xCORE: Multicore theory, hardware and programming

MULTICORE MICROCONTROLLERS

Agenda

xS
1. Introduction to xCORE
Architecture and Technology
Products

xCORE

xS

2. Introducing xSOFTip
Software IP and Applications

xSOFTip

xT

3. Introducing xTIMEcomposer
xCORE Development Environment

xTOOLS

xC

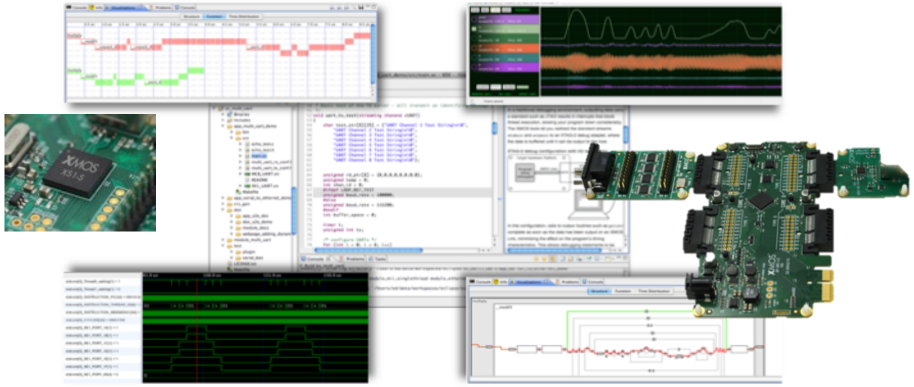
4. Languages and parallelism
Learn high level programming in xC

xC

xK

5. Introducing startKIT
Your startKIT development board

xKITS



Introduction to xCORE

xCORE › xSOFTip › xTOOLS › xC › xKITS

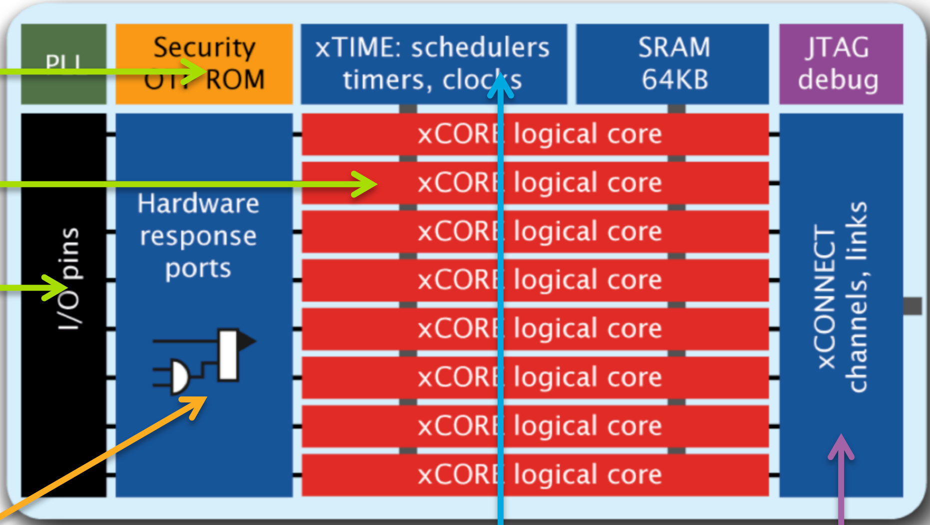
Why Multicore…

- …system functionality is partitioned into discrete tasks
- …each core runs independently, unaffected by events on other cores
- …which makes your software code completely timing-predictable
- …and achieves up to x100 faster response times to external events

xCORE : multicore microcontrollers

xCORE

Up to 8 logical 32bit cores per tile
- 4-16 core devices
- Parallel multi-tasking
- >60MIPS per core
- 64KB RAM per tile
- DSP processing
- Security (OTP)

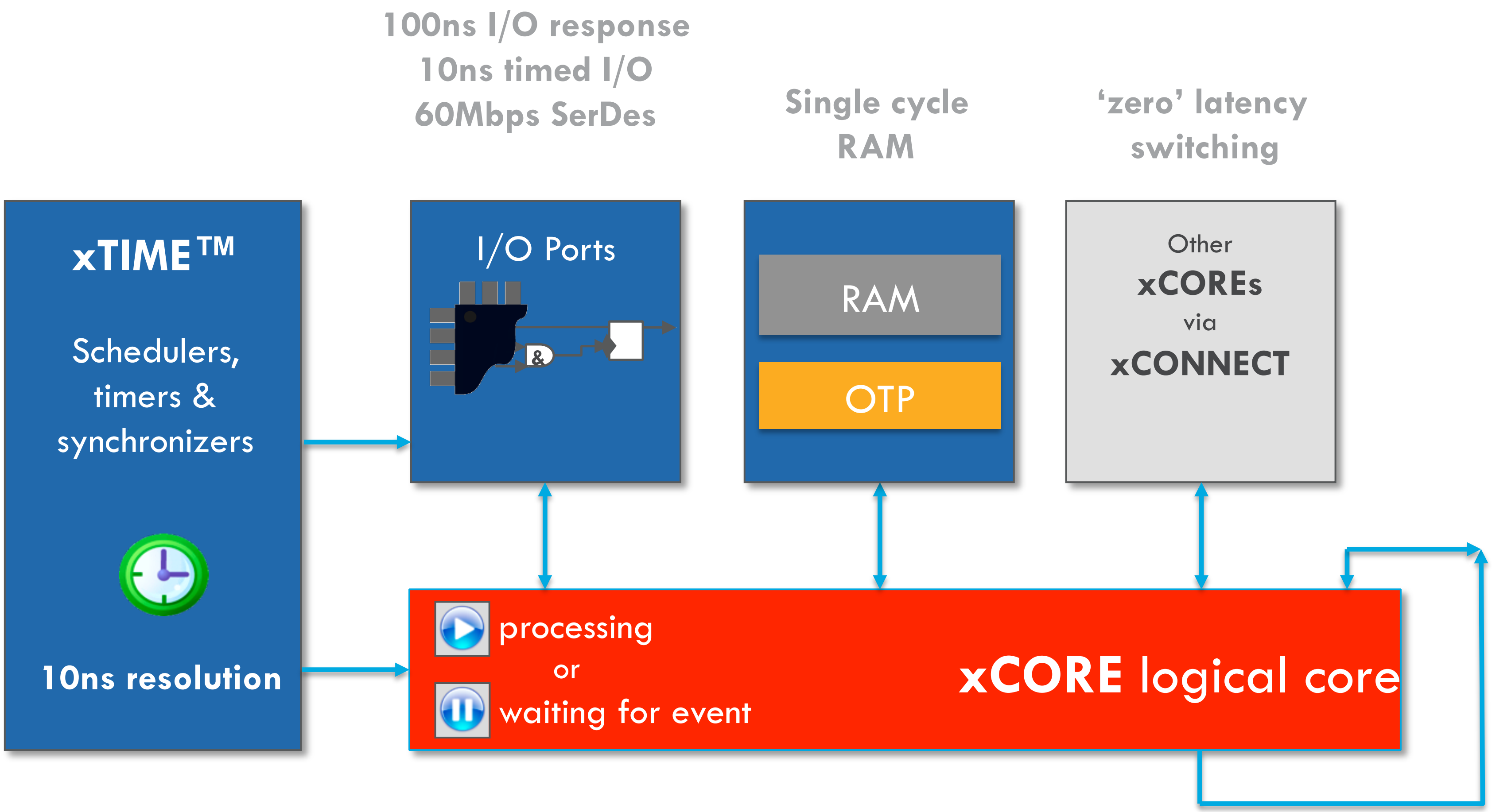


Hardware Response^{TM} I/O
- Guaranteed I/O capture
- 100x faster response
- Real-time logic processing
- Automated serialize/deserialize
- Software defined peripherals

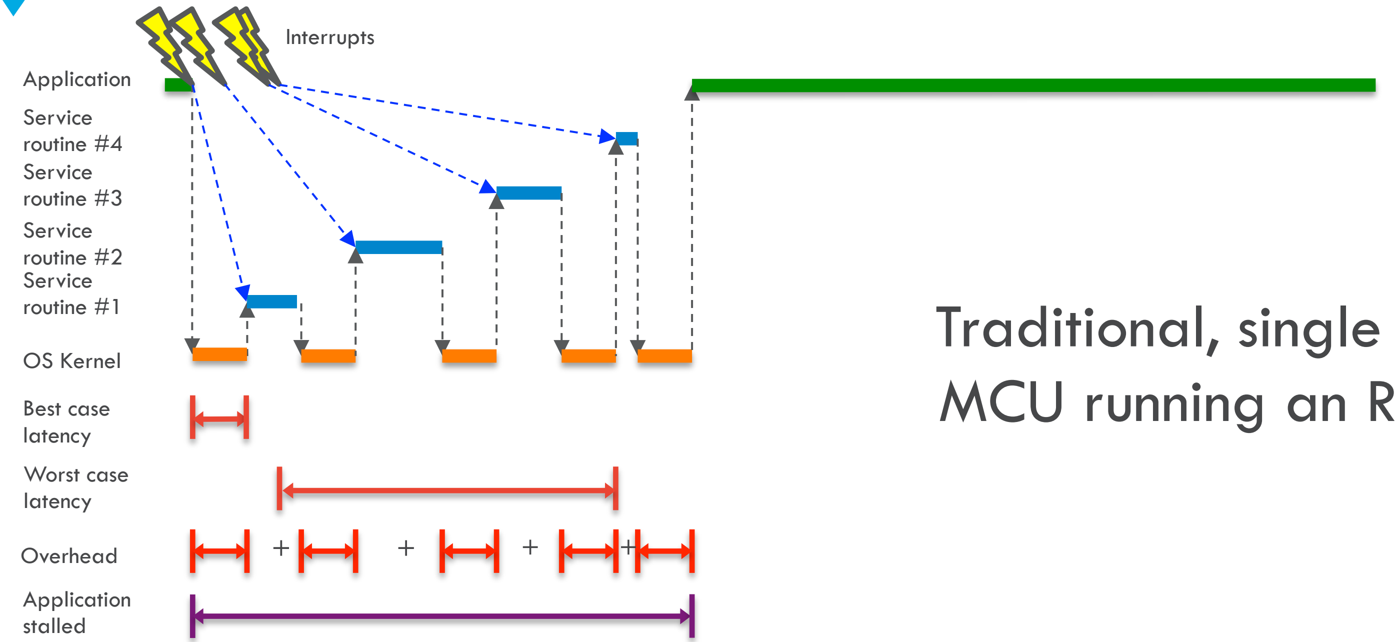
xTIME scheduler
- Ensures 100% timing determinism
- 10ns resolution

Zero latency switch

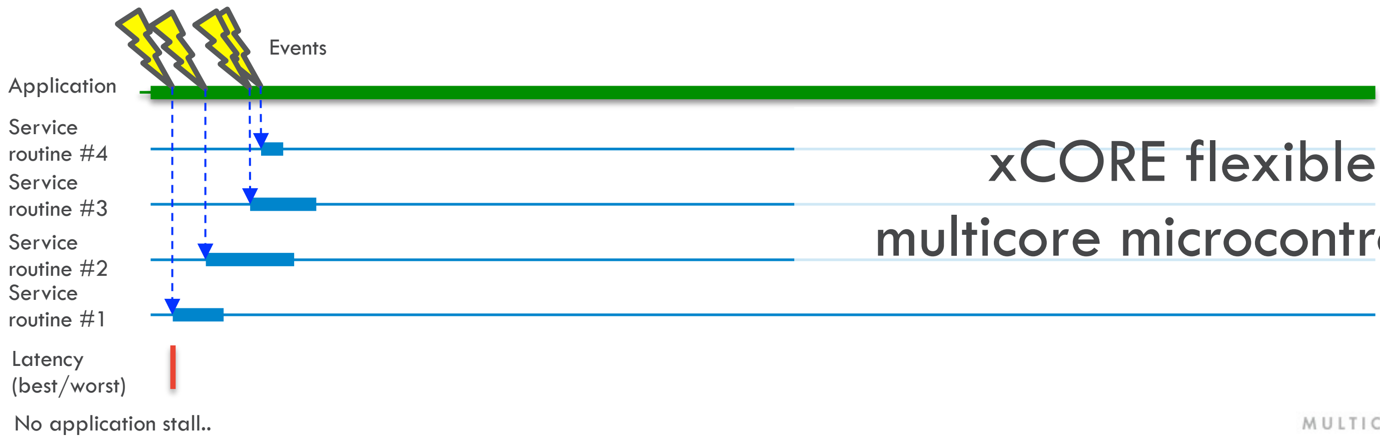
xCORE timing determinism



xCORE Hardware-Response technology

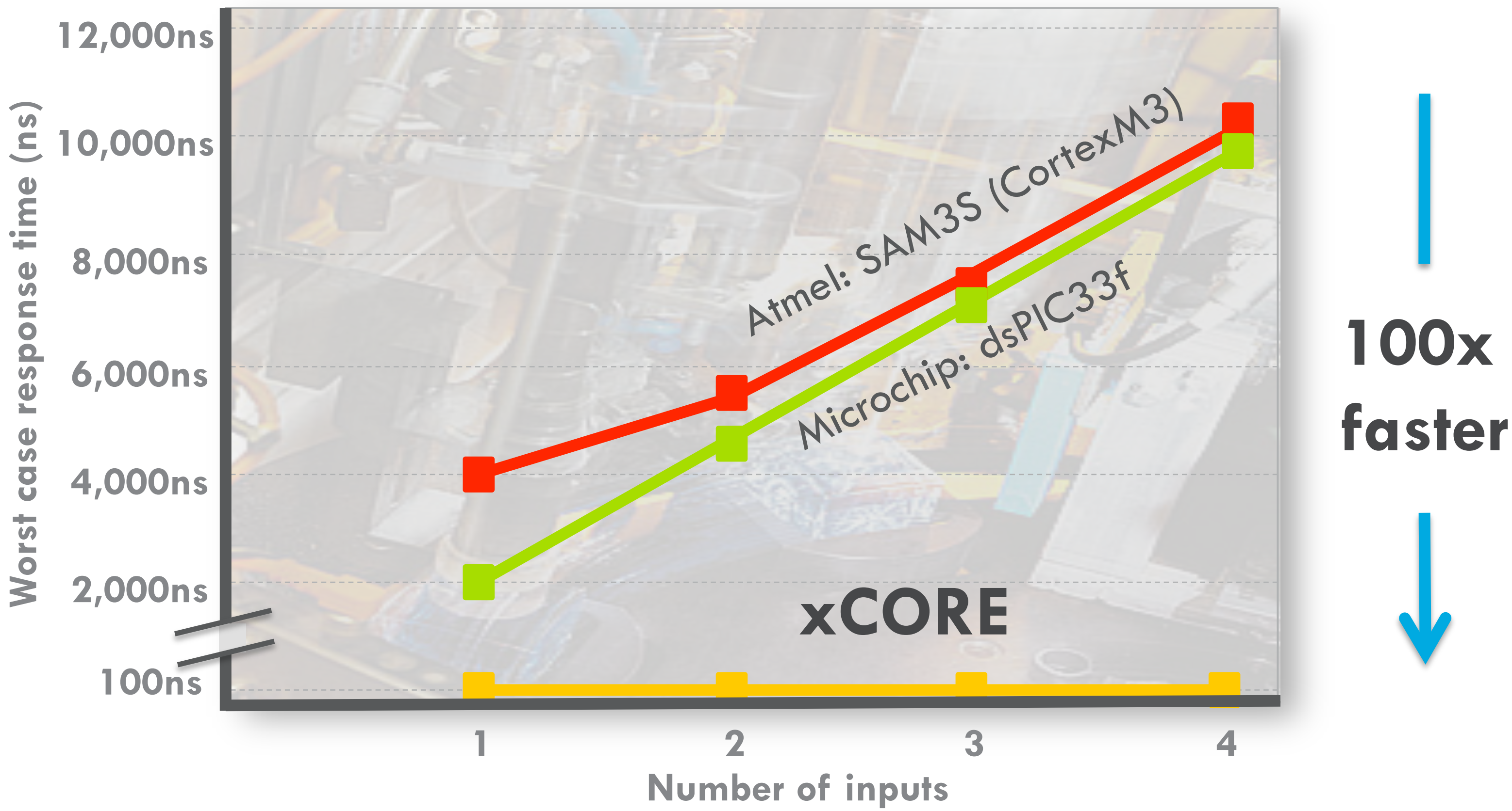


Traditional, single core MCU running an RTOS



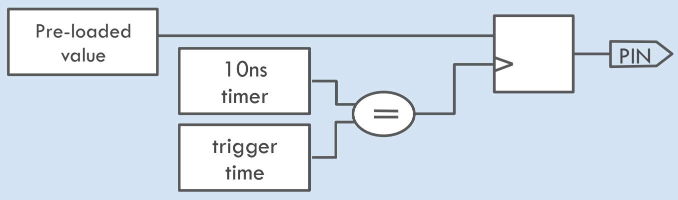
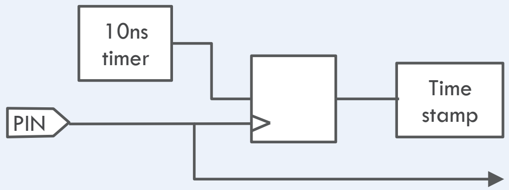
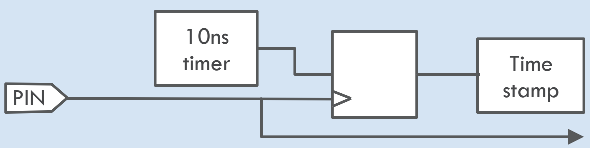
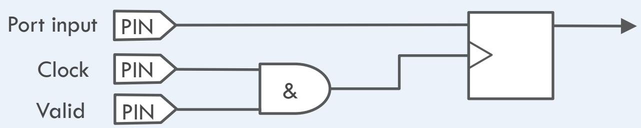
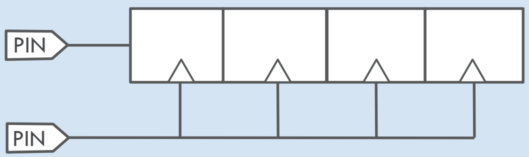
xCORE flexible multicore microcontroller

100x faster in the real world



- Response time to asynchronous external I/O events
- Results normalized for 100MHz clock for fair comparison
- Atmel & Microchip devices running freeRTOS
- Link to white paper for full details

I/O ports: offload processing

	Timed Input / Output	• Generate arbitrary timed output waveforms under software control • Sample signals at specific times • 10ns resolution
	Time-stamping	• Reads the current time in the port • Provides an origin for timed I/O • 10ns timer
	Predicated Input	• Core waits for given pin state • Responsive software state machines
	Clocked port with strobe	• Synchronise to external clock domains • Support for throttled data
	Dedicated serialization hardware	• 32bit serialize/deserialize data • Extends pin data rate beyond instruction rate - 60Mbps

xCORE products

General Purpose

- 4 – 16 Logical Cores
- 64KB – 128KB RAM
- Hardware-Response I/O
- *46 family members*

USB

- 8 – 16 Logical Cores
- 64KB – 128KB RAM
- Integrated USB 2.0 PHY
- Hardware-Response I/O
- *12 family members*

Analog

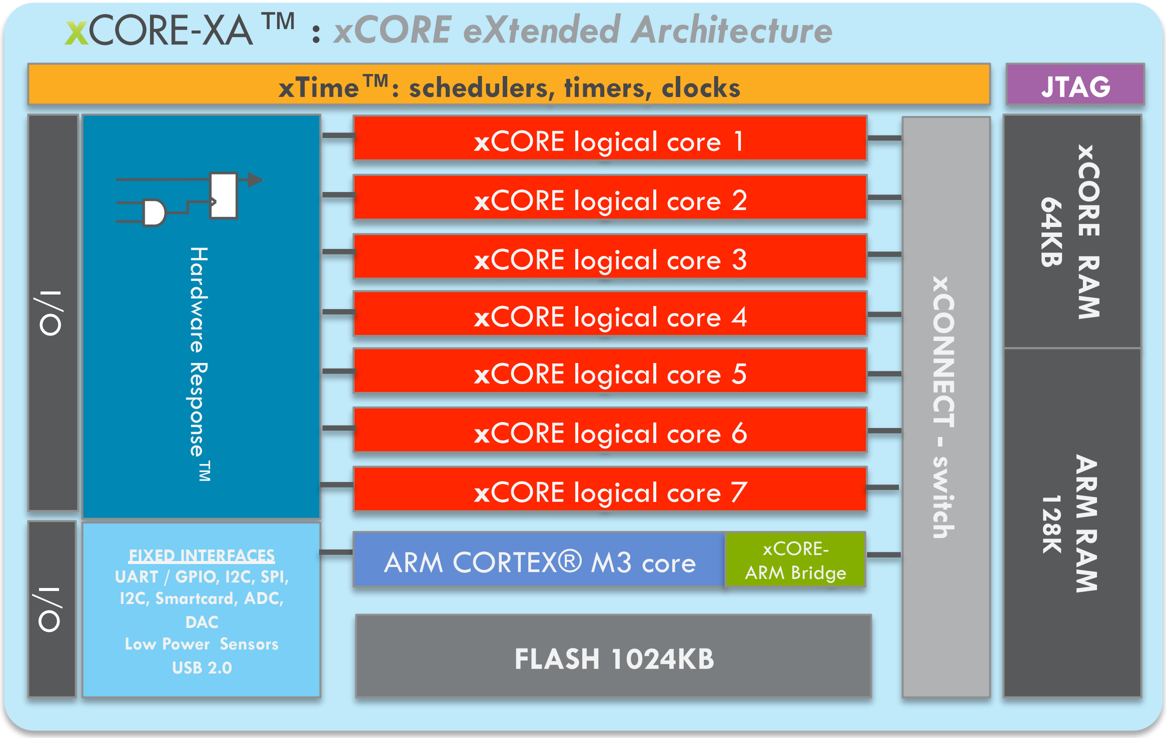
- 6 – 16 Logical Cores
- 64KB – 128KB RAM
- Hardware-Response I/O plus Analog I/O
- Low Power
- *24 family members*

eXtended Architecture

- 7 Logical Cores + CORTEX M3
- 192KB RAM + 1024 FLASH
- Hardware-Response I/O + ARM GPIO
- Dedicated low power peripherals
- Ultra Low Power
- *2 family members*

xCORE-XA

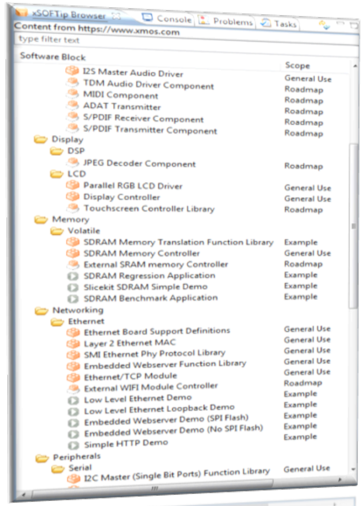
- Multicore microcontroller
  - Eight 32bit cores
  - Core dedicated to ARM bridge
  - 500MIPS performance
  - DSP and security processing
- Unique I/O flexibility
  - Hardware Response I/O ports and ARM GPIO
  - Low energy fixed peripherals
  - Configurable xCORE I/O
- Integrated system memory
  - 192KB SRAM
  - 1024KB Flash
- Extremely low energy
  - Down to 100nA



XS1 Product Families

Family	General Purpose	USB	Analog	XA
Device	XS1-L	XS1-U	XS1-A	XS1-XA
Logical Cores	4, 6, 8, 10, 12, 16	6, 8, 10, 12, 16	6, 8, 10, 12, 16	8
MIPS	400-1000	500-1000	500-1000	500
Total SRAM	64, 128, 256 KBytes	64,128 KBytes	64,128 KBytes	192 KBytes
Total Flash	-	-	-	512, 1024 KBytes
Supplies	3v3, 1v0	3v3 or 5v0 core, 3v3 I/O	3v3 or 5v0 core, 3v3 I/O	3v3, 1v0
Low power mode	15mW	<500µW	<500µW	100nA
Analog functions	-	USB, ADC, POR, DC-DC, Decouple	ADC, POR, DC-DC, RTC, Decouple	ADC, DAC, Analog Comparators, USB *(option)*
Fixed low power peripherals	-	-	-	3 Universal (UART/SPI/Smartcard/I2S), 2 USART, 2 UART, 2 I2C
Packages (Hardware Response I/O)	TQFP48 (28) QFP64 (36) QFP128 (64) QFN124(84)	BGA96 (37) + 4 Analog inputs BGA217(73) + 8 Analog inputs	BGA96 (37) + 4 Analog inputs BGA217(73) + 8 Analog inputs	FBGA265 (38)

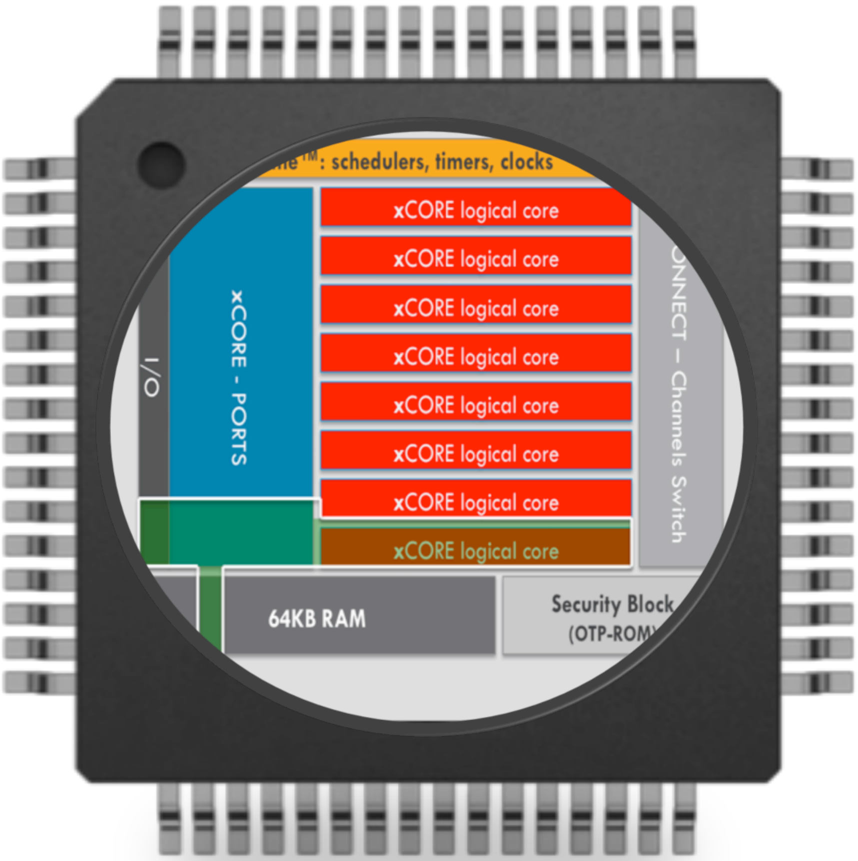
† Power measured under typical conditions



xSOFTip AND APPLICATIONS

xCORE › **xSOFTip** › xTOOLS › xC › xKITS

xSOFTip



USB 2.0
Asynchronous
480Mbps

Time stamped
Ethernet
AVB
100Mbps

Audio
I2S
SPDIF

DSP
BiQuad
FIR

UART
SPI
I2C

Industrial
PWMs
PID

Automotive
CANbus
AVB

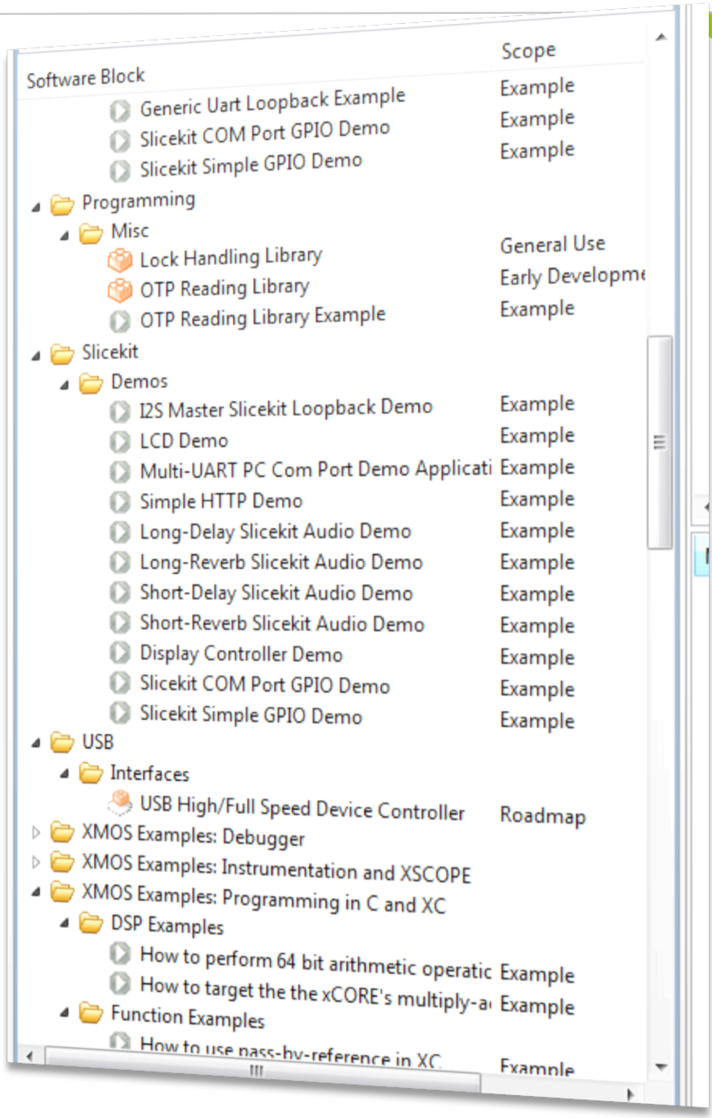
+ *many more…*

MULTICORE MICROCONTROLLERS

xSOFTip Examples

- **Audio & DSP**
  - Reverb
  - Loudness enhancement
  - Biquad
- **Networking**
  - Ethernet MAC
  - Webserver
  - TCP/IP client
- **Industrial**
  - COM port
  - Multi UART
  - CAN
- **General Purpose**
  - LCD display
  - SDRAM
  - USB

And many more!



xSOFTip Applications

General Connectivity

USB-to-things | Ethernet-to-things

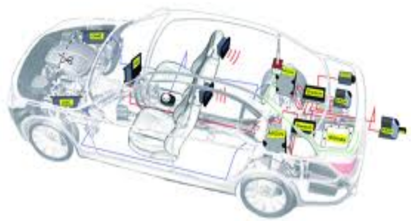
android

Made for iPod iPhone iPad

Multichannel Audio Connectivity

ETHERNET S/PDIF MIDI CERTIFIED HI-SPEED USB

Automotive networking



AVnu lin CAN

Robotics

ROS
Open Source Robotics Foundation

XMOS: bridging things–to–things

I2C | SPI | UART | PWM | A/D | …

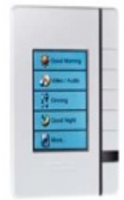
USB–to–things

USB 2.0
USB 1.1
Host/slave

Serial | GPIO

Fieldbus

CAN Modbus

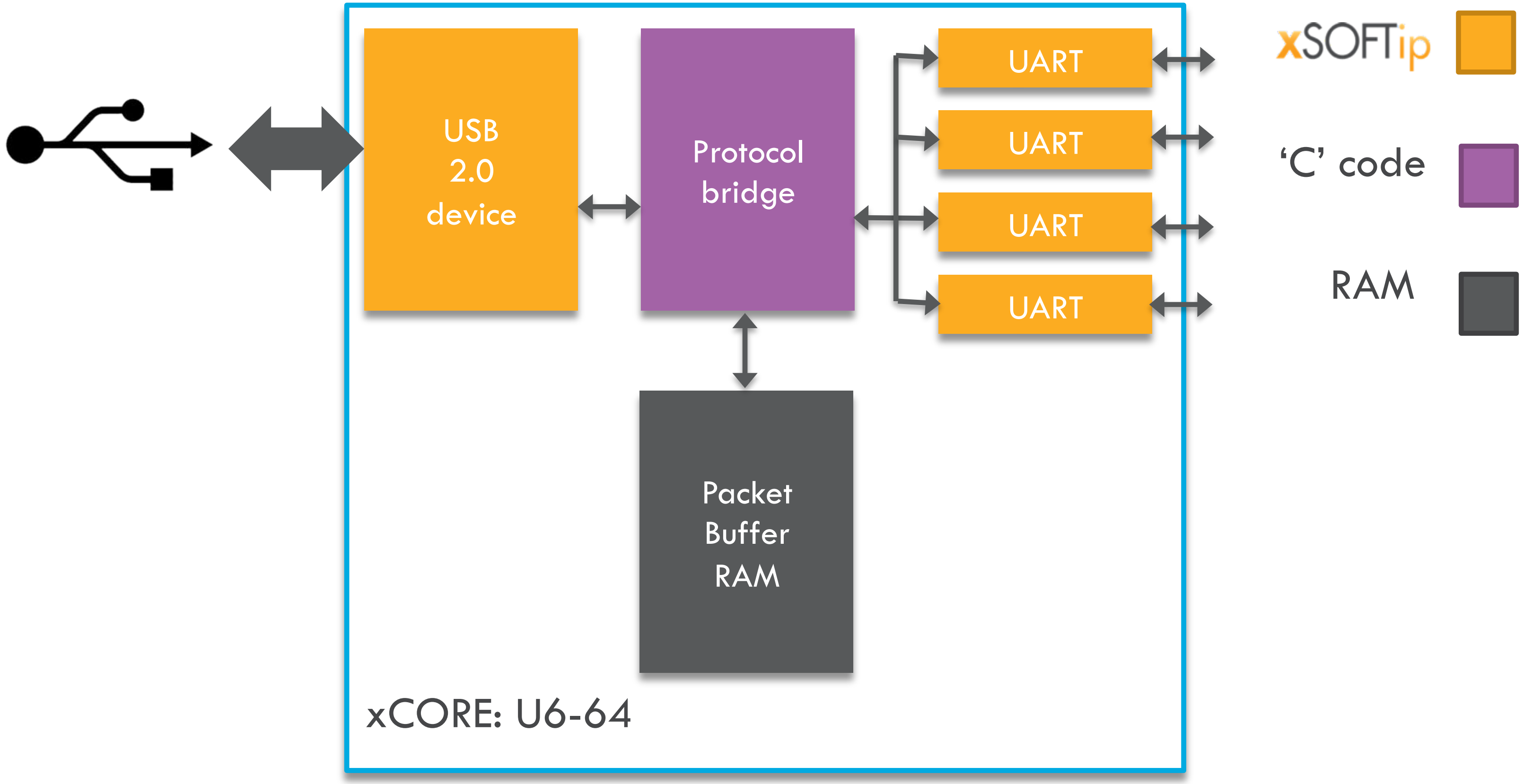


Ethernet–to–things

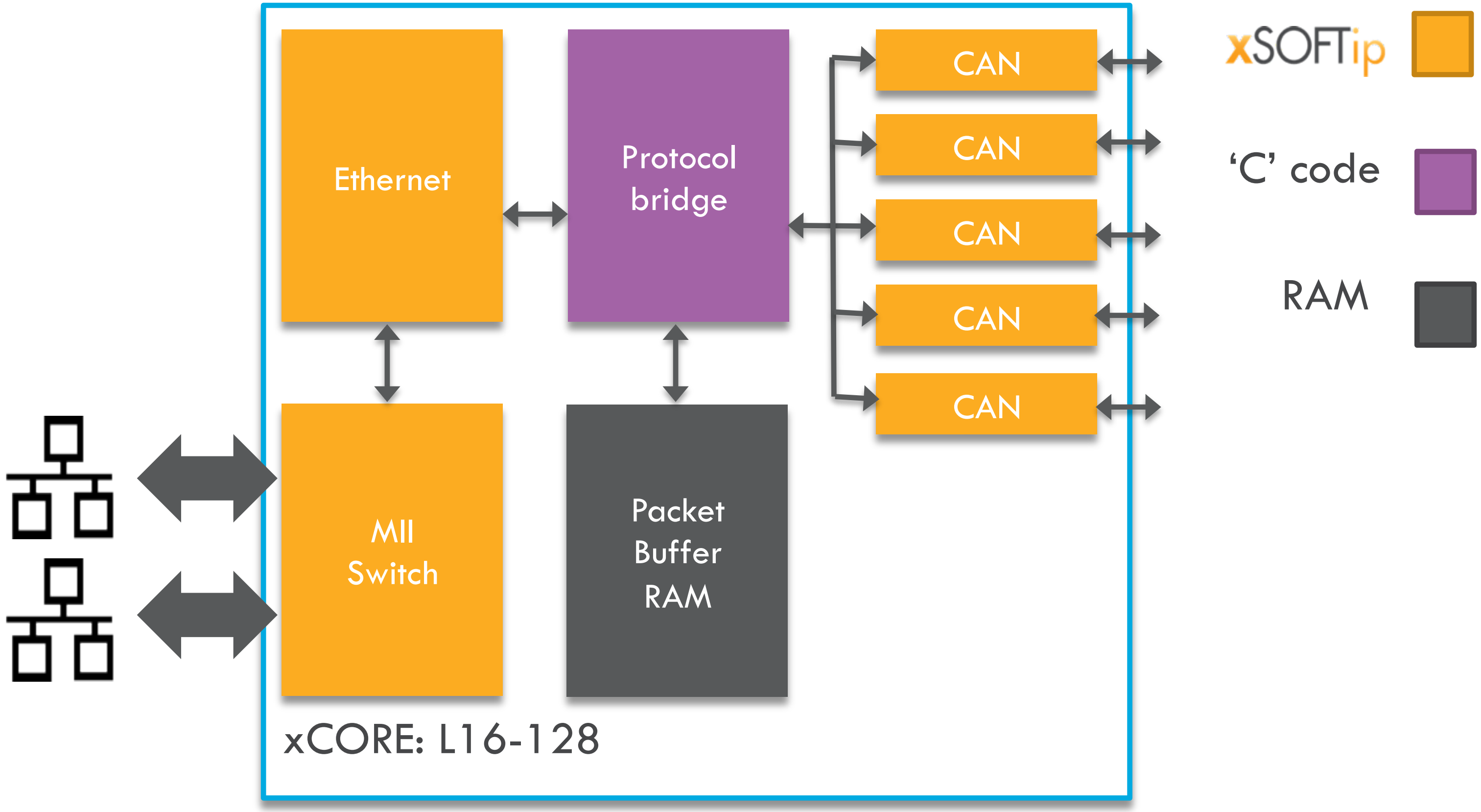
100 Mbps MAC
TCP/IP
Daisy chain, Redundancy, Packet pass-through

MULTICORE MICROCONTROLLERS

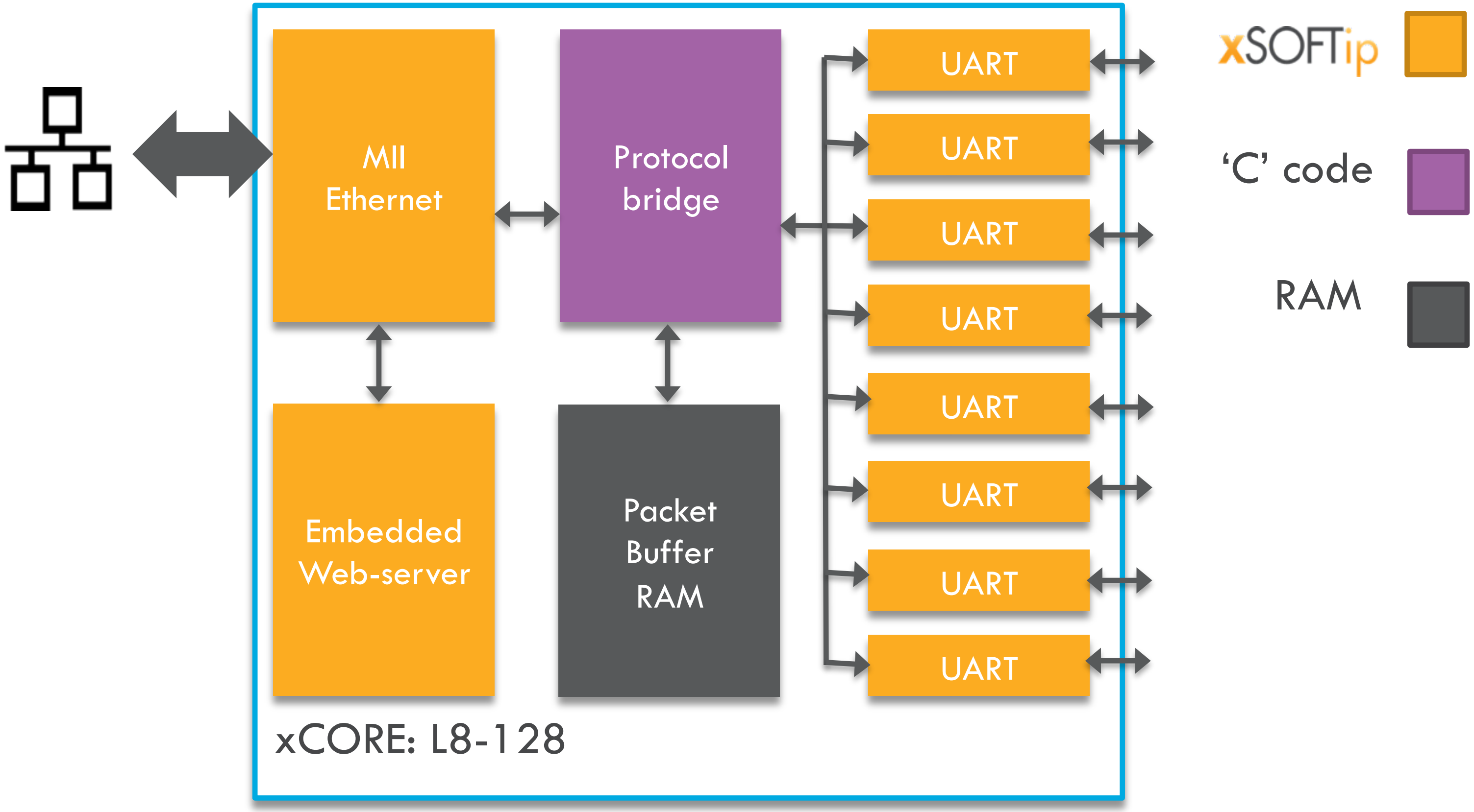
Connectivity: USB 2.0–to–Quad-UART



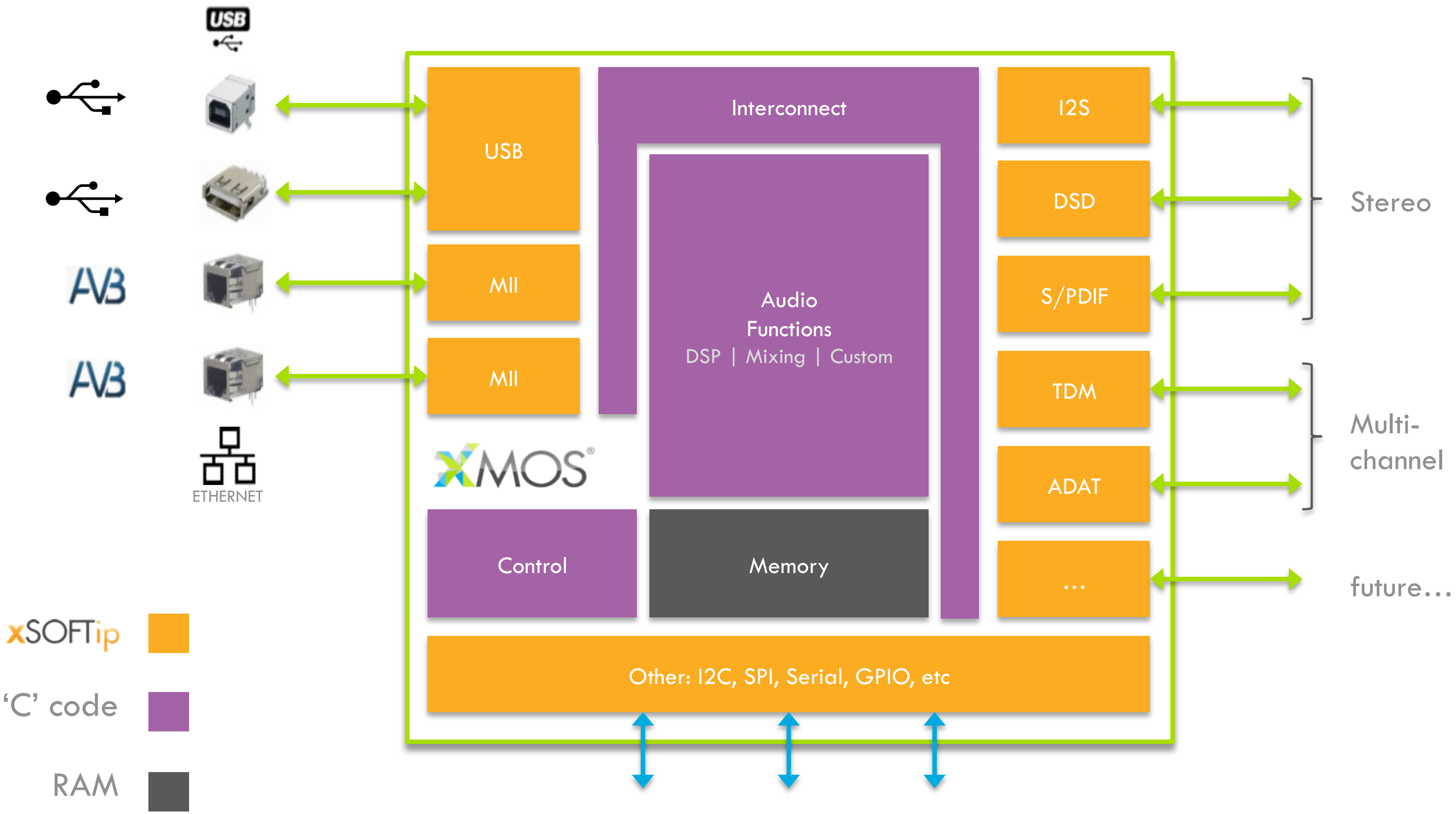
Connectivity: Ethernet DC–to–CAN



Connectivity: Ethernet–to–Octal-UART

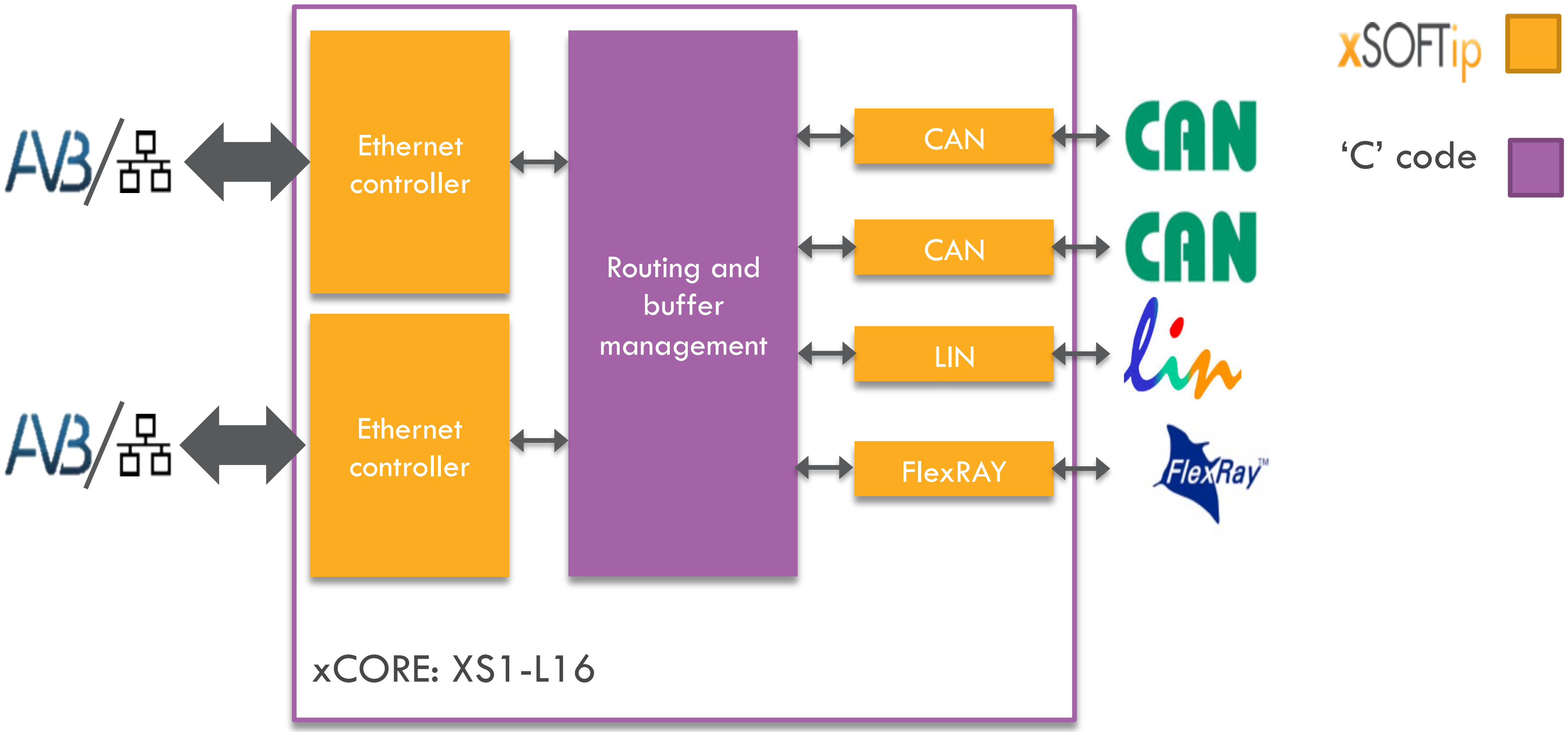


Audio connectivity: capability

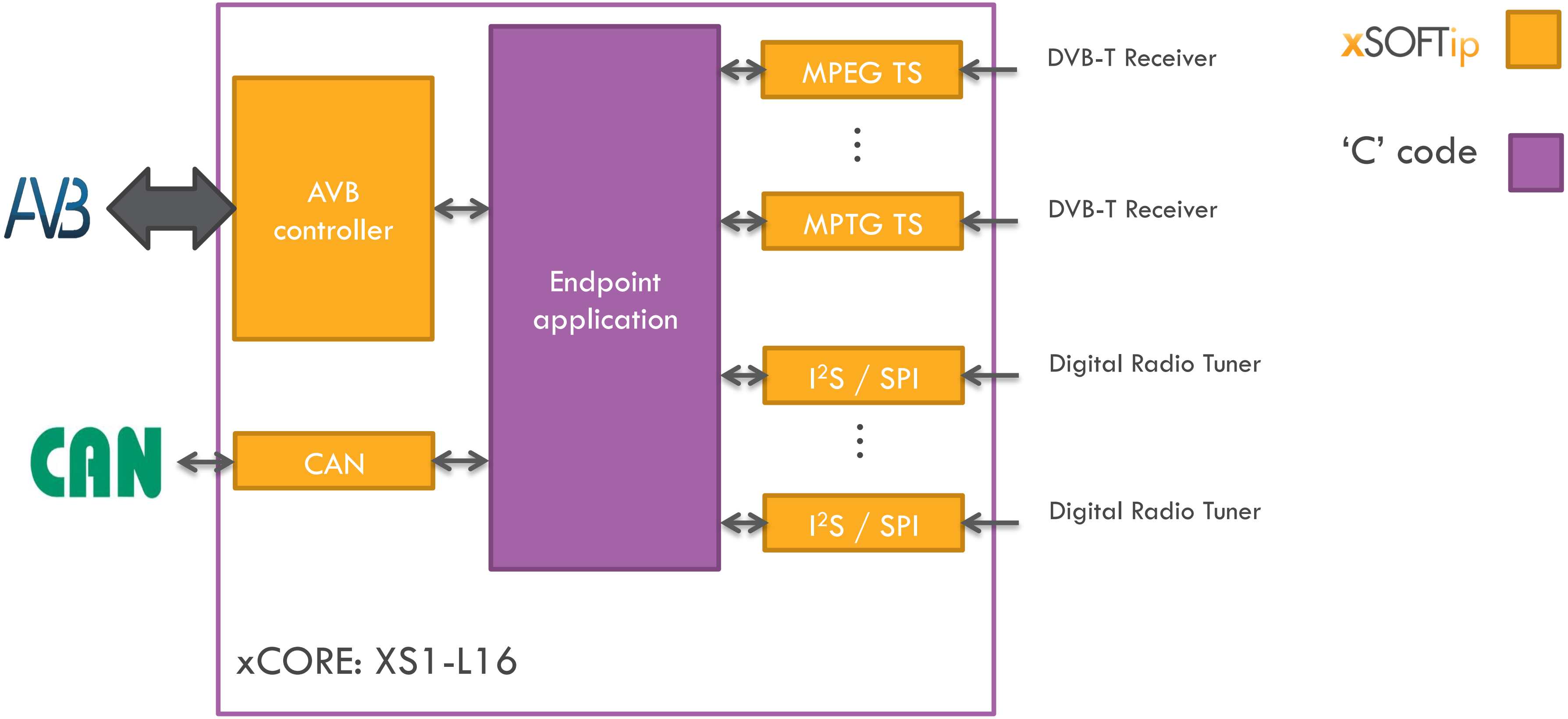


Scalable, flexible, multi-standard multi-channel audio connectivity

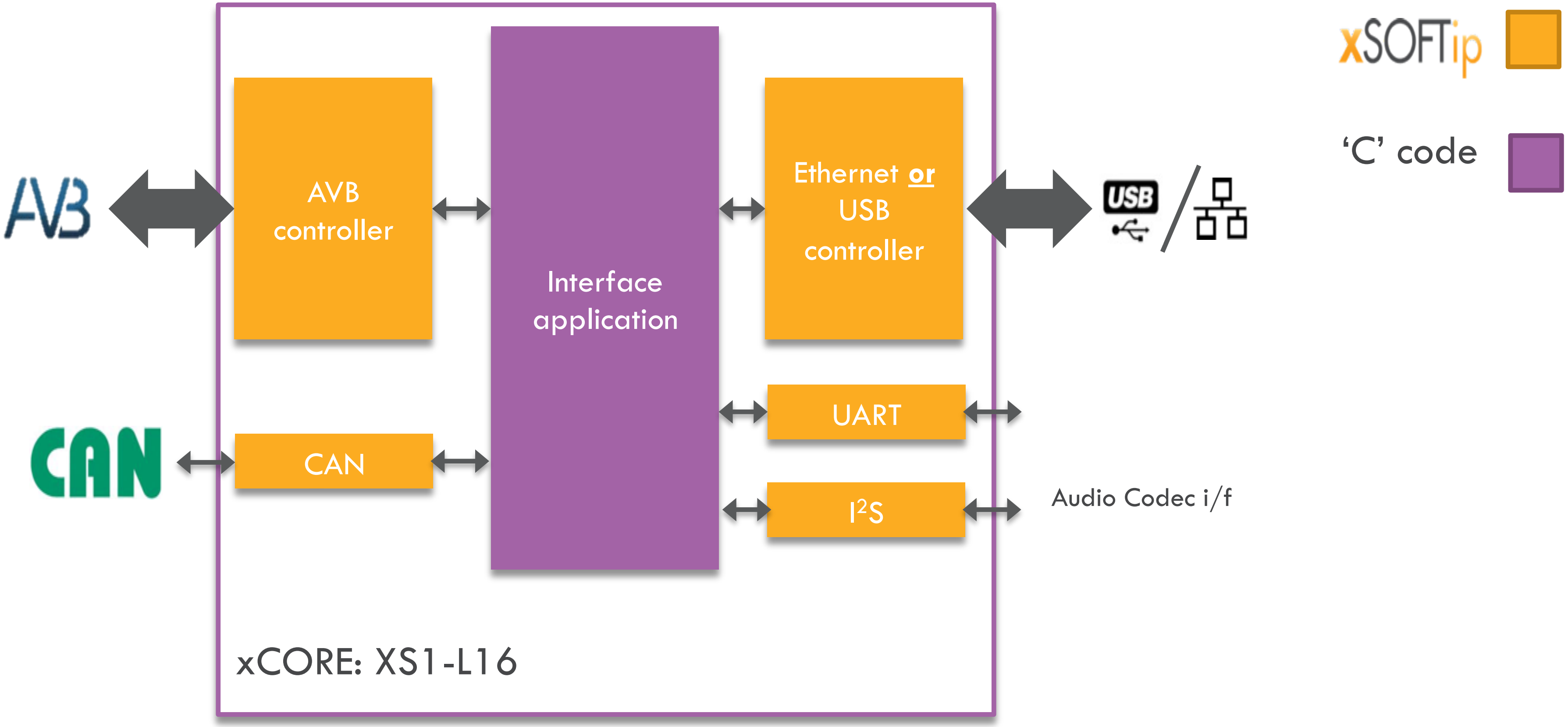
Automotive: network gateway



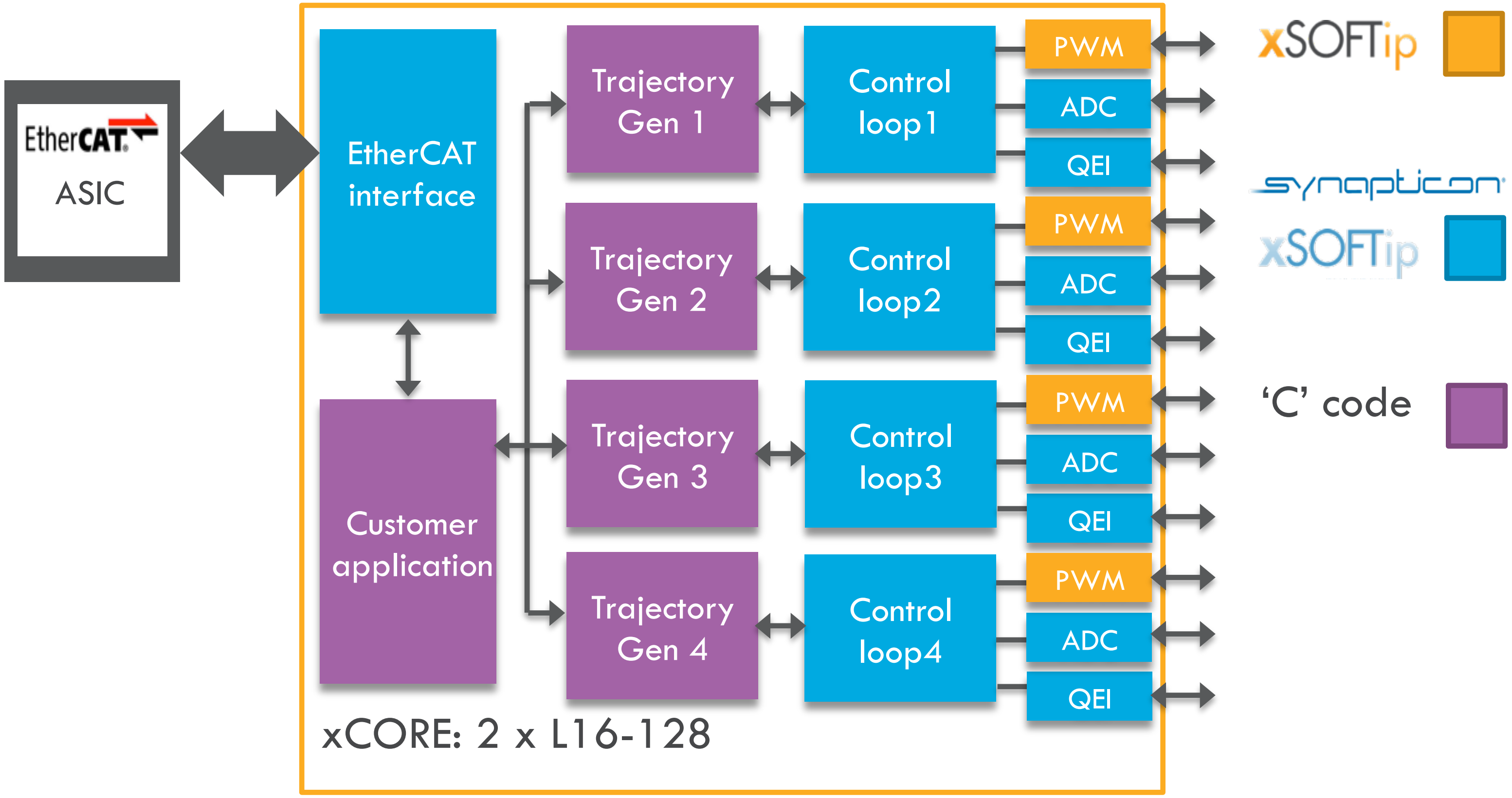
Automotive: AVB Audio/Video endpoint



Automotive: telematics unit interface



Robotics: 4-axis mobile platform



Development TOOLS

xCORE | xSOFTip | xTOOLS | xC | xKITS

xTIMEcomposer Editions

Enterprise Edition

- Enterprise class tools
- Stable and verified xSOFTip
- Licensed and maintained
- Bug-fixes provided against historical Enterprise releases
- Supported via xmos.com

Community Edition

- Rolling tools beta program
- Alpha & beta xSOFTip available
- New features introduced first
- User testing pre enterprise
- Fixes provided against latest release
- Supported via xcore.com

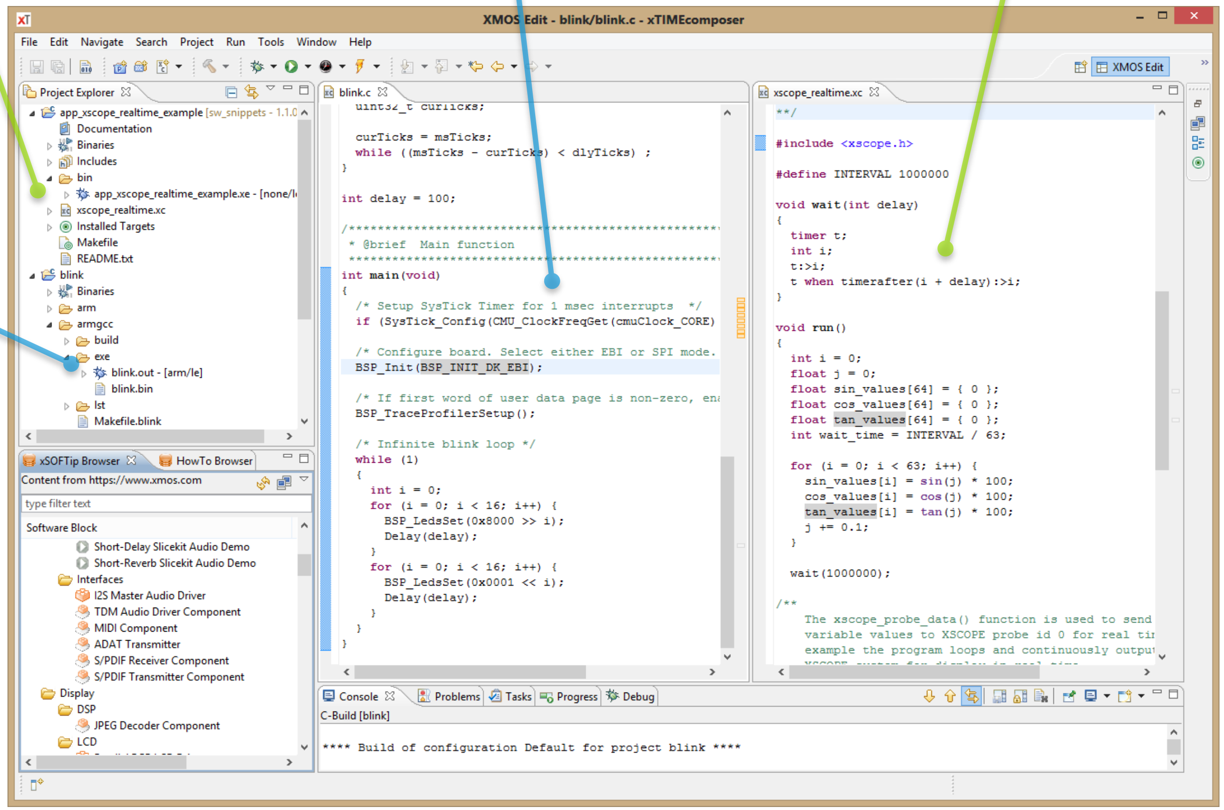
Programmable SoC - all in software

XMOS xCORE target makefile project

ARM C code editing within xTIMEcomposer

xCORE C code editing within xTIMEcomposer

ARM Cortex™-M3 target makefile project



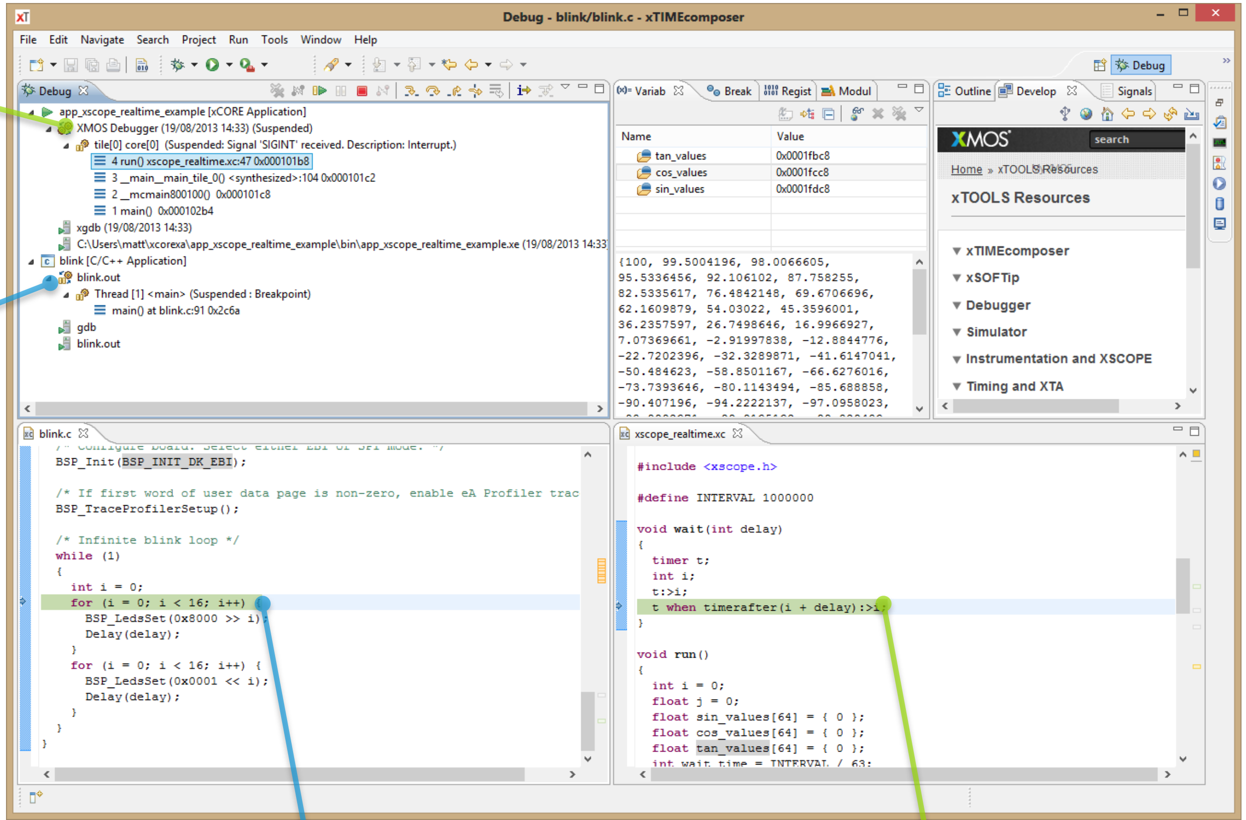
MULTICORE MICROCONTROLLERS

xCORE | xSOFTip | xTOOLS | xC | xKITS

Advanced debug

xCORE debugger

ARM CORTEX-M3 debugger

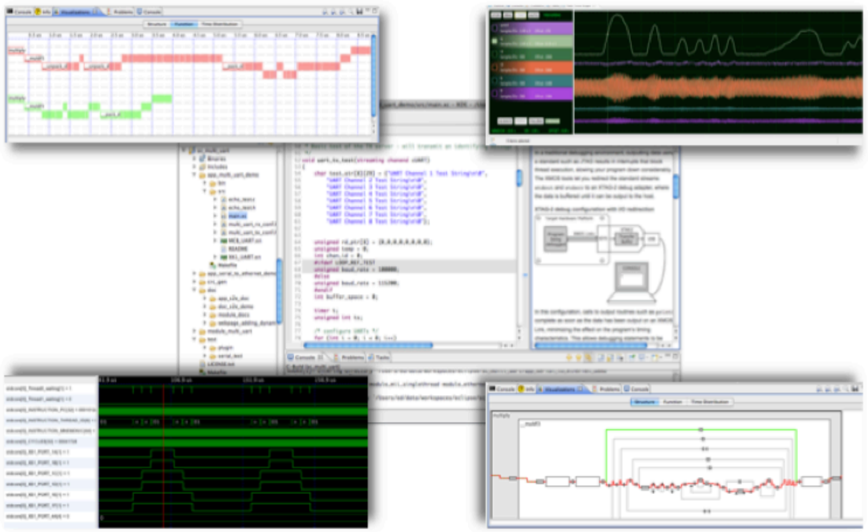


Parallel debugging of xCORE and ARM

ARM Cortex™-M3 source line shown inside debug perspective

xCORE source line shown inside debug perspective

xTIMEcomposer



Debug adapter

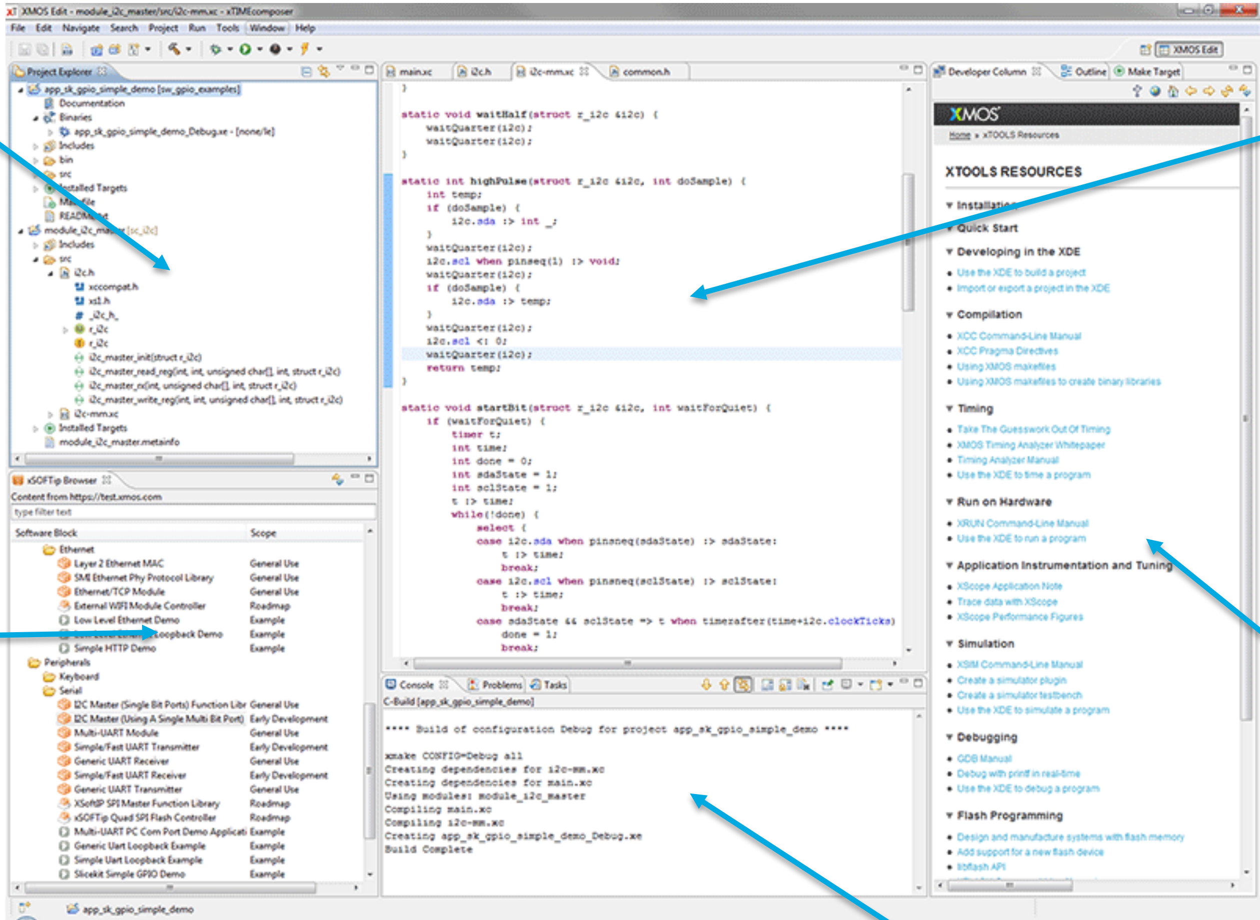
- **Eclipse graphical environment**
  - Plus command line tool option
- **Industry leading LLVM C and C++ compilers**
  - C language extensions for fine-grain real-time control
- **xSOFTip Explorer fully integrated**
- **XTA™**
  - Static timing analyzer
- **xSCOPE™**
  - In circuit instrumentation
  - Real-time logic analyzer
- **xSIM™**
  - Cycle accurate simulator
- **xDEBUG™**
  - GDB xCORE debugger
- **Free Download**

  | Windows | Apple OS-X | Linux |

xTIMEcomposer

Project Explorer

Code Editor



xSOFTip Browser

Developer Column

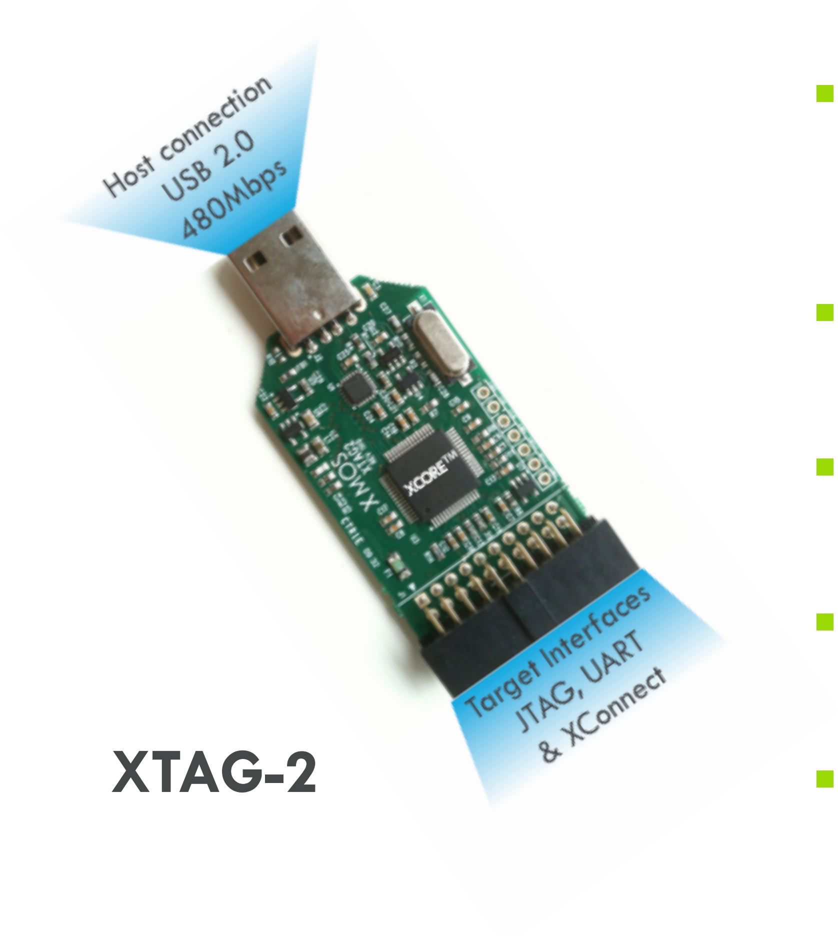
Documentation

Console and xSCOPE

Compiler output and printf output from xCORE

Debugging: XGDB with xTAG

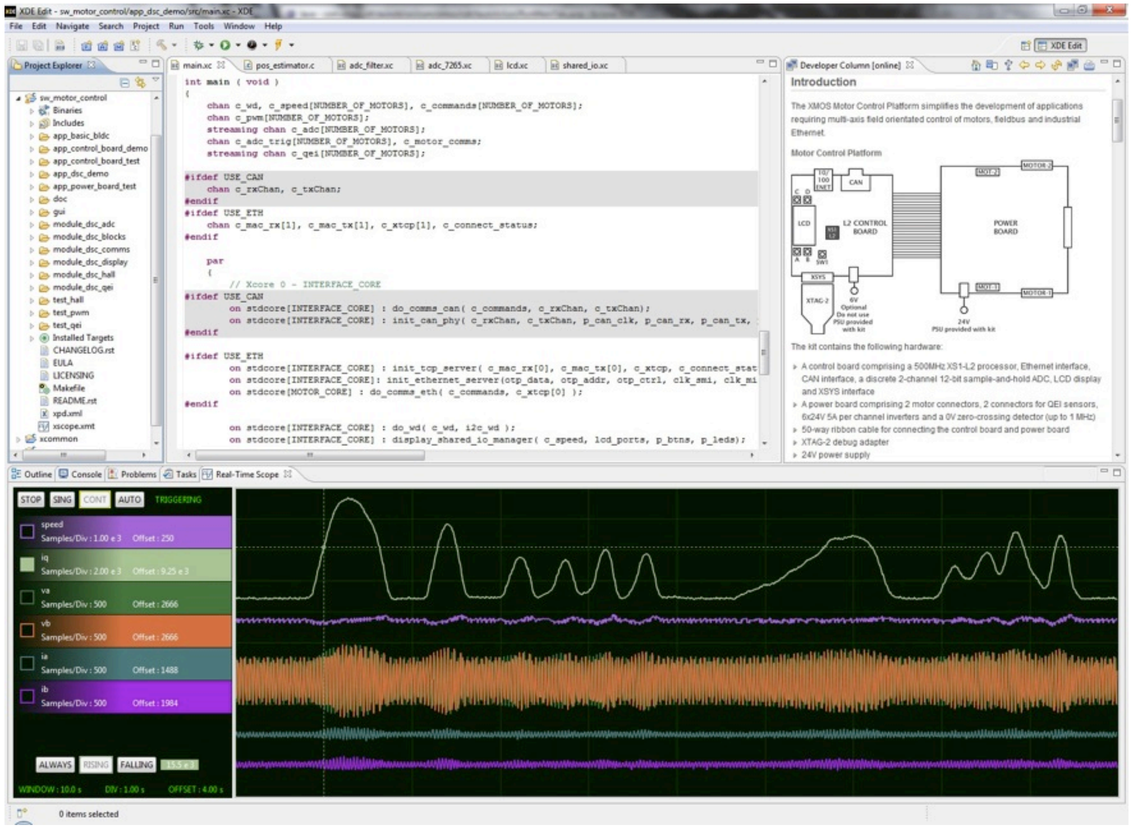
XGDB



XTAG-2

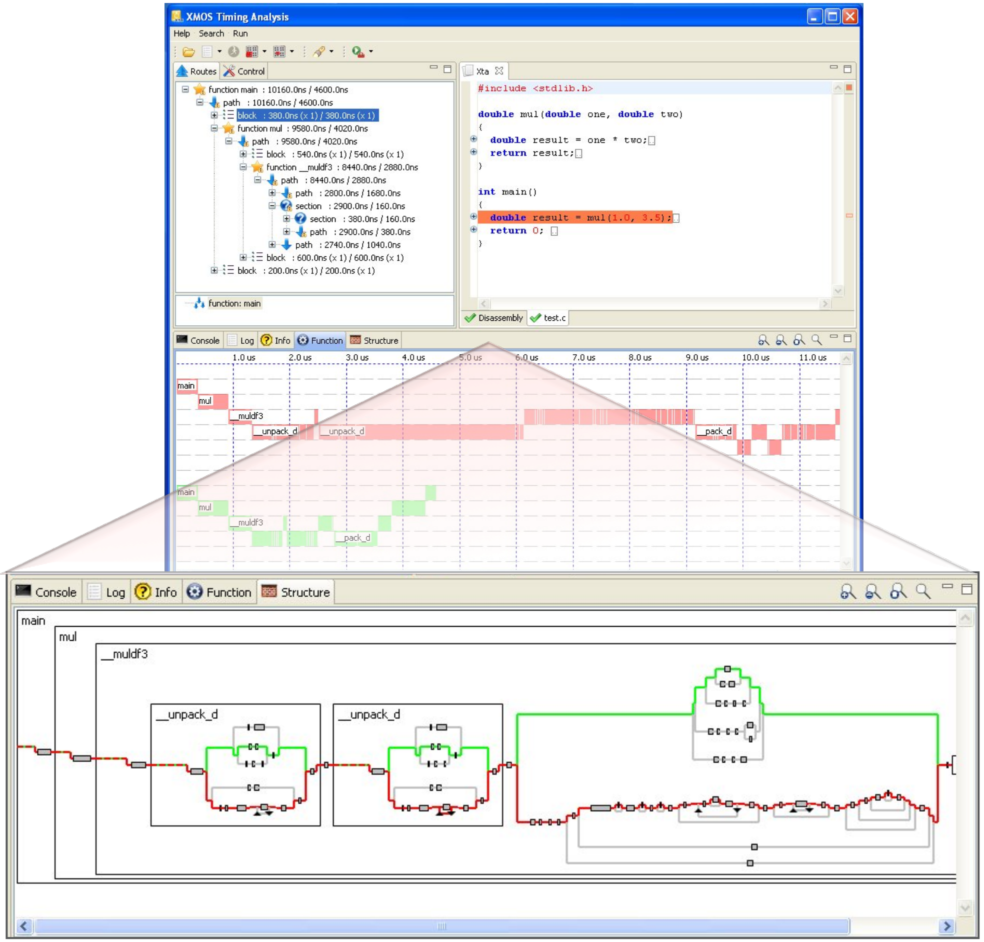
- Based on Gnu GDB, with GUI support in xTIMEcomposer
- Set breakpoints
- Multicore debug support
- Command scripting
- Connects via 4-pin xCONNECT interface

xSCOPE



- xSCOPE library distributed with xTIMEcomposer tools
  - Real-time view in tools
  - Stream data to file for post analysis
- Capture and view meaningful data
  - Includes 10ns timestamp
- Very low intrusiveness
  - Up to 1MSPS capture rate
  - Includes fast printf feature
- Connects via 4-pin xCONNECT interface

XMOS Timing Analyzer (XTA)

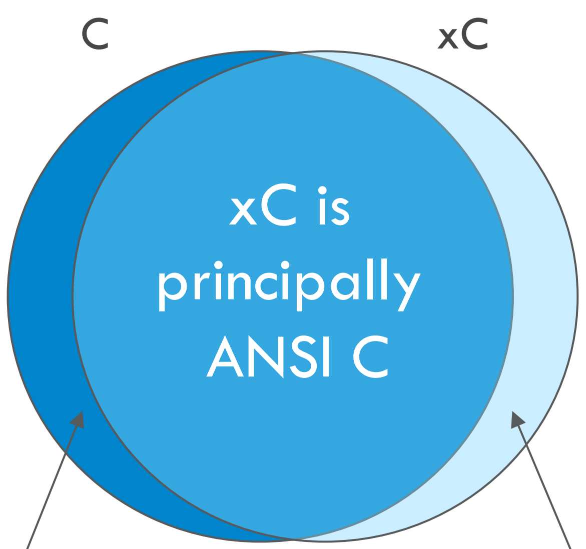


- Static timing analysis of xCORE based applications
- Checks timing for routes through code
- Highlights best/worst timing
- Graphical display of routes through code, including structural view
- Command line verification mode
- Test timing assertions using a script
- Very fast operation
- Analyzes code blocks and functions in seconds
- Integrate with build systems to check timing at compile time
- Unique tool due to determinism of XMOS architecture

PROGRAMMING IN xC

xCORE | xSOFTip | xTOOLS | **xC** | xKITS

How does xC compare with C?



Pointers
Volatile

Support for communication, IO timing, events and parallelism

Types
port
clock
timer
chan

Parallelism
par

Events
select

in port
:> :> >>
@ (time)
buffered

out port
<: <: >>
@ (time)
buffered

chan
:> <:
streaming
transaction
master
slave

select
case
ordered

Use C, xC or a mixture of both

xC : Learning Outcomes

- ✓ **xC multichannel extensions to C**
  - ✓ Adds keywords for concurrency, I/O and timing
  - Handles multiple asynchronous events deterministically
- ✓ **xCORE offers a range of intelligent hardware resources**
  - ✓ Logical Cores to run tasks
  - ✓ Channels and Channel Ends enable communication between tasks
  - ✓ Timers create events with precise timing or measure the precise timing of external events
  - ✓ Hardware Response™ I/O Ports interact with the outside world
- ✓ **Hardware Response™ I/O Ports are powerful and flexible**
  - ✓ Create complex and fast I/O protocols
  - ✓ Create bit-level and cycle-level protocols without overloading a logical core
- ✓ **xTIMEcomposer How-to examples show how to write xC**

xC extends C

Additional data types

```
port
timer
chan, chanend
```

Additional constructs

`:>`	*input*
`<:`	*output*
`select`	*event-driven processing*
`par`	*fork-join*

These features ***simplify*** programming.

Following slides show xC keywords in **bold green** text.

Logical cores and concurrency

- The **`par`** statement creates a number of task
- Each task consumes one or more logical cores
- Example

```
int main() {
  par {
    f(); // run tasks
    g(); // concurrently
  }
  return 1;
}
```

Logical cores and concurrency

- Par can be used to extend to multiple xCORE tiles
- The **on** statement specifies on which tile to run a task

Multiple xCORE tiles are always defined as arrays

```
int main() {
  par {
    on tile[0] : f(); // run tasks in
    on tile[1] : g(); // concurrently
  }
  return 1;
}
```

- f() runs on tile 0
- g() runs on tile 1

Channels: communication between cores

- Tasks communicate using channels
  - Synchronous point-to-point communication
  - Use the input (:>) and output (<:) operators
  - Default data size is 32 bits

- Example

```
chan c;    /* declare a channel */
c :> x;    /* input from c and store in x */
c <: y;    /* output y on c */
```

Interfaces

- xC interfaces provide a logical abstraction
- Allows easy partitioning between tiles/chips
- Lower overhead and latency to shared memory

Ports

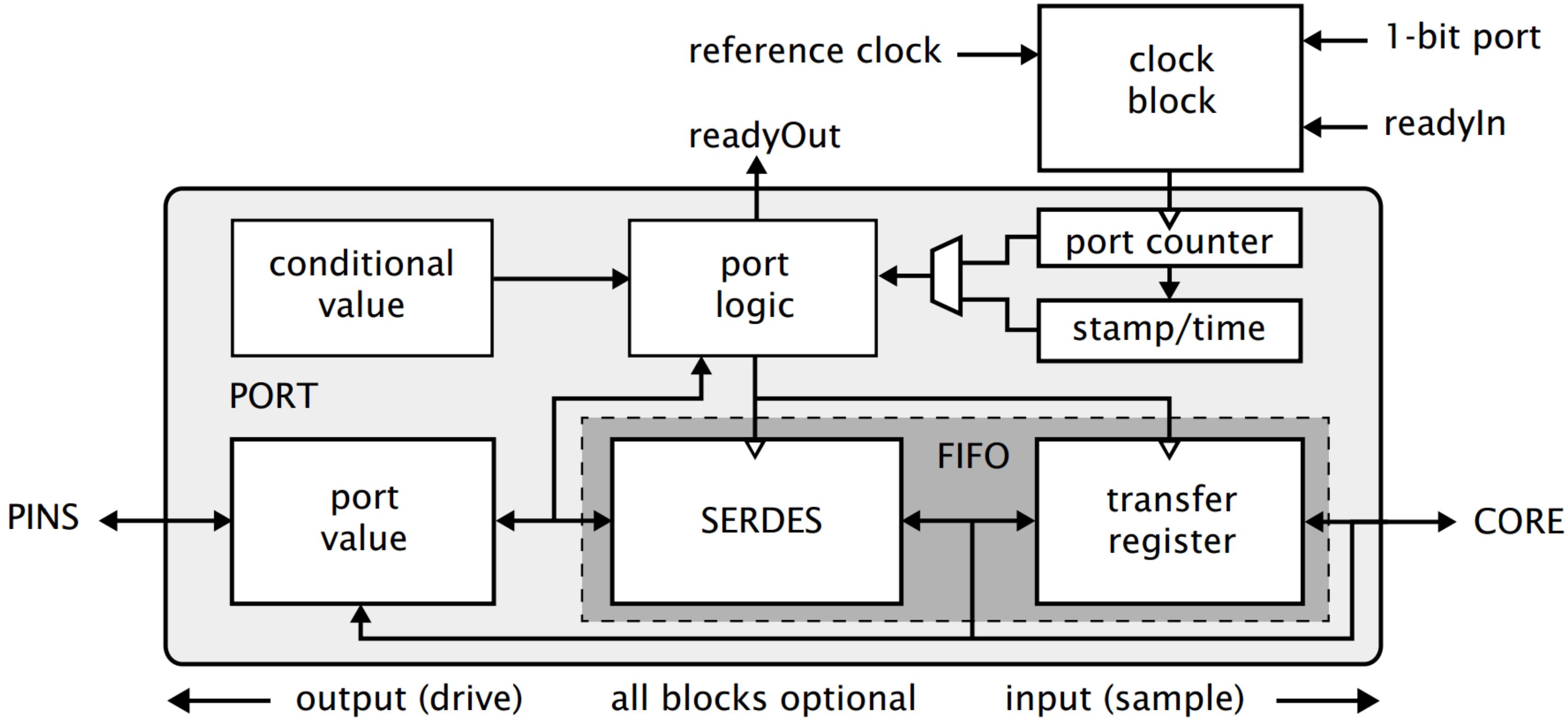
- Ports provide direct access to pins
  - *More powerful, faster and flexible than traditional 'GPIO'*

- xC port declaration
  - `port p = XS1_ PORT_8A; // interface to 8 pins`

- xC input operation on port
  - `p :> x;`

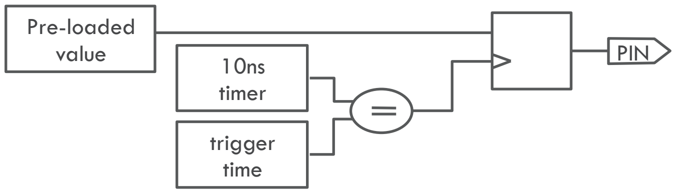
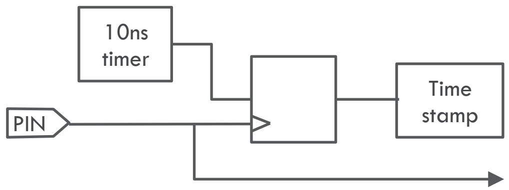
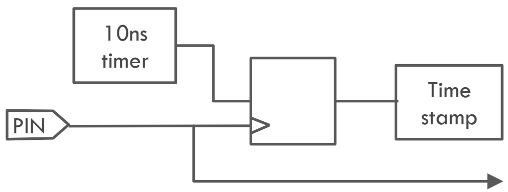
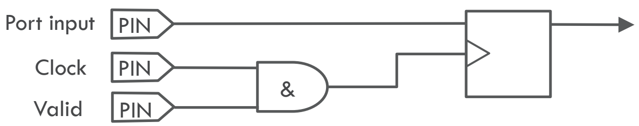
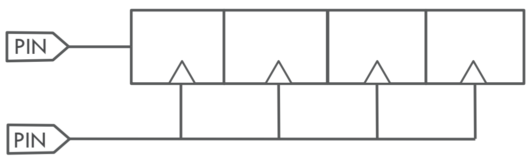
- xC output operation on port
  - `p <: x;`

I/O ports: architecture

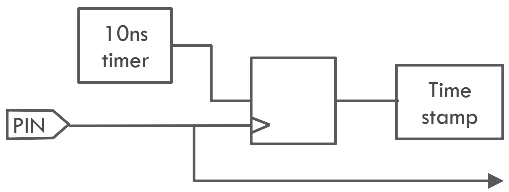
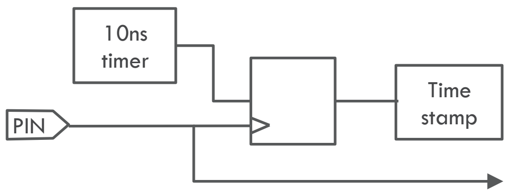
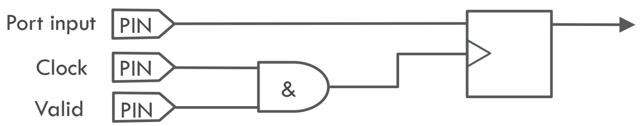
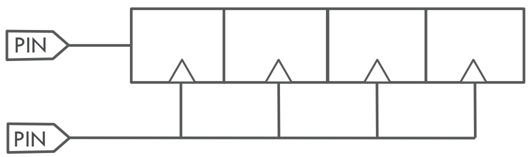


http://www.xmos.com/published/xs1-ports-introduction

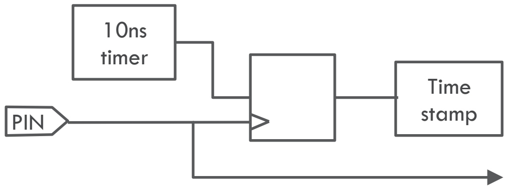
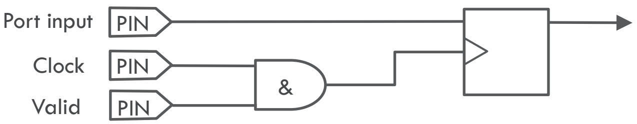
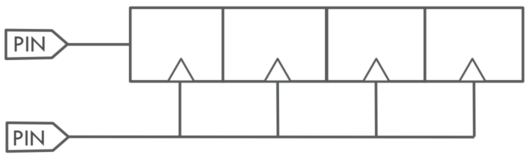
xCORE Ports – advanced functions

	Timed output	• Generate timed waveforms such as LCD, PWM, MII etc..
	Time-stamped input	• Decode timed protocols such as IEEE 1588 (801.1AS)
	Conditional input	• Logical core waits for given pin state • Very efficient state machines
	Clocked port with data ready	• Synchronize to external clocks • Support for throttled data
	Dedicated serialization hardware	• Serialize/deserialize data • Extends pin data rate beyond instruction rate

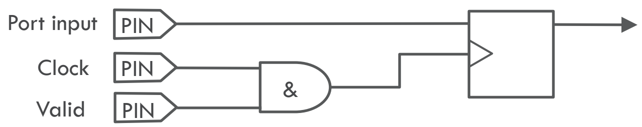
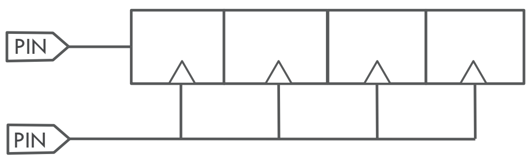
xCORE Ports – advanced functions

`p @ t <: x;`	Timed output	• Generate timed waveforms such as LCD, PWM, MII etc..
	Time-stamped input	• Decode timed protocols such as IEEE 1588 (801.1AS)
	Conditional input	• Logical core waits for given pin state • Very efficient state machines
	Clocked port with data ready	• Synchronize to external clocks • Support for throttled data
	Dedicated serialization hardware	• Serialize/deserialize data • Extends pin data rate beyond instruction rate

xCORE Ports – advanced functions

`p @ t <: x;`	Timed output	• Generate timed waveforms such as LCD, PWM, MII etc..
`p :> x @ t;`	Time-stamped input	• Decode timed protocols such as IEEE 1588 (801.1AS)
	Conditional input	• Logical core waits for given pin state • Very efficient state machines
	Clocked port with data ready	• Synchronize to external clocks • Support for throttled data
	Dedicated serialization hardware	• Serialize/deserialize data • Extends pin data rate beyond instruction rate

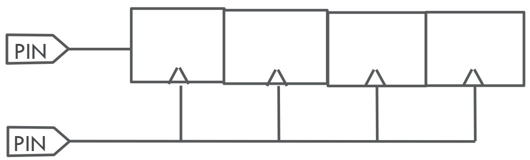
xCORE Ports – advanced functions

`p @ t <: x;`	Timed output	• Generate timed waveforms such as LCD, PWM, MII etc..
`p :> x @ t;`	Time-stamped input	• Decode timed protocols such as IEEE 1588 (801.1AS)
`p when pinseq(0) :> x;`	Conditional input	• Logical core waits for given pin state • Very efficient state machines
	Clocked port with data ready	• Synchronize to external clocks • Support for throttled data
	Dedicated serialization hardware	• Serialize/deserialize data • Extends pin data rate beyond instruction rate

xCORE Ports – advanced functions

`p @ t <: x;`	Timed output	• Generate timed waveforms such as LCD, PWM, MII etc..
`p :> x @ t;`	Time-stamped input	• Decode timed protocols such as IEEE 1588 (801.1AS)
`p when pinseq(0) :> x;`	Conditional input	• Logical core waits for given pin state • Very efficient state machines

`in port p, r;`
`clock c;`
`configure_ in_strobed_slave(p, r, c);`
`p :> x;`
	Clocked port with data ready	• Synchronize to external clocks
		• Support for throttled data

	Dedicated serialization hardware	• Serialize/deserialize data • Extends pin data rate beyond instruction rate

MULTICORE MICROCONTROLLERS

xCORE Ports – advanced functions

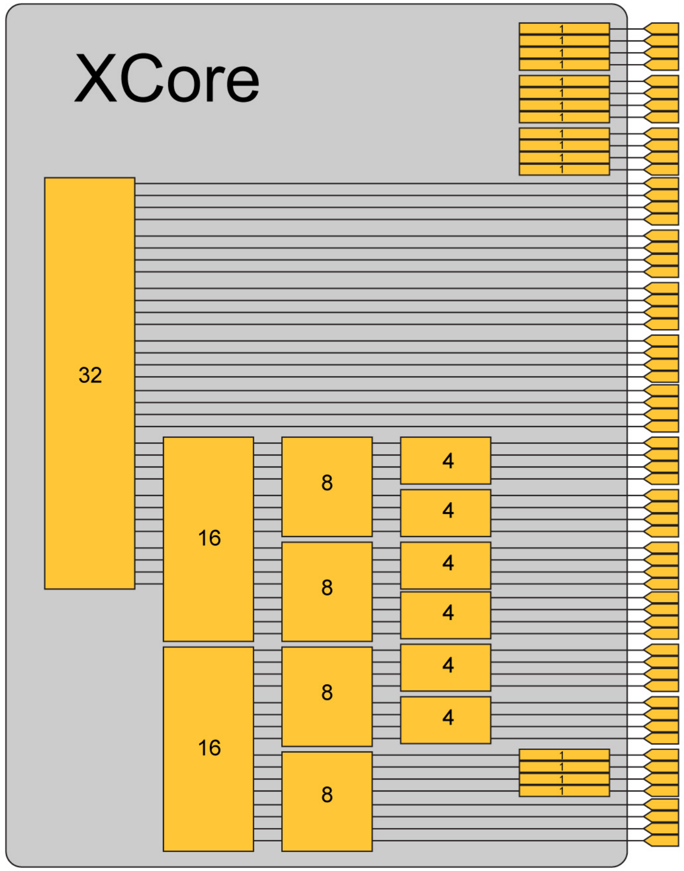
`p @ t <: x;`	Timed output	• Generate timed waveforms such as LCD, PWM, MII etc..
`p :> x @ t;`	Time-stamped input	• Decode timed protocols such as IEEE 1588 (801.1AS)
`p when pinseq(0) :> x;`	Conditional input	• Logical core waits for given pin state • Very efficient state machines

`in port p, r;`
`clock c;`
`configure_ in_strobed_slave(p, r, c);`
`p :> x;`
	Clocked port with data ready	• Synchronize to external clocks
		• Support for throttled data

`buffered port p:8 = XS1_PORT_1A;`
 	
`p <: 0b00110101;`
	Dedicated serialization hardware	• Serialize/deserialize data
		• Extends pin data rate beyond instruction rate

Port I/O Mapping

- 64 I/O pins per xCORE
  - Mix of 1, 4, 8, 16, 32 bit ports
  - Up to 256 I/O pins
- High performance
  - Tightly integrated into the xCORE
  - Precise I/O timing, 10ns resolution
- Time aware
  - All I/O can be synchronized to internal or external reference clock
- Advanced modes
  - Logic to offload threads



Port I/O Mapping

Pin	Link	Port 1bit	Port 4bit	Port 8bit	Port 16bit	Port 32bit	Tile 0
XnD00		P1A0					
XnD01	LA4o	P1B0					
XnD02	LA3o		P4A0	P8A0	P16A0	P32A20	
XnD03	LA2o		P4A1	P8A1	P16A1	P32A21	
XnD04	LA1o		P4B0	P8A2	P16A2	P32A22	
XnD05	LA0o		P4B1	P8A3	P16A3	P32A23	
XnD06	LA0i		P4B2	P8A4	P16A4	P32A24	
XnD07	LA1i		P4B3	P8A5	P16A5	P32A25	
XnD08	LA2i		P4A2	P8A6	P16A6	P32A26	
XnD09	LA3i		P4A3	P8A7	P16A7	P32A27	
XnD10	LA4i	P1C0					
XnD11		P1D0					
XnD12		P1E0					
XnD13	LB4o	P1F0					
XnD14	LB3o		P4C0	P8B0	P16A8	P32A28	
XnD15	LB2o		P4C1	P8B1	P16A9	P32A29	
XnD16	LB1o		P4D0	P8B2	P16A10		
XnD17	LB0o		P4D1	P8B3	P16A11		
XnD18	LB0i		P4D2	P8B4	P16A12		
XnD19	LB1i		P4D3	P8B5	P16A13		
XnD20	LB2i		P4C2	P8B6	P16A14	P32A30	
XnD21	LB3i		P4C3	P8B7	P16A15	P32A31	
XnD22	LB4i	P1G0					
XnD23		P1H0					
XnD24		P1I0					
XnD25		P1J0					
XnD26			P4E0	P8C0	P16B0		
XnD27			P4E1	P8C1	P16B1		
XnD28			P4F0	P8C2	P16B2		
XnD29			P4F1	P8C3	P16B3		
XnD30			P4F2	P8C4	P16B4		
XnD31			P4F3	P8C5	P16B5		
XnD32			P4E2	P8C6	P16B6		
XnD33			P4E3	P8C7	P16B7		
XnD34		P1K0					
XnD35		P1L0					
XnD36		P1M0		P8D0	P16B8		
XnD37		P1N0		P8D1	P16B9		
XnD38		P1O0		P8D2	P16B10		
XnD39		P1P0		P8D3	P16B11		
XnD40				P8D4	P16B12		
XnD41				P8D5	P16B13		
XnD42				P8D6	P16B14		
XnD43				P8D7	P16B15		

- **Naming**
  - Each port is named like
    - ‘P1D0’ - 1-bit port “D”, bit 0
    - ‘P8A4’ - 8 bit port “A”, bit 4
- **Mapping**
  - I/O Ports are mapped to specific xCORE I/O pins (e.g. X0D22)
- **Independence**
  - Every port is an independent hardware resource
- **Ports may be overloaded**
  - Overlapping ports may be activated together
  - Lowest bit-width port takes precedence
- **Devices and Portmaps**
  - Each xCORE device has a different portmap
  - Check www.xmos.com/resources

Timers

- A timer is a special port used for inputting the time
- A timer runs at the frequency of the reference clock
- `timer t;   /* declare a timer */`
- `t :> x;/* input current time into x */`

- Example: Function that waits for period of time

```
void wait(timer t, int period) {
  t :> x;
  t when timerafter(x+period) :> void;
    .
    do something
    .
}
```

Events

- Generated by ports, timers or channel ends
- Used for scheduling within a logical core
- Use "`select`" construct – hardware equivalent of "`switch`"

```
select {
  case p_input :> X:
   ...
   break;
  case t when timerafter (t + PERIOD):
   ...
   break;
}
```

- Waits for one of the events to occur before continuing
- Assign the event priority in the select-case by adding the `[[ordered]]` attribute

Putting it all together

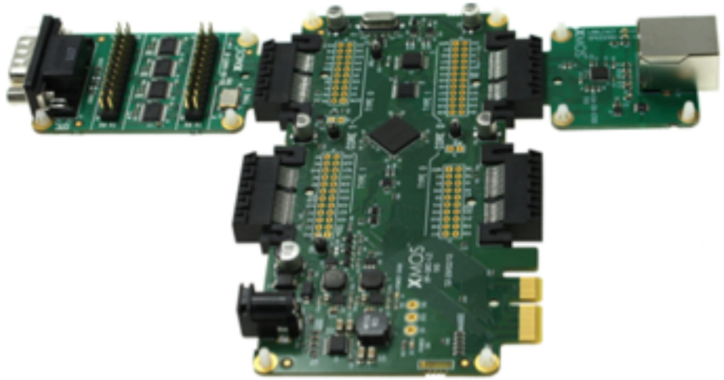
- Combine port events, timer events, channel events in a single select statement
- Take control of your environment!

```
select {
  case p_input :> X:
   ...
   break;
  case t when timerafter (t + PERIOD):
    ...
    break;
}
```

Putting it all together - example

- Combine port events, timer events, channel events in a single select statement
- Take control of your environment!

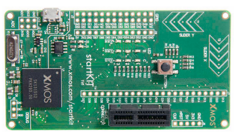
```
while(1) {
   select {
      case chnlend_a :> chnl_input_a :
         printstr("Channel Input A Received ");
         printintln(chnl_input_a);
         break;
     case chnlend_b :> chnl_input_b :
         printstr("Channel Input B Received ");
         printintln(chnl_input_b);
         break;
     case port_a when pinseq(1) :> port_input_a:
         printstr("Port Input A Recieved ");
         printintln(port_input_a);
        break;
     case tmr_a when timerafter(start_time + 10000) :> start_time:
        printstrln("Timer A Fired ");
        break;
    }
 }
```

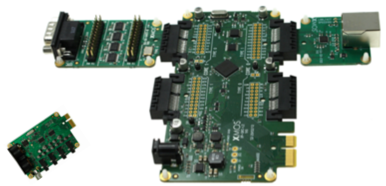


xKITS

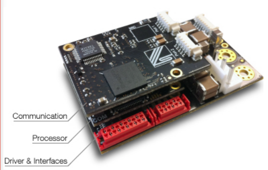
XMOS → xSOFTip → xTOOLS → xC → **xKITS**

xKITS

startKIT



sliceKIT

USB 2.0 Audio

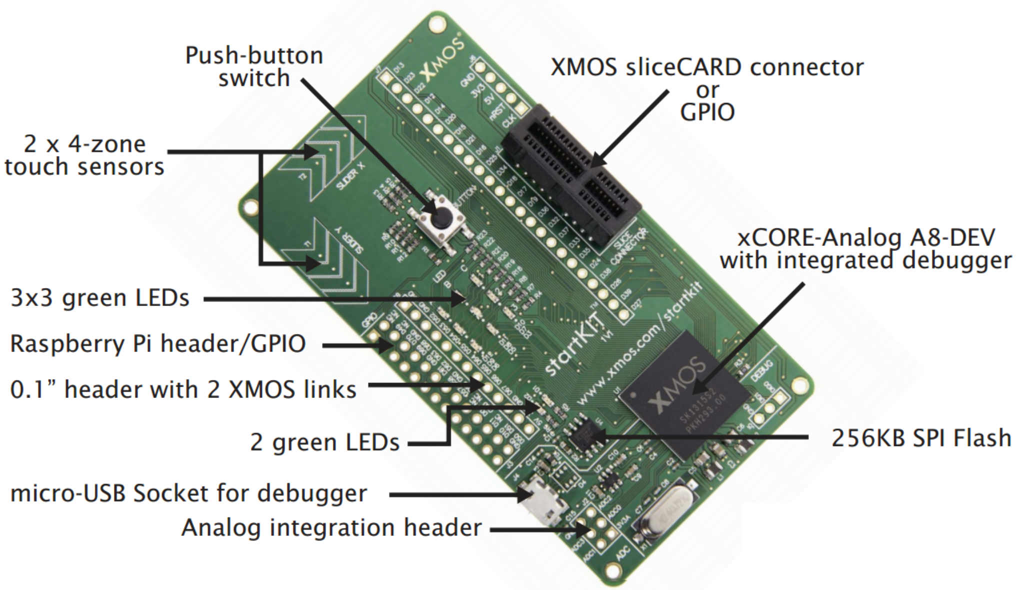
Motor control


Ethernet AVB

- startKIT
- sliceKIT
  - Modular development system
  - Up to four sliceCARDS for your exact configuration
- Hardware platforms with reference software
  - USB Audio
  - Motor Control
  - Ethernet AVB

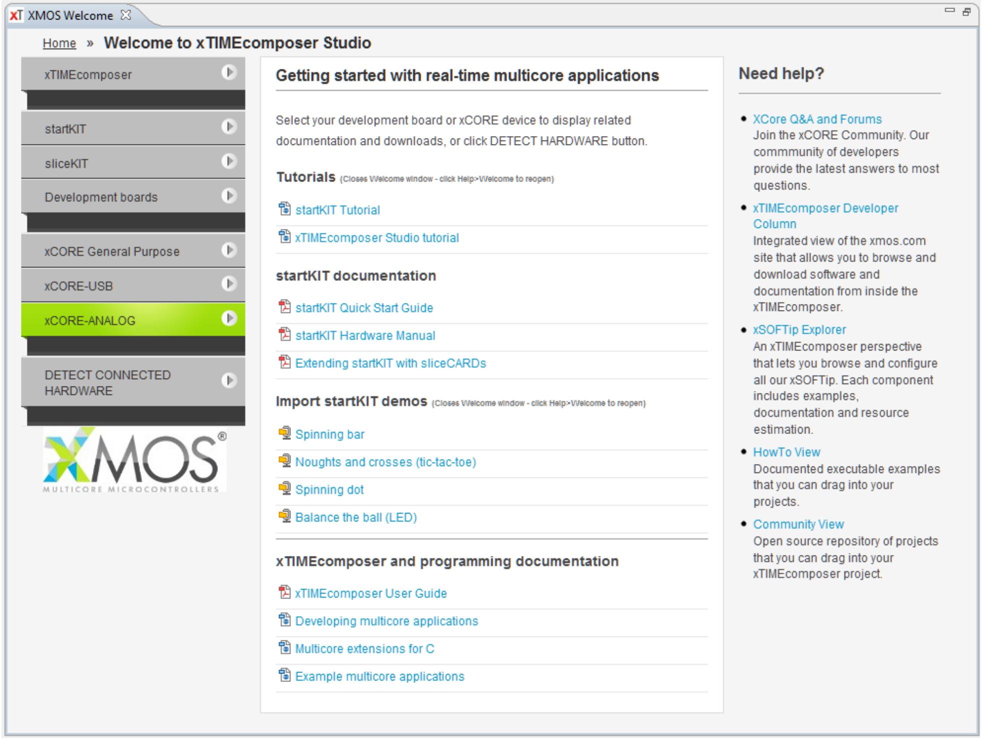
startKIT

- Single XS1-A8-DEV device
  - 500 MIPS / 8 cores available to user
  - 64KB RAM
  - 8KB OTP memory
- 256KB SPI FLASH for standalone operation
- Powered directly from the USB connector

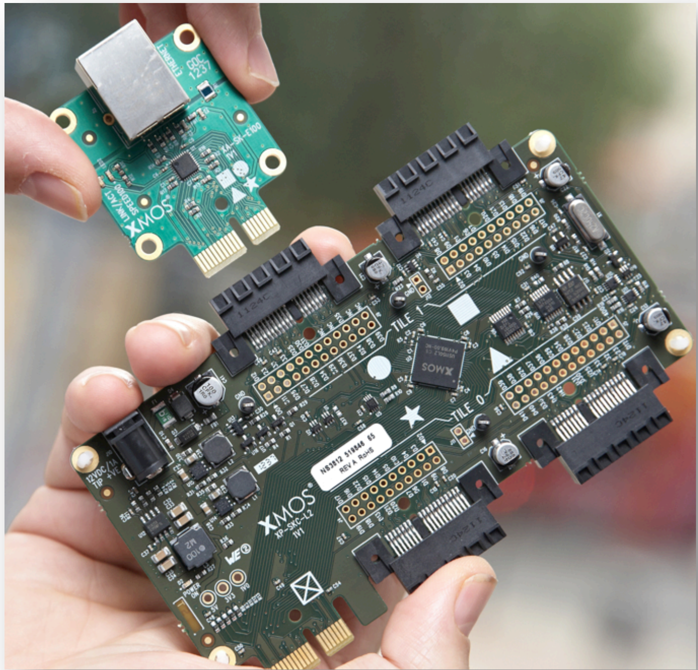


Programming your startKIT

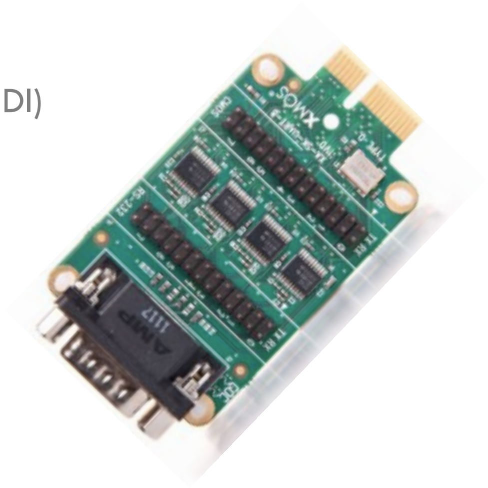
- **Download the startKIT documentation and examples**
  - http://www.xmos.com/startkit/started
- **Run xTIMEcomposer and run through the startKIT Tutorial**



sliceKIT



- Modular development hardware system
- Core boards with xCORE device:
  - XS1-L16, XS1-U16, XS1-A16 core boards
- Range of slices:
  - Ethernet
  - Multi-UART
  - LCD Display
  - GPIO
  - ADC
  - Audio (digital, analog & MIDI)
  - SDRAM
  - ISBUS (CAN, LIN, RS485)
  - WiFi
  - *Range expanding rapidly…*
  - *Third party slices*
    - *BroadR-reach*



Where to go next

- Download xTIMEcomposer and run the tutorials
  - https://www.xmos.com/support/downloads/xtimecomposer
- Purchase some hardware
  - sliceKIT – https://www.xmos.com/products/xkits/slicekit
  - startKIT – http://www.xmos.com/startkit
- Start developing
- HowTo Examples: http://www.xmos.com/support/documentation, or in the xTIMEcomposer 'HowTo Browser' view
- xCORE Community – http://www.xcore.com/
  - Online support
  - Question and Answer section
  - Large user community
  - Peripheral IP downloads
  - Software libraries and open-source code
  - Design Partner Program
- Github - https://github.com/xcore
  - Software libraries and open-source code

XMOS®
MULTICORE MICROCONTROLLERS

xCORE
xSOFTip
xTOOLS
xC
xKITS

THANK YOU!