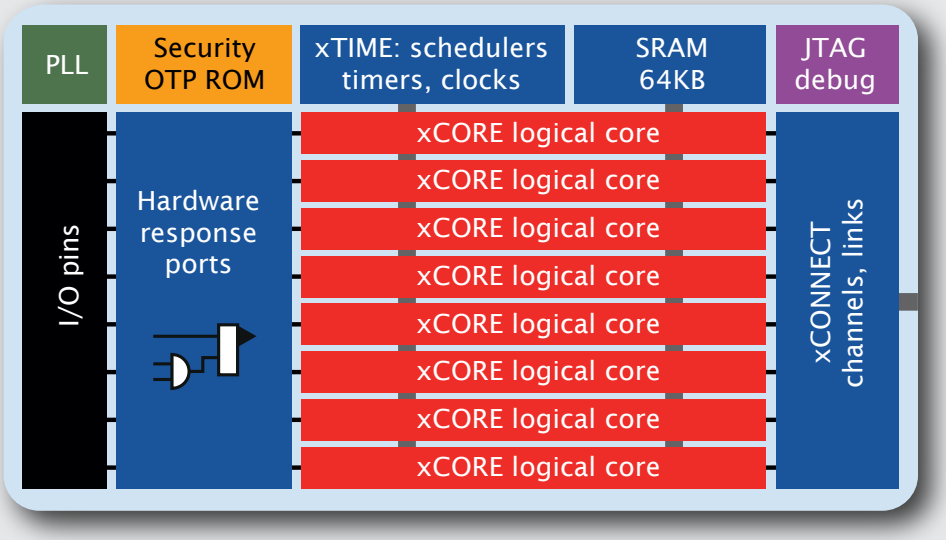
xCORE

xCORE ARCHITECTURE

xCORE multicore microcontrollers are unique. Like traditional MCUs they are programmed in C/C++, but unlike traditional products they have multiple processor cores and can execute several tasks concurrently and independently. They have incredibly flexible I/O and a unique timing-deterministic architecture with an intimate connection between the cores and the outside world. xCORE lets you deliver complex real-time projects using a simple design process that is both flexible and scalable.



A HARDWARE REAL-TIME OPERATING SYSTEM

Today's embedded applications require a deterministic microcontroller architecture that detects and resolves task changes as they occur. The architecture must process multiple concurrent tasks quickly, minimize jitter and handle combinations of fast, complex interfaces. It must provide easy integration of a wide range of components and be scalable.

Traditional real-time systems are interrupt-driven, relying on an RTOS to schedule tasks and handle communication. But they face challenges such as handling complex I/O streams within an acceptable time window, interrupt latency, kernel processing time, jitter and memory contention. They are hard to verify, and the RTOS imposes significant processor and memory overhead.

The xCORE architecture solves these problems by removing all of the features of a traditional MCU that introduce uncertainty. Instructions execute in a single cycle; there are no interrupts; no traditional bus structure; no pipeline; and no cache. Many of the features of an RTOS are integrated in hardware: system events are handled using a scheduled single-cycle context switch, so applications do not suffer interrupt jitter.

With multiple processing cores executing independently, tasks can be guaranteed to complete within a strict timing window. Tasks can share data without using caches. Each core can run I/O, DSP and application code, and activities in one task do not affect other tasks. An intimate connection between processor resource and hardware ports passes I/O events directly to tasks, yielding response times up to 100x faster than traditional microcontrollers.

Because xCORE devices are deterministic, designers can create systems that are predictable, with the exact combination of peripheral interfaces they require. xCORE delivers all the features required by today's embedded developers, with performance unmatched by traditional interrupt-driven systems.

- **Multiple processing cores**
  - Runs multiple tasks concurrently
  - Guaranteed performance
- **RTOS features in hardware**
  - Scheduler, timers, communications
  - Low latency
  - Predictable repeatable behavior
- **Immediate response time**
  - 100x faster than other MCUs
- **Flexible ports and peripherals**
  - Sophisticated port logic
  - Supports fast complex interfaces
  - Multiple peripherals in one chip
- **Integrated development tools**
  - Programmed in C and C++
  - Instrumentation/trace libraries
  - Static timing analyzer
- **Multicore extensions for C**
  - Concurrent tasks, timing, communication, I/O, memory management

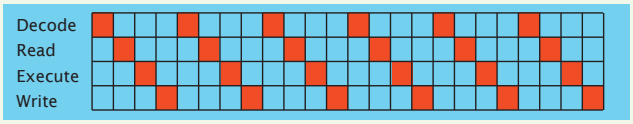
xTIME™ SCHEDULER

Each xCORE device has one or more tiles. Each tile has up to eight independent 32-bit logical cores that run in parallel without interruption from other cores. A tile also includes the xTIME scheduler, the xCONNECT switch, ports and SRAM.

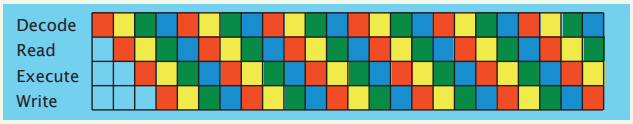
Active cores are guaranteed a minimum level of MIPS. Cores that are idle are not scheduled to the processing resource.

Cores access instructions directly, unlike traditional microcontrollers that use memory mapping. All instructions complete in a single core cycle, or pause the core.

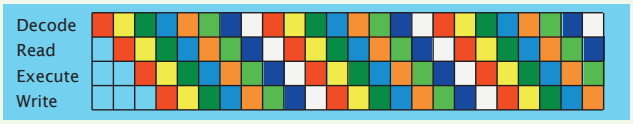
Cores are triggered by events that are managed by the xTIME scheduler. Events that occur at I/O pins are fed directly to a core by the Hardware Response ports. Events can also be generated by timers and tasks, and serviced by the scheduler, with guaranteed behavior.



Single core running: executes every 4 clock ticks (f/4 MHz)



Four cores running: executes every 4 clock ticks (f/4 MHz)



Eight cores running: executes every 8 clock ticks (f/8 MHz)

EVENTS AND INTERRUPTS

Traditional devices take a number of instruction cycles to respond to an interrupt, during which time they store the state associated with the running task and then load the new state associated with the higher priority task that needs to be started.

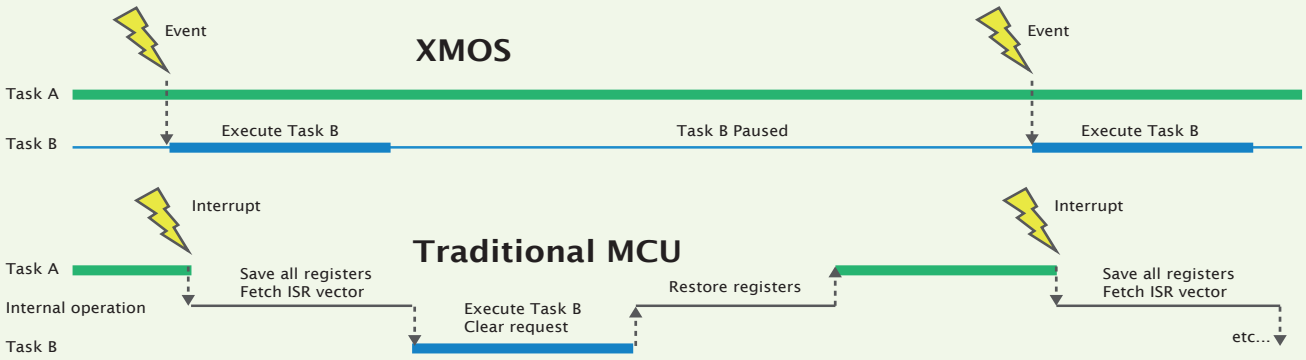
XMOS devices respond to events triggered by I/O pins, timers and tasks, rather than interrupts. Since an xCORE device can run multiple tasks in parallel there is no need for one task to interrupt another. A task can handle an event by running in parallel with other tasks and waiting for the event to happen.

The advantages of the XMOS approach include:

- Response time to events is dramatically improved (in conjunction with the multi-core xCORE architecture).
- Reasoning about worst-case execution time (WCET) is easier since code cannot be interrupted during its execution.

Independent tests show that an xCORE device can respond to single events within 10ns and handle multiple asynchronous real-time events within a worst-case response time of 100ns; this is 100x times faster than conventional real-time systems and critically, scales to larger, more integrated systems.

(http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6389416)



TASK PRIORITY

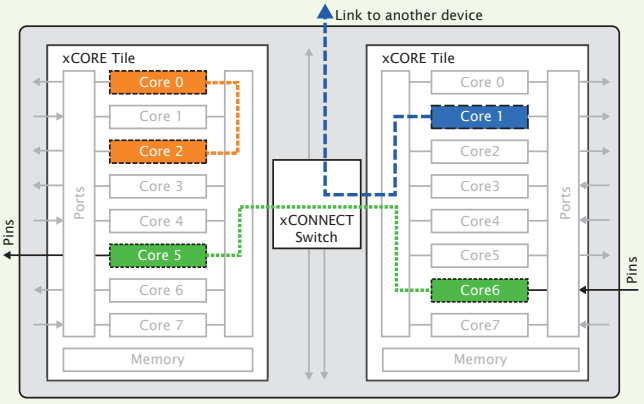
By default, each task in an xCORE application is placed on a different logical core. This means the task runs independently of the others and has incredibly quick response time when it is waiting on events. In RTOS terms, each task running on its own core enjoys the highest priority scheduling by the hardware. To reduce resource requirements, lower priority tasks can share a core via co-operative multitasking. Co-operative multitasking is supported directly in the multicore extensions to C that enable task parallelization on xCORE devices.

TASK COMMUNICATION

The high-speed xCONNECT™ network ensures that all tasks can communicate with each other on the same tile or other tiles in the same device, with very low overhead. This network can also be extended to allow multiple xCORE devices to be connected together so that larger multi-processor real-time systems can be created.

Each task runs in parallel across the logical cores of the xCORE device. The compiler automatically checks the number of cores per tile used, and allocates the required amount of stack and data memory to each task.

Tasks communicate via transactions or software interfaces that define the kind of transactions that can occur between the tasks and the data that is passed with them.

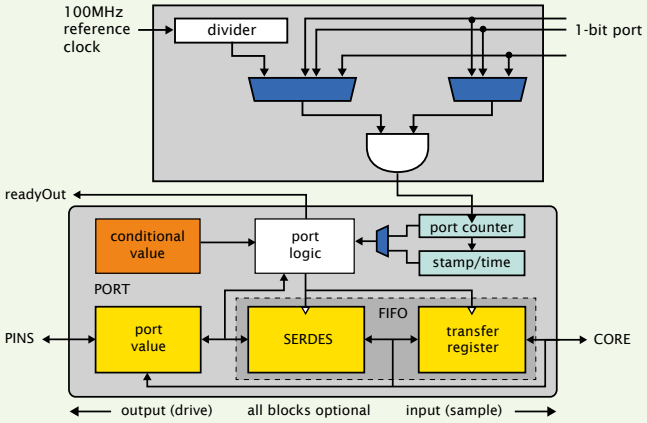


HARDWARE RESPONSE™ PORTS

The GPIO pins of the xCORE device are managed by port logic that can efficiently drive external pins high and low, and sample values.

Ports are available in different widths (1/4/8/16/32bit) depending on the device package. They are driven by clocks or timers, and data can be buffered, serialized and timestamped.

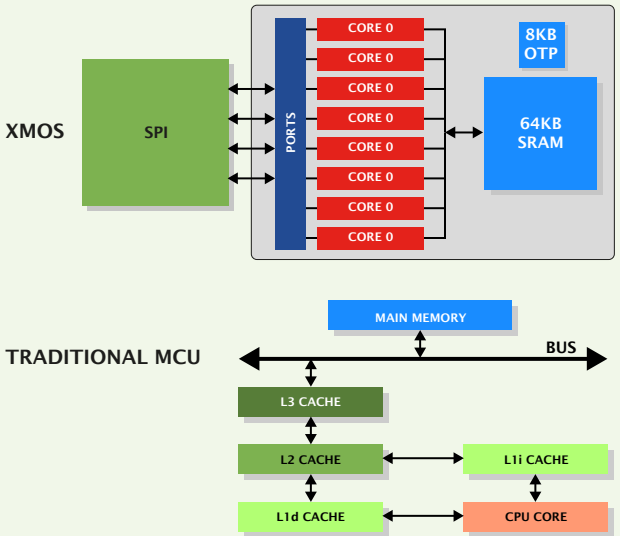
xCORE devices can communicate with fast, complex interfaces that would not be possible using standard bit-banging techniques required by other microcontrollers. As well as running real-time parallel applications, XMOS microcontrollers allow complex I/O protocols and combinations of peripherals to be implemented via the ports within a single device.



MEMORY SYSTEM

Many microcontrollers have a single large memory system with caches, while others split the memory into blocks for running code and storing data. These systems require careful engineering to avoid data trashing or contention. The xCORE memory system is integrated with the xTIME scheduler and is fully deterministic:

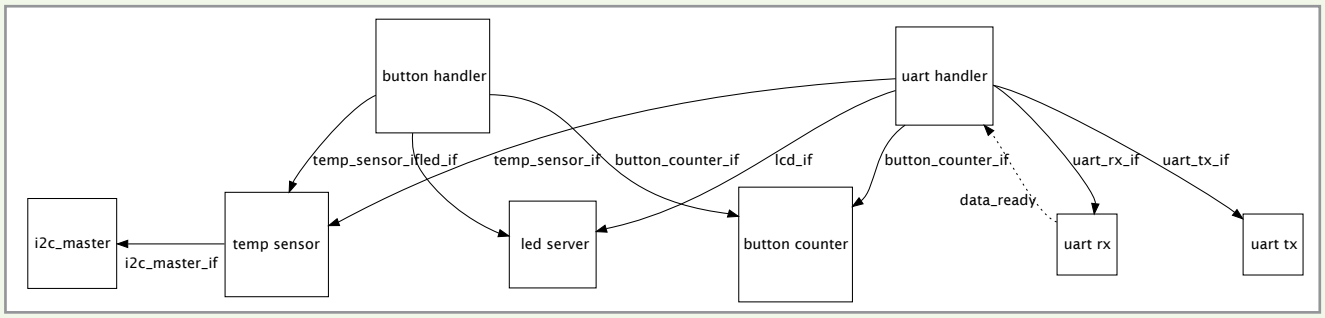
- Each tile contains 64KB local memory, which is shared between all cores on that tile for code and data.
- Each core has a slot to access the memory in a single cycle; there is no requirement for a cache.
- Synchronized tasks can share data structures in local memory, or pass data directly to other tasks.
- Tasks communicate explicitly with external FLASH/SDRAM memory using I/O ports, separate to the local tile memory; no additional memory manager is required.
- Tasks on different tiles communicate via inter-task communication channels over the xCONNECT switch.
- Each tile also has 8KB one-time programmable memory for secure boot code and encryption keys.



PROGRAMMING MULTICORE APPLICATIONS

On xCORE devices, programs are composed of multiple tasks running in parallel. The concurrent tasks manage their own state and resources and interact by performing transactions with each other.

The compiler maps tasks onto the logical cores of the device (under user direction in the code). The ability of the xCORE architecture to run code independently in parallel allows tasks to be very responsive to events that occur in the system.



MULTICORE EXTENSIONS TO C

To help programmers access the real-time hardware features of the architecture, some easy-to-use, yet powerful, multicore language extensions for C have been added. These extensions form a programming language called xC which includes features for:

- Task based parallelism
- Task communication
- Accurate timing and timestamping
- Ports and I/O
- Safe memory management

Software projects can mix C source files with or without the multicore extensions enabled. The xTIMEcomposer compiler automatically enables the extensions based on the file extension.

To help programmers write real-time concurrent applications, XMOS provides numerous modules and blocks of xSOFTip code that implement common tasks and interfaces. xSOFTip components can be quickly configured in the software development tools and integrated into applications.

```
[[combinable]]
void task1(interface my_interface client c)
{
  timer tmr;
  int time;
  int count = 0;
  tmr :> time;
  while (1) {
    select {
    case tmr when timerafter(time) :> int now:
      count++;
      if (count > 5) {
        c.finish();
        return;
      }
      // c is the client end of the connection,
      // let's send a message to the other end.
      c.msgA(count, 10);
      time += 1000;
      break;
    }
  }
}
```

SOFTWARE DEVELOPMENT ENVIRONMENT

The hardware features are complemented by a software development environment, which makes it easy to define real-time tasks as a scalable parallel system. The xTIMEcomposer tools include fully standards-compliant C and C++ compilers plus the standard language libraries, an IDE, simulator, symbolic debugger, runtime instrumentation and trace libraries and a static code timing analyzer (XTA).

All of the components are aware of the real-time multicore nature of the programs, giving a fully integrated approach. As a result the xCORE tools are able to support parallel code descriptions, report on memory and resource usage, let you debug multicore programs and determine the exact timing of your software code.