core 0

Bits	Perm	Init	Description
31:0	CRO		Value.

C.18 SR of logical core 1: 0x61

Value of the SR of logical core 1

0x61: SR of logical core 1

Bits	Perm	Init	Description
31:0	CRO		Value.

C.19 SR of logical core 2: 0x62

Value of the SR of logical core 2

D Node Configuration

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

Number	Perm	Description
0x00	RO	Device identification
0x01	RO	System switch description
0x04	RW	Switch configuration
0x05	RW	Switch node identifier
0x06	RW	PLL settings
0x07	RW	System switch clock divider
0x08	RW	Reference clock
0x09	R	System JTAG device ID register
0x0A	R	System USERCODE register
0x0C	RW	Directions 0-7
0x0D	RW	Directions 8-15
0x10	RW	DEBUG_N configuration, tile 0
0x11	RW	DEBUG_N configuration, tile 1
0x1F	RO	Debug source
0x20 .. 0x28	RW	Link status, direction, and network
0x40 .. 0x47	RO	PLink status and network
0x80 .. 0x88	RW	Link configuration and initialization
0xA0 .. 0xA7	RW	Static link configuration

Figure 35: Summary

D.1 Device identification: 0x00

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

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

0x00: Device identification

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

0x40 .. 0x47: PLink status and network

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

These registers contain configuration and debugging information specific to external links. The link speed and width can be set, the link can be initialized, and the link status can be monitored. The registers control links 0..7.

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

0x80 .. 0x88: Link configuration and initialization

0x20: UIFM Sticky flags

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

F.10 UIFM port masks: 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
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.
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.
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.
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.

F.11 UIFM SOF value: 0x28

USB Start-Of-Frame counter

0x28: UIFM SOF value

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

F.12 UIFM PID: 0x2C

The last USB packet identifier received

H Schematics Design Check List

☑ This section is a checklist for use by schematics designers using the XEF216-512-FB236. Each of the following sections contains items to check for each design.

H.1 Power supplies

☐ VDDIO and OTP_VCC supply is within specification before the VDD (core) supply is turned on. Specifically, the VDDIO and OTP_VCC supply is within specification before VDD (core) reaches 0.4V (Section 13).

☐ The VDD (core) supply ramps monotonically (rises constantly) from 0V to its final value (0.95V - 1.05V) within 10ms (Section 13).

☐ The VDD (core) supply is capable of supplying 700 mA (Section 13 and Figure 21).

☐ PLL_AVDD is filtered with a low pass filter, for example an RC filter, see Section 13 .

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 13).

☐ A bulk decoupling capacitor of at least 10uF is placed on each supply (Section 13).

H.3 Power on reset

☐ The RST_N and TRST_N pins are asserted (low) during or after power up. The device is not used until these resets have taken place.

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.

J Associated Design Documentation

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

K Related Documentation

Document Title	Information	Document Number
The XMOS XS1 Architecture	ISA manual	X7879
XS1 Port I/O Timing	Port timings	X5821
xCONNECT Architecture	Link, switch and system information	X4249
XS1-UEF Link Performance and Design Guidelines	Link timings	
XS1-UEF Clock Frequency Control	Advanced clock control	