

## 3 Describing Hardware Platforms

---

### In This Chapter

- ▶ Using XN files to describe hardware platforms
  - ▶ Modifying XN files in the XML editor
  - ▶ The XN hierarchy of elements
- 

XMOS development boards are described using XN. An XN file provides an XML description of the hardware platform, which includes components such as XMOS devices, named ports and links, external flash memory and an oscillator.

XMOS tools use the XN data to generate a platform-specific header file `<platform.h>`, and to compile, boot and debug multi-node systems. The tools support the following network topologies.

---

Network Topology	Supported Configurations
Line	XS1-L devices only, up to a maximum of 16 nodes
Hypercube	Degree-2 (pair of nodes) Degree-3 (ring of 4 nodes) Degree-3 (cube of 8 nodes) Degree-4 (canonical cube of 16 nodes)

---

### 3.1 An Example XN file

The XN file in Figure 10 describes a hardware platform consisting of two L1 devices arranged in a line. It is described below:

**Line 7** The board contains two XCores. The declaration “`core stdcore[2];`” is exported to the header file `<platform.h>`.

**L 11, 24** Nodes “M” and “S” are XS1-L1A devices in TQ128 packages. These devices are clocked by a 20MHz oscillator.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <Network xmlns="http://www.xmos.com"
3         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4         xsi:schemaLocation="http://www.xmos.com http://www.xmos.com">
5
6   <Declarations>
7     <Declaration>core stdcore[2]</Declaration>
8   </Declarations>
9
10  <Nodes>
11    <Node Id="M" Type="XS1-L1A-TQ128" Oscillator="20MHz"
12          SystemFrequency="400MHz" ReferenceFrequency="100MHz">
13      <Boot>
14        <Source Location="SPI:bootFlash"/>
15        <Bootee NodeId="S" Core="0"/>
16      </Boot>
17      <Core Number="0" Reference="stdcore[0]">
18        <Port Location="XS1_PORT_1A" Name="PORT_SPI_MISO"/>
19        <Port Location="XS1_PORT_1B" Name="PORT_SPI_SS"/>
20        <Port Location="XS1_PORT_1C" Name="PORT_SPI_CLK"/>
21        <Port Location="XS1_PORT_1D" Name="PORT_SPI_MOSI"/>
22        <Port Location="XS1_PORT_4A" Name="PORT_LED"/>
23      </Core>
24    </Node>
25    <Node Id="S" Type="XS1-L1A-TQ128" Oscillator="20Mhz">
26      <Boot>
27        <Source Location="XMOSLINK"/>
28      </Boot>
29      <Core Number="0" Reference="stdcore[1]">
30        <Port Location="XS1_PORT_1K" Name="PORT_BUTTON"/>
31      </Core>
32    </Node>
33  </Nodes>
34
35  <Links>
36    <Link Encoding="2wire" Delays="4,4">
37      <LinkEndpoint NodeId="M" Link="XOLD"/>
38      <LinkEndpoint NodeId="S" Link="XOLC"/>
39    </Link>
40  </Links>
41
42  <Devices>
43    <Device NodeId="M" Core="0" Class="SPIFlash" Name="bootFlash" Type="AT25FS010">
44      <Attribute Name="PORT_SPI_MISO" Value="PORT_SPI_MISO"/>
45      <Attribute Name="PORT_SPI_SS" Value="PORT_SPI_SS"/>
46      <Attribute Name="PORT_SPI_CLK" Value="PORT_SPI_CLK"/>
47      <Attribute Name="PORT_SPI_MOSI" Value="PORT_SPI_MOSI"/>
48    </Device>
49  </Devices>
50
51  <JTAGChain>
52    <JTAGDevice NodeId="M" Position="0"/>
53    <JTAGDevice NodeId="S" Position="1"/>
54  </JTAGChain>
55
56 </Network>

```

**Figure 10:**  
An example  
XN file.

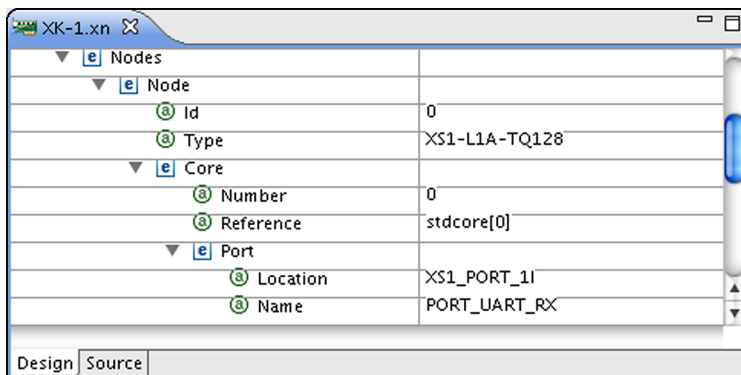
- L 13, 42** Node “M” is booted from an SPI device named “bootFlash” which has the class “SPIFlash”.
- L 14, 26** Node “M” boots Node “S” over an XMOS Link.
- L 16** Declares XCore 0 and associates it with stdcore[0].
- L 17-21** Provides symbolic names for ports. These declarations are exported to the header file <platform.h>.
- L 35-38** Defines a 2-wire XMOS link with intra- and inter-symbol delays of 4 clock periods, which connects Node “M” on link XOLD to Node “S” on link XOLC.

- L 41 Specifies a list of components on the board that are connected to XMOS devices.
- L 42-46 Declares a device named “bootFlash” which is connected to XCore 0 on Node “M”, and associates the four SPI pins with ports. The class “SPIFlash” is recognised by XFLASH.
- L 50-53 Specifies the JTAG scan chain.

## 3.2 Modifying XN files

When you create a new project, the XN file for your selected development board is added to the *Project Explorer*. If you add additional components to your board, or if you connect multiple boards together, you can extend the XN file to describe your new platform.

To modify the contents of the XN file, double-click on the file in the *Project Explorer* to open it in the XML editor, as shown in Figure 11.



**Figure 11:**  
The XN editor.

Expand the XML nodes to see the full description of your development board. The data is organised into the hierarchy given in Figure 12. For further details, see Appendix B.

Here are some other things you can do using the XN editor:

- ✓ **Add a new element:** Right-click on an element at the same level of the hierarchy you wish to add and choose **Add Before ► New Element...** or **Add After ► New Element...** from the contextual menu. A dialog box appears. Enter an element name and click **OK**.
- ✓ **Add an attribute to an existing element:** Right-click on an element and choose **Add Attribute ► New Attribute** from the contextual menu. A dialog box appears. Enter a name and value, then click **OK**.

Node	Number	Description	Section
Network	1	An XMOS network	
Declarations	0+		
Declaration	1+	XCore declaration	<a href="#">§B.1</a>
Nodes	1		
Node	1+	Node declaration	<a href="#">§B.2</a>
Core	1+	An XCore	<a href="#">§B.2.1</a>
Port	0+	An XCore symbolic port name	<a href="#">§B.2.2</a>
Boot	0 or 1	Boot method	<a href="#">§B.2.3</a>
Source	1	Binary location	<a href="#">§B.2.4</a>
Bootee	0+	Nodes booted	<a href="#">§B.2.5</a>
Links	0 or 1		
Link	1+	XMOS Link declaration	<a href="#">§B.3</a>
LinkEndpoint	2	XMOS Link endpoint	<a href="#">§B.3.1</a>
Package	0+	Chip package	<a href="#">§B.4</a>
Component	1+	Adds a nodes to the package	<a href="#">§B.4.1</a>
ExternalDevices	0 or 1		
ExternalDevice	1+	External device	<a href="#">§B.5</a>
Attribute	0+	A device attribute	<a href="#">§B.5.1</a>
JTAGChain	0 or 1		
JTAGDevice	1+	A device in the JTAG chain	<a href="#">§B.6</a>
JTAGSpace	0+	A space in the JTAG chain	<a href="#">§B.7</a>

**Figure 12:**  
The XN  
hierarchy of  
elements.

- ✓ **Add a child element:** Right-click on a variable and choose **Add Child ► New Element...** from the contextual menu. A dialog box appears. Enter an element name and click **OK**.
- ✓ **Delete an element:** Right-click on an element and choose **Remove** from the contextual menu.
- ✓ **View the source code:** Click the **Source** tab at the bottom-left of the editor window.
- ✓ **View the definitions available in <platform.h>.** Click the **Header** tab at the bottom-left of the editor window.

## B XN Elements

An XN file starts with the following XML declaration.

```
<?xml version="1.0" encoding="UTF-8"?>
```

The XMOS tools require a single network declaration.

```
<?xml version="1.0" encoding="UTF-8"?>
<Network xmlns="http://www.xmos.com"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.xmos.com http://www.xmos.com">
```

The network contains the hierarchy of elements described in [Figure 12](#).

### B.1 Declaration

A Declaration element declares a symbolic name for one or more XCores, which can be used in an XC program. Its body must have one of the following two forms:

- core *identifier*
- core *identifier* [ *constant-expression* ]

An XC declaration is exported to the header file `<platform.h>`. The name is also used by XMOS tools such as the debugger to refer to a particular XCore. A core variable is assigned to a physical XCore by a Core element (see [§B.2.1](#)).

#### Example

```
<Declaration>core master</Declaration>
```

```
<Declaration>core slave[8]</Declaration>
```

## B.2 Node

A Node element defines a set of XCores in the network, all of which are connected to a single switch.

Attribute	Required	Type	Description
Id	No	String	An identifier that names the node. All node identifiers must be unique in the network.
Type	Yes	String	The name of an XML file that describes the node. The tools search for the file <code>config_type.xml</code> in the path specified by <code>XCC_DEVICE_PATH</code> .
Oscillator	No	String	The PLL oscillator frequency, specified as an integer followed by either MHz, KHz or Hz. If omitted, no PLL programming is performed on boot.
SystemFrequency	Yes	String	A system frequency, specified as an integer followed by either MHz, KHz or Hz.
ReferenceFrequency	Yes	String	A reference clock frequency, specified as an integer followed by either MHz, KHz or Hz.

If the system and reference clock frequencies cannot be generated precisely, the tools issue a warning and report the actual frequencies achieved.



A network may contain either XS1-L devices or XS1-G devices, but not a mixture of both.

### Example

```
<Node Id="Master" Type="XS1-L1A-TQ128" Oscillator="20Mhz"
      SystemFrequency="410MHz" ReferenceFrequency="98.5Mhz">
```

The node named `Master` is described in the file `config_XS1-L1A-TQ128.xml`.

## B.2.1 Core

A `Core` element describes the properties of a single XCore.

Attribute	Required	Type	Description
Number	Yes	Integer	A unique number for the core in the node. Must be a value between 0 and $n-1$ where $n$ is the number of cores as defined in the node's XML file.
Reference	No	String	Associates the core with an identifier in a Declaration. A core may be associated with at most one identifier.

### Example

```
<Core Number="0" Reference="stdcore[0]" />
```

## B.2.2 Port

A `Port` element provides a symbolic name for a port.

Attribute	Required	Type	Description
Location	Yes	String	A port identifier defined in the standard header file <code>&lt;xs1.h&gt;</code> .
Name	Yes	String	A valid C preprocessor identifier. All port names declared in the network must be unique.

### Example

```
<Port Location="XS1_PORT_1I" Name="PORT_UART_TX" />
<Port Location="XS1_PORT_1J" Name="PORT_UART_RX" />
```

## B.2.3 Boot

A `Boot` element defines the how the node is booted. It contains one `Source` element and zero or more `Bootee` elements. If the source specifies an XMOS Link, no `Bootee` elements may be specified.



In a line of XS1-L devices, if the node at JTAG position  $x$  boots from SPI, the node at position  $m$  may boot the node at position  $n$  over an XMOS Link only if  $x$  is outside the range  $m..n$ .



The XMOS tools require a `Boot` element to be able to load bootable programs into flash memory (see §7.2).

### B.2.4 Source

A `Source` element specifies the location from which the node boots. It has the following attributes.

Attribute	Required	Type	Description
Location	Yes	String	Has the form <code>SPI:&lt;device-name&gt;</code> or <code>XMOSLINK</code> . The <code>device-name</code> must be declared in the set of <code>Device</code> elements.



Only XMOS XS1-L devices can be configured to boot over XMOS Links.

#### Example

```
<Source Location="SPI:bootFlash"/>
```

### B.2.5 Bootee

A `Bootee` element specifies another nodes in the system that this node boots (via an XMOS Link). If more than one XMOS Link is configured between this node and one if its bootees (see §B.3 and §B.3.1), the tools pick one to use for booting.

Attribute	Required	Type	Description
Name	Yes	String	A valid identifier for another node.

#### Example

```
<Bootee NodeId="S">
```

## B.3 Link

A `Link` element describes the characteristics of an XMOS Link, the details of which can be found in the XS1 system specification [1]. A `Link` element must contain exactly two `LinkEndpoint` children.

Attribute	Required	Type	Description
Encoding	Yes	String	Must be either <i>2wire</i> or <i>5wire</i> .
Delays	Yes	String	Of the form <i>x,y</i> where <i>x</i> specifies the inter delay value for the endpoint, and <i>y</i> specifies the intra delay value for the endpoint. If a value for <i>y</i> is omitted, <i>x</i> is used. If both values are omitted, 1,1 is used.

**Example**

```
<Link Encoding="2wire" Delays="4,4">
```

**B.3.1 LinkEndpoint**

A `LinkEndpoint` describes one end of an XMOS Link, the details of which can be found in the XS1 system specification [11]. Each endpoint associates a node identifier to a physical XMOS Link.

Attribute	Required	Type	Description
NodeID	Yes	String	A valid node identifier.
Link	Yes	String	Must be either an integer value from 0 to 15 inclusive, or a name in the form <b>XnLm</b> where <i>n</i> denotes a core number and <i>m</i> the link letter.

**Example**

```
<LinkEndpoint NodeId="0" Link="XOLD"/>
<LinkEndpoint NodeId="1" Link="XOLC"/>
```

**B.4 Package**

A `Package` element refers to a package file that describes the mapping from XCore ports and links to the pins on the package. It requires one or more `Component` children, which provide a mapping between nodes in the network and nodes in the package.

A package description enables the compiler to warn if multiple ports or links are mapped to the same pin. It also exports the the I/O pad ring, which is required to write testbenches that interface with the XMOS simulator (see §??).

Attribute	Required	Type	Description
Id	Yes	String	An identifier that names the package. All package identifiers must be unique in the network.
Type	Yes	String	The name of the XML package. The tools search for the file <code>type.pkg</code> in the path specified by <code>XCC_DEVICE_PATH</code> .

**Example**

```
<Package id="L2" Type="XS1-L2A-QF124">
```

The package named L2 is described in the file `XS1-L2A-QF64.xml`.

### B.4.1 Component

A `Component` element provides a mapping between a node described in the network and a node in a package. Nodes inside XMOS package files are named "0", "1", "2" and so on up the the number of nodes present.

Attribute	Required	Type	Description
NodeID	Yes	String	A valid node identifier in the network.
InPackage	Yes	String	A valid node identifier in the package file.

#### Example

```
<Package id="L2" Type="XS1-L2A-QF124">
  <Component NodeId="M" InPackage="0">
  <Component NodeId="S" InPackage="1">
</Package>
```

Node "M" in the network maps to node "0" in the package file and node "S" in the network maps to node "1" in the package file.

## B.5 ExternalDevice

An `ExternalDevice` element describes a device attached to an XCores that is not connected directly to an XMOS Link.

Attribute	Required	Type	Description
Name	Yes	String	An identifier that names the device.
Node	Yes	String	The identifier for the node that the device is connected to.
Core	Yes	Integer	The core in the node that the device is connected to.
Class	Yes	String	The class of the device.
Type	No	String	The type of the device.

The XMOS tools recognise the class "SPIFlash".

### B.5.1 Attribute

An `Attribute` element describes one aspect of an `ExternalDevice`.

Attribute	Required	Type	Description
Name	Yes	String	Specifies an attribute of the device.
Value	Yes	String	Specifies a value associated with the attribute.

The XMOS tools support the following attribute names for the device class “SPI-Flash”:

`PORT_SPI_MISO` SPI Master In Slave Out signal.  
`PORT_SPI_SS` SPI Slave Select signal.  
`PORT_SPI_CLK` SPI Clock signal.  
`PORT_SPI_MOSI` SPI Master Out Slave In signal.

Example :

```
<Attribute Name="PORT_SPI_MISO" Value="PORT_SPI_MISO"/>
```

## B.6 JTAGDevice

The XMOS tools require a JTAG scan chain to load and debug programs on hardware. The `JTAGChain` element describes a device in the JTAG chain.

Attribute	Required	Type	Description
NodeID	Yes	String	A valid node identifier.
Position	Yes	Integer	The position of the node in the JTAG chain.

**Example**

```
<JTAGDevice NodeId="M" Position="0"/>
<JTAGDevice NodeId="S" Position="1"/>
```

## B.7 JTAGSpace

The XMOS tools can bypass non-devices in the JTAG chain.

Attribute	Required	Type	Description
Position	Yes	Integer	The position of the space in the JTAG chain.
Devices	Yes	Integer	The number of devices represented by the space.
BitLength	Yes	Integer	The number of bits used in this space.

Thesupport spaces in the JTAG chain only at its start or end positions.