XU316-1024-FB265 Datasheet

Publication Date: 2025/1/13
Document Number: XM-014035-PC v2.0.0

XMOS

IN THIS DOCUMENT

1 xCORE Multicore Microcontrollers

The xcore.ai series is a comprehensive range of 32-bit multicore microcontrollers that bring 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 when executing from internal memory, you can write software to implement functions that traditionally require dedicated hardware.

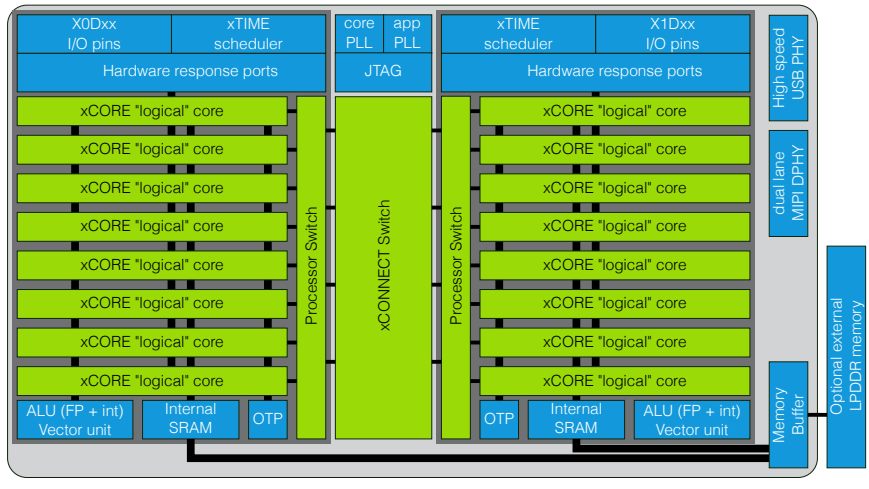


Fig. 1: XU316-1024-FB265 block diagram

Key features of the XU316-1024-FB265 include:

- **Tiles**: Devices consist of one or more xCORE tiles. Each tile contains between five and eight 32-bit logical cores with highly integrated I/O and on-chip memory.
- **Logical cores** Each logical core can execute tasks such as computational code, DSP code, Floating point operations, Vector operations, control software (including logic decisions and executing a state machine) or software that handles I/O. See *Product Overview*
- **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. See *xTIME Scheduler*
- **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. See *Channels and Channel Ends*
- **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. See *xCONNECT Switch and Links*

- **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. See *Hardware Response Ports*
- **Clock blocks** xCORE devices include a set of programmable clock blocks that can be used to govern the rate at which ports execute. *Clock Blocks*
- **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. A memory buffer can be used to interface to an optional external LPDDR memory, or to implement software defined memory. See *Memory*
- **Dual PLL** One PLL is used to create a high-speed processor clock given a low speed external oscillator. A secondary PLL is for user application. *Oscillator, Clocks, and PLLs*
- **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. See *USB PHY*
- **MIPI** The MIPI D-PHY receiver provides a high speed communication link to single or dual lane MIPI devices. See *MIPI PHY*
- **JTAG** The JTAG module can be used for loading programs, boundary scan testing, in-circuit source-level debugging and programming the OTP memory. See *JTAG*

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, Voice, Ethernet, PWM, graphics driver, and audio EQ to your applications.

1.2 Tools: XTC

The XTC 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 XTC: uniquely in the embedded world, XTC therefore includes a cycle-accurate simulator, and high-speed in-circuit instrumentation.

The tools are supported on Windows, Linux and MacOS X and available at no cost from xmos.com/software-tools/ . Information on using the tools is provided in the XTC Tools User Guide .

2 XU316-1024-FB265 Features

- **Multicore Microcontroller with Advanced Multi-Core RISC Architecture**
  - 16 real-time logical cores on 2 xCORE tiles
  - Cores share up to 1600 MIPS
    - Up to 3200 MIPS in dual issue mode
    - Up to 1600 MFLOPS
  - Each logical core has:
    - Guaranteed throughput of between 1/5 and 1/8 of tile MIPS
    - 16x32bit dedicated registers
  - 229 high-density 16/32-bit instructions
    - All have single clock-cycle execution (except for divide)
    - 32x32 -> 64-bit MAC instructions for DSP, arithmetic and cryptographic functions
  - Vector unit, capable of:
    - up to eight word, 16 half-word, or 32 byte multiply-adds.
    - quad complex multiply, or 256 bit-wide multiply-adds.
- **USB PHY, fully compliant with USB 2.0 specification**
- **MIPI receiver, up to two lanes, up to 1.5 Gbit/s**
- **Application PLL with fractional control**
- **Programmable I/O**
  - 128 general-purpose I/O pins, configurable as input or output
    - Up to 32 x 1-bit port, 12 x 4-bit port, 8 x 8-bit port, 4 x 16-bit port, 2 x 32-bit 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
  - 1.8V/3.3V IO with programmable drive strength
- **Memory**
  - 1024KB internal single-cycle SRAM (512KB per tile) for code and data storage
  - LPDDR-1 interface for optional external memory.
  - 8KB internal OTP (shared between tiles or split providing 4KB per tile) for application boot code
- **Hardware resources**
  - 12 clock blocks (6 per tile)
  - 20 timers (10 per tile)
  - 8 locks (4 per tile)
- **JTAG Module for On-Chip Debug**
- **Security Features**
  - Programming lock disables debug and prevents read-back of memory contents
  - AES bootloader ensures secrecy of IP held on external flash memory
- **Ambient Temperature Range**
  - Commercial qualification: 0 °C to 70 °C
  - Industrial qualification: -40 °C to 85 °C

- **Speed Grade**
  - 24: 600 MHz; up to 2400 MIPS, 1200 MFLOP/s, 38.4 GMACC/s
  - 32: 800 MHz; up to 3200 MIPS, 1600 MFLOP/s, 51.2 GMACC/s
- **Power Consumption**
  - Active: 203 mW at 600 MHz (typical)
  - Active: 270 mW at 800 MHz (typical)
  - Standby: 5 mA (typical)
- **265-pin FBGA package 0.8 mm pitch**

3 Pin Configuration

The pin layout of FB265 is shown in Fig. 2. Any pin marked NC should not be connected to any net.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
A	4B0 X0D04	VSS	4E2 X0D32 DQ2	4F2 X0D30 DQ4	1K0 X1D34 DQ6	4E2 X1D32 LDQS	4F3 X1D31 LDM	VDDIOT	32A16 X0D67 RASN	32A14 X0D65 WEN	32A10 X0D61 CKE	VSS	VSS	32A0 X0D49 A7	32A2 X0D51 A5	8D5 X0D41 A11	1P0 X0D39 A13
B	4B3 X0D07	4B2 X0D06	1N0 X0D34 DQ0	4E3 X0D33 DQ1	4F3 X0D31 DQ3	1L0 X1D35 DQ5	4E3 X1D33 DQ7	VSS	32A15 X0D66 CASN	32A13 X0D64 BA0	VSS	VDDIOT	32A1 X0D50 A6	32A3 X0D52 A4	8D7 X1D43 A8	8D4 X0D40 A12	VDDIOT
C	1B0 X0D01	4B1 X0D05	VSS	VDDIOT	4F1 X1D29 DQ14	4E1 X1D27 DQ12	1J0 X1D25 DQ10	32A19 X0D70 DQ8	32A17 X0D68 UDM	32A12 X0D63 BA1	32A9 X0D58 CK	32A8 X0D57 CKN	32A6 X0D55 A1	32A4 X0D53 A3	8D6 X0D42 A10	VSS	VSS
D	1A0 X0D00 XL4in3	1C0 X0D10 XL4in4	4C1 X0D15 XL4in0	4C0 X0D14 XL4in1	4F2 X1D30 DQ15	4F2 X1D28 DQ13	4B3 X1D26 DQ11	1N X1D24 DQ9	32A18 X0D69 UDQS	VSS	32A11 X0D62 CSN	32A7 X0D56 A0	32A5 X0D54 A2	8D7 X0D43 A9	VSS	1N0 X0D37 XL3out3	1D0 X0D38 XL3out4
E	4D2 X0D18 XL4out2	1D0 X0D11 XL4in2	4D1 X0D17 XL4out1	4D0 X0D16 XL4out0	VSS	VDD	VSS	VDD	VDDIOT	VDD	VSS	VDD	VSS	4B2 X0D29 XL3out0	1L0 X0D35 XL3out1	4E0 X0D26 XL3in2	1M0 X0D36 XL3out2
F	4C2 X0D20 XL4out4	4D3 X0D19 XL4out3	1P0 X1D39 XL5in0	1O0 X1D38 XL5in1	VDD								VDD	4E1 X0D27 XL3in1	4F0 X0D28 XL3in0	1P X0D24 XL3in4	1J0 X0D25 XL3in3
G	1M0 X1D36 XL5in3	4C3 X0D21 XL5in4	4D3 X1D41 XL5out1	4D3 X1D40 XL5out0	VDDIOL		VSS	VSS	VSS	VSS	VSS		VDDIOR	32A15 X1D66 XL2out3	32A13 X1D67 XL2out1	32A19 X1D69 XL2out5	32A18 X1D70 XL2out4
H	8D5 X1D42 XL5out2	1N0 X1D37 XL5in2	4A0 X1D02 XL5out3	4A1 X1D03 XL5out4	VDD		VSS	VSS	VSS	VSS	VSS		VDD	32A13 X1D64 XL2in1	32A14 X1D65 XL2in0	32A12 X1D63 XL2in2	32A17 X1D68 XL2out2
J	4B1 X1D04 XL6in4	4B0 X1D05 XL6in3	VDDIOL	VSS	VDD		VSS	VSS	VSS	VSS	VSS		VDD	VSS	VDDIOR	32A16 X1D61 XL2in4	32A11 X1D62 XL2in5
K	4A3 X1D09 XL6out2	4B2 X1D06 XL6in2	4A2 X1D08 XL6in0	4B3 X1D07 XL6in1	VDD		VSS	VSS	VSS	VSS	VSS		VDD	VSS	VSS	32A8 X1D57 XL1out3	32A9 X1D58 XL1out4
L	1D0 X1D11 XL6out4	1C0 X1D10 XL6out3	1D0 X1D01 XL6out1	4A3 X1D00 XL6out0	VSS		VSS	VSS	VSS	VSS	VSS		VSS	32A5 X1D54 XL1out0	32A6 X1D55 XL1out1	32A4 X1D51 XL1in2	32A7 X1D56 XL1out2
M	MIPI_VDD18	MIPI_GND09	VSS	VSS	VDDIOL								VDDIOR	32A3 X1D52 XL1in1	32A4 X1D53 XL1in0	32A0 X1D49 XL1in4	32A1 X1D50 XL1in3
N	MIPI_DN2	MIPI_DP2	MIPI_VDD09	VSS	VSS	VDD	VDDIOB18	VDD	VDD	VDD	VDDIOB18	VDD	VSS	4D2 X1D18 XL0out0	4D3 X1D19 XL0out1	4C3 X1D21 XL0out3	1D0 X1D22 XL0out4
P	MIPI_DN1	MIPI_DP1	VSS	NC	LV_L_N	LV_T_N	1E0 X0D12 XL7in4	1G0 X0D22 XL7in5	VDDIOB18	OTP_VCC	NC	VSS	VSS	4D0 X1D16 XL0in1	4D1 X1D17 XL0in0	4C1 X1D15 XL0in2	4C2 X1D20 XL0out2
R	MIPI_DN0	MIPI_DP0	VSS	RST_N	TRST_N	DEBUG_N	1F3 X0D13 XL7in3	1H0 X0D23 XL7in1	VSS	LV_R_N	NC	NC	VSS	VSS	VSS	1F3 X1D13 XL0in4	4C0 X1D14 XL0in3
T	VSS	POR_DISABLE	VDDIOB18	TDO	PLL_AGND	PLL_AGND2	TMS	1A0 X0D02 XL7in0	4A3 X0D08 XL7out1	1E0 X1D12 XL7out3	NC	VSS	USB_ID	USB_VDD33	USB_GND18	VSS	VDDIOR
U	XOUT	XIN	VSS	TDI	PLL_AVDD	PLL_AVDD2	TCK	4A1 X0D03 XL7out5	4A3 X0D09 XL7out2	1H0 X1D23 XL7out4	VSS	VSS	USB_DM	USB_DP	USB_VDD18	VSS	VSS

Fig. 2: FB265 pin configuration

4 Signal Description and GPIO

This section lists the signals and I/O pins available on the XU316-1024-FB265. See *Signal List* for a list of pins sorted by pin number. 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.
- ST: The IO pin has a Schmitt Trigger on its input.
- IOL, IOB, IOR, IOT: The IO pin is powered from VDDIOL, VDDIOB18, VDDIOR, and VDDIOT respectively.

Note that all GPIO have optional pull-down, pull-up, and Schmitt triggers. The GPIO functions are as follows:

- $XLI_{in/out}{}^{N}$: this pin can be used for xlink *I* wire *N*, input or output.
- NX^{m}: this pin can be used by bit *m* of *N*-bit port *X*
- Any other signal name refers to how this pin can be used for the LPDDR interface

4.1 Power and Ground Pins

Signal	FB265	Function	Type	Properties
MIPI_GND09	M2	MIPI Analog ground	GND	
MIPI_VDD09	N3	MIPI Analog power	PWR	
MIPI_VDD18	M1	MIPI Analog power	PWR	
OTP_VCC	P10	OTP power supply	PWR	
PLL_AGND	T5	Analog ground for PLL	GND	
PLL_AGND2	T6	Analog ground for secondary PLL	GND	
PLL_AVDD	U5	Analog power for PLL	PWR	
PLL_AVDD2	U6	Analog power for secondary PLL	PWR	
USB_GND18	T15	USB Analog ground	GND	
USB_VDD18	U15	USB Analog power	PWR	
USB_VDD33	T14	USB Analog power	PWR	
VDD	mult	Digital tile power	PWR	
VDDIOB18	mult	Digital I/O power (bottom)	PWR	
VDDIOL	mult	Digital I/O power (left)	PWR	
VDDIOR	mult	Digital I/O power (right)	PWR	
VDDIOT	mult	Digital I/O power (top)	PWR	
VSS	mult	Digital ground	GND	

4.2 I/O Pins

Signal	FB265	XL	1	4	8	16	32	Type	Properties
X0D00	D1	$XL4_{in}^{3}$	$1A^{0}$					I/O	IOL
X0D01	C1		$1B^{0}$					I/O	IOL
X0D02	T8	$XL7_{in}^{0}$		$4A^{0}$	$8A^{0}$	$16A^{0}$	$32A^{20}$	I/O	IOB
X0D03	U8	$XL7_{out}^{0}$		$4A^{1}$	$8A^{1}$	$16A^{1}$	$32A^{21}$	I/O	IOB
X0D04	A1			$4B^{0}$	$8A^{2}$	$16A^{2}$	$32A^{22}$	I/O	IOL
X0D05	C2			$4B^{1}$	$8A^{3}$	$16A^{3}$	$32A^{23}$	I/O	IOL
X0D06	B2			$4B^{2}$	$8A^{4}$	$16A^{4}$	$32A^{24}$	I/O	IOL
X0D07	B1			$4B^{3}$	$8A^{5}$	$16A^{5}$	$32A^{25}$	I/O	IOL
X0D08	T9	$XL7_{out}^{1}$		$4A^{2}$	$8A^{6}$	$16A^{6}$	$32A^{26}$	I/O	IOB
X0D09	U9	$XL7_{out}^{2}$		$4A^{3}$	$8A^{7}$	$16A^{7}$	$32A^{27}$	I/O	IOB
X0D10	D2	$XL4_{in}^{4}$	$1C^{0}$					I/O	IOL
X0D11	E2	$XL4_{in}^{2}$	$1D^{0}$					I/O	IOL
X0D12	P7	$XL7_{in}^{4}$	$1E^{0}$					I/O	IOB
X0D13	R7	$XL7_{in}^{3}$	$1F^{0}$					I/O	IOB
X0D14	D4	$XL4_{in}^{1}$		$4C^{0}$	$8B^{0}$	$16A^{8}$	$32A^{28}$	I/O	IOL
X0D15	D3	$XL4_{in}^{0}$		$4C^{1}$	$8B^{1}$	$16A^{9}$	$32A^{29}$	I/O	IOL
X0D16	E4	$XL4_{out}^{0}$		$4D^{0}$	$8B^{2}$	$16A^{10}$		I/O	IOL
X0D17	E3	$XL4_{out}^{1}$		$4D^{1}$	$8B^{3}$	$16A^{11}$		I/O	IOL
X0D18	E1	$XL4_{out}^{2}$		$4D^{2}$	$8B^{4}$	$16A^{12}$		I/O	IOL
X0D19	F2	$XL4_{out}^{3}$		$4D^{3}$	$8B^{5}$	$16A^{13}$		I/O	IOL
X0D20	F1	$XL4_{out}^{4}$		$4C^{2}$	$8B^{6}$	$16A^{14}$	$32A^{30}$	I/O	IOL
X0D21	G2	$XL5_{in}^{4}$		$4C^{3}$	$8B^{7}$	$16A^{15}$	$32A^{31}$	I/O	IOL
X0D22	P8	$XL7_{in}^{2}$	$1G^{0}$					I/O	IOB
X0D23	R8	$XL7_{in}^{1}$	$1H^{0}$					I/O	IOB
X0D24	F16	$XL3_{in}^{4}$	$1I^{0}$					I/O	IOR
X0D25	F17	$XL3_{in}^{3}$	$1J^{0}$					I/O	IOR
X0D26	E16	$XL3_{in}^{2}$		$4E^{0}$	$8C^{0}$	$16B^{0}$		I/O	IOR
X0D27	F14	$XL3_{in}^{1}$		$4E^{1}$	$8C^{1}$	$16B^{1}$		I/O	IOR
X0D28	F15	$XL3_{in}^{0}$		$4F^{0}$	$8C^{2}$	$16B^{2}$		I/O	IOR
X0D29	E14	$XL3_{out}^{0}$		$4F^{1}$	$8C^{3}$	$16B^{3}$		I/O	IOR
X0D30	A4	DQ4		$4F^{2}$	$8C^{4}$	$16B^{4}$		I/O	IOT
X0D31	B5	DQ3		$4F^{3}$	$8C^{5}$	$16B^{5}$		I/O	IOT
X0D32	A3	DQ2		$4E^{2}$	$8C^{6}$	$16B^{6}$		I/O	IOT
X0D33	B4	DQ1		$4E^{3}$	$8C^{7}$	$16B^{7}$		I/O	IOT
X0D34	B3	DQ0	$1K^{0}$					I/O	IOT
X0D35	E15	$XL3_{out}^{1}$	$1L^{0}$					I/O	IOR
X0D36	E17	$XL3_{out}^{2}$	$1M^{0}$		$8D^{0}$	$16B^{8}$		I/O	IOR
X0D37	D16	$XL3_{out}^{3}$	$1N^{0}$		$8D^{1}$	$16B^{9}$		I/O	IOR
X0D38	D17	$XL3_{out}^{4}$	$1O^{0}$		$8D^{2}$	$16B^{10}$		I/O	IOR
X0D39	A17	A13	$1P^{0}$		$8D^{3}$	$16B^{11}$		I/O	IOT
X0D40	B16	A12			$8D^{4}$	$16B^{12}$		I/O	IOT
X0D41	A16	A11			$8D^{5}$	$16B^{13}$		I/O	IOT
X0D42	C15	A10			$8D^{6}$	$16B^{14}$		I/O	IOT

continues on next page

Table 1 – continued from previous page

Signal	FB265	XL	1	4	8	16	32	Type	Properties
X0D43	D14	A9			$8D^{7}$	$16B^{15}$		I/O	IOT
X0D49	A14	A7					$32A^{0}$	I/O	IOT
X0D50	B13	A6					$32A^{1}$	I/O	IOT
X0D51	A15	A5					$32A^{2}$	I/O	IOT
X0D52	B14	A4					$32A^{3}$	I/O	IOT
X0D53	C14	A3					$32A^{4}$	I/O	IOT
X0D54	D13	A2					$32A^{5}$	I/O	IOT
X0D55	C13	A1					$32A^{6}$	I/O	IOT
X0D56	D12	A0					$32A^{7}$	I/O	IOT
X0D57	C12	CKN					$32A^{8}$	I/O	IOT
X0D58	C11	CK					$32A^{9}$	I/O	IOT
X0D61	A11	CKE					$32A^{10}$	I/O	IOT
X0D62	D11	CSN					$32A^{11}$	I/O	IOT
X0D63	C10	BA1					$32A^{12}$	I/O	IOT
X0D64	B10	BA0					$32A^{13}$	I/O	IOT
X0D65	A10	WEN					$32A^{14}$	I/O	IOT
X0D66	B9	CASN					$32A^{15}$	I/O	IOT
X0D67	A9	RASN					$32A^{16}$	I/O	IOT
X0D68	C9	UDM					$32A^{17}$	I/O	IOT
X0D69	D9	UDQS					$32A^{18}$	I/O	IOT
X0D70	C8	DQ8					$32A^{19}$	I/O	IOT
X1D00	L4	$XL6_{out}{}^{0}$	$1A^{0}$					I/O	IOL
X1D01	L3	$XL6_{out}{}^{1}$	$1B^{0}$					I/O	IOL
X1D02	H3	$XL5_{out}{}^{3}$		$4A^{0}$	$8A^{0}$	$16A^{0}$	$32A^{20}$	I/O	IOL
X1D03	H4	$XL5_{out}{}^{4}$		$4A^{1}$	$8A^{1}$	$16A^{1}$	$32A^{21}$	I/O	IOL
X1D04	J1	$XL6_{in}{}^{4}$		$4B^{0}$	$8A^{2}$	$16A^{2}$	$32A^{22}$	I/O	IOL
X1D05	J2	$XL6_{in}{}^{3}$		$4B^{1}$	$8A^{3}$	$16A^{3}$	$32A^{23}$	I/O	IOL
X1D06	K2	$XL6_{in}{}^{2}$		$4B^{2}$	$8A^{4}$	$16A^{4}$	$32A^{24}$	I/O	IOL
X1D07	K4	$XL6_{in}{}^{1}$		$4B^{3}$	$8A^{5}$	$16A^{5}$	$32A^{25}$	I/O	IOL
X1D08	K3	$XL6_{in}{}^{0}$		$4A^{2}$	$8A^{6}$	$16A^{6}$	$32A^{26}$	I/O	IOL
X1D09	K1	$XL6_{out}{}^{2}$		$4A^{3}$	$8A^{7}$	$16A^{7}$	$32A^{27}$	I/O	IOL
X1D10	L2	$XL6_{out}{}^{3}$	$1C^{0}$					I/O	IOL
X1D11	L1	$XL6_{out}{}^{4}$	$1D^{0}$					I/O	IOL
X1D12	T10	$XL7_{out}{}^{3}$	$1E^{0}$					I/O	IOB
X1D13	R16	$XL0_{in}{}^{4}$	$1F^{0}$					I/O	IOR
X1D14	R17	$XL0_{in}{}^{3}$		$4C^{0}$	$8B^{0}$	$16A^{8}$	$32A^{28}$	I/O	IOR
X1D15	P16	$XL0_{in}{}^{2}$		$4C^{1}$	$8B^{1}$	$16A^{9}$	$32A^{29}$	I/O	IOR
X1D16	P14	$XL0_{in}{}^{1}$		$4D^{0}$	$8B^{2}$	$16A^{10}$		I/O	IOR
X1D17	P15	$XL0_{in}{}^{0}$		$4D^{1}$	$8B^{3}$	$16A^{11}$		I/O	IOR
X1D18	N14	$XL0_{out}{}^{0}$		$4D^{2}$	$8B^{4}$	$16A^{12}$		I/O	IOR
X1D19	N15	$XL0_{out}{}^{1}$		$4D^{3}$	$8B^{5}$	$16A^{13}$		I/O	IOR
X1D20	P17	$XL0_{out}{}^{2}$		$4C^{2}$	$8B^{6}$	$16A^{14}$	$32A^{30}$	I/O	IOR
X1D21	N16	$XL0_{out}{}^{3}$		$4C^{3}$	$8B^{7}$	$16A^{15}$	$32A^{31}$	I/O	IOR

continues on next page

Table 1 – continued from previous page

Signal	FB265	XL	1	4	8	16	32	Type	Properties
X1D22	N17	$XL0_{out}^{4}$	$1G^{0}$					I/O	IOR
X1D23	U10	$XL7_{out}^{4}$	$1H^{0}$					I/O	IOB
X1D24	D8	DQ9	$1I^{0}$					I/O	IOT
X1D25	C7	DQ10	$1J^{0}$					I/O	IOT
X1D26	D7	DQ11		$4E^{0}$	$8C^{0}$	$16B^{0}$		I/O	IOT
X1D27	C6	DQ12		$4E^{1}$	$8C^{1}$	$16B^{1}$		I/O	IOT
X1D28	D6	DQ13		$4F^{0}$	$8C^{2}$	$16B^{2}$		I/O	IOT
X1D29	C5	DQ14		$4F^{1}$	$8C^{3}$	$16B^{3}$		I/O	IOT
X1D30	D5	DQ15		$4F^{2}$	$8C^{4}$	$16B^{4}$		I/O	IOT
X1D31	A7	LDM		$4F^{3}$	$8C^{5}$	$16B^{5}$		I/O	IOT
X1D32	A6	LDQS		$4E^{2}$	$8C^{6}$	$16B^{6}$		I/O	IOT
X1D33	B7	DQ7		$4E^{3}$	$8C^{7}$	$16B^{7}$		I/O	IOT
X1D34	A5	DQ6	$1K^{0}$					I/O	IOT
X1D35	B6	DQ5	$1L^{0}$					I/O	IOT
X1D36	G1	$XL5_{in}^{3}$	$1M^{0}$		$8D^{0}$	$16B^{8}$		I/O	IOL
X1D37	H2	$XL5_{in}^{2}$	$1N^{0}$		$8D^{1}$	$16B^{9}$		I/O	IOL
X1D38	F4	$XL5_{in}^{1}$	$1O^{0}$		$8D^{2}$	$16B^{10}$		I/O	IOL
X1D39	F3	$XL5_{in}^{0}$	$1P^{0}$		$8D^{3}$	$16B^{11}$		I/O	IOL
X1D40	G4	$XL5_{out}^{0}$			$8D^{4}$	$16B^{12}$		I/O	IOL
X1D41	G3	$XL5_{out}^{1}$			$8D^{5}$	$16B^{13}$		I/O	IOL
X1D42	H1	$XL5_{out}^{2}$			$8D^{6}$	$16B^{14}$		I/O	IOL
X1D43	B15	A8			$8D^{7}$	$16B^{15}$		I/O	IOT
X1D49	M16	$XL1_{in}^{4}$					$32A^{0}$	I/O	IOR
X1D50	M17	$XL1_{in}^{3}$					$32A^{1}$	I/O	IOR
X1D51	L16	$XL1_{in}^{2}$					$32A^{2}$	I/O	IOR
X1D52	M14	$XL1_{in}^{1}$					$32A^{3}$	I/O	IOR
X1D53	M15	$XL1_{in}^{0}$					$32A^{4}$	I/O	IOR
X1D54	L14	$XL1_{out}^{0}$					$32A^{5}$	I/O	IOR
X1D55	L15	$XL1_{out}^{1}$					$32A^{6}$	I/O	IOR
X1D56	L17	$XL1_{out}^{2}$					$32A^{7}$	I/O	IOR
X1D57	K16	$XL1_{out}^{3}$					$32A^{8}$	I/O	IOR
X1D58	K17	$XL1_{out}^{4}$					$32A^{9}$	I/O	IOR
X1D61	J16	$XL2_{in}^{4}$					$32A^{10}$	I/O	IOR
X1D62	J17	$XL2_{in}^{3}$					$32A^{11}$	I/O	IOR
X1D63	H16	$XL2_{in}^{2}$					$32A^{12}$	I/O	IOR
X1D64	H14	$XL2_{in}^{1}$					$32A^{13}$	I/O	IOR
X1D65	H15	$XL2_{in}^{0}$					$32A^{14}$	I/O	IOR
X1D66	G14	$XL2_{out}^{0}$					$32A^{15}$	I/O	IOR
X1D67	G15	$XL2_{out}^{1}$					$32A^{16}$	I/O	IOR
X1D68	H17	$XL2_{out}^{2}$					$32A^{17}$	I/O	IOR
X1D69	G16	$XL2_{out}^{3}$					$32A^{18}$	I/O	IOR
X1D70	G17	$XL2_{out}^{4}$					$32A^{19}$	I/O	IOR

4.3 MIPI Pins

Signal	FB265	Function	Type	Properties
MIPI_DN0	R1	MIPI lane 0, negative	Input	
MIPI_DN1	P1	MIPI lane 1, negative	Input	
MIPI_DN2	N1	MIPI lane 2, negative	Input	
MIPI_DP0	R2	MIPI lane 0, positive	Input	
MIPI_DP1	P2	MIPI lane 1, positive	Input	
MIPI_DP2	N2	MIPI lane 2, positive	Input	

4.4 Power Control Pins

Signal	FB265	Function	Type	Properties
LV_L_N	P5	Select low voltage VDDIOL, active low	Input	PU IOB
LV_R_N	R10	Select low voltage VDDIOR, active low	Input	PU IOB
LV_T_N	P6	Select low voltage VDDIOT, active low	Input	PU IOB

4.5 JTAG Pins

Signal	FB265	Function	Type	Properties
POR_DISABLE	T2	Disable on chip Power-On-Reset	Input	PD IOB
RST_N	R4	Global reset input, active low	Input	ST PU IOB
TCK	U7	Test clock	Input	PD ST IOB
TDI	U4	Test data input	Input	PU IOB
TDO	T4	Test data output	Output	IOB
TMS	T7	Test mode select	Input	PU IOB
TRST_N	R5	Test reset input, active low	Input	ST PU IOB

4.6 System Service Pins

Signal	FB265	Function	Type	Properties
DEBUG_N	R6	Multi-chip debug, active low	I/O	PU IOB

4.7 USB Pins

Signal	FB265	Function	Type	Properties
USB_DM	U13	USB Data-	I/O	
USB_DP	U14	USB Data+	I/O	
USB_ID	T13	USB Identification	Input	

4.8 Analog Pins

Signal	FB265	Function	Type	Properties
XIN	U2	Crystal in or clock input	Input	IOB
XOUT	U1	Crystal out	Output	IOB

4.9 IO Properties

The device has four IO power domains, three of which can be run from either 3.3V or 1.8V nominal supplies. Three pins, LV_L_N, LV_T_N, and LV_R_N specify which voltage is used on the left, top, and right power domains. These pins should be tied low to specify a domain uses a 1.8V nominal supply, and should be tied high or left floating to specify the domain uses a 3.3V nominal supply. The table above states which GPIO pin is powered from which IO domain. Note that the bottom IO domain, which includes JTAG and the crystal oscillator, is always at 1.8V.

The GPIO pins have software programmable drive strengths, slew rate control, and Schmitt trigger:

- When a port is used for output, the default drive settings for each IO pin are to drive at 4 mA nominally, with no slew rate control (fast edge). When a port is used as input, the default settings when you use a port as an input port is to not have a Schmitt-trigger, and not have a pull resistor. From software, the drive strength can be reduced to 2 mA in order to reduce EMI, or they can be driven at 8 or 12 mA in order to increase speed. The total current that can be supplied by each IO domain is limited and specified in *DC Characteristics - VDDIO=1V8*.
- When used as an input, IO pins can be programmed to have a Schmitt trigger enabled, and two programmable pull resistors can be set to either provide a weak pull-down, a weak pull-up, or a bus keep function where the current level is kept until it is changed by a strong low or a strong high. Pins that are not in use have a weak pull-down enabled to keep them in a defined state.
- The controls are set on a per-port basis by either using the API functions, or by setting six bits using the SETC instruction.

5 Example Application Diagram

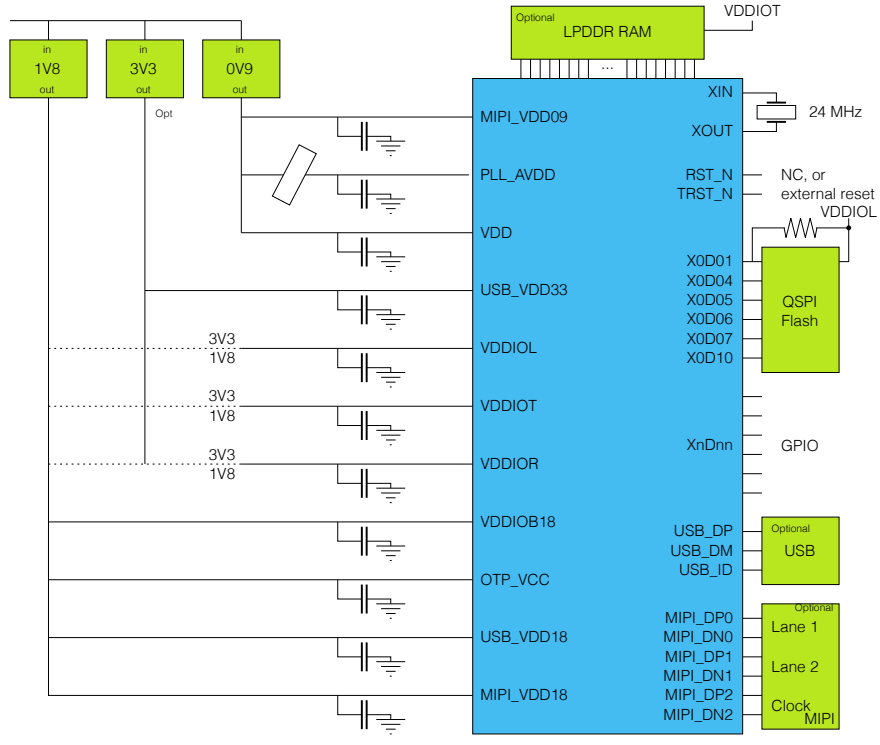


Fig. 3: Simplified Reference Schematic

- see *Integration* for details on the power supplies and PCB design
- the dotted lines on 3V3/1V8 indicate that the device can be powered from either, provided that the LV_x_N pins are tied correctly, see *Integration*.
- see *USB PHY* for details on the USB PHY
- see *MIPI PHY* for details on the MIPI D-PHY receiver
- see *Oscillator, Clocks, and PLLs* for details on oscillator frequencies

6 Product Overview

6.1 Logical cores

Each tile has up to 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). The table below shows the guaranteed core performance depending on the number of cores used.

Speed	active logical cores:		1	2	3	4	5	6	7	8
Grade	MIPS	Frequency	Minimum issue rate per logical core							
24	2400 MIPS	600 MHz	120	120	120	120	120	100	86	75
32	3200 MIPS	800 MHz	160	160	160	160	160	133	114	100

When executing code from internal memory, 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 XU316-1024-FB265, and the software running on it. A combination of 1bit, 4bit, 8bit, 16bit and 32bit ports are available. All pins of a port provide either output or input. Signals in different directions cannot be mapped onto the same port.

The port logic can drive its pins high or low, or it can sample the value on its pins, optionally waiting for a particular condition. Ports are accessed using dedicated instructions that are executed in a single processor cycle. xcore.ai IO pins can be used as *open-drain*

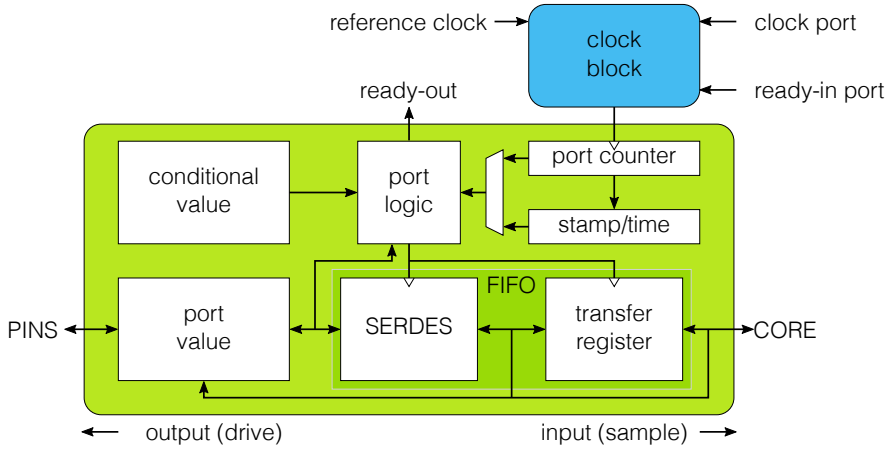


Fig. 4: Port block diagram

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

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

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

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

6.4 Clock Blocks

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

A clock block can use a 1-bit port as its clock source allowing external application clocks to be used to drive the input and output interfaces. xcore.ai 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.

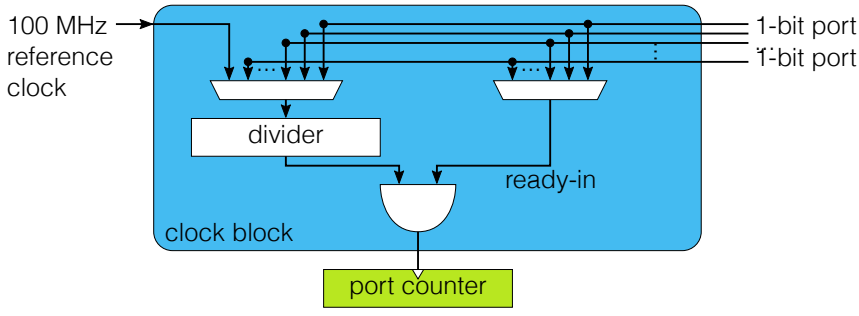


Fig. 5: Clock block diagram

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.

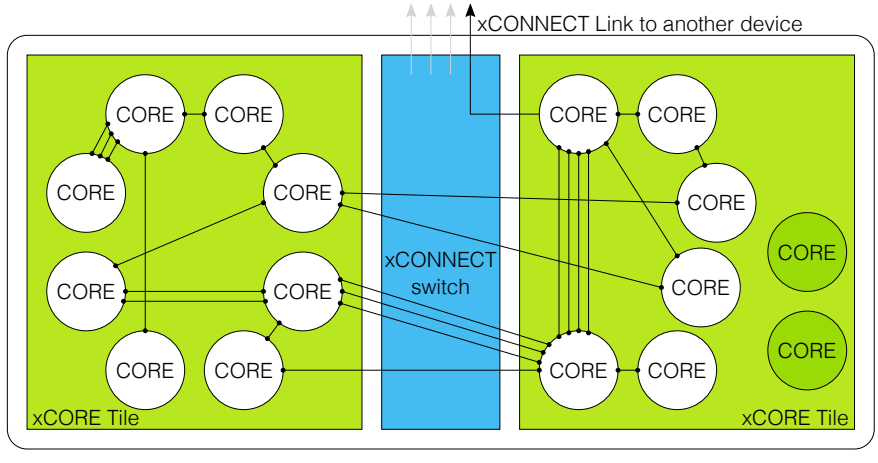


Fig. 6: Switch, links and channel ends

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 xCOREs, 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 xCONNECT Architecture.

7 Oscillator, Clocks, and PLLs

The device executes using a clock that is scaled up by two on-chip PLLs: a *core-PLL* that provides a clock for the digital logic, and a secondary fractional-N PLL for application use. Both PLLs are driven from an oscillator on the XIN and XOUT pins. If you use a crystal, you must use a 24 MHz crystal (+/- 500 ppm or better). Otherwise you can supply a clock between 8 and 30 MHz, with an accuracy governed by your application. Note that the USB PHY only supports limited frequencies, see *USB UTMI Config USB_PHY_CFG0 0xF008*.

The clock structure of the device is shown in Fig. 7. The main purpose of the core PLL is to generate the clocks needed for the digital blocks of the device, including the two processing cores and the switch. The main purpose of the secondary PLL is to provide an application clock if required.

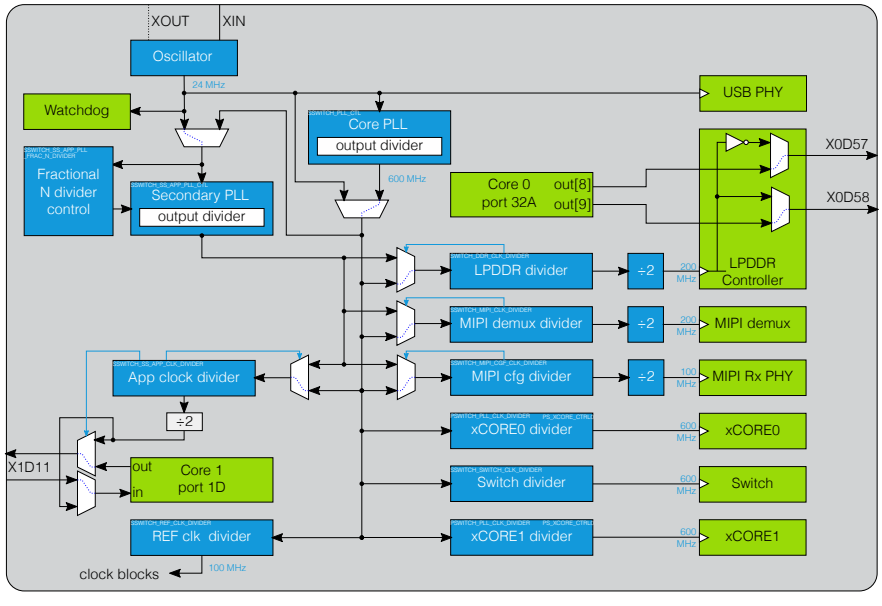


Fig. 7: Clock structure

The frequencies are typical frequencies used when the device operates at 600 MHz. The 100 MHz reference frequency can be used by software to time software and interfaces. The core and switch clocks can be clocked down as required to save power, independent of the reference clock. In very low power modes, both PLLs can be placed in a low-

power mode, and the whole chip executed directly from the oscillator. In this case, the reference can no longer operate at 100 MHz. The labels list the registers in *Processor Status Configuration*, *Tile Configuration*, and *Node Configuration*, that are used to control the clocks.

7.1 Core PLL

The core PLL creates a high-speed clock that is used for the switch, tile, and reference clock. The initial PLL multiplication value is:

Oscillator Frequency	Tile Boot Frequency	PLL Ratio	PLL settings		
			OD	*F*	*R*
8-30 MHz	133-500 MHz	16.667	2	99	0

This table lists the oscillator frequency range, and the values of *OD*, *F* and *R*, which are the registers that define the ratio of the tile frequency to the oscillator frequency:

Fcore = Fosc x (F+1)/2 x 1/(R+1) x 1/(OD+1)

OD, *F* and *R* must be chosen so that *0 <= R <= 63*, *1 <= F <= 8191*, *0 <= OD <= 7*, and *360 MHz <= Fosc x (F+1)/2 x 1/(R+1) <= 1800 MHz*. The *OD*, *F*, and *R* values can be modified by writing to the digital node PLL configuration register, see *PLL settings PLL_CTL 0x06*.

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 AN02022: xcore.ai Clock Frequency Control.

7.2 Secondary PLL

The secondary PLL can be used for generating clocks inside the device, or to create an *application clock* out of the device. When used as an application clock, the output is routed to pin X1D11 and port 1D on core 1 as is shown in Fig. 8. The clock output is divided down to between 171 Hz and 200 MHz. When enabled, tile 1 can input the clock on port 1D. If the clock is required on other tiles, then the clock should be routed to one-bit ports on those tiles over the PCB. An output divider (*Application clock divider SS_APP_CLK_DIVIDER 0x0E*) can be programmed in even steps.

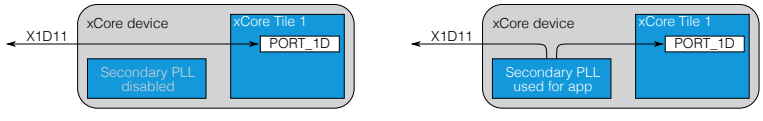


Fig. 8: Secondary PLL connectivity.

The secondary PLL is configured using the register documented in *Secondary PLL settings SS_APP_PLL_CTL 0x0F*. The output frequency of the secondary PLL is

Fpll2 = Fpll2in x (F+1)/2 x 1/(R+1) x 1/(OD+1)

OD, *F* and *R* must be chosen so that *0 <= R <= 63*, *1 <= F <= 8191*, *0 <= OD <= 7*, and *360 MHz <= Fosc x (F+1)/2 x 1/(R+1) <= 1800 MHz*. A flag allows the user to choose between

two input frequencies, *Fpll2in* can be set to either the oscillator (*Fosc*) or the output of the core PLL (*Fcore*).

The secondary PLL has an optional fractional divider (*Secondary PLL Fractional N Divider SS_APP_PLL_FRAC_N_DIVIDER 0x12*). When enabled, the fractional divider will count a period of input clocks, and over part of this period it will cause the secondary PLL to use a divider *F+1* rather than *F*. The period *p* and fraction *f* are set through the control register for the fractional divider, and will result in an output frequency:

Fpll2 = Fpll2in x (F+ 1 + (f+1)/(p+1))/2 x 1/(R+1) x 1/(OD+1)

The use of fractional control adds flexibility to create arbitrary frequencies at the expense of extra jitter. The fractional divider only works for *f < p*.

7.3 Oscillator Circuit

The device has an on-chip oscillator. To use this, you need to connect a crystal, two capacitors, and damping and feedback resistors to the device as shown in Fig. 9. Instead of using a crystal, you can supply a 1V8 clock input on the XIN pin. The clock must be running when the chip gets out of reset.

Fig. 9: Example circuits using a crystal (left), or external oscillator (right).

Rf should be 1M Ohm. Values for *Cl1*, *Cl2* and *Rd* depend on the crystal characteristics. We recommend that you use a crystal with characteristics as specified in the following table. These have an ESR of at most 60 Ohm, have a load capacitance of 12 pF, and all resonate at their fundamental frequency:

Name	Freq	Load	max ESR	Power	Rd	Cl1/2
Seiko Epson FA-238 24.0000MD30X-W5						
	24 MHz	12 pF	60 R	10-200uW	680 R	22 pF
Multicomp MCSJK-7U-24.00-12-10-60-B-10						
	24 MHz	12 pF	60 R	1-200uW	680 R	22 pF
IQD LFXTAL032813						
	24 MHz	12 pF	40 R	< 500uW	680 R	22 pF
TKC 7M-24.000MAAE-T						
	24 MHz	12 pF	30 R	1-500uW	680 R	22 pF

7.4 Low Power Use

For systems that need to run in a low-power mode, the following sequence of operations can be taken:

- set the core clock divider to an appropriately high value. This will reduce performance and power

- set the PLL to a low frequency. This will reduce power consumption.
- provide a clock into the XIN pin instead of using the oscillator circuit.

8 Reset Logic

The device has an on-chip Power-on-Reset (POR). This keeps the chip in reset whilst the supplies are coming up, as shown in Fig. 10. The device assumes that the supplies come up monotonically to reach their minimum operating voltages within the times specified in *Reset Timing*. The POR resets the whole device to a defined state, including the PLL configuration, the JTAG logic, the PHYs, and the cores. When in reset, all GPIO pins have a pull-down enabled.

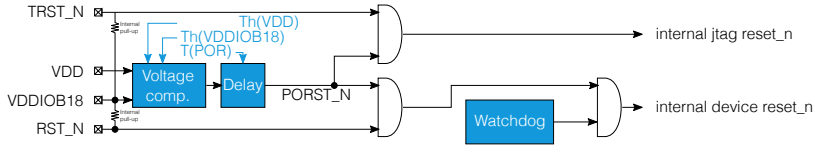


Fig. 10: Simplified reset circuit

When the device comes out of reset, the boot procedure starts (*Boot Procedure*). The chip can be reset externally using the RST_N pin. If required, the JTAG state machine can be reset to its idle state by clocking TCK five times whilst TMS is high, or TRST_N can be asserted.

If the chip needs to be reset at a later stage, this can be done from software using the PLL control register (*PLL settings PLL_CTL 0x06*). This soft resets everything except for the PLL logic. It is therefore possible to reset keeping the current PLL settings.

When the device comes out of reset, the processor will attempt to boot within a very short period of time. If booting from external flash, ensure that there is enough time between before RST_N coming up for the external flash to settle.

An independent watchdog runs from the input clock pin XIN. It can be set to take the chip into reset when the watchdog has not been updated or cleared in time. The 12-bit watchdog timer with a 16-bit divider provides accuracies of between 1 input clock and 65536 input clocks, and a time-out of between 1 input clock and 268,435,456 input clocks (just over 11 seconds with a 24 MHz input crystal). The watchdog is set-up through the watchdog registers (*Watchdog Config WATCHDOG_CFG 0xF020*-*Watchdog Status WATCHDOG_STATUS 0xF024*)

9 Boot Procedure

The xCORE Tile Tile boot procedure is illustrated in Fig. 11. If the secure-boot bit of the security register (which resides at pre-defined locations in OTP, see *OTP*) is set, the device boots from OTP. Otherwise it boots from external device(s) according to boot source pin values X0D04, X0D05, and X0D06 (see the table below). The boot pins are sampled shortly after reset with the internal weak pull-downs enabled on those pins. In typical use, a boot mode other than QSPI Flash can be selected by using one or more pull-ups on those pins. Care should be taken if other external devices are connected to this port that the boot mode is selected correctly.

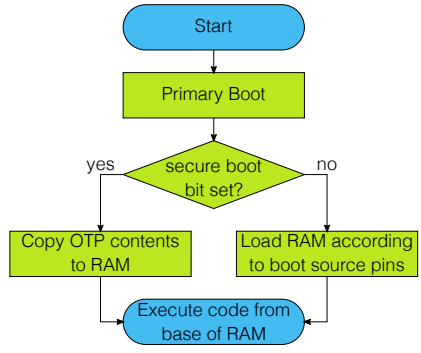


Fig. 11: Boot procedure

X0D06	X0D05	X0D04	Tile 0 boot	Enabled Links	Other tiles
0	0	0	QSPI flash	None	Channel end 0
0	0	1	SPI flash	None	Channel end 0
0	1	0	SPI slave	None	Channel end 0
0	1	1	SPI slave	None	SPI slave
1	0	0	Channel end 0	XL0 (2w)	Channel end 0
1	0	1	Channel end 0	XL4-XL7 (5w)	Channel end 0
1	0	1	Channel end 0	XL1/2/5/6 (5w)	Channel end 0
1	0	1	Channel end 0	XL0-XL3 (5w)	Channel end 0

The boot image provided by an external device has the following format:

- A 32-bit program size *s* in words.
- Program consisting of *s x 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.

9.1 Boot from QSPI Flash

If set to boot from QSPI flash, the processor enables the six pins specified in the table below, and drives the SPI clock. A Quad I/O READ command (0xEB) is issued with three address bytes (0x00) and one dummy byte. Boot data is then expected from the flash and input into the device. The clock polarity and phase are 0 / 0. The flash is assumed to be ready within 300 us after power-up, if the flash takes longer than 300 us the chip should be held in reset using RST_N until the flash is ready. The flash is assumed to be in its power-up state, where QSPI-mode accesses will succeed. In particular, the flash

device must be set into quad mode or similar. If the flash is set to an alternate mode, for example QPI, and the xCORE device is reset, then the subsequent boot will fail.

Pin	Signal	Description
X0D01	SS	Slave Select
X0D04	SPIO0	Data0
X0D05	SPIO1	Data1
X0D06	SPIO2	Data2
X0D07	SPIO3	Data3
X0D10	SCLK	Clock

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

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

The boot sequence up to the start of the QSPI boot is outlined in Fig. 12

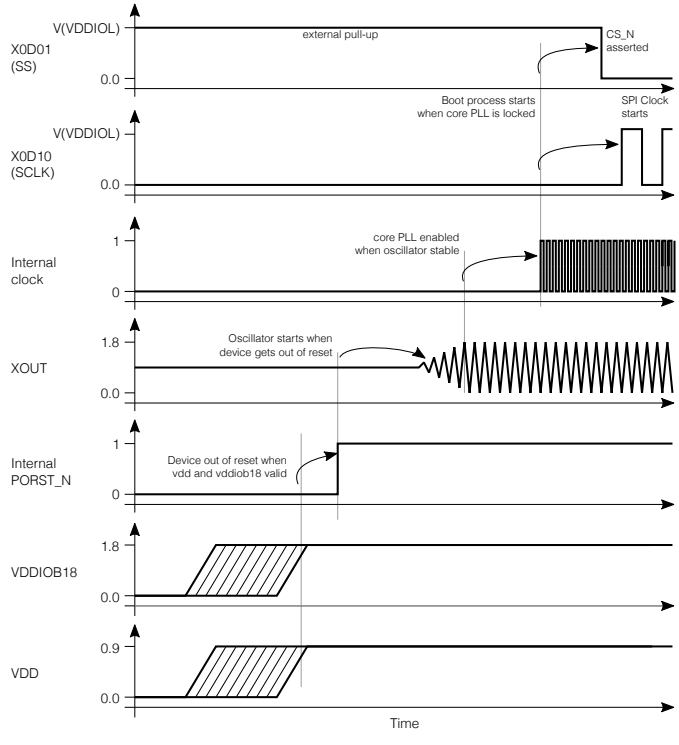


Fig. 12: Outline boot sequence

9.2 Boot from SPI Flash

If set to boot from SPI master, the processor enables the four pins specified in the table below, and drives the SPI clock. A READ command (0x03) is issued with three address bytes (0x00), no dummy, then the data is expected from the flash. The clock polarity and phase are 0 / 0.

Pin	Signal	Description
X0D00	MISO	Master In Slave out (data)
X0D01	SS	Slave Select
X0D10	SCLK	Clock
X0D11	MOSI	Master Out Slave In (cmd)

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

If a large boot image is to be read in, it is faster to first load a small boot-loader that reads the large image using a faster SPI clock, for example 50 MHz or as fast as the flash device supports.

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

The boot sequence up to the start of the SPI boot is outlined in Fig. 12

9.3 Boot as SPI Slave

If set to boot from SPI slave, the processor enables the three pins specified in the table below and expects a boot image to be clocked in. There is no command sequence, data is input directly from the first rising edge of clock. The supported clock polarity and phase are 0/0 and 1/1.

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

The xCORE Tile expects each byte to be transferred with the *least-significant bit first*. The pins used for SPI boot are hardcoded in the boot ROM and cannot be changed. If required, an SPI boot program can be burned into OTP that uses different pins.

9.4 Boot from xConnect Link

If set to boot from an xConnect Link, the processor enables its link(s) shortly after the boot process starts. Enabling the Link switches off the pull-down resistors on the link, drives all the TX wires low (the initial state for the link), and monitors the RX pins for boot-traffic; they must be low at this stage. If the internal pull-down is too weak to drain any residual charge, external pull-downs may be required on those pins.

The boot-rom on the core will then:

1. Allocate channel-end 0.
2. Input a word on channel-end 0. It will use this word as a channel to acknowledge the boot. Provide the null-channel-end 0x0000FF02 if no acknowledgment is required.
3. Input the boot image specified above, including the CRC.
4. Input an END control token.
5. Output an END control token to the channel-end received in step 2.
6. Free channel-end 0.
7. Jump to the loaded code.

9.5 Boot from OTP

If an xCORE tile is set to use secure boot (see Fig. 11), the boot image is read from address 0 of the OTP memory in the security module of the tile.

This feature can be used to implement a secure bootloader which loads an encrypted image from external flash, decrypts and CRC checks it with the processor, and discontinues the boot process if the decryption or CRC check fails. XMOS provides a default secure bootloader that can be written to the OTP along with secret decryption keys.

Each tile can be configured to have its own individual OTP memory, and hence some tiles can be booted from OTP while others are booted from SPI or the channel interface. This enables systems to be partially programmed, dedicating one or more tiles to perform a particular function, leaving the other tiles user-programmable.

10 Memory

The address space as seen by each core is shown in Fig. 13. This address space comprises internal RAM (*SRAM*), an external RAM (*LPDDR Memory Interface*), a software defined memory (*Software-defined Memory*), and the boot ROM.

Outside the normal address space, the device contains a one-time-programmable memory (*OTP*). The OTP memory cannot be read and written directly from the instruction set; instead, it is accessed through a library.

10.1 SRAM

Each xCORE Tile integrates a single 512KB SRAM bank for both instructions and data. All internal memory is 256 bits wide, and instructions are either 16-bit or 32-bit. Byte (8-bit), half-word (16-bit), word (32-bit), double-word (64-bit) and vector (256-bit) accesses are supported and are executed within one tile clock cycle.

10.2 LPDDR Memory Interface

The xCORE can be connected to an LPDDR memory through the pins in the VDDIOT power domain. The GPIO pins on the VDDIOT domain are overlaid onto a JEDEC compatible LPDDR interface with 14 address pins (A13..A0) and 16 data pins (DQ15..DQ0), enabling a memory of 16-128 MByte to be interfaced. This pin muxing is shown in Fig. 14.

DDR speeds of up to 100 MHz are supported, and care should be taken with the PCB design. See appnote AN02021 Using external memory with xcore.ai for a reference layout.

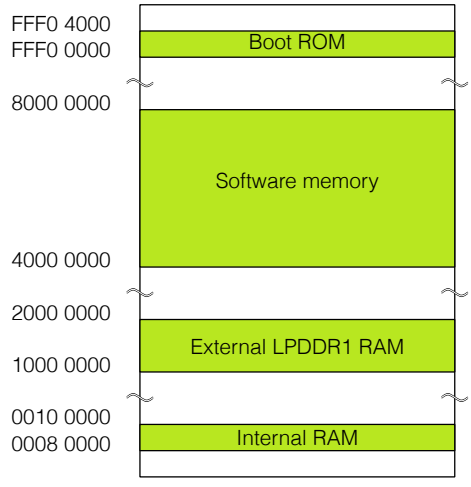


Fig. 13: Address space

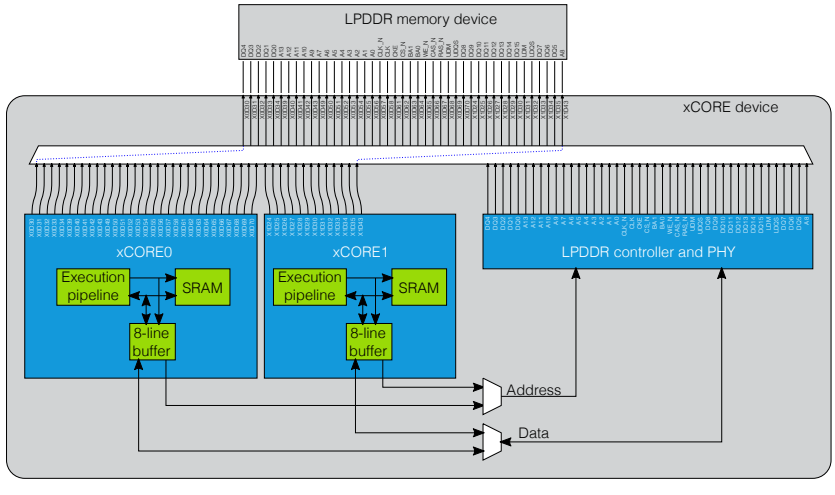


Fig. 14: LPDDR pin muxing.

Logically, the memory can be connected to either Tile 0 or Tile 1; it is up to the programmer to decide which one. The memory is connected by the tile enabling the external memory interface through a process-status control register, see *xCORE Tile control XCORE_CTRL0 0x02*. Only one tile should enable the external memory interface. A small buffer decouples the LPDDR memory from the device. The memory is addressed in the enabled device from address 0x1000 0000 - 0x1FFF FFFF.

Details on external memory can be found in the application note on "xcore.ai external memory", X14230

10.3 Software-defined Memory

The device can map any memory into the address space under software control. For example, a QSPI flash can be mapped into the address space (to execute code from), or serial RAM devices can be connected. The software memory is in address 0x4000 0000 - 0x7FFF FFFF. Refer to the XS3 ISA specification for details on how to use software memory.

10.4 OTP

The device integrates 4KB of one-time programmable (OTP) memory per tile. This memory contains some global information about the chip behaviour and, optionally, code and data that can be used for, for example, secure boot.

The OTP can be set to be used in unified mode (single OTP) or in split mode (where there is two half OTPs). In split mode, the top half of the OTP is inaccessible.

The memory map of the unified OTP is shown below:

Address	Name	Meaning
0x000	SECURITY_CONFIG_TILE_0	The security configuration word for tile 0 Individual bits determine which features are disabled, and these are documented later in this section
0x001	SECURITY_CONFIG_TILE_1	The security configuration word for tile 1 in unified mode. Individual bits determine which features are disabled, and these are documented later in this section
0x002..003		Reserved
0x004	OTP_JTAG_USER_WORD	Bits 13:0 are copied into the JTAG_USERCODE[31:18]
0x005..7ff		User code and/or data in unified mode

In split mode, elements 5..0x7ff have the following meanings:

Address	Name	Meaning
0x005..3ff		User code and/or data for tile 0 in split mode
0x400		Reserved
0x401	SECURITY_CONFIG_TILE_1	The security configuration word for tile 1 in split mode. Individual bits determine which features are disabled, and these are documented later in this section
0x402..403		Reserved
0x404		Reserved
0x405..7ff		User code and/or data in split mode

The OTP memory is programmed using three special I/O ports. Programming is performed through `lib_otp3` and `xburn`.

Feature	Bits	Description
Disable JTAG	0	Set to 1 to disable the JTAG interface to the tile. This makes it impossible for the tile state or memory content to be accessed via the JTAG interface.
Disable JTAG to PLL	4	Set to 1 to disable JTAG access to the PLL configuration register.
Secure Boot	5	Set to 1 to force the xCORE Tile to boot from address 0 of the OTP
Unified mode	7	Set to 1 to create one unified OTP rather than two half OTPs for each tile. This disables registers 0x400-0x404, and enables register 0x001.
Write disable	8	Disable programming.
Reserved	9	Set to 0
Disable Global Debug	14	Disables access to the DEBUG_N pin.

11 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 (*USB UTMI Config USB_PHY_CFG0 0xF008-USB Shim configuration USB_SHIM_CFG 0xF00C*), and data is communicated through ports on the digital node. A library, lib_xud , is provided to implement the MAC layer and full *USB-device* functionality.

The USB PHY is connected to the ports as shown in Fig. 15. Enabling the USB PHY on a tile will connect the ports shown to the USB PHY. These ports will not be available for GPIO on that tile. All other IO pins and ports are unaffected. The USB PHY should not be enabled on both tiles. Two clock blocks can be used to clock the USB ports. One clock block for the TXDATA path, and one clock block for the RXDATA path. Details on how to connect those ports are documented in an application note on USB for xcore.ai.

11.1 USB VBUS

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

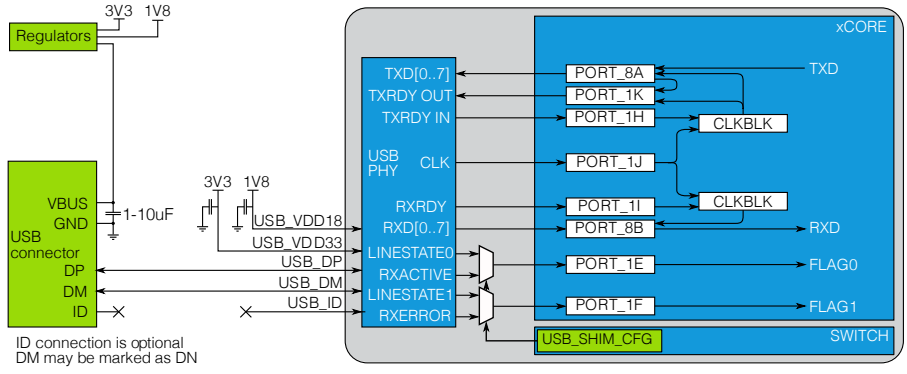


Fig. 15: Bus powered *USB-device*

D+/D- pins when VBus is not present, "USB Back Voltage Test"). This requires a GPIO pin XnDnn to be connected to the VBUS pin of the USB connector as is shown in Fig. 16; lib_xud needs to be configured to use the chosen GPIO pin to enable/disable the D+/D- pull-ups.

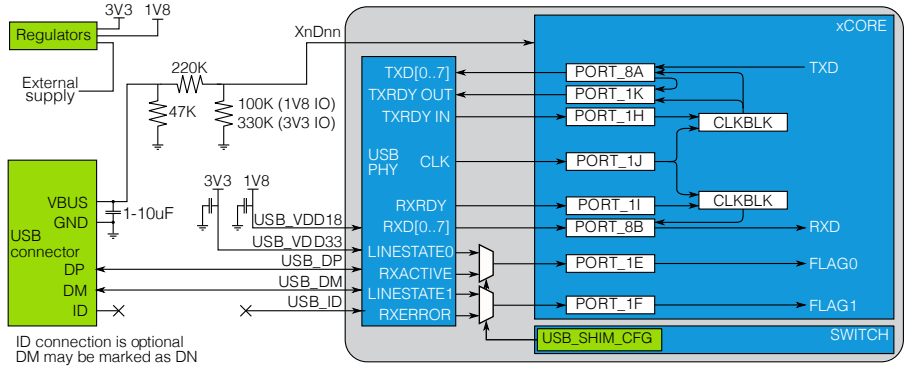


Fig. 16: Self powered *USB-device*

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 Fig. 16 ensures that the transient does not damage the device. The 220k series resistor and 1-10uF capacitor ensure than any input transient is filtered and does not reach the device. A resistor to ground divides the 5V VBUS voltage, and makes sure that the signal on the GPIO pin is not more than the IO voltage. It should be 100K for a 1.8V IO domain, or 330K for a 3.3V IO domain. 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.

11.2 Logical Core Requirements

The XMOS XUD library lib_xud 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.

12 MIPI PHY

The device has a two Data Lane MIPI D-PHY receiver on board, capable of receiving MIPI data at up to 1.5 Gbps. The MIPI D-PHY has three differential pairs. By default, DP0/DN0 are lane one, DP1/DN1 are the clock, and DP2/DN2 are an optional second lane. The lanes can be configured (and the lanes/clocks swapped around) using the MIPI lane configuration register, see *MIPI D-PHY lane config MIPI_DPHY_CFG3 0xE01B*.

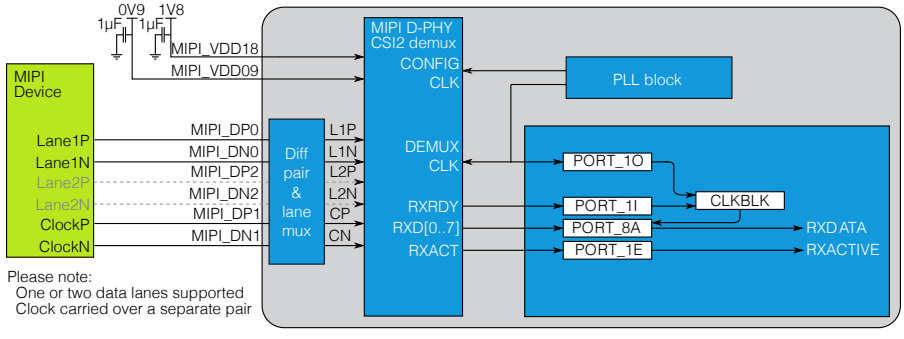


Fig. 17: Connecting a *MIPI-device*

The MIPI receiver has a decoder for common CSI-2 packed formats, and is connected to the ports shown in Fig. 17. The MIPI block is clocked from its own clock source that can either be driven from the system PLL (divide by 4 min), or from the secondary PLL. See *Oscillator, Clocks, and PLLs* on how to set the clocks.

13 JTAG

The JTAG module can be used for loading programs, boundary scan testing, and in-circuit source-level debugging. JTAG can be used for programming flash devices and the OTP by loading code onto the device that will program the flash and/or OTP. All JTAG signals use a 1.8V supply.

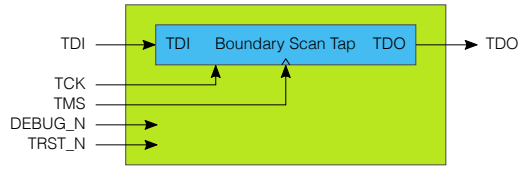


Fig. 18: JTAG chain structure

The JTAG chain structure is illustrated in Fig. 18. It comprises a single IEEE 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 can be left not connected, or used to reset the JTAG module.

The JTAG module can be reset by holding TMS high for five clock cycles.

The DEBUG_N pin is used to synchronize the debugging of multiple xCOREs. 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 Ohm 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 below:

Bit 31	Device Identification Register						Bit 0
Version	Part Number				Manufacturer Identity		
0000	0000	0000	0000	0110	0110	0011	0011
0	0	0	0	6	6	3	3

The JTAG usercode register can be read by using the USERCODE instruction. Its contents are specified in the table below. The OTP User ID field is read from bits [13:0] of the OTP_JTAG_USER_WORD on xCORE Tile 0, see *OTP* (all zero on unprogrammed devices). The OTP User ID field is set by the boot ROM when it executes after the device reset has been de-asserted, so its value is not available to read when the device is in reset.

Bit 31	Usercode Register						Bit 0
OTP USER ID					Silicon Revision		
0000	0000	0000	0000	0000	1010	0000	0100
0	0	0	0	0	A	0	4

You can program the PLL and reset the device over JTAG. When IR is set to eight, the DR value is shifted directly into the PLL settings register (*PLL settings PLL_CTL 0x06*), which includes bits for resetting the device and for setting the “boot-from-JTAG” bit. Note that if TCK is not free running then at least 100 TCK clocks must be provided after shifting the value into DR for the write to take effect.

14 Integration

The device has power and ground pins for different supplies. Several pins of each type may be provided to minimize the effect of inductance within the package, all of which must be connected.

- VDD pins for the xCORE Tile. The VDD supply should be well decoupled at high frequencies. Place many (at least 12) 100 nF low inductance multi-layer ceramic capacitors close to the chip between the supplies and GND.
- VDDIO pins for the I/O lines. Separate I/O supplies are provided for the left, bottom, top, and right side of the package; different I/O voltages may be supplied on those.

The signal description (*Signal Description and GPIO*) specifies which I/O is powered from which power supply.

The 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, one 100nF 0402 low inductance MLCCs on each supply pin.

If you use 1.8V for any of the VDDIOL, VDDIOT, or VDDIOR domains, then you must strap the corresponding LV_L_N, LV_T_N, or LV_R_N pins to GROUND

- PLL_AVDD pin for the PLL, with an associated PLL_AGND. 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 1 uF multi-layer ceramic capacitor and a ferrite of 600 ohm at 100MHz and DCR < 1 ohm, eg, Taiyo Yuden BKH1005LM601-T) is recommended on this pin.
- PLL_AVDD2 pins for the secondary PLL. This should be filtered the same way as PLL_AVDD.
- PLL_AGND2 associated with PLL_AVDD2
- OTP_VCC pins for the OTP
- A MIPI_VDD09 pin for the analogue core supply to the MIPI D-PHY. This supply needs a 1 uF decoupler close to the pin. Connect MIPI_VDD09 to ground if MIPI is not used in the design.
- A MIPI_GND09 pin for the analogue ground to the MIPI D-PHY.
- A MIPI_VDD18 pin for the analogue 1.8V supply to the MIPI D-PHY. This supply needs a 1 uF decoupler close to the pin. Connect MIPI_VDD18 to ground if MIPI is not used in the design.
- A USB_VDD18 pin for the analogue 1.8V supply to the USB-PHY. You can leave USB_VDD18 unconnected if USB is not used in the design.
- A USB_GND18 pin for the analogue ground of the USB-PHY.
- A USB_VDD33 pin for the analogue 3.3V supply to the USB-PHY. You can leave USB_VDD33 unconnected if USB is not used in the design.
- GND for all other supplies, including VDD and VDDIO.

All ground pins must be connected directly to the board ground. 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 VDD and VDDIO supplies.

The power supplies must be brought up monotonically, and input voltages must not exceed the specifications at any time.

Power sequencing is summarised in Fig. 19. VDDIO and VDD can ramp up independently. In order to reduce stresses on the device, it is preferable to make them ramp up within a short time of each other, no more than 50 ms apart. You must ensure that the VDDIOL, VDDIOT, and VDDIOR domains are valid before the device is taken out of reset, as the boot pins are on VDDIOL. If you use a single 1.8V VDDIO power supply, then the on-chip power-on-reset will ensure that reset stays low until all supplies are valid. If you use multiple power supplies, then you must either ensure that RST_N stays asserted until the VDDIOL/R/T domains are valid, or ensure that VDDIOL/R/T are valid by the time that VDDIOB18 and VDD are valid.

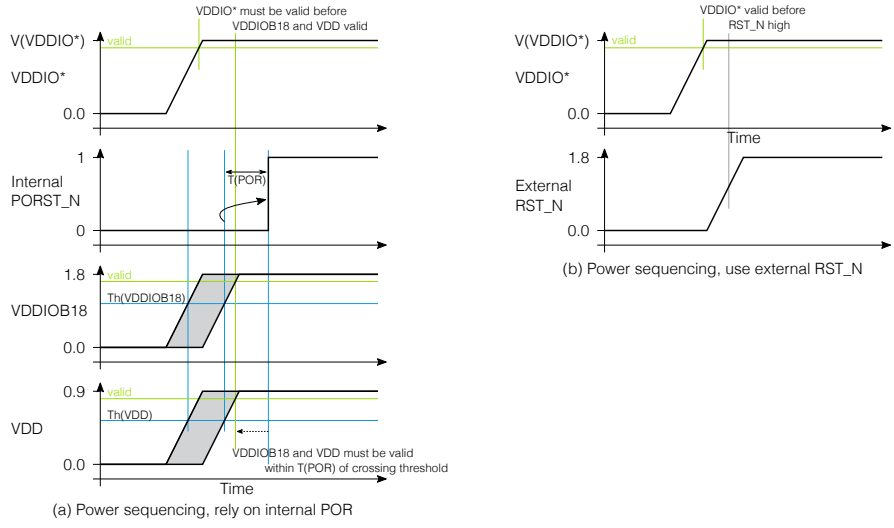


Fig. 19: Sequencing of power supplies and RST_N (if used)

In the condition where the VDDIOB18 supply is off (below spec minimum) and the VDD supply is on (above spec minimum) then the VDDIOL/R/T supplies default to being in 1.8V nominal mode and the absolute maximum ratings for that mode apply (see AMR table). This means that, for systems with a 3.3V VDDIO supply, while the VDDIOB18 supply is off, the 3.3V supply should not rise above 1.98V unless the VDD supply is off. A simple way to meet these conditions is to ensure the VDDIOB18 supply is not turned on last. Most systems can meet this requirement without explicit power sequencing.

14.1 Differential Pair Signal Routing and Placement

If you are using the USB PHY and/or the MIPI D-PHY, then you should route the differential pair marked D+ and D- carefully in order to ensure signal integrity. The D+ and D- lines are the positive and negative data polarities of a high-speed 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 D+ and D- are tightly matched. In addition, the differential impedance of D+ and D- must meet its specifications. We route MIPI D-PHY signals as loosely coupled pairs. Fig. 20 and the table below shows guidelines on how to space and stack the board when routing differential pairs.

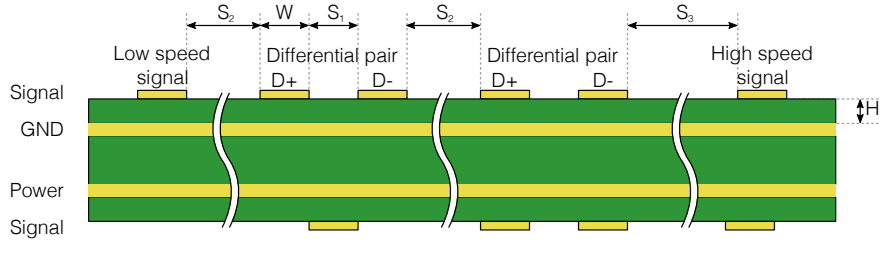


Fig. 20: Spacings of a low speed signal, two differential pairs and a high speed signal

Parameter	USB	MIPI
Impedance	90 ohm	2x 50 ohm
W: trace width	0.12 mm	0.125 mm
S1: spacing between D+/D-	0.10 mm	0.275 mm
S2: spacing between diff pairs	0.51 mm	0.625 mm
S3: spacing to high speed signal	1.27 mm	1.27 mm
H: di-electric height	0.10 mm	0.10 mm
Skew between D+/D-	1 mm	0.5 mm
Skew between clock/data	N/A	2 mm

14.2 General Routing and Placement Guidelines

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

For best results, most of the routing should be done on the top layer (assuming the devices 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 differential pairs are routed first before any other routing. When routing high speed signals, the following guidelines should be followed:

- High speed differential pairs should be routed together.
- High-speed signal pair traces should be trace-length matched.
- Ensure that high speed signals (clocks, 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 a distance *S3* away from D+/D- (see Fig. 20 and the table above).
- Low-speed and non-periodic signal traces that run parallel should be at least *S_2* away from D+/D- (see Fig. 20 and the table above).
- Route high speed signals on the top of the PCB wherever possible.
- Route high speed 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 x h* rule; keep traces *20 x h* (the height above the power plane) away from the edge of the power plane.
- Use a minimum of vias in high speed 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 differential pair traces near clock sources, clocked circuits or magnetic devices.
- Avoid stubs on high speed signals.

In order to optimise MIPI routing, the D+/D- pairs can be swapped, and the lane/clock differential pairs can be reassigned; see *MIPI D-PHY lane config MIPI_DPHY_CFG3 0xE01B*.

14.3 Land Patterns and Solder Stencils

The package is a 265 ball Fine Ball Grid Array (FBGA) on a 0.8 mm pitch.

The land patterns and solder stencils will depend on the PCB manufacturing process. We recommend you design them 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 *Package Information* specify the dimensions and tolerances.

14.4 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. The central ground balls form the main thermal path for heat dissipation, and you should aim to use one via per BGA ball into the ground plane.

14.5 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. Examples of 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-033D.

14.6 Reflow

The package is RoHS compliant and uses Pb-free solder balls for connection to the system PCB. For this reason, a Pb-free solder paste and reflow profile should be used to generate a reliable interconnect.

You should ensure that the board assembly process is optimised for the design; for details of the recommended reflow profile, please refer to the Joint IPC/JEDEC standard J-STD-020.

15 Electrical Characteristics

15.1 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	typ	max	units	notes
VDD	Tile DC supply voltage	-0.5		1.05	V	
PLL_AVDD*	PLL analog supplies	-0.5		1.05	V	
VDDIOB18	I/O supply voltage	-0.5		1.98	V	
OTP_VCC	OTP supply voltage	-0.5		1.98	V	
Tj	Active junction temperature	-40		125	°C	
Tstg	Storage temperature	-65		150	°C	
V(Vin)	Voltage applied to any IO pin	-0.5		VDDIO+0.5	V	
I(XxDxx)	Current per GPIO pin	-25		25	mA	A
I(VDDIOL)	Sum of current for VDDIOL			252	mA	B C D
I(VDDIOR)	Sum of current for VDDIOR			378	mA	B C D
I(VDDIOT)	Sum of current for VDDIOT			504	mA	B C D
I(VDDIOB18)	Sum of current for VDDIOB18			126	mA	B C D
VDDIOL (1V8 nom)	I/O supply voltage	-0.5		1.98	V	
VDDIOR (1V8 nom)	I/O supply voltage	-0.5		1.98	V	
VDDIOT (1V8 nom)	I/O supply voltage	-0.5		1.98	V	
VDDIOL (3V3 nom)	I/O supply voltage	-0.5		3.63	V	E
VDDIOR (3V3 nom)	I/O supply voltage	-0.5		3.63	V	E
VDDIOT (3V3 nom)	I/O supply voltage	-0.5		3.63	V	E

- A: At 1.8V
- B: Exceeding these current limits will result in premature aging and reduced lifetime.
- C: This current consumption must be evenly distributed over all VDDIO pins.
- D: All main power (VDD, VDDIO) and ground (VSS) pins must always be connected.
- E: Refer to *Integration* for sequencing information

15.2 Operating Conditions

Symbol	Parameter	min	typ	max	units	notes
VDD	Tile DC supply voltage	0.855	0.900	0.945	V	
VDDIOL (1V8 nom)	I/O supply voltage	1.62	1.80	1.98	V	
VDDIOT (1V8 nom)	I/O supply voltage	1.62	1.80	1.98	V	
VDDIOR (1V8 nom)	I/O supply voltage	1.62	1.80	1.98	V	
VDDIOB18	I/O supply voltage	1.62	1.80	1.98	V	
VDDIOL (3V3 nom)	I/O supply voltage	2.97	3.30	3.63	V	
VDDIOT (3V3 nom)	I/O supply voltage	2.97	3.30	3.63	V	
VDDIOR (3V3 nom)	I/O supply voltage	2.97	3.30	3.63	V	
USB_VDD33	USB tile analog supply	3.00	3.30	3.60	V	
USB_VDD18	USB tile analog supply	1.62	1.80	1.98	V	
PLL_AVDD*	PLL analog supplies	0.855	0.90	0.945	V	
MIPI_VDD09	MIPI 0.9V analog supply	0.855	0.90	0.99	V	
MIPI_VDD18	MIPI 1.8V analog supply	1.62	1.80	1.98	V	
Ta	Ambient operating temperature (C24, C32)	0		70	°C	
Ta	Ambient operating temperature (I24, I32)	-40		85	°C	

15.3 DC Characteristics - VDDIO=1V8

Symbol	Parameter	min	typ	max	units	notes
V(IH)	Input high voltage	0.65 x VDDIO		VDDIO + 0.3	V	A
V(IL)	Input low voltage	-0.3		0.35 x VDDIO	V	A
V(T+)	Hysteresis threshold up	0.4 x VDDIO		0.7 x VDDIO	V	B
V(T-)	Hysteresis threshold down	0.3 x VDDIO		0.6 x VDDIO	V	B
V(HYS)	Input hysteresis voltage	0.1 x VDDIO		0.4 x VDDIO	V	B
V(OH)	Output high voltage	1.35			V	C
V(OL)	Output low voltage			0.24	V	C
I(PU)	Internal pull-up current (Vin=0V)	-35			uA	D
I(PD)	Internal pull-down current (Vin=VDDIO)			32	uA	D
I(LC)	Input leakage current		11	248	nA	
Ci	Input capacitance		6		pF	

- A: All pins except power supply pins.
- B: When Schmitt-Trigger enabled
- C: Measured with 2 mA drivers sourcing 2 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.

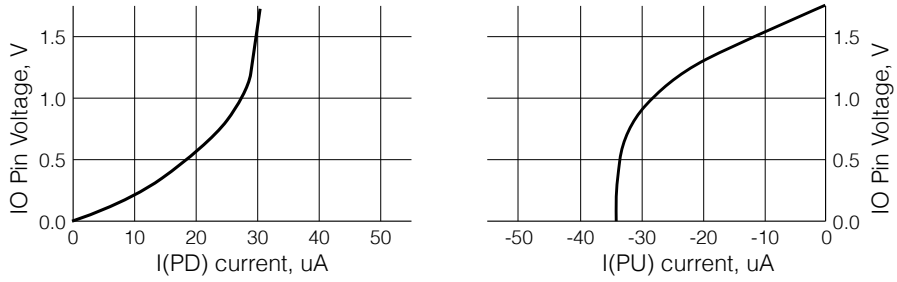


Fig. 21: Typical internal pull-down and pull-up currents at 1V8

15.4 DC Characteristics, VDDIO=3V3

Symbol	Parameter	min	typ	max	units	notes
V(IH)	Input high voltage	2		VDDIO+0.3	V	A
V(IL)	Input low voltage	-0.3		0.8	V	A
V(T+)	Hysteresis threshold up	0.9		2.1	V	B
V(T-)	Hysteresis threshold down	0.7		1.9	V	B
V(HYS)	Input hysteresis voltage	0.2		1.4	V	B
V(OH)	Output high voltage	2.68			V	C
V(OL)	Output low voltage			0.23	V	C
I(PU)	Internal pull-up current (Vin=0V)	-65			uA	D
I(PD)	Internal pull-down current (Vin=VDDIO)			54	uA	D
I(LC)	Input leakage current		12	317	nA	
Ci	Input capacitance		6		pF	

- A: All pins except power supply pins.
- B: When Schmitt-Trigger enabled
- C: Measured with 2 mA drivers sourcing 2 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.

15.5 ESD Stress Voltage

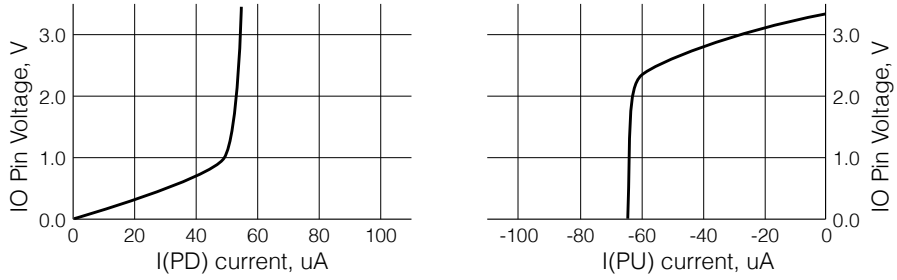


Fig. 22: Typical internal pull-down and pull-up currents at 3V3

Symbol	Parameter	min	typ	max	units	notes
HBM	Human body model	-2000		2000	V	
CDM	Charged Device Model	-500		500	V	

15.6 Reset Timing

Symbol	Parameter	min	typ	max	units	notes
T(RST)	Reset pulse width	5			us	
Vth(VDD)	POR threshold for VDD	0.722		0.798	V	
Vth(VDDIOB18)	POR threshold for VDDIOB18	1.425		1.575	V	
T(INIT)	Initialization time		290	480	us	A

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

15.7 Power Consumption

Symbol	Parameter	min	typ	max	units	notes
Iddq(VDD)	Quiescent VDD current		5		mA	A B C
PD	Tile power dissipation		0.4	1.2	mW/MHz	A D E
I(VDD)	Active VDD current (C24, I24)		225	830	mA	A F
I(VDD)	Active VDD current (C32, I32)		300	1110	mA	A F
P(VDD)	Active VDD power (C24, I24)		203	750	mW	A F
P(VDD)	Active VDD power (C32, I32)		270	1000	mW	A F
I(PLL_AVDD)	PLL_AVDD current	0.2	5		mA	G
I(USB_VDD33) (hs)	VDD33 current in HS mode		0.8	1	mA	
I(USB_VDD33) (fs tx)	VDD33 current on FS transmission		23	25	mA	H
I(USB_VDD18) (hs)	VDD18 current in HS mode		30	36	mA	
I(USB_VDD18) (fs tx)	VDD18 current on FS transmission		6.8	8.2	mA	
I(VDD) (hs)	VDD current in hs mode		6	9	mA	
I(VDD) (fs tx)	VDD current for USB FS tx		1.6	6.5	mA	
I(MIPI_VDD09A)	MIPI_VDD09A current		2.5		mA	
I(MIPI_VDD18A)	MIPI_VDD18A current		5		mA	

- A: Use for budgetary purposes only.
- B: Assumes typical tile and I/O voltages with no switching activity.
- C: Excludes PLL current.
- D: Assumes typical tile and I/O voltages with nominal switching activity.
- E: PD(TYP) value is the usage power consumption under typical operating conditions.
- F: Measurement conditions: VDD = 0.9 V, VDDIO = 1.8 V, 25 °C.
- G: PLL_AVDD = 0.9 V
- H: Full-speed values are for a 3m USB cable

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 AN02023: xcore.ai Power Consumption Estimation.

15.8 Clock

Symbol	Parameter	min	typ	max	units	notes
f	Input frequency	8	24	30	MHz	
SR(CLK)	Slew rate, clock	0.1			V/ns	
TJ(LT)	Long term input jitter (pk-pk)			2	%	A B
f(MAX)	Core clock frequency (C24, I24)			600	MHz	C
f(MAX)	Core clock frequency (C32, I32)			800	MHz	C

- A: Percentage of CLK period.
- B: When used with an external oscillator on XIN
- C: Assumes typical tile and I/O voltages with nominal activity.

Further details can be found in AN02022: xcore.ai Clock Frequency Control.

15.9 xCORE Tile I/O AC Characteristics

The 10%-90% rise and fall times on output pins are shown below.

I/O AC characteristics 1V8:

Symbol	Parameter	min	typ	max	units	notes
Trise	Rise time of output pins	0.81	1.18	2.43	ns	A
Tfall	Fall time of output pins	0.74	1.20	2.41	ns	A

- A: With a 5 pf Load @ 4mA drive strength

I/O AC characteristics 3V3:

Symbol	Parameter	min	typ	max	units	notes
Trise	Rise time of output pins	0.92	1.64	3.41	ns	A
Tfall	Fall time of output pins	0.98	1.54	3.12	ns	A

- A: With a 5 pf Load @ 4mA drive strength

Information on interfacing with high-speed interfaces can be found in xcore.ai I/O timings.

15.10 xConnect 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.

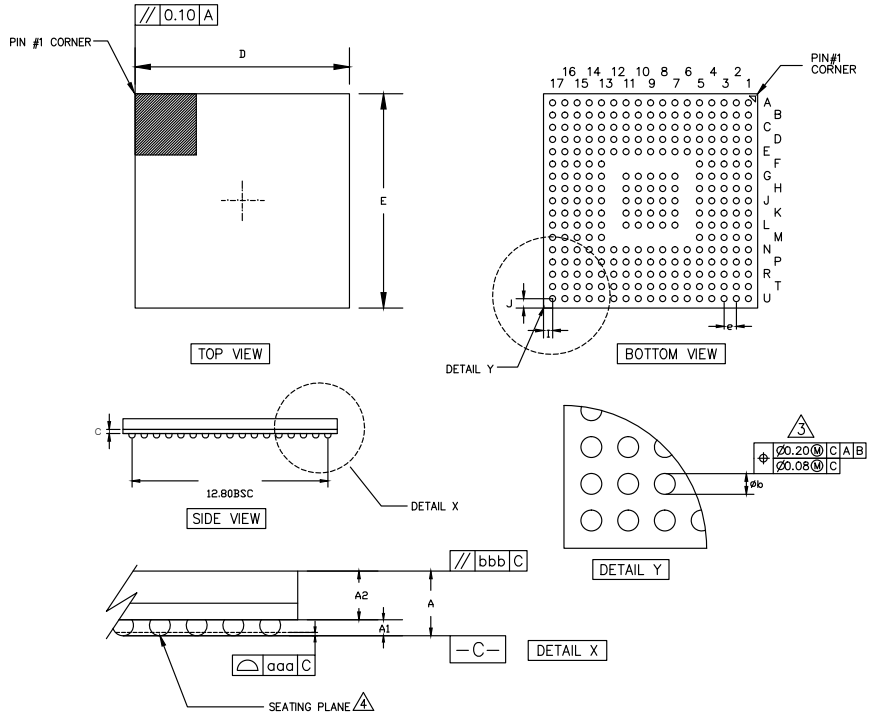
15.11 JTAG Timing

Symbol	Parameter	min	typ	max	units	notes
f(TCK_D)	TCK frequency (debug)			25	MHz	
f(TCK_B)	TCK frequency (boundary scan)			25	MHz	

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

16 Package Information

The mechanical drawings for FB265 are shown in Fig. 23.



SYMBOL	MIN.	NOM.	MAX.
A	1.17	1.27	1.37
A1	0.26	0.31	0.36
A2	0.91	0.96	1.01
D	13.90	14.00	14.10
E	13.90	14.00	14.10
I	0.60 REF.		
J	0.60 REF.		
M	17X17 <DEPOPULATED>		
aaa			0.12
bbb			0.10
b	0.35	0.40	0.45
e	0.80 TYP.		
c	0.26 REF.		

NOTE:

1. "e" REPRESENTS THE BASIC SOLDER BALL GRID PITCH.
2. "M" REPRESENTS THE MAXIMUM SOLDER BALL MATRIX SIZE.
3. DIMENSION "b" IS MEASURED AT THE MAXIMUM SOLDER BALL DIAMETER PARALLEL TO PRIMARY DATUM -C- .
4. PRIMARY DATUM -C- AND SEATING PLANE ARE DEFINED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.
5. ALL DIMENSIONS ARE IN MILLIMETERS.
6. DIMENSIONING AND TOLERANCING PER ASME Y14.5M–1994
7. AFTER REFLOW, DIMENSION "b" IS 0.420

Fig. 23: Package information for FB265

The part marking scheme is detailed in Fig. 24.

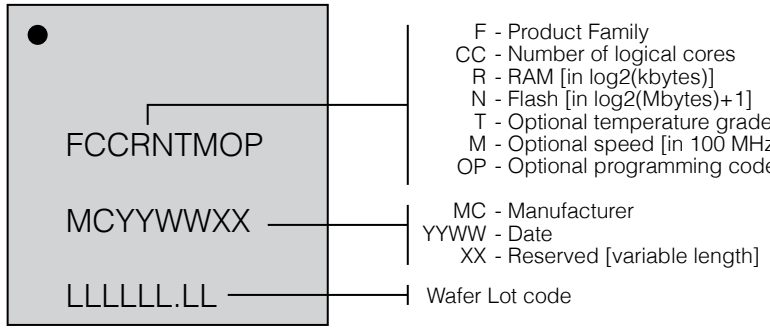


Fig. 24: Part marking scheme

17 Ordering Information

Product Code	Marking	Qualification	Speed Grade
XU316-1024-FB265-C24	`V16A0 MCYYWW`	Commercial	2400 MIPS
XU316-1024-FB265-C32	`V16A0C8 MCYYWW`	Commercial	3200 MIPS
XU316-1024-FB265-I24	`V16A0I6 MCYYWW`	Industrial	2400 MIPS
XU316-1024-FB265-I32	`V16A0I8 MCYYWW`	Industrial	3200 MIPS

A Configuration of the XU316-1024-FB265

The device is configured through banks of registers, as shown in Fig. 25.

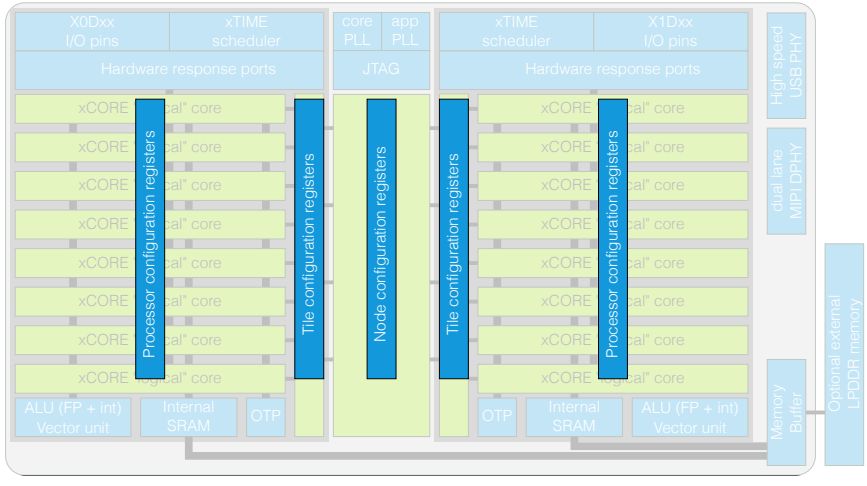


Fig. 25: Registers

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.

Registers are addressed by a number, for each register a symbolic constant is defined in the `xs1.h` include file which has one of the following three names:

- `XS1_PS_<NAME>` for processor status registers.
- `XS1_PSWITCH_<NAME>_NUM` for tile configuration registers.
- `XS1_SSWITCH_<NAME>_NUM` for node configuration registers.

Each register typically comprises a set of *bit-fields* that control individual functions. These bitfields are specified in the tables in subsequent appendices. Macros are defined in the `xs1.h` include file which perform the following support functions:

- `XS1_<NAME>(x)` The value of the bitfield extracted from a word `x`.
- `x = XS1_<NAME>_SET(x, v)` Setting the bitfield in a word `x` to the value `v`.

Registers and bit-fields have permissions as follows:

[RO]
: read-only

[RW]
: read and write

[D..]
Only works when the processor is in Debug mode.

[C..]
Conditional permission, see *PSwitch permissions DBG_CTRL 0x04*.

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 `__builtin_getps(reg)` and `__builtin_setps(reg,value)` can be used from C.

A.2 Accessing an xCORE Tile Configuration Register

xCORE Tile configuration registers can be accessed through the interconnect using the functions `write_pswitch_reg(get_local_tile_id(), ...)` and `read_pswitch_reg(get_local_tile_id(), ...)`, where `get_local_tile_id()` can be replaced with the number of the tile. 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 tokens (data tokens unless specified otherwise):

- Control token 192 (signifies write to configuration register)
- Bits 31..24 of channel-end that response should be sent to, set to 0xff for blind write
- Bits 23..16 of channel-end that response should be sent to, set to 0xff for blind write
- Bits 15..8 of channel-end that response should be sent to, set to 0xff for blind write
- Bits 15..8 of register number that should be written
- Bits 7..0 of register number that should be written
- Bits 31..24 of data to be written
- Bits 23..16 of data to be written
- Bits 15..8 of data to be written
- Bits 7..0 of data to be written
- Control token 1 (signifies end of message)

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 tokens (data tokens unless specified otherwise):

- Control token 193 (signifies read from configuration register)

- Bits 31..24 of channel-end that response should be sent to
- Bits 23..16 of channel-end that response should be sent to
- Bits 15..8 of channel-end that response should be sent to
- Bits 15..8 of register number that should be read
- Bits 7..0 of register number that should be read
- Control token 1 (signifies end of message)

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_sswitch_reg(get_local_tile_id(), ...)` and `read_sswitch_reg(get_local_tile_id(), ...)`, where `get_local_tile_id()` can be replaced with the number of the node. These functions implement the protocols described above.

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.

The message structure is identical to the tile configuration messages above.

B Processor Status Configuration

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

The identifiers for the registers need 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	*Core Cell Ring Oscillator Value*	RING_OSC_DATA0
0x08	RO	*Core Wire Ring Oscillator Value*	RING_OSC_DATA1
0x09	RO	*Peripheral Cell Ring Oscillator Value*	RING_OSC_DATA2
0x0A	RO	*Peripheral Wire 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
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
0xA0	RO	*The number of cache misses*	CACHE_MISS_CNT
0xA1	RO	*The total number of cache accesses*	CACHE_ACCESS_CNT

B.1 RAM base address RAM_BASE 0x00

This register contains the base address of the RAM. It is initialized to 0x00080000.

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

Bits	Perm	Init	Description, Identifier
31:19	RW		The event and interrupt vectors. VECTOR_BASE
18:0	RO	0	Reserved

B.3 xCORE Tile control XCORE_CTRL0 0x02

Register to control features in the xCORE tile

Bits	Perm	Init	Description, Identifier
31:13	RO	0	Reserved
12:11	RW	3	Specify size of a connected LPDDR device (options are: 128,256,512Mbits, 1Gbit), XCORE_CTRL0_EXTMEM_DEVICE_SIZE
10	RW	0	Disable RAMs to save power (contents will be lost) XCORE_CTRL0_RAMSHUTDOWN
9	RW	0	Enable memory auto-sleep feature XCORE_CTRL0_MEMSLEEP_ENABLE
8	RW	0	Enable MIPI interface periph ports XCORE_CTRL0_MIPI_ENABLE
7:6	RO	0	Reserved
5	RO	0	Reserved
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
3:2	RO	0	Reserved
1	RO	0	Reserved
0	RW	0	Enable External memory interface XCORE_CTRL0_EXTMEM_ENABLE

B.4 xCORE Tile boot status BOOT_CONFIG 0x03

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

Bits	Perm	Init	Description, Identifier
31:24	RO	0	Reserved
23:16	RO		Processor number. BOOT_CONFIG_PROCESSOR
15:9	RO	0	Reserved
8	RO		Overwrite BOOT_MODE. BOOT_CONFIG_SECURE_BOOT
7:5	RO	0	Reserved
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.

Bits	Perm	Init	Description, Identifier
31	RW		Disables write permission on this register SECUR_CFG_DISABLE_ACCESS
30:15	RO	0	Reserved
14	RW		Disable access to XCore's global debug SECUR_CFG_DISABLE_GLOBAL_DEBUG
13:10	RO	0	Reserved
9	RW		Disable read access to OTP. SECUR_CFG_OTP_READ_LOCK
8	RW		Prevent access to OTP SBPI interface to prevent programming and other functions. SECUR_CFG_OTP_PROGRAM_DISABLE
7	RW		Combine OTP into a single address-space for reading. SECUR_CFG_OTP_COMBINED
6	RO	0	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	0	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 two subsequent registers. The ring oscillators are asynchronous to the xCORE tile clock and can be used as a source of random bits.

Bits	Perm	Init	Description, Identifier
31:2	RO	0	Reserved
1	RW	0	Core ring oscillator enable. RING_OSC_CORE_ENABLE
0	RW	0	Set to 1 to enable the core peripheral ring oscillator. RING_OSC_PERPH_ENABLE

B.7 Core Cell 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.

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

B.8 Core Wire 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.

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

B.9 Peripheral Cell 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.

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

B.10 Peripheral Wire 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.

Bits	Perm	Init	Description, Identifier
31:16	RO	0	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

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

B.12 Debug SSR DBG_SSR 0x10

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

Bits	Perm	Init	Description, Identifier
31:11	RO	0	Reserved
10	DRW		1 if in high priority mode SR_QUEUE
9	DRW		1 if, on kernel entry, the thread will switch to dual issue. SR_KEDI
8	DRW		1 when in dual issue mode. SR_DI
7	DRW		1 when the thread is in fast mode and will continually issue. SR_FAST
6	DRW		1 when the thread is paused waiting for events, a lock or another resource. SR_WAITING
5	RO	0	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		1 when interrupts are enabled for the thread. SR_IEBLE
0	DRW		1 when 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.

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.

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.

Bits	Perm	Init	Description, Identifier
31:8	RO	0	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

Bits	Perm	Init	Description, Identifier
31:5	RO	0	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.

Bits	Perm	Init	Description, Identifier
31:18	RO	0	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	0	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.

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.

Bits	Perm	Init	Description, Identifier
31:8	RO	0	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*.

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.

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.

Bits	Perm	Init	Description, Identifier
31:24	RO	0	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	0	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 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. Condition *A* of a watchpoint is met if the effective address of an instruction is greater than or equal to the value in this register. The CTRL register for the watchpoint will dictate whether the watchpoint triggers on stores only or on loads and stores, and whether it requires either condition *A* or *B*, or both *A* and *B*.

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. Condition *B* of a watchpoint is met if the effective address of an instruction is less than or equal to the value in this register. The CTRL register for the watchpoint will dictate whether the watchpoint triggers on stores only or on loads and stores, and whether it requires either condition *A* or *B*, or both *A* and *B*.

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.

Bits	Perm	Init	Description, Identifier
31:24	RO	0	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	0	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 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.

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.

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.

Bits	Perm	Init	Description, Identifier
31:24	RO	0	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	0	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 breakpoint is enabled. BRK_ENABLE

B.29 The number of cache misses CACHE_MISS_CNT 0xA0

This is a free running, unresetable, read-only counter incremented on every cache miss by any thread to either SWMEM or EXTMEM.

Bits	Perm	Init	Description, Identifier
31:0	RO		Value. ALL_BITS

B.30 The total number of cache accesses CACHE_ACCESS_CNT 0xA1

This is a free running, unresetable, read-only counter incremented on every cache access by any thread to either SWMEM or EXTMEM.

Bits	Perm	Init	Description, Identifier
31:0	RO		Value. ALL_BITS

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`"

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	*PSwitch permissions*	DBG_CTRL
0x05	CRW	*Cause debug interrupts*	DBG_INT
0x06	CRW	*xCORE Tile clock divider*	PLL_CLK_DIVIDER
0x07	CRO	*Switch security configuration*	SECU_CONFIG
0x20..0x27	CRW	*Switch 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

Bits	Perm	Init	Description, Identifier
31:24	CRO		The least significant byte of the unique Tile ID. Note byte reversal within register. This is derived from the Node ID and the tile index. DEVICE_ID0_PID
23:16	CRO		The most significant byte of the unique Tile ID. Note byte reversal within register. This is derived from the Node ID and the tile index. 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.

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.

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

C.4 PSwitch permissions 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.

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	0	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.

Bits	Perm	Init	Description, Identifier
31:2	RO	0	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*

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	0	Reserved
15:0	CRW	0	Clock divider. PLL_CLK_DIVIDER

C.7 Switch security configuration SECU_CONFIG 0x07

Copy of the security register as read from OTP.

Bits	Perm	Init	Description, Identifier
31	CRO		Disables write permission on this register SECUR_CFG_DISABLE_ACCESS
30:15	RO	0	Reserved
14	CRO		Disable access to XCore's global debug SECUR_CFG_DISABLE_GLOBAL_DEBUG
13:10	RO	0	Reserved
9	CRO		Disable read access to OTP. SECUR_CFG_OTP_READ_LOCK
8	CRO		Prevent access to OTP SBPI interface to prevent programming and other functions. SECUR_CFG_OTP_PROGRAM_DISABLE
7	CRO		Combine OTP into a single address-space for reading. SECUR_CFG_OTP_COMBINED
6	RO	0	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	0	Reserved
0	CRO		Disable access to XCore's JTAG debug TAP SECUR_CFG_DISABLE_XCORE_JTAG

C.8 Switch 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*.

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.

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.

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.

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.

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.

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.

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.

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.

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

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

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

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

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

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

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

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

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`"

Number	Perm	Description	Register Identifier
0x00	RO	*Switch 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	RO	*System JTAG device ID register*	JTAG_DEVICE_ID
0x0A	RO	*System USERCODE register*	JTAG_USERCODE
0x0B	RW	*LPDDR clock*	DDR_CLK_DIVIDER
0x0C	RW	*Directions 0-7*	DIMENSION_DIRECTION0
0x0D	RW	*Directions 8-15*	DIMENSION_DIRECTION1
0x0E	RW	*Application clock divider*	SS_APP_CLK_DIVIDER
0x0F	RW	*Secondary PLL settings*	SS_APP_PLL_CTL
0x10	RW	*DEBUG_N configuration, tile 0*	XCORE0_GLOBAL_DEBUG_CONFIG
0x11	RW	*DEBUG_N configuration, tile 1*	XCORE1_GLOBAL_DEBUG_CONFIG
0x12	RW	*Secondary PLL Fractional N Divider*	SS_APP_PLL_FRAC_N_DIVIDER
0x13	RW	*LPDDR Controller configuration*	SS_LPDDR_CONTROLLER_CONFIG
0x14	RW	*MIPI shim clock config*	MIPI_CLK_DIVIDER
0x15	RW	*MIPI PHY clock config*	MIPI_CFG_CLK_DIVIDER
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
0xC000	RW	*LPDDR enable IID transactions*	LPDDR_IID_ENABLE
0xC001	RW	*LPDDR queue assignment for data*	LPDDR_IID_0_7
0xC002	RW	*LPDDR queue assignment for instructions*	LPDDR_IID_8_15
0xC003	RW	*LPDDR Queue Control*	LPDDR_QUEUE_CONT
0xC008	RW	*LPDDR Arbiter RO priority data*	LPDDR_RO_COMMAND_QUEUE_PRIORITY
0xC009	RW	*LPDDR Arbiter RW priority data*	LPDDR_RW_COMMAND_QUEUE_PRIORITY
0xC00A	RW	*LPDDR Arbiter timeout data*	LPDDR_ARBITRATION_TIMEOUT
0xC01D	RW	*LPDDR PHY control*	LPDDR_PHY_CONTROL
0xC01E	RW	*LPDDR LMR config*	LPDDR_LMR_OPCODE
0xC01F	RW	*LPDDR EMR config*	LPDDR_EMR_OPCODE
0xC020	RW	*LPDDR timings 1*	LPDDR_PROTOCOL_ENGINE_CONF_0

continues on next page

Table 3 – continued from previous page

Number	Perm	Description	Register Identifier
0xC021	RW	*LPDDR timings 2*	LPDDR_PROTOCOL_ENGINE_CONF_1
0xD000	RW	*Padcontrol LPDDR CLK and CLK_N*	PADCTRL_CLK
0xD001	RW	*Padcontrol LPDDR CKE*	PADCTRL_CKE
0xD002	RW	*Padcontrol LPDDR CS_N*	PADCTRL_CS_N
0xD003	RW	*Padcontrol LPDDR WE_N*	PADCTRL_WE_N
0xD004	RW	*Padcontrol LPDDR CAS_N*	PADCTRL_CAS_N
0xD005	RW	*Padcontrol LPDDR RAS_N*	PADCTRL_RAS_N
0xD006	RW	*Padcontrol LPDDR A0-A13*	PADCTRL_ADDR
0xD007	RW	*Padcontrol LPDDR BA0/BA1*	PADCTRL_BA
0xD008	RW	*Padcontrol LPDDR DQ0-DQ15*	PADCTRL_DQ
0xD009	RW	*Padcontrol LPDDR UDQS/LDQS*	PADCTRL_DQS
0xD00A	RW	*Padcontrol LPDDR UDM/LDM*	PADCTRL_DM
0xE013	RW	*Mipi status*	MIPI_STATUS0
0xE014	RW	*Mipi shim status*	MIPI_SHIM_STATUS
0xE018	RW	*MIPI D-PHY reset config*	MIPI_DPHY_CFG0
0xE01B	RW	*MIPI D-PHY lane config*	MIPI_DPHY_CFG3
0xE01C	RW	*Mipi phy congif 4*	MIPI_DPHY_CFG4
0xE01F	RW	*MIPI shim configuration*	MIPI_SHIM_CFG0
0xF008	RW	*USB UTMI Config*	USB_PHY_CFG0
0xF00A	RW	*USB reset*	USB_PHY_CFG2
0xF00C	RW	*USB Shim configuration*	USB_SHIM_CFG
0xF011	RO	*USB Phy Status*	USB_PHY_STATUS
0xF020	RW	*Watchdog Config*	WATCHDOG_CFG
0xF021	RO	*Watchdog Prescaler*	WATCHDOG_PRESCALER
0xF022	RW	*Watchdog Prescaler wrap*	WATCHDOG_PRESCALER_WRAP
0xF023	RW	*Watchdog Count*	WATCHDOG_COUNT
0xF024	RO	*Watchdog Status*	WATCHDOG_STATUS

D.1 Switch device identification DEVICE_ID0 0x00

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

Bits	Perm	Init	Description, Identifier
31:24	RO	0	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.

Bits	Perm	Init	Description, Identifier
31:24	RO	0	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.

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	0	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	0	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, which uniquely identifies this node in a network. This is also used to derive the Tile ID of each tile on the Node. Each Tile ID is set to the value of the Node ID, with the lowest 1 bit replaced with the tile index.

Bits	Perm	Init	Description, Identifier
31:16	RO	0	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.

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 boot mode to boot from JTAG SS_TEST_MODE_BOOT_JTAG
28	DW		If set to 1, set the PLL to be bypassed SS_TEST_MODE_PLL_BYPASS
27	RO		Reserved
26	RO	0	Reserved
25:23	RW		Output divider value range from 0 to 7. OD value. SS_PLL_CTL_POST_DIVISOR
22:21	RO	0	Reserved
20:8	RW		Feedback multiplication ratio, range from 1 (0x0001) to 8191 (0x1FFF). F value. SS_PLL_CTL_FEEDBACK_MUL
7:6	RO	0	Reserved
5:0	RW		Oscilator input divider value range from 0 (0x00) to 63 (0x3F). 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.

Bits	Perm	Init	Description, Identifier
31:16	RO	0	Reserved
15:0	RW	0	SSwitch clock divider 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.

Bits	Perm	Init	Description, Identifier
31:16	RO	0	Reserved
15:0	RW	3	Software reference clock divider SS_SSWITCH_REF_CLK_DIV

D.8 System JTAG device ID register JTAG_DEVICE_ID 0x09

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

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 LPDDR clock DDR_CLK_DIVIDER 0x0B

Sets the ratio of the PLL/APP PLL clock and the LPDDR clock. There is a divide by 2 permanently after the clock divider to create a matched mark space ratio. The LPDDR clock needs to be set to be twice the frequency required.

Bits	Perm	Init	Description, Identifier
31	RW	0	If set to 1, the secondary PLL is used as a source for the LPDDR clock divider. By default, the output of the core PLL is used. SS_DDR_CLK_FROM_APP_PLL
30:17	RO	0	Reserved
16	RW	1	LPDDR clock divider disable. When set to 0, the divider is enabled. SS_DDR_CLK_DIV_DISABLE
15:0	RW	0	LPDDR clock divider. When set to *X* the input clock is divided by *2(X+1)*. SS_DDR_CLK_DIV

D.11 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.

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.12 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

D.13 Application clock divider SS_APP_CLK_DIVIDER 0x0E

The clock divider and output of the secondary PLL can be set in this register

Bits	Perm	Init	Description, Identifier
31	RW	0	If set to 1, the secondary PLL is used as a source for the application clock divider. By default, the output of the core PLL is used. SS_APP_CLK_FROM_APP_PLL
30:17	RO	0	Reserved
16	RW	1	Application clock divider disable. When set to 0, the divider is enabled, and pin X1D11 will be connected to the application clock rather than to port 1D. SS_APP_CLK_DIV_DISABLE
15:0	RW	0	Application clock divider. When set to *X*, the output of the secondary PLL will be divided by *2(X+1)* in order to form the output on the output pin SS_APP_CLK_DIV

D.14 Secondary PLL settings SS_APP_PLL_CTL 0x0F

A secondary on-chip PLL multiplies the input clock up to a higher frequency clock. See *Secondary PLL*.

Bits	Perm	Init	Description, Identifier
31:30	RO	0	Reserved
29	DW		If set to 1, set the APP PLL to be bypassed SS_APP_PLL_BYPASS
28	DW		If set to 1, use the output of the core PLL as input, otherwise use the crystal oscillator as input. SS_APP_PLL_INPUT_FROM_SYS_PLL
27	DW	0	If set to 1, enable the secondary PLL SS_APP_PLL_ENABLE
26	RO	0	Reserved
25:23	RW		Output divider value range from 0 to 7. OD value. SS_PLL_CTL_POST_DIVISOR
22:21	RO	0	Reserved
20:8	RW		Feedback multiplication ratio, range from 1 (0x0001) to 8191 (0x1FFF). F value. SS_PLL_CTL_FEEDBACK_MUL
7:6	RO	0	Reserved
5:0	RW		Oscilator input divider value range from 0 (0x00) to 63 (0x3F). R value. SS_PLL_CTL_INPUT_DIVISOR

D.15 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	0	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

D.16 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	0	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

D.17 Secondary PLL Fractional N Divider SS_APP_PLL_FRAC_N_DIVIDER 0x12

Controls an optional fractional N Divider on the secondary PLL. When enabled, the multiplier *F* for the secondary PLL will effectively become *F+(f+1)/(p+1)*, *f* must be less than

p. This is achieved by running the PLL with a divider *F* for the first part of the fractional period, and then *F+1* for the remainder of the period. The period is measured in input clocks divided by *R+1*.

Bits	Perm	Init	Description, Identifier
31	DW	0	When set to 1, the secondary PLL will be a fractional *N* divided PLL SS_FRAC_N_ENABLE
30:16	RO	0	Reserved
15:8	DW		The *f* value for the fractional divider. The number of clock cycles in the period that a divider *F+1* is used is *f+1*. SS_FRAC_N_F_HIGH_CYC_CNT
7:0	DW		The *p* value for the fractional divider. The period over which the fractional N divider oscillates between *F* and *F+1* is *p+1* SS_FRAC_N_PERIOD_CYC_CNT

D.18 LPDDR Controller configuration SS_LPDDR_CONTROLLER_CONFIG 0x13

Controls whether LPDDR Controller is enabled, and which core it is accessible to through the mux.

Bits	Perm	Init	Description, Identifier
31:2	RO	0	Reserved
1	DW		Defines which xCORE has access to the LPDDR controller via the mux SS_LPDDR_MUXTO_CORE1
0	DW		When set to 1 this will allow the LPDDR controller to access the pads SS_LPDDR_ENABLE

D.19 MIPI shim clock config MIPI_CLK_DIVIDER 0x14

Configures the clock to the MIPI shim, the hardware block interfacing the MIPI PHY to the xCORE.

Bits	Perm	Init	Description, Identifier
31	RW	0	If set to 1, the secondary PLL is used as a source for the MIPI shim clock divider. By default, the output of the core PLL is used. SS_MIPI_CLK_FROM_APP_PLL
30:17	RO	0	Reserved
16	RW	1	MIPI clock divider disable. When set to 0, the divider is enabled. SS_SSWITCH_MIPI_CLK_DIV_DISABLE
15:0	RW	3	MIPI shim clock divider. When set to *X* the input clock is divided by *2(X+1)*. SS_SSWITCH_MIPI_CLK_DIV

D.20 MIPI PHY clock config MIPI_CFG_CLK_DIVIDER 0x15

Configures the clock to the MIPI PHY.

Bits	Perm	Init	Description, Identifier
31	RW	1	If set to 1, the secondary PLL is used as a source for the MIPI PHY clock divider. By default, the output of the core PLL is used. SS_MIPI_CFG_CLK_FROM_APP_PLL
30:17	RO	0	Reserved
16	RW	1	MIPI PHY clock divider disable. When set to 0, the divider is enabled. SS_MIPI_CFG_CLK_DIV_DISABLE
15:0	RW	0	MIPI PHY clock divider. When set to *X*, the input clock will be divided by *2(X+1)*. SS_MIPI_CFG_CLK_DIV

D.21 Debug source GLOBAL_DEBUG_SOURCE 0x1F

Contains the source of the most recent debug event.

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

D.22 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.

Bits	Perm	Init	Description, Identifier
31:26	RO	0	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	0	Reserved
11:8	RW	0	The direction that this link operates in. LINK_DIRECTION
7:6	RO	0	Reserved
5:4	RW	0	Determines the network to which this link belongs, reset as 0. LINK_NETWORK
3	RO	0	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.23 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	0	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	0	Reserved
5:4	RO	0	Determines the network to which this link belongs, reset as 0. LINK_NETWORK
3	RO	0	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.24 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	0	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	0	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

D.25 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.

Bits	Perm	Init	Description, Identifier
31	RW	0	Enable static forwarding. XSTATIC_ENABLE
30:9	RO	0	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	0	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

D.26 USB UTMI Config USB_PHY_CFG0 0xF008

This register configures the UTMI signals to the USB PHY. See the UTMI specification for more details. The oscillator speed should be set to match the crystal on XIN/XOUT.

Bits	Perm	Init	Description, Identifier
31	RO	0	Reserved
30:15	RO	0	Reserved
14:12	RW	1	Oscillator freqeuncy. Set to: 0 (10MHz), 1 (12MHz), 2 (25MHz), 3 (30MHz), 4 (19.2MHz), 5 (24MHz), 6 (27MHz), or 7 (40MHz). USB_PHY_CFG0_XTLSEL
11	RW	0	Set to 1 to enable the ID PAD USB_PHY_CFG0_IDPAD_EN
10	RW	0	Set to 1 to enable USB LPM USB_PHY_CFG0_LPM_ALIVE
9	RW	0	Set to 1 to enable the USB PLL USB_PHY_CFG0_PLL_EN
8	RW	0	Set to 1 to enable USB Tx BitStuffing USB_PHY_CFG0_TXBITSTUFF_EN
7	RW	0	Set to 1 to enable the DM Pulldown USB_PHY_CFG0_DMPULLDOWN
6	RW	0	Set to 1 to enable the DP Pulldown USB_PHY_CFG0_DPPULLDOWN
5	RW	1	Value of the UTMI SuspendM signal to the USB Phy USB_PHY_CFG0_UTMI_SUSPENDM
4:3	RW	1	Value of the UTMI OpMode signals to the USB Phy USB_PHY_CFG0_UTMI_OPMODE
2	RW	1	Value of the UTMI Terminal Select signal to the USB Phy USB_PHY_CFG0_UTMI_TERMSELECT
1:0	RW	1	Value of the UTMI XCVRSelect signals to the USB Phy USB_PHY_CFG0_UTMI_XCVRSELECT

D.27 USB reset USB_PHY_CFG2 0xF00A

Bits	Perm	Init	Description, Identifier
31:2	RO	0	Reserved
1	RW	1	UTMI reset, set to 0 to take UTMI out of reset USB_PHY_CFG2_UTMI_RESET
0	RW	0	USB PHY reset, set to 1 to take the PHY out of reset USB_PHY_CFG2_PONRST

D.28 USB Shim configuration USB_SHIM_CFG 0xF00C

This register contains the hardware interfacing the USB PHY and the xCORE. It governs how the rxActive, rxValid, and line-state signals are mapped onto two one-bit ports.

Bits	Perm	Init	Description, Identifier
31:2	RO	0	Reserved
1	RW	0	USB flag mode selection: 1 selects linestate; 0 selects RxActive and RxValid USB_SHIM_CFG_FLAG_MODE
0	RW	0	When enabled RxValid output to xCore is AND'd with RxActive USB_SHIM_CFG_AND_RXV_RXA

D.29 USB Phy Status USB_PHY_STATUS 0xF011

Bits	Perm	Init	Description, Identifier
31:21	RO	0	Reserved
20:5	RO	0	Reserved
4	RO	0	1 if BIST succeeded USB_PHY_STATUS_BIST_OK
3	RO	0	1 if resistance of IDPAD to ground is > 100 kOhm (mini B plug) USB_PHY_STATUS_IDPAD
2	RO	0	Set to 1 if no peripheral is connected USB_PHY_STATUS_HOSTDISCONNECT
1:0	RO	0	The UTMI line state; 0: SE0, 1: J, 2: K, 3: SE1 USB_PHY_STATUS_UTMI_LINESTATE

D.30 Watchdog Config WATCHDOG_CFG 0xF020

Register to control the watchdog. By default the watchdog is neither counting, nor triggering. When used as a watchdog it should be set to both count and trigger a reset on reaching 0. It can be set to just count for debugging purposes

Bits	Perm	Init	Description, Identifier
31:2	RO	0	Reserved
1	RW	0	Set this bit to 1 to enable the watchdog to actually reset the chip. WATCHDOG_TRIGGER_ENABLE
0	RW	0	Set this bit to 1 to enable the watchdog counter. WATCHDOG_COUNT_ENABLE

D.31 Watchdog Prescaler WATCHDOG_PRESCALER 0xF021

Register to read out the current divider counter. Can be used to implement a timer that is independent of the PLL.

Bits	Perm	Init	Description, Identifier
31:16	RO	0	Reserved
15:0	RO	0	This is the current count of the prescaler. One is added one every input clock edge on the oscillator (XIN). When it reaches the prescaler wrap value (see below), it resets to zero and one is subtracted from the watchdog count (see below). WATCHDOG_PRESCALER_VALUE

D.32 Watchdog Prescaler wrap WATCHDOG_PRESCALER_WRAP 0xF022

Register to set the watchdog pre-scale divider value.

Bits	Perm	Init	Description, Identifier
31:16	RO	0	Reserved
15:0	RW	0xFFFF	This is the prescaler divider. The input clock on XIN is divided by this value plus one, before being used to adjust the watchdog count (see below). WATCHDOG_PRESCALER_WRAP_VALUE

D.33 Watchdog Count WATCHDOG_COUNT 0xF023

Register to set the value at which the watchdog timer should time out. This register must be overwritten regularly to stop the watchdog from resetting the chip.

Bits	Perm	Init	Description, Identifier
31:12	RO	0	Reserved
11:0	RW	0xFFF	This is the watchdog counter. It counts down every PRESCALER_WRAP_VALUE input clock edges. When it reaches zero the chip is reset. The maximum time for the watchdog is $2^{12} \times 2^{16} = 2^{28} = 268{,}435{,}456$ input clocks. WATCHDOG_COUNT_VALUE

D.34 Watchdog Status WATCHDOG_STATUS 0xF024

Register that can be used to inspect whether the watchdog has triggered.

Bits	Perm	Init	Description, Identifier
31:1	RO	0	Reserved
0	RO	0	When 1, the watchdog has been triggered. This bit is only reset to 0 on a power-on-reset. WATCHDOG_HAS_TRIGGERED

D.35 Mipi status MIPI_STATUS0 0xE013

Bits	Perm	Init	Description, Identifier
31	RO	0	Reserved
30:15	RO	0	Reserved
14	RO		Lane 1 is in the stop state MIPI_STATUS0_STOPSTATE_LAN1
13	RO		Lane 0 is in the stop state MIPI_STATUS0_STOPSTATE_LAN0
12	RO		Clock lane is in the stop state MIPI_STATUS0_STOPSTATE_CLK
11:6	RO		Test mode da cdphy r100 control0 2d1c MIPI_STATUS0_DA_CDPHY_R100_CTRL0_2D1C
5	RO		Test mode data correct lan2 MIPI_STATUS0_DATA_CORRECT_LAN2
4	RO		Test mode data correct lan1 MIPI_STATUS0_DATA_CORRECT_LAN1
3	RO		Test mode data correct lan0 MIPI_STATUS0_DATA_CORRECT_LAN0
2	RO		Test mode bit clk greater than 2400G MIPI_STATUS0_BIT_CLK_GREATER_THAN_2400G
1	RO		Test mode osc clock ready MIPI_STATUS0_OSC_CLK_READY
0	RO		Test mode osc clock act MIPI_STATUS0_OSC_CLK_ACT

D.36 Mipi shim status MIPI_SHIM_STATUS 0xE014

This register provides status for the MIPI demuxing logic

Bits	Perm	Init	Description, Identifier
31:1	RO	0	Reserved
0	RW	0	Set to 1 if an overflow has been detected in the DEMUXER. This is not recoverable, and indicates that the MIPI_CLK is too slow for the rate at which data is received. MIPI_SHIM_STATUS_REG

D.37 MIPI D-PHY reset config MIPI_DPHY_CFG0 0xE018

Controls the reset signals to the MIPI D-PHY

Bits	Perm	Init	Description, Identifier
31:2	RO	0	Reserved
1	RW	0	Set to 1 MIPI_DPHY_CFG0_RSTB09_ALWAYS_ON
0	RW	0	Reset, set to 1 to take the MIPI PHY out of reset MIPI_DPHY_CFG0_HW_RSTN

D.38 MIPI D-PHY lane config MIPI_DPHY_CFG3 0xE01B

Configures the settings for the three lanes, in particular, where the wires appear on the physical interfaces and which ones are enabled.

Bits	Perm	Init	Description, Identifier
31	RO	0	Reserved
30:15	RO	0	Reserved
14	RW	1	Set to 0 to disable lane 1 receiver MIPI_DPHY_CFG3_ENABLE_LAN1
13	RW	1	Set to 0 to disable lane 0 receiver MIPI_DPHY_CFG3_ENABLE_LAN0
12	RW	1	Set to 0 to disable the clock lane receiver MIPI_DPHY_CFG3_ENABLE_CLK
11	RW	0	Set to 1 to swap the DN/DP pair on the lane 1 MIPI_DPHY_CFG3_DPDN_SWAP_LAN1
10	RW	0	Set to 1 to swap the DN/DP pair on the lane 0 MIPI_DPHY_CFG3_DPDN_SWAP_LAN0
9	RW	0	Set to 1 to swap the DN/DP pair on the clock lane MIPI_DPHY_CFG3_DPDN_SWAP_CLK
8:6	RW	2	The DP/DN pair over which to input lane 1 (if two lanes are needed) MIPI_DPHY_CFG3_LANE_SWAP_LAN1
5:3	RW	0	The DP/DN pair over which to input lane 0 MIPI_DPHY_CFG3_LANE_SWAP_LAN0
2:0	RW	1	The DP/DN pair over which to input the clock MIPI_DPHY_CFG3_LANE_SWAP_CLK

D.39 Mipi phy congif 4 MIPI_DPHY_CFG4 0xE01C

Bits	Perm	Init	Description, Identifier
31:24	RO	0	Reserved
23:16	RW	0xA	MIPI dphy Tclk-settle in lane 1 MIPI_DPHY_CFG4_PRECOUNTER_IN_LAN1
15:8	RW	0xA	MIPI dphy Tclk-settle in lane 0 MIPI_DPHY_CFG4_PRECOUNTER_IN_LAN0
7:0	RW	0xB	MIPI dphy Tclk-settle for clock MIPI_DPHY_CFG4_PRECOUNTER_IN_CLK

D.40 MIPI shim configuration MIPI_SHIM_CFG0 0xE01F

This register is used to configure the MIPI shim, the hardware block interfacing the MIPI D-PHY to the xCORE. By default the MIPI shim just passes the data from the MIPI D-PHY straight through to the receiver. This register enables you to demultiplex 10-bit, 12-bit, 14-bit and 565-data into 16-bit and 8-bit values. When the demultiplexer is enabled, you must specify the CSI-2 packet type that demultiplexing should apply to. Optionally, you can choose to align add an extra fourth byte for RGB formats, or you can choose to bias the data so that all the data values are signed.

Bits	Perm	Init	Description, Identifier
31:27	RO	0	Reserved
26	RW	0	MIPI shim config0 sel debug MIPI_SHIM_CFG0_SEL_DEBUG
25	RW	0	MIPI shim config0 sel debug out MIPI_SHIM_CFG0_SEL_DEBUG_OUT
24	RO	0	Reserved
23	RW	0	Set to 1 to offset the output pixels with -0x80 (for 8-bit outputs) or -0x8000 (for 16-bit outputs). This can be used to make unsigned data signed around zero. MIPI_SHIM_BIAS
22	RW	0	Set to 1 to add an extra data byte after every RGB565 or RGB888 pixel. This will align pixels to a 32-bit word. MIPI_SHIM_DEMUX_STUFF
21:16	RW	0	Specifies how the demultiplexer operates. The modes supported are 10to16, 12to16, 14to16, rgb565to888, rgb888to888. MIPI_SHIM_CFG0_PIXEL_DEMUX_MODE
15:8	RW	0	This field needs to be set to the CSI-2 packet type that needs to be demuxed. Only packets with a matching type are demultiplexed. MIPI_SHIM_CFG0_PIXEL_DEMUX_DATATYPE
7:1	RO	0	Reserved
0	RW	0	Set to 1 to enable the MIPI shim to demultiplex data according to the demux mode and stuff fields. Demuxing is only applied to packets that have the correct datatype. MIPI_SHIM_CFG0_PIXEL_DEMUX_EN

D.41 LPDDR enable IID transactions LPDDR_IID_ENABLE 0xC000

This register is used to enable one or more threads to route its requests through specified queues. There are three queues (one read-only queue, RO, and two read-write queues, RW0/RW1) and for each thread instruction accesses and data accesses can be routed through specified queues.

Bits	Perm	Init	Description, Identifier
31	RO	0	Reserved
30:15	RO	0	Reserved
15:0	RW	0	Two 8-bit masks, one bit per thread. Top eight bits enable instructions to be routed through a specified queue, bottom eight bits enable data to be routed through a specified queue. LPDDR_IID_ENABLE

D.42 LPDDR queue assignment for data LPDDR_IID_0_7 0xC001

For each thread, this register specifies which queue a data access should be routed through.

Bits	Perm	Init	Description, Identifier
31:0	RW	0	Four bits per thread. Top bit sets the queue type that this thread should be using (0: RO, 1: RW), further three bits the number of the queue. Valid values for the further three bits are 000 for RO queues, and 000/001 for a RW queue. LPDDR_IID_0_7

D.43 LPDDR queue assignment for instructions LPDDR_IID_8_15 0xC002

For each thread, this register specifies which queue an instruction access should be routed through.

Bits	Perm	Init	Description, Identifier
31:0	RW	0	Four bits per thread. Top bit sets the queue type that this thread should be using (0: RO, 1: RW), further three bits the number of the queue. Valid values for the further three bits are 000 for RO queues, and 000/001 for a RW queue. LPDDR_IID_8_15

D.44 LPDDR Queue Control LPDDR_QUEUE_CONT 0xC003

Bits	Perm	Init	Description, Identifier
31:1	RO	0	Reserved
0	RW	0	Slow sys clock. Set this bit if the tile clock is less than the LPDDR clock. LPDDR_QUEUE_CONT

D.45 LPDDR Arbiter RO priority data LPDDR_RO_COMMAND_QUEUE_PRIORITY 0xC008

Bits	Perm	Init	Description, Identifier
31:3	RO	0	Reserved
2:0	RW	7	Priority for RO queue. Zero is lowest priority. LPDDR_RO_PRI

D.46 LPDDR Arbiter RW priority data LPDDR_RW_COMMAND_QUEUE_PRIORITY 0xC009

Bits	Perm	Init	Description, Identifier
31:6	RO	0	Reserved
5:3	RW	0	Priority for RW queue 1. Zero is lowest priority. LPDDR_RW1_PRI
2:0	RW	5	Priority for RW queue 0. Zero is lowest priority. LPDDR_RW0_PRI

D.47 LPDDR Arbiter timeout data LPDDR_ARBITRATION_TIMEOUT 0xC00A

Setting this to a non-zero value guarantees that each queue is served at least every *N* transactions and prevents starvation.

Bits	Perm	Init	Description, Identifier
31:4	RO	0	Reserved
3:0	RW	4	Maximum number of transactions until a queue is served. Set to 0 to disable a timeout LPDDR_TOUT

D.48 LPDDR PHY control LPDDR_PHY_CONTROL 0xC01D

Bits	Perm	Init	Description, Identifier
31:14	RO	0	Reserved
13:0	RW	0x2101	PHY Control LPDDR_PHY_CONTROL

D.49 LPDDR LMR config LPDDR_LMR_OPCODE 0xC01E

Bits	Perm	Init	Description, Identifier
31:14	RO	0	Reserved
13:0	RW	0x0034	LMR opcode LPDDR_LMR_OPCODE

D.50 LPDDR EMR config LPDDR_EMR_OPCODE 0xC01F

Bits	Perm	Init	Description, Identifier
31:14	RO	0	Reserved
13:0	RW	0x0000	EMR opcode LPDDR_EMR_OPCODE

D.51 LPDDR timings 1 LPDDR_PROTOCOL_ENGINE_CONF_0 0xC020

Register used to set the tREFI, tRAS, tXSR, and tWR timings, all measured in terms of LPDDR clocks

Bits	Perm	Init	Description, Identifier
31:24	RO	0	Reserved
23:21	RW	1	LPDDR tWR clock count LPDDR_PE_TWR_CNT
20:15	RW	39	LPDDR tXSR clock count LPDDR_PE_TXSR_CNT
14:11	RW	8	LPDDR tRAS clock count LPDDR_PE_TRAS_CNT
10:0	RW	779	LPDDR tREFI clock count LPDDR_PE_TREFI_CNT

D.52 LPDDR timings 2 LPDDR_PROTOCOL_ENGINE_CONF_1 0xC021

Register used to set the tRRC, tRCD, tRP, tRFC, and tRRD timings, all measured in terms of LPDDR clocks. This register is also used to configure the use of 256 bit memories.

Bits	Perm	Init	Description, Identifier
31:18	RO	0	Reserved
17	RW	0	Enable 256 Mbit device LPDDR_PE_EN_256M_DEV_SIZE
16:15	RW	1	LPDDR tRRD clock count LPDDR_PE_TRRD_CNT
14:10	RW	27	LPDDR tRFC clock count LPDDR_PE_TRFC_CNT
9:7	RW	4	LPDDR tRP clock count LPDDR_PE_TRP_CNT
6:4	RW	4	LPDDR tRCD clock count LPDDR_PE_TRCD_CNT
3:0	RW	11	LPDDR tRC clock count LPDDR_PE_TRC_CNT

D.53 Padcontrol LPDDR CLK and CLK_N PADCTRL_CLK 0xD000

When LPDDR is enabled, this register controls the PAD properties for the CLK and CLK_N pins

Bits	Perm	Init	Description, Identifier
31:7	RO	0	Reserved
6	RW	0	Set to 1 to enable the schmitt trigger PADCTRL_SCHMITT_TRIGGER_ENABLE
5	RW	0	Set to 1 to enable slew-rate control PADCTRL_SLEW_RATE_CONTROL
4:3	RW	10	Pad drive strength: 00 for 2 mA; 01 for 4 mA; 10 for 8 mA; or 11 for 12 mA. PADCTRL_DRIVE_STRENGTH
2:1	RW	00	Pull resistor: 00 for none; 01 for weak pull-up; 10 for weak pull-down; or 11 for weak bus-keep. PADCTRL_PULL
0	RW	0	Set to 1 to enable the input receiver PADCTRL_RECEIVER_ENABLE

D.54 Padcontrol LPDDR CKE PADCTRL_CKE 0xD001

When LPDDR is enabled, this register controls the PAD properties for the CKE pin

Bits	Perm	Init	Description, Identifier
31:7	RO	0	Reserved
6	RW	0	Set to 1 to enable the schmitt trigger PADCTRL_SCHMITT_TRIGGER_ENABLE
5	RW	0	Set to 1 to enable slew-rate control PADCTRL_SLEW_RATE_CONTROL
4:3	RW	10	Pad drive strength: 00 for 2 mA; 01 for 4 mA; 10 for 8 mA; or 11 for 12 mA. PADCTRL_DRIVE_STRENGTH
2:1	RW	00	Pull resistor: 00 for none; 01 for weak pull-up; 10 for weak pull-down; or 11 for weak bus-keep. PADCTRL_PULL
0	RW	0	Set to 1 to enable the input receiver PADCTRL_RECEIVER_ENABLE

D.55 Padcontrol LPDDR CS_N PADCTRL_CS_N 0xD002

When LPDDR is enabled, this register controls the PAD properties for the CS_N pin

Bits	Perm	Init	Description, Identifier
31:7	RO	0	Reserved
6	RW	0	Set to 1 to enable the schmitt trigger PADCTRL_SCHMITT_TRIGGER_ENABLE
5	RW	0	Set to 1 to enable slew-rate control PADCTRL_SLEW_RATE_CONTROL
4:3	RW	10	Pad drive strength: 00 for 2 mA; 01 for 4 mA; 10 for 8 mA; or 11 for 12 mA. PADCTRL_DRIVE_STRENGTH
2:1	RW	00	Pull resistor: 00 for none; 01 for weak pull-up; 10 for weak pull-down; or 11 for weak bus-keep. PADCTRL_PULL
0	RW	0	Set to 1 to enable the input receiver PADCTRL_RECEIVER_ENABLE

D.56 Padcontrol LPDDR WE_N PADCTRL_WE_N 0xD003

When LPDDR is enabled, this register controls the PAD properties for the WE_N pin

Bits	Perm	Init	Description, Identifier
31:7	RO	0	Reserved
6	RW	0	Set to 1 to enable the schmitt trigger PADCTRL_SCHMITT_TRIGGER_ENABLE
5	RW	0	Set to 1 to enable slew-rate control PADCTRL_SLEW_RATE_CONTROL
4:3	RW	10	Pad drive strength: 00 for 2 mA; 01 for 4 mA; 10 for 8 mA; or 11 for 12 mA. PADCTRL_DRIVE_STRENGTH
2:1	RW	00	Pull resistor: 00 for none; 01 for weak pull-up; 10 for weak pull-down; or 11 for weak bus-keep. PADCTRL_PULL
0	RW	0	Set to 1 to enable the input receiver PADCTRL_RECEIVER_ENABLE

D.57 Padcontrol LPDDR CAS_N PADCTRL_CAS_N 0xD004

When LPDDR is enabled, this register controls the PAD properties for the CAS_N pin

Bits	Perm	Init	Description, Identifier
31:7	RO	0	Reserved
6	RW	0	Set to 1 to enable the schmitt trigger PADCTRL_SCHMITT_TRIGGER_ENABLE
5	RW	0	Set to 1 to enable slew-rate control PADCTRL_SLEW_RATE_CONTROL
4:3	RW	10	Pad drive strength: 00 for 2 mA; 01 for 4 mA; 10 for 8 mA; or 11 for 12 mA. PADCTRL_DRIVE_STRENGTH
2:1	RW	00	Pull resistor: 00 for none; 01 for weak pull-up; 10 for weak pull-down; or 11 for weak bus-keep. PADCTRL_PULL
0	RW	0	Set to 1 to enable the input receiver PADCTRL_RECEIVER_ENABLE

D.58 Padcontrol LPDDR RAS_N PADCTRL_RAS_N 0xD005

When LPDDR is enabled, this register controls the PAD properties for the RAS_N pin

Bits	Perm	Init	Description, Identifier
31:7	RO	0	Reserved
6	RW	0	Set to 1 to enable the schmitt trigger PADCTRL_SCHMITT_TRIGGER_ENABLE
5	RW	0	Set to 1 to enable slew-rate control PADCTRL_SLEW_RATE_CONTROL
4:3	RW	10	Pad drive strength: 00 for 2 mA; 01 for 4 mA; 10 for 8 mA; or 11 for 12 mA. PADCTRL_DRIVE_STRENGTH
2:1	RW	00	Pull resistor: 00 for none; 01 for weak pull-up; 10 for weak pull-down; or 11 for weak bus-keep. PADCTRL_PULL
0	RW	0	Set to 1 to enable the input receiver PADCTRL_RECEIVER_ENABLE

D.59 Padcontrol LPDDR A0-A13 PADCTRL_ADDR 0xD006

When LPDDR is enabled, this register controls the PAD properties for the A0-A13 pins

Bits	Perm	Init	Description, Identifier
31:7	RO	0	Reserved
6	RW	0	Set to 1 to enable the schmitt trigger PADCTRL_SCHMITT_TRIGGER_ENABLE
5	RW	0	Set to 1 to enable slew-rate control PADCTRL_SLEW_RATE_CONTROL
4:3	RW	10	Pad drive strength: 00 for 2 mA; 01 for 4 mA; 10 for 8 mA; or 11 for 12 mA. PADCTRL_DRIVE_STRENGTH
2:1	RW	00	Pull resistor: 00 for none; 01 for weak pull-up; 10 for weak pull-down; or 11 for weak bus-keep. PADCTRL_PULL
0	RW	0	Set to 1 to enable the input receiver PADCTRL_RECEIVER_ENABLE

D.60 Padcontrol LPDDR BA0/BA1 PADCTRL_BA 0xD007

When LPDDR is enabled, this register controls the PAD properties for the BA0 and BA1 pins

Bits	Perm	Init	Description, Identifier
31:7	RO	0	Reserved
6	RW	0	Set to 1 to enable the schmitt trigger PADCTRL_SCHMITT_TRIGGER_ENABLE
5	RW	0	Set to 1 to enable slew-rate control PADCTRL_SLEW_RATE_CONTROL
4:3	RW	10	Pad drive strength: 00 for 2 mA; 01 for 4 mA; 10 for 8 mA; or 11 for 12 mA. PADCTRL_DRIVE_STRENGTH
2:1	RW	00	Pull resistor: 00 for none; 01 for weak pull-up; 10 for weak pull-down; or 11 for weak bus-keep. PADCTRL_PULL
0	RW	0	Set to 1 to enable the input receiver PADCTRL_RECEIVER_ENABLE

D.61 Padcontrol LPDDR DQ0-DQ15 PADCTRL_DQ 0xD008

When LPDDR is enabled, this register controls the PAD properties for the DQ0-DQ15 pins

Bits	Perm	Init	Description, Identifier
31:7	RO	0	Reserved
6	RW	0	Set to 1 to enable the schmitt trigger PADCTRL_SCHMITT_TRIGGER_ENABLE
5	RW	0	Set to 1 to enable slew-rate control PADCTRL_SLEW_RATE_CONTROL
4:3	RW	10	Pad drive strength: 00 for 2 mA; 01 for 4 mA; 10 for 8 mA; or 11 for 12 mA. PADCTRL_DRIVE_STRENGTH
2:1	RW	00	Pull resistor: 00 for none; 01 for weak pull-up; 10 for weak pull-down; or 11 for weak bus-keep. PADCTRL_PULL
0	RW	0	Set to 1 to enable the input receiver PADCTRL_RECEIVER_ENABLE

D.62 Padcontrol LPDDR UDQS/LDQS PADCTRL_DQS 0xD009

When LPDDR is enabled, this register controls the PAD properties for the UDQS and LDQS pins

Bits	Perm	Init	Description, Identifier
31:7	RO	0	Reserved
6	RW	0	Set to 1 to enable the schmitt trigger PADCTRL_SCHMITT_TRIGGER_ENABLE
5	RW	0	Set to 1 to enable slew-rate control PADCTRL_SLEW_RATE_CONTROL
4:3	RW	10	Pad drive strength: 00 for 2 mA; 01 for 4 mA; 10 for 8 mA; or 11 for 12 mA. PADCTRL_DRIVE_STRENGTH
2:1	RW	00	Pull resistor: 00 for none; 01 for weak pull-up; 10 for weak pull-down; or 11 for weak bus-keep. PADCTRL_PULL
0	RW	0	Set to 1 to enable the input receiver PADCTRL_RECEIVER_ENABLE

D.63 Padcontrol LPDDR UDM/LDM PADCTRL_DM 0xD00A

When LPDDR is enabled, this register controls the PAD properties for the UDM and LDM pins

Bits	Perm	Init	Description, Identifier
31:7	RO	0	Reserved
6	RW	0	Set to 1 to enable the schmitt trigger PADCTRL_SCHMITT_TRIGGER_ENABLE
5	RW	0	Set to 1 to enable slew-rate control PADCTRL_SLEW_RATE_CONTROL
4:3	RW	10	Pad drive strength: 00 for 2 mA; 01 for 4 mA; 10 for 8 mA; or 11 for 12 mA. PADCTRL_DRIVE_STRENGTH
2:1	RW	00	Pull resistor: 00 for none; 01 for weak pull-up; 10 for weak pull-down; or 11 for weak bus-keep. PADCTRL_PULL
0	RW	0	Set to 1 to enable the input receiver PADCTRL_RECEIVER_ENABLE

E Signal List

E.1 FB265 signal list

A list of pins by function can be found in *Pin Configuration*

Signal	Pin	Function	Type	Properties
X0D04	A1	$4B^{0}$; $8A^{2}$; $16A^{2}$; $32A^{22}$	I/O	IOL
VSS	A2	Digital ground	GND	
X0D32	A3	DQ2; $4E^{2}$; $8C^{6}$; $16B^{6}$	I/O	IOT
X0D30	A4	DQ4; $4F^{2}$; $8C^{4}$; $16B^{4}$	I/O	IOT
X1D34	A5	DQ6; $1K^{0}$	I/O	IOT
X1D32	A6	LDQS; $4E^{2}$; $8C^{6}$; $16B^{6}$	I/O	IOT
X1D31	A7	LDM; $4F^{3}$; $8C^{5}$; $16B^{5}$	I/O	IOT
VDDIOT	A8	Digital I/O power (top)	PWR	
X0D67	A9	RASN; $32A^{16}$	I/O	IOT
X0D65	A10	WEN; $32A^{14}$	I/O	IOT
X0D61	A11	CKE; $32A^{10}$	I/O	IOT
VSS	A12	Digital ground	GND	
VSS	A13	Digital ground	GND	
X0D49	A14	A7; $32A^{0}$	I/O	IOT
X0D51	A15	A5; $32A^{2}$	I/O	IOT
X0D41	A16	A11; $8D^{5}$; $16B^{13}$	I/O	IOT
X0D39	A17	A13; $1P^{0}$; $8D^{3}$; $16B^{11}$	I/O	IOT
X0D07	B1	$4B^{3}$; $8A^{5}$; $16A^{5}$; $32A^{25}$	I/O	IOL
X0D06	B2	$4B^{2}$; $8A^{4}$; $16A^{4}$; $32A^{24}$	I/O	IOL
X0D34	B3	DQ0; $1K^{0}$	I/O	IOT
X0D33	B4	DQ1; $4E^{3}$; $8C^{7}$; $16B^{7}$	I/O	IOT
X0D31	B5	DQ3; $4F^{3}$; $8C^{5}$; $16B^{5}$	I/O	IOT
X1D35	B6	DQ5; $1L^{0}$	I/O	IOT
X1D33	B7	DQ7; $4E^{3}$; $8C^{7}$; $16B^{7}$	I/O	IOT
VSS	B8	Digital ground	GND	
X0D66	B9	CASN; $32A^{15}$	I/O	IOT
X0D64	B10	BA0; $32A^{13}$	I/O	IOT
VSS	B11	Digital ground	GND	
VDDIOT	B12	Digital I/O power (top)	PWR	
X0D50	B13	A6; $32A^{1}$	I/O	IOT
X0D52	B14	A4; $32A^{3}$	I/O	IOT
X1D43	B15	A8; $8D^{7}$; $16B^{15}$	I/O	IOT
X0D40	B16	A12; $8D^{4}$; $16B^{12}$	I/O	IOT
VDDIOT	B17	Digital I/O power (top)	PWR	
X0D01	C1	$1B^{0}$	I/O	IOL
X0D05	C2	$4B^{1}$; $8A^{3}$; $16A^{3}$; $32A^{23}$	I/O	IOL
VSS	C3	Digital ground	GND	
VDDIOT	C4	Digital I/O power (top)	PWR	

continues on next page

Table 4 – continued from previous page

Signal	Pin	Function	Type	Properties
X1D29	C5	DQ14; 4F[1]; 8C[3]; 16B[3]	I/O	IOT
X1D27	C6	DQ12; 4E[1]; 8C[1]; 16B[1]	I/O	IOT
X1D25	C7	DQ10; 1J[0]	I/O	IOT
X0D70	C8	DQ8; 32A[19]	I/O	IOT
X0D68	C9	UDM; 32A[17]	I/O	IOT
X0D63	C10	BA1; 32A[12]	I/O	IOT
X0D58	C11	CK; 32A[9]	I/O	IOT
X0D57	C12	CKN; 32A[8]	I/O	IOT
X0D55	C13	A1; 32A[6]	I/O	IOT
X0D53	C14	A3; 32A[4]	I/O	IOT
X0D42	C15	A10; 8D[6]; 16B[14]	I/O	IOT
VSS	C16	Digital ground	GND	
VSS	C17	Digital ground	GND	
X0D00	D1	$XL4_{in}$[3]; 1A[0]	I/O	IOL
X0D10	D2	$XL4_{in}$[4]; 1C[0]	I/O	IOL
X0D15	D3	$XL4_{in}$[0]; 4C[1]; 8B[1]; 16A[9]; 32A[29]	I/O	IOL
X0D14	D4	$XL4_{in}$[1]; 4C[0]; 8B[0]; 16A[8]; 32A[28]	I/O	IOL
X1D30	D5	DQ15; 4F[2]; 8C[4]; 16B[4]	I/O	IOT
X1D28	D6	DQ13; 4F[0]; 8C[2]; 16B[2]	I/O	IOT
X1D26	D7	DQ11; 4E[0]; 8C[0]; 16B[0]	I/O	IOT
X1D24	D8	DQ9; 1I[0]	I/O	IOT
X0D69	D9	UDQS; 32A[18]	I/O	IOT
VSS	D10	Digital ground	GND	
X0D62	D11	CSN; 32A[11]	I/O	IOT
X0D56	D12	A0; 32A[7]	I/O	IOT
X0D54	D13	A2; 32A[5]	I/O	IOT
X0D43	D14	A9; 8D[7]; 16B[15]	I/O	IOT
VSS	D15	Digital ground	GND	
X0D37	D16	$XL3_{out}$[3]; 1N[0]; 8D[1]; 16B[9]	I/O	IOR
X0D38	D17	$XL3_{out}$[4]; 1O[0]; 8D[2]; 16B[10]	I/O	IOR
X0D18	E1	$XL4_{out}$[2]; 4D[2]; 8B[4]; 16A[12]	I/O	IOL
X0D11	E2	$XL4_{in}$[2]; 1D[0]	I/O	IOL
X0D17	E3	$XL4_{out}$[1]; 4D[1]; 8B[3]; 16A[11]	I/O	IOL
X0D16	E4	$XL4_{out}$[0]; 4D[0]; 8B[2]; 16A[10]	I/O	IOL
VSS	E5	Digital ground	GND	
VDD	E6	Digital tile power	PWR	
VSS	E7	Digital ground	GND	
VDD	E8	Digital tile power	PWR	
VDDIOT	E9	Digital I/O power (top)	PWR	
VDD	E10	Digital tile power	PWR	
VSS	E11	Digital ground	GND	
VDD	E12	Digital tile power	PWR	
VSS	E13	Digital ground	GND	

continues on next page

Table 4 – continued from previous page

Signal	Pin	Function	Type	Properties
X0D29	E14	$XL3_{out}^{0}$; $4F^{1}$; $8C^{3}$; $16B^{3}$	I/O	IOR
X0D35	E15	$XL3_{out}^{1}$; $1L^{0}$	I/O	IOR
X0D26	E16	$XL3_{in}^{2}$; $4E^{0}$; $8C^{0}$; $16B^{0}$	I/O	IOR
X0D36	E17	$XL3_{out}^{2}$; $1M^{0}$; $8D^{0}$; $16B^{8}$	I/O	IOR
X0D20	F1	$XL4_{out}^{4}$; $4C^{2}$; $8B^{6}$; $16A^{14}$; $32A^{30}$	I/O	IOL
X0D19	F2	$XL4_{out}^{3}$; $4D^{3}$; $8B^{5}$; $16A^{13}$	I/O	IOL
X1D39	F3	$XL5_{in}^{0}$; $1P^{0}$; $8D^{3}$; $16B^{11}$	I/O	IOL
X1D38	F4	$XL5_{in}^{1}$; $1O^{0}$; $8D^{2}$; $16B^{10}$	I/O	IOL
VDD	F5	Digital tile power	PWR	
VDD	F13	Digital tile power	PWR	
X0D27	F14	$XL3_{in}^{1}$; $4E^{1}$; $8C^{1}$; $16B^{1}$	I/O	IOR
X0D28	F15	$XL3_{in}^{0}$; $4F^{0}$; $8C^{2}$; $16B^{2}$	I/O	IOR
X0D24	F16	$XL3_{in}^{4}$; $1I^{0}$	I/O	IOR
X0D25	F17	$XL3_{in}^{3}$; $1J^{0}$	I/O	IOR
X1D36	G1	$XL5_{in}^{3}$; $1M^{0}$; $8D^{0}$; $16B^{8}$	I/O	IOL
X0D21	G2	$XL5_{in}^{4}$; $4C^{3}$; $8B^{7}$; $16A^{15}$; $32A^{31}$	I/O	IOL
X1D41	G3	$XL5_{out}^{1}$; $8D^{5}$; $16B^{13}$	I/O	IOL
X1D40	G4	$XL5_{out}^{0}$; $8D^{4}$; $16B^{12}$	I/O	IOL
VDDIOL	G5	Digital I/O power (left)	PWR	
VSS	G7	Digital ground	GND	
VSS	G8	Digital ground	GND	
VSS	G9	Digital ground	GND	
VSS	G10	Digital ground	GND	
VSS	G11	Digital ground	GND	
VDDIOR	G13	Digital I/O power (right)	PWR	
X1D66	G14	$XL2_{out}^{0}$; $32A^{15}$	I/O	IOR
X1D67	G15	$XL2_{out}^{1}$; $32A^{16}$	I/O	IOR
X1D69	G16	$XL2_{out}^{3}$; $32A^{18}$	I/O	IOR
X1D70	G17	$XL2_{out}^{4}$; $32A^{19}$	I/O	IOR
X1D42	H1	$XL5_{out}^{2}$; $8D^{6}$; $16B^{14}$	I/O	IOL
X1D37	H2	$XL5_{in}^{2}$; $1N^{0}$; $8D^{1}$; $16B^{9}$	I/O	IOL
X1D02	H3	$XL5_{out}^{3}$; $4A^{0}$; $8A^{0}$; $16A^{0}$; $32A^{20}$	I/O	IOL
X1D03	H4	$XL5_{out}^{4}$; $4A^{1}$; $8A^{1}$; $16A^{1}$; $32A^{21}$	I/O	IOL
VDD	H5	Digital tile power	PWR	
VSS	H7	Digital ground	GND	
VSS	H8	Digital ground	GND	
VSS	H9	Digital ground	GND	
VSS	H10	Digital ground	GND	
VSS	H11	Digital ground	GND	
VDD	H13	Digital tile power	PWR	
X1D64	H14	$XL2_{in}^{1}$; $32A^{13}$	I/O	IOR
X1D65	H15	$XL2_{in}^{0}$; $32A^{14}$	I/O	IOR
X1D63	H16	$XL2_{in}^{2}$; $32A^{12}$	I/O	IOR

continues on next page

Table 4 – continued from previous page

Signal	Pin	Function	Type	Properties
X1D68	H17	$XL2_{out}^{2}$; $32A^{17}$	I/O	IOR
X1D04	J1	$XL6_{in}^{4}$; $4B^{0}$; $8A^{2}$; $16A^{2}$; $32A^{22}$	I/O	IOL
X1D05	J2	$XL6_{in}^{3}$; $4B^{1}$; $8A^{3}$; $16A^{3}$; $32A^{23}$	I/O	IOL
VDDIOL	J3	Digital I/O power (left)	PWR	
VSS	J4	Digital ground	GND	
VDD	J5	Digital tile power	PWR	
VSS	J7	Digital ground	GND	
VSS	J8	Digital ground	GND	
VSS	J9	Digital ground	GND	
VSS	J10	Digital ground	GND	
VSS	J11	Digital ground	GND	
VDD	J13	Digital tile power	PWR	
VSS	J14	Digital ground	GND	
VDDIOR	J15	Digital I/O power (right)	PWR	
X1D61	J16	$XL2_{in}^{4}$; $32A^{10}$	I/O	IOR
X1D62	J17	$XL2_{in}^{3}$; $32A^{11}$	I/O	IOR
X1D09	K1	$XL6_{out}^{2}$; $4A^{3}$; $8A^{7}$; $16A^{7}$; $32A^{27}$	I/O	IOL
X1D06	K2	$XL6_{in}^{2}$; $4B^{2}$; $8A^{4}$; $16A^{4}$; $32A^{24}$	I/O	IOL
X1D08	K3	$XL6_{in}^{0}$; $4A^{2}$; $8A^{6}$; $16A^{6}$; $32A^{26}$	I/O	IOL
X1D07	K4	$XL6_{in}^{1}$; $4B^{3}$; $8A^{5}$; $16A^{5}$; $32A^{25}$	I/O	IOL
VDD	K5	Digital tile power	PWR	
VSS	K7	Digital ground	GND	
VSS	K8	Digital ground	GND	
VSS	K9	Digital ground	GND	
VSS	K10	Digital ground	GND	
VSS	K11	Digital ground	GND	
VDD	K13	Digital tile power	PWR	
VSS	K14	Digital ground	GND	
VSS	K15	Digital ground	GND	
X1D57	K16	$XL1_{out}^{3}$; $32A^{8}$	I/O	IOR
X1D58	K17	$XL1_{out}^{4}$; $32A^{9}$	I/O	IOR
X1D11	L1	$XL6_{out}^{4}$; $1D^{0}$	I/O	IOL
X1D10	L2	$XL6_{out}^{3}$; $1C^{0}$	I/O	IOL
X1D01	L3	$XL6_{out}^{1}$; $1B^{0}$	I/O	IOL
X1D00	L4	$XL6_{out}^{0}$; $1A^{0}$	I/O	IOL
VSS	L5	Digital ground	GND	
VSS	L7	Digital ground	GND	
VSS	L8	Digital ground	GND	
VSS	L9	Digital ground	GND	
VSS	L10	Digital ground	GND	
VSS	L11	Digital ground	GND	
VSS	L13	Digital ground	GND	
X1D54	L14	$XL1_{out}^{0}$; $32A^{5}$	I/O	IOR

continues on next page

Table 4 – continued from previous page

Signal	Pin	Function	Type	Properties
X1D55	L15	$XL1_{out}^{1}$; $32A^{6}$	I/O	IOR
X1D51	L16	$XL1_{in}^{2}$; $32A^{2}$	I/O	IOR
X1D56	L17	$XL1_{out}^{2}$; $32A^{7}$	I/O	IOR
MIPI_VDD18	M1	MIPI Analog power	PWR	
MIPI_GND09	M2	MIPI Analog ground	GND	
VSS	M3	Digital ground	GND	
VSS	M4	Digital ground	GND	
VDDIOL	M5	Digital I/O power (left)	PWR	
VDDIOR	M13	Digital I/O power (right)	PWR	
X1D52	M14	$XL1_{in}^{1}$; $32A^{3}$	I/O	IOR
X1D53	M15	$XL1_{in}^{0}$; $32A^{4}$	I/O	IOR
X1D49	M16	$XL1_{in}^{4}$; $32A^{0}$	I/O	IOR
X1D50	M17	$XL1_{in}^{3}$; $32A^{1}$	I/O	IOR
MIPI_DN2	N1	MIPI lane 2, negative	Input	
MIPI_DP2	N2	MIPI lane 2, positive	Input	
MIPI_VDD09	N3	MIPI Analog power	PWR	
VSS	N4	Digital ground	GND	
VSS	N5	Digital ground	GND	
VDD	N6	Digital tile power	PWR	
VDDIOB18	N7	Digital I/O power (bottom)	PWR	
VDD	N8	Digital tile power	PWR	
VDD	N9	Digital tile power	PWR	
VDD	N10	Digital tile power	PWR	
VDDIOB18	N11	Digital I/O power (bottom)	PWR	
VDD	N12	Digital tile power	PWR	
VSS	N13	Digital ground	GND	
X1D18	N14	$XL0_{out}^{0}$; $4D^{2}$; $8B^{4}$; $16A^{12}$	I/O	IOR
X1D19	N15	$XL0_{out}^{1}$; $4D^{3}$; $8B^{5}$; $16A^{13}$	I/O	IOR
X1D21	N16	$XL0_{out}^{3}$; $4C^{3}$; $8B^{7}$; $16A^{15}$; $32A^{31}$	I/O	IOR
X1D22	N17	$XL0_{out}^{4}$; $1G^{0}$	I/O	IOR
MIPI_DN1	P1	MIPI lane 1, negative	Input	
MIPI_DP1	P2	MIPI lane 1, positive	Input	
VSS	P3	Digital ground	GND	
LV_L_N	P5	Select low voltage VDDIOL, active low	Input	PU IOB
LV_T_N	P6	Select low voltage VDDIOT, active low	Input	PU IOB
X0D12	P7	$XL7_{in}^{4}$; $1E^{0}$	I/O	IOB
X0D22	P8	$XL7_{in}^{2}$; $1G^{0}$	I/O	IOB
VDDIOB18	P9	Digital I/O power (bottom)	PWR	
OTP_VCC	P10	OTP power supply	PWR	
VSS	P12	Digital ground	GND	
VSS	P13	Digital ground	GND	
X1D16	P14	$XL0_{in}^{1}$; $4D^{0}$; $8B^{2}$; $16A^{10}$	I/O	IOR
X1D17	P15	$XL0_{in}^{0}$; $4D^{1}$; $8B^{3}$; $16A^{11}$	I/O	IOR

continues on next page

Table 4 – continued from previous page

Signal	Pin	Function	Type	Properties
X1D15	P16	$XL0_{in}^{2}$; $4C^{1}$; $8B^{1}$; $16A^{9}$; $32A^{29}$	I/O	IOR
X1D20	P17	$XL0_{out}^{2}$; $4C^{2}$; $8B^{6}$; $16A^{14}$; $32A^{30}$	I/O	IOR
MIPI_DN0	R1	MIPI lane 0, negative	Input	
MIPI_DP0	R2	MIPI lane 0, positive	Input	
VSS	R3	Digital ground	GND	
RST_N	R4	Global reset input, active low	Input	ST PU IOB
TRST_N	R5	Test reset input, active low	Input	ST PU IOB
DEBUG_N	R6	Multi-chip debug, active low	I/O	PU IOB
X0D13	R7	$XL7_{in}^{3}$; $1F^{0}$	I/O	IOB
X0D23	R8	$XL7_{in}^{1}$; $1H^{0}$	I/O	IOB
VSS	R9	Digital ground	GND	
LV_R_N	R10	Select low voltage VDDIOR, active low	Input	PU IOB
VSS	R13	Digital ground	GND	
VSS	R14	Digital ground	GND	
VSS	R15	Digital ground	GND	
X1D13	R16	$XL0_{in}^{4}$; $1F^{0}$	I/O	IOR
X1D14	R17	$XL0_{in}^{3}$; $4C^{0}$; $8B^{0}$; $16A^{8}$; $32A^{28}$	I/O	IOR
VSS	T1	Digital ground	GND	
POR_DISABLE	T2	Disable on chip Power-On-Reset	Input	PD IOB
VDDIOB18	T3	Digital I/O power (bottom)	PWR	
TDO	T4	Test data output	Output	IOB
PLL_AGND	T5	Analog ground for PLL	GND	
PLL_AGND2	T6	Analog ground for secondary PLL	GND	
TMS	T7	Test mode select	Input	PU IOB
X0D02	T8	$XL7_{in}^{0}$; $4A^{0}$; $8A^{0}$; $16A^{0}$; $32A^{20}$	I/O	IOB
X0D08	T9	$XL7_{out}^{1}$; $4A^{2}$; $8A^{6}$; $16A^{6}$; $32A^{26}$	I/O	IOB
X1D12	T10	$XL7_{out}^{3}$; $1E^{0}$	I/O	IOB
VSS	T12	Digital ground	GND	
USB_ID	T13	USB Identification	Input	
USB_VDD33	T14	USB Analog power	PWR	
USB_GND18	T15	USB Analog ground	GND	
VSS	T16	Digital ground	GND	
VDDIOR	T17	Digital I/O power (right)	PWR	
XOUT	U1	Crystal out	Output	IOB
XIN	U2	Crystal in or clock input	Input	IOB
VSS	U3	Digital ground	GND	
TDI	U4	Test data input	Input	PU IOB
PLL_AVDD	U5	Analog power for PLL	PWR	
PLL_AVDD2	U6	Analog power for secondary PLL	PWR	
TCK	U7	Test clock	Input	PD ST IOB
X0D03	U8	$XL7_{out}^{0}$; $4A^{1}$; $8A^{1}$; $16A^{1}$; $32A^{21}$	I/O	IOB
X0D09	U9	$XL7_{out}^{2}$; $4A^{3}$; $8A^{7}$; $16A^{7}$; $32A^{27}$	I/O	IOB
X1D23	U10	$XL7_{out}^{4}$; $1H^{0}$	I/O	IOB

continues on next page

Table 4 – continued from previous page

Signal	Pin	Function	Type	Properties
VSS	U11	Digital ground	GND	
VSS	U12	Digital ground	GND	
USB_DM	U13	USB Data-	I/O	
USB_DP	U14	USB Data+	I/O	
USB_VDD18	U15	USB Analog power	PWR	
VSS	U16	Digital ground	GND	
VSS	U17	Digital ground	GND	

F Resource Configuration

This section documents how many of each resources are present, and how the SETC instruction is used to configure the resource. For all other information on resources, please refer to the XS3 ISA specification.

The SETC operand is a number with the following bit fields that have been organised so that frequently used modes can be encoded in an immediate 6-bit operand.

- [31..16] Reserved
- [15..12] Long mode setting
- [11..3] Value
- [2..0] Mode setting, set to 0x7 to denote a long mode.

The meaning of the bits is resource dependent.

F.1 Port Resources

There are:

32 x 1-bit port, 12 x 4-bit port, 8 x 8-bit port, 4 x 16-bit port, 2 x 32-bit port

The following controls can be set using SETC:

▶ INUSE_OFF ▶ INUSE_ON	Mode bits 0x0000. Switches the port resource on (value 1) and off (value 0). Before using a port it must be switched on.
▶ COND_NONE ▶ COND_EQ ▶ COND_NEQ	Mode bits 0x0001. Sets the port condition. Value 1 sets up a test for equal, and value 2 sets up a test for not equal. An input of a port with a condition will only succeed when the condition matches. SETD is used to set the test operand.
▶ IE_MODE_EVENT ▶ IE_MODE_INTERRUPT	Mode bits 0x0002. Sets the resource to generate events (value 0) or interrupts (value 1). By default it generates events.
▶ DRIVE_DRIVE ▶ DRIVE_PULL_DOWN ▶ DRIVE_PULL_UP	Mode bits 0x0003. Sets the drive mode of the port. Value 1 sets the drive transistor to just drive the high side and enable a weak pull-down, Value 2 sets the drive transistors to just drive the low side and enable a weak pull-up
▶ MODE_SETPADCTRL	Mode bits 0x0006. Sets the pad options according to the value of bits 23..18. Bits 19 and 18 set the pull resistor (00 for none; 01 for weak pull-up; 10 for weak pull-down; or 11 for weak bus-keep.). Bits 21 and 20 set the drive strength (00 for 2mA; 01 for 4mA; 10 for 8mA; or 11 for 12mA). Bit 22 enables slew-rate control. Bit 23 enables the Schmitt-Trigger.
▶ RUN_CLRBUF	Mode bits 0x0007, value 2: clears the port buffer
▶ MS_MASTER ▶ MS_SLAVE	Mode bits 0x1007. Sets the port to master mode (value 0) or slave mode (value 1).
▶ BUF_NOBUFFERS ▶ BUF_BUFFERS	Mode bits 0x2007. Sets the port to be buffered (value 1) or unbuffered (value 0). Unbuffered is the default.
▶ RDY_NOREADY ▶ RDY_STROBED ▶ RDY_HANDSHAKE	Mode bits 0x3007. Sets the port to use data strobes (value 1) or full handshaking (value 2). Default is no ready wires.
▶ SDELAY_NOSDELAY ▶ SDELAY_SDELAY	Mode bits 0x4007. Sets the port to optionally capture data on the falling edge (value 1)
▶ PORT_DATAPORT ▶ PORT_CLOCKPORT ▶ PORT_READYPORT	Mode bits 0x5007. Sets the port to be a clock (value 1) or ready signal (value 2). By default the port is a data port. This can only be applied to 1-bit ports.
▶ INV_NOINVERT ▶ INV_INVERT	Mode bits 0x6007. Sets the port to optionally invert the signal (value 1).
▶ PAD_DELAY	Mode bits 0x7007, value must be in the range 0..4. Delays the input signals by a set number of core clock ticks. Defaults to 0.

F.2 Timer Resources

There are 20 timers (10 per tile) timers. The following controls can be set using SETC:

▶ COND_NONE ▶ COND_AFTER	Mode bits 0x0001. Sets the timer to have to only be ready after the given time (value 1). Set the time for comparison using SETD
▶ IE_MODE_EVENT ▶ IE_MODE_INTERRUPT	Mode bits 0x0002. Sets the resource to generate events (value 0) or interrupts (value 1). By default it generates events.

F.3 Channel-end Resources

There are 64 channel ends (32 per tile). The following controls can be set using SETC:

▶ IE_MODE_EVENT ▶ IE_MODE_INTERRUPT	Mode bits 0x0002. Sets the resource to generate events (value 0) or interrupts (value 1). By default it generates events.

F.4 Synchronizer Resources

There are 14 sychronizers (7 per tile). They cannot be configured using SETC.

F.5 Thread Resources

There are 16 threads (8 per tile). They cannot be configured using SETC.

F.6 Lock Resources

There are 8 locks (4 per tile). They cannot be configured using SETC.

F.7 Clock Block Resources

There are 12 clock blocks (6 per tile). The following controls can be set using SETC:

▶ INUSE_OFF ▶ INUSE_ON	Mode bits 0x0000. Switches the clock block on (value 1) and off (value 0). Before using a port it must be switched on.
▶ RUN_STOPR ▶ RUN_STARTR	Mode bits 0x0007. Starts the clock running (value 1). Once it is running, the clock block cannot be reconfigured.
▶ FALL_DELAY	Mode bits 0x8007, value 0..511. Delays the falling edge of the clock block by this many core clock cycles. The clock block cannot delay beyond the rising input clock edge.
▶ RISE_DELAY	Mode bits 0x9007, value 0..511. Delays the rising edge of the clock block by this many core clock cycles. The clock block cannot delay beyond the falling input clock edge.

F.8 Software-defined Memory Resources

There are two software-defined memory resources in each tile: the read miss resource and the write miss resource. The following controls can be set using SETC:

▶ INUSE_OFF ▶ INUSE_ON	Mode bits 0x0000. Switches the software memory on (value 1) or off (value 0). When on, the software memory address space will be routed to the mini-cache, and misses will cause an event/interrupt on this resource.
▶ IE_MODE_EVENT ▶ IE_MODE_INTERRUPT	Mode bits 0x0002. Sets the resource to generate events (value 0) or interrupts (value 1). By default it generates events.
▶ RUN_STARTR	Mode bits 0x0007, data 1. This operation signals to the hardware that the software memory miss has been serviced by software.

G JTAG, xSCOPE and Debugging

If you intend to design a board that can be used with the XMOS toolchain and xTAG4 debugger, you will need an xSYS2 connection on your board. There are three physical xSYS2 connections that XMOS uses:

- In its smallest form you can put 6 testpoints and three through-holes on the PCB and use a *TAG-connect* cable to connect to an XTAG4.
- You can use a half-sized header (approximately 7 mm wide) that supports just JTAG, which is cabled to an XTAG4.
- You can use a full sized header (approximately 13 mm wide) that supports both JTAG and XSCOPE, again cabled to an XTAG4.

Note that the xSYS2 header has a different form-factor than the xSYS header used on older devices. This is because the signal levels are different (1.8V rather than 3.3V). Only use 1.8V XTAG adapters to program this device. It needs to be an XTAG4 or above.

Fig. 26 shows a decision diagram which explains what type of xSYS2 connectivity you need. The three subsections below explain the options in detail.

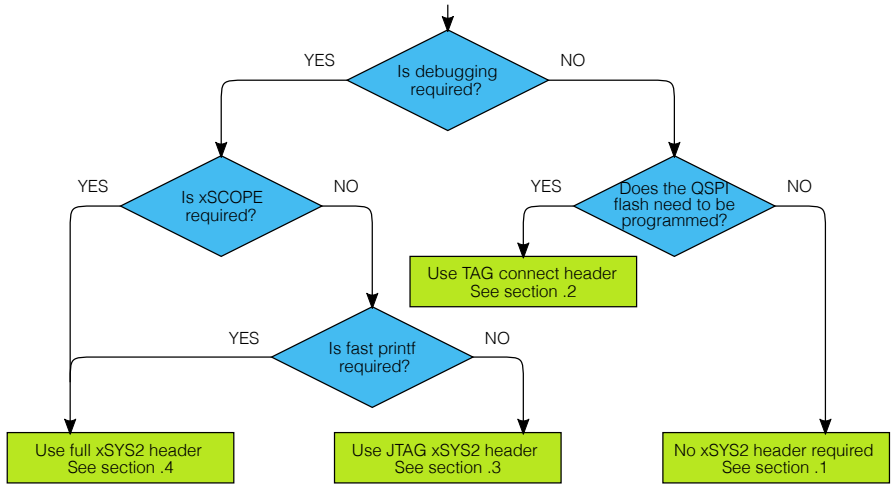


Fig. 26: Decision diagram for the xSYS2 header

G.1 No xSYS2 Connection

The use of an xSYS2 connection is optional, and may not be required for volume production designs. However, the XMOS toolchain expects the xSYS2 connection; if you do not have an xSYS2 connection, then you must provide your own method for writing to flash/OTP and for debugging.

G.2 JTAG-only TAG-connect Header

This header requires six test-points on the PCB with three through holes for registration, see Fig. 27. These connect to a TC2030-IDC cable from Tag-Connect, which in turn is plugged into an XTAG4. For details on the foot-print and on the cable see https://www.tag-connect.com. Use the following pin-out:

Fig. 27: Foot print for tag-connect header

- pin 1: TCK
- pin 2: GND
- pin 3: TMS
- pin 4: TDI
- pin 5: VREF
- pin 6: TDO

G.3 JTAG-only xSYS2 Header

Connect the following pins of the 0.05” header:

- pins 3, 5, 7, and 9 to GROUND
- pin 1 to VDDIOB18 (with a decoupler)
- pin 2 to TMS
- pin 4 to TCK
- pin 6 to TDO
- pin 8 to TDI
- pin 10 to RST_N

The pin-out of this header is shown in the blue section of Fig. 28.

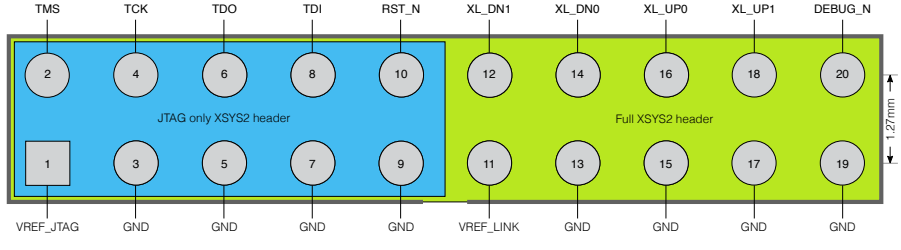


Fig. 28: xSYS2 header pin-out, as seen from above

G.4 Full xSYS2 Header

For a full xSYS2 header you will need to connect the pins as discussed in *JTAG-only xSYS2 Header*, and then connect a 2-wire xCONNECT Link to the xSYS2 header. The pin-out of this header is shown in Fig. 28.

The links can be found in the Signal description table (*Signal Description and GPIO*): they are labelled XL0, XL1, etc in the function column. The 2-wire link comprises two inputs and outputs. For example, if you choose to use XL0 for xSCOPE I/O, you need to connect up $XL0_{out}{}^{1}$, $XL0_{out}{}^{0}$, $XL0_{in}{}^{0}$, $XL0_{in}{}^{1}$ as follows:

- $XL0_{out}{}^{1}$ (X1D19) to pin 18 of the xSYS2 header with a 43R series resistor close to the device.
- $XL0_{out}{}^{0}$ (X1D18) to pin 16 of the xSYS2 header with a 43R series resistor close to the device.
- $XL0_{in}{}^{0}$ (X1D17) to pin 14 of the xSYS2 header.
- $XL0_{in}{}^{1}$ (X1D16) to pin 12 of the xSYS2 header.
- Connect pin 11 to the VDDIO that is used to power the link, with a decoupler. In this case, that will be VDDIOR, as that is the IO supply for X1D16..X1D19.

For links 0..3 you will need to connect pin 11 to VDDIOR, for links 4..6 connect it to VDDIOL, and for link 7 use VDDIOB18.

H Schematics Design Checklist

[X]

This section is a checklist for use by schematics designers using the XU316-1024-FB265. Each of the following sections contains items to check for each design.

H.1 Power supplies

[ ]

The VDD (core) supply is capable of supplying the required current, see *Integration* and *Operating Conditions*.

[ ]

PLL_AVDD is filtered with a low pass filter, for example an RC filter, see *Integration*.

[ ]

PLL_AVDD2 is filtered with a low pass filter, for example an RC filter, see *Integration*.

[ ]

If any of the VDDIOL, VDDIOT, or VDDIOR domains are at 1V8, then then the corresponding LV_L_N, LV_T_N, or LV_R_N pin has been strapped to GROUND (*Integration*).

[ ]

If any of the VDDIOL, VDDIOT, or VDDIOR domains are at 3V3, then power supplies have been sequenced as specified in *Integration*.

H.2 Decoupling of the power supplies

[ ]

The design has multiple decoupling capacitors per supply, as specified in *Integration*.

[ ]

A bulk decoupling capacitor of at least 10uF is placed on each supply (*Integration*).

H.3 Power on reset

[ ]

At least one of these two conditions is true:

1. All VDDIO pins are supplied by the same 1.8V supply (the on-chip power-on-reset will operate correctly); or
2. RST_N is kept low until all VDDIO are valid, and RST_N is fast enough to meet USB timings.

See *Integration*.

H.4 Clocking

[ ]

If you put a crystal between XIN/XOUT you followed the guidelines in *Oscillator Circuit*.

[ ]

If you supply a clock directly onto XIN, then it is 1.8V, low jitter, and has monotonic edges.

[ ]

You have chosen an input clock frequency that is supported by the device (*Oscillator, Clocks, and PLLs*).

[ ]

If you use USB, then your clock frequency is one of 12 or 24 MHz (*Oscillator, Clocks, and PLLs*).

H.5 Boot

[ ]

The device is connected to a QSPI flash for booting, connected to X0D01, X0D04..X0D07, and X0D10 (*Boot Procedure*). If not, you must boot the device through OTP or JTAG, or set it to boot from SPI and connect an SPI flash.

[ ]

The Flash that you have chosen is supported by the tools.

H.6 JTAG, XScope, and Debugging

[ ]

You have decided as to whether you need an xSYS2 header or not (*JTAG, xSCOPE and Debugging*)

[ ]

If you included an xSYS2 header, you are using the smaller 0.05" header (*JTAG, xSCOPE and Debugging*)

[ ]

If you have not included an xSYS2 header, you have devised a method to program the SPI-flash or OTP (*JTAG, xSCOPE and Debugging*).

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 high and low appropriately (*Boot Procedure*)

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 xlink have, for example, X0D06 pulled high and have link XL0 connected to a device to boot from (*Boot Procedure*).

I PCB Layout Design Checklist

[X]

This section is a checklist for use by PCB designers using the XU316-1024-FB265. 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. (*Ground and Thermal Vias*)

I.2 Power Supply Decoupling

[ ]

The decoupling capacitors are all placed close to a supply pin (*Integration*).

[ ]

The decoupling capacitors are spaced around the device (*Integration*).

[ ]

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

I.3 PLL_AVDD

[ ]

The PLL_AVDD filter (especially the capacitor) is placed close to the PLL_AVDD pin (*Integration*).

[ ]

The PLL_AVDD2 filter (especially the capacitor) is placed close to the PLL_AVDD2 pin (*Integration*).

J Associated Design Documentation

- AN02023: xcore.ai Power Consumption Estimation.
- XTC Tools Documentation: compilers, assembler and linker/mapper, xScope, debugger, Flash and OTP programming utilities.

K Related Documentation

- The XMOS XS3 Architecture: ISA manual.
- xcore.ai I/O timings: port timings.
- AN02021 : Using external memory on xcore.ai
- xCONNECT Architecture: link, switch, and system information.
- XS1-L Link Performance/Design Guidelines: link timings.
- AN02022: xcore.ai Clock Frequency Control: advanced clock control.

L Revision History

Date	Description
2025-01-13	2.0.0: Semantic versioning and HTML rendering
2024-12-02	Fixed documentation links; fixed link to USB clock frequency
2024-11-26	Fixed error in MIPI connectivity (TQ128, FB265 only)
2024-11-11	Fixed error in XTAG2 connectivity
2024-11-10	New layout tools (includes HTML)
2024-09-05	Added 800 MHz variants
2024-09-05	Release of combined FB265/TQ128/QF60B/QF60A datasheet
2024-09-05	Minor changes to electrical characteristics, new layout
2023-08-29	Release of Industrial QF60B
2022-09-28	Added statement on power sequencing for 3v3 VDDIO
2022-08-15	Industrial grade TQ128 and FB265 introduced, updated max power
2022-01-20	Fixes to links
2022-01-19	TQ128 datasheet introduced
2021-07-14	QF60B datasheet introduced
2021-06-23	Characterisation data completed for all parts
2020-10-10	Fixed boot table
2020-08-05	Preliminary release

XMOS