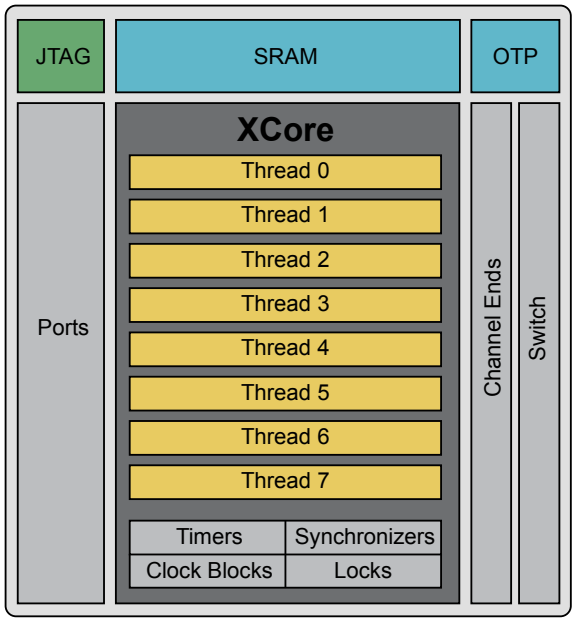
XMOS Architecture and XS1 Devices

XMOS has developed the industry's first 32-bit user-defined embedded processors. These powerful, deterministic processors significantly lower development time and system bill of materials cost. The XCore® event-driven processor combines the code efficiency of a RISC processor, the computational performance of a DSP and the unique flexibility of implementing all peripherals through user-defined "C" software.

- **Multi-threaded processor with OS features built into hardware**

An XS1 core executes up to eight concurrent threads. Each thread has its own registers in hardware. Each core has instruction set support for thread scheduling, communication, input-output and timing; this makes the cost of these operations the same as arithmetic, memory operations and branches. Each XS1 core has a single unified memory system, which is shared for program and data by all of the threads executed by the core. This means that multiple threads can share the same program in memory. Memory access is single cycle, giving high performance without caches.



Communication between threads is performed using hardware channels.

The processor executes one instruction per clock cycle, giving up to 500 MIPS performance per core. If four or less threads are active, each thread gets a quarter of the available processing MIPS. If more than four threads are active, the processing power is divided equally between the active threads; for example, if eight threads are active, each gets 62.5 MIPS and executes an instruction every 16ns.

- **Peripherals implemented in high-level 'C' instead of silicon gates**

The XCore has a tightly integrated set of input-output ports that provide connections to physical pins and are controlled directly by instructions. Data is transferred directly between processor registers and ports, avoiding the use of memory and minimizing latency. Ports can serialize and deserialize data enabling the processor to handle high-speed data streams. Ports can also timestamp data arrival and accurately control the time at which data is transferred to or from the pins. IP can be downloaded free of charge at xmos.com.

The software running on the threads implements peripherals. This allows the designer to chose the exact mix of interfaces required in the application, for example, SPI, I2C, I2S, S/PDIF, ULPI for high speed USB and MII for Ethernet.

- **Tuned for deterministic real-time performance**

The instruction pipeline has four instructions in flight at any one time. These instructions are from four different threads making them completely independent and avoiding the need for any forwarding. Thread execution is deterministic and the time taken to execute a sequence of instructions can be predicted at compile time, making it possible for software executing on an XCore to perform many functions usually done by hardware, especially DSP and input-output.

XMOS

- **Efficient event-driven processing without interrupt overhead**

On each processor clock cycle, the XCore thread scheduler issues an instruction from the next thread that is not waiting. In its simplest form, a thread waits if it attempts to input data before the data is available; a thread can also wait for one of a set of events. An event can come from a port, channel or timer. When an event occurs, it signifies that a thread needs to service the port/channel/timer.

Device	XS1-L1	XS1-L2	XS1-G4
XCores	1	2	4
Threads	8	16	32
MIPS	400/500	800/1000	1600
SRAM	64KB per core		
OTP	8KB per core		
I/O	3v3	3v3	3v3 (5v tolerant)
Power	15-200mW	30-400mW	200-1200mW
Packages (I/O)	QFP48 (24) QFP64 (36) QFP128(64)	QFN124(84)	BGA144 (88) BGA512 (256)

If a thread is waiting for one or more events and an event occurs, the thread immediately resumes at the event target with all registers and memory in a known state. This combination of threads and event-driven processing largely eliminates the need for interrupts. A thread can wait for one or more events that would have interrupted a traditional processor, and deal with them without having to save or restore context.

- **Energy-efficiency designed into the architecture**

XCores and threads do not need to consume energy when waiting for events. The compact instruction encoding minimizes power to access instructions. Direct transfer of data between thread registers and channels or ports avoids transferring data via memory. Data is transferred via the links using a transition-based encoding to minimize energy consumption and the links only use energy when they are actually transferring data.

- **Highly efficient network of inter-communicating cores and chips**

Multiple XCores can be connected on chips, substrates, boards and distributed systems. Each XMOS device includes a high-bandwidth low-latency switch, which routes messages between the XCores using communication links. As XCores are added, computational performance increases, memory increases, communication throughput increases and event-handling throughput increases. The same instruction sequence can be used whether the threads are executed by the same XCore or by different XCores.

- **Secure bootloader to protect developer IP**

XMOS devices contain on-chip one-time programmable (OTP) memory that can be used to store encrypted programs. This means that programs can be made secure, authenticated against personal keys and are difficult to reverse engineer.

- **Development tools to go from concept to production**

XMOS development tools are based on a standard embedded software flow that supports C, C++ and XC. XC is an extended version of C which allows developers to access the real time features of the XCore. As well as providing compilers and a debugger, the tools validate that hard real-time constraints are met on target devices.

Tools are available at no charge for Microsoft Windows, Mac OS and Linux platforms.