Connecting Multiple XDKs using XN

(VERSION 9.5)

2009/05/12

Authors:

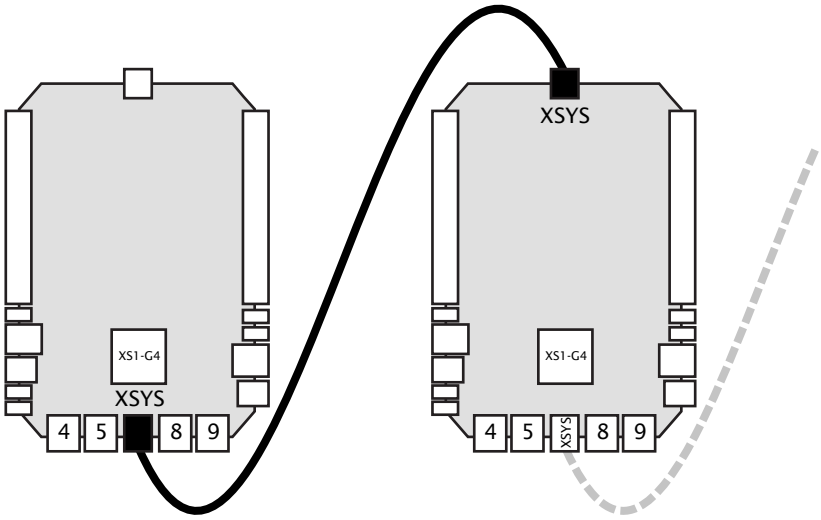
XMOS LTD.

1 Introduction

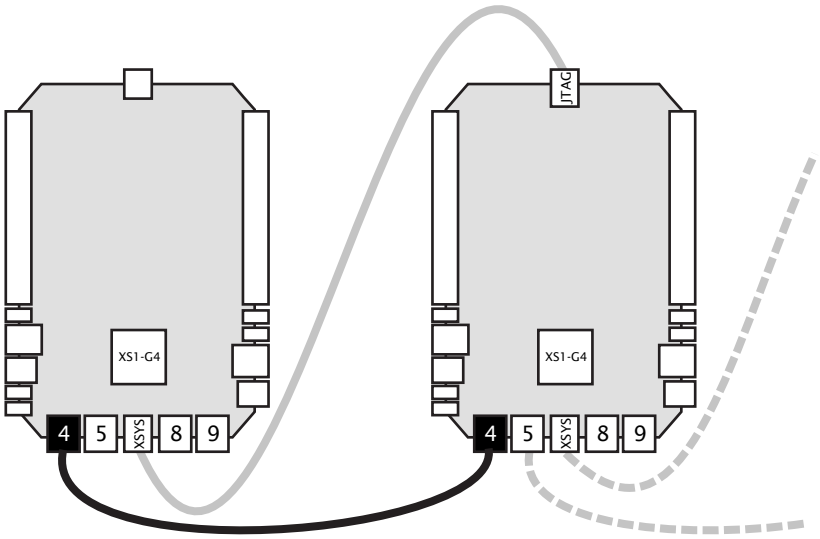
You can build up a network of XCores by connecting multiple XDKs together and producing an XN file that describes their connectivity in a format that the compiler toolchain can use.

The following example shows how to link two XDKs together and flash the LEDs on them in sequence.

1. Connect the XSYS UP socket on the bottom of the first XDK to the XSYS DOWN socket on top of the second XDK. This cable carries the JTAG signals used for booting and debugging multi-node designs.



2. Connect the two XDKs together using an XMOS Link cable, plugging the cable into the leftmost socket on the bottom of each device. Note that the leftmost sockets are identified as link 4, the other links left-to-right are 5, 8, 9. These numbers are used to identify the XMOS links when connecting one XDK to another.



3. Open a text editor and write an XN file called `xdk-2.xn` as follows:

```
<Network>

  <Declarations>
    <Declaration>core stdcore[8]</Declaration>
  </Declarations>

  <Nodes>
    <Node Number="0" Type="XS1-G4A-FB512">
      <Core Number="0" Reference="stdcore[0]"/>
        <Port Location="XS1_PORT_1E" Name="X0LEDA"/>
        <Port Location="XS1_PORT_1F" Name="X0LEDB"/>
        <Port Location="XS1_PORT_1G" Name="X0LEDC"/>
      <Core Number="1" Reference="stdcore[1]"/>
        <Port Location="XS1_PORT_1E" Name="X1LEDA"/>
        <Port Location="XS1_PORT_1F" Name="X1LEDB"/>
        <Port Location="XS1_PORT_1G" Name="X1LEDC"/>
      <Core Number="2" Reference="stdcore[2]"/>
        <Port Location="XS1_PORT_1E" Name="X2LEDA"/>
        <Port Location="XS1_PORT_1F" Name="X2LEDB"/>
```

```
      <Port Location="XS1_PORT_1G" Name="X2LEDC"/>
    <Core Number="3" Reference="stdcore[3]"/>
      <Port Location="XS1_PORT_1E" Name="X3LEDA"/>
      <Port Location="XS1_PORT_1F" Name="X3LEDB"/>
      <Port Location="XS1_PORT_1G" Name="X3LEDC"/>
  </Node>
  <Node Number="1" Type="XS1-G4A-FB512">
    <Core Number="0" Reference="stdcore[4]"/>
      <Port Location="XS1_PORT_1E" Name="X4LEDA"/>
      <Port Location="XS1_PORT_1F" Name="X4LEDB"/>
      <Port Location="XS1_PORT_1G" Name="X4LEDC"/>
    <Core Number="1" Reference="stdcore[5]"/>
      <Port Location="XS1_PORT_1E" Name="X5LEDA"/>
      <Port Location="XS1_PORT_1F" Name="X5LEDB"/>
      <Port Location="XS1_PORT_1G" Name="X5LEDC"/>
    <Core Number="2" Reference="stdcore[6]"/>
      <Port Location="XS1_PORT_1E" Name="X6LEDA"/>
      <Port Location="XS1_PORT_1F" Name="X6LEDB"/>
      <Port Location="XS1_PORT_1G" Name="X6LEDC"/>
    <Core Number="3" Reference="stdcore[7]"/>
      <Port Location="XS1_PORT_1E" Name="X7LEDA"/>
      <Port Location="XS1_PORT_1F" Name="X7LEDB"/>
      <Port Location="XS1_PORT_1G" Name="X7LEDC"/>
  </Node>
</Nodes>

<Links>
  <Link Encoding="2wire" Delays="15,15">
    <LinkEndpoint Node="0" Link="4"/>
    <LinkEndpoint Node="1" Link="4"/>
  </Link>
</Links>

<Boot>
  <JTAGChain>
     <JTAGDevice Node="0"/>
```

```
      <JTAGDevice Node="1"/>
    </JTAGChain>
    <SPIBoot Node="0"/>
  </Boot>

</Network>
```

This makes the eight cores available from the XC main function.

The ports are given user-friendly names (X0LEDA, X0LEDB, X1LEDA, X1LEDB and so on) to make it easier to relate to components connected to the XCores and provide easier portability between devices.

NOTE: Names must be unique in the XN file across all cores, and once defined the port ID is fixed to a particular core.

The Link tag specifies that Link 4 on the first XDK is connected to Link 4 on the second XDK.

The Boot tag specifies that cores 0 to 3 on the Node 0 (the XDK with the debug cable plugged into the bottom) are booted first, followed by cores 4 to 7 on Node 1.

The chain can be extended to include as many nodes and cores are required.

2 Using XN files

To use the definitions in the file you need to generate a header file that can be included in your XC/C source files. Use the command line tool XNC to create the XN file. Note that in the initial release of XN the XNC tool is not included in the tools path so you must run it from the *libexec* directory.

1. Pass the `xdk-2.xn` file to the XNC compiler and run:

   ```
   XMOS_9.5.0\libexc\xnc -H  flash-led.xn
   ```

2. Copy the output `xdk-2.h` file into the *XMOS_9.5.0\target\include* directory.

3. Write an XC file that includes the `xdk-2.h` file and uses the core identifiers defined in the XN file. For example the following program flashes the 24 LEDS on the two XDKs in a circular sequence:

```
#include <xdk-2.h>

#define TOKEN  1
#define PERIOD 20000000

#define CIRCLE(i) i==0  ? 0 : \
                  i==1  ? 0 : \
                  i==2  ? 0 : \
                  i==3  ? 3 : \
                  i==4  ? 3 : \
                  i==5  ? 3 : \
                  i==6  ? 4 : \
                  i==7  ? 4 : \
                  i==8  ? 4 : \
                  i==9  ? 7 : \
                  i==10 ? 7 : \
                  i==11 ? 7 : \
                  i==12 ? 6 : \
                  i==13 ? 6 : \
                  i==14 ? 6 : \
                  i==15 ? 5 : \
                  i==16 ? 5 : \
                  i==17 ? 5 : \
                  i==18 ? 2 : \
                  i==19 ? 2 : \
                  i==20 ? 2 : \
                  i==21 ? 1 : \
                  i==22 ? 1 : \
                  i==23 ? 1 : 1000

out port leds[24] = { on stdcore[0] : X0LEDA,
                      on stdcore[0] : X0LEDB,
```

```
                        on stdcore[0] : X0LEDC,
                        on stdcore[3] : X3LEDA,
                        on stdcore[3] : X3LEDB,
                        on stdcore[3] : X3LEDC,
                        on stdcore[4] : X4LEDA,
                        on stdcore[4] : X4LEDB,
                        on stdcore[4] : X4LEDC
                        on stdcore[7] : X7LEDA,
                        on stdcore[7] : X7LEDB,
                        on stdcore[7] : X7LEDC,
                        on stdcore[6] : X6LEDA,
                        on stdcore[6] : X6LEDB,
                        on stdcore[6] : X6LEDC,
                        on stdcore[5] : X5LEDA,
                        on stdcore[5] : X5LEDB,
                        on stdcore[5] : X5LEDC,
                        on stdcore[2] : X2LEDA,
                        on stdcore[2] : X2LEDB,
                        on stdcore[2] : X2LEDC,
                        on stdcore[1] : X1LEDA,
                        on stdcore[1] : X1LEDB,
                        on stdcore[1] : X1LEDC,
};

void flashLED(chanend left, chanend right,
  out port led, int period, int m) {
  timer tmr;
  unsigned t;
  if (m)
    right <: TOKEN;
  while (1) {
    left :> void;
    led <: 0;
    tmr :> t;
    tmr when timerafter(t+period) :> void;
    led <: 1;
```

```
    right <: TOKEN;
  }
  return;
}

int main(void) {
  chan c[24];

  par (int i=0; i<24; i++) {
    on stdcore[RING(i)] : flashLED(c[i],
      c[(i+1)%24], leds[i], PERIOD, i==0);
  }
  return 0;
}
```

The replicator creates a field of 24 separate threads. The macro CIRCLE provides a mapping between replicator indices and the core on which the corresponding LED is illuminated.