# xCORE-200 Clock Frequency Control

IN THIS DOCUMENT



## 1 PLL and Clock Divider Overview

A low frequency external clock is used to drive the internal phase locked loop (PLL) of xCORE-200 devices and obtain the system clock. A number of system clock dividers are then used on the system clock to derive the clocks for the xCORE tiles, the RGMII unit, the switch and the reference clock.

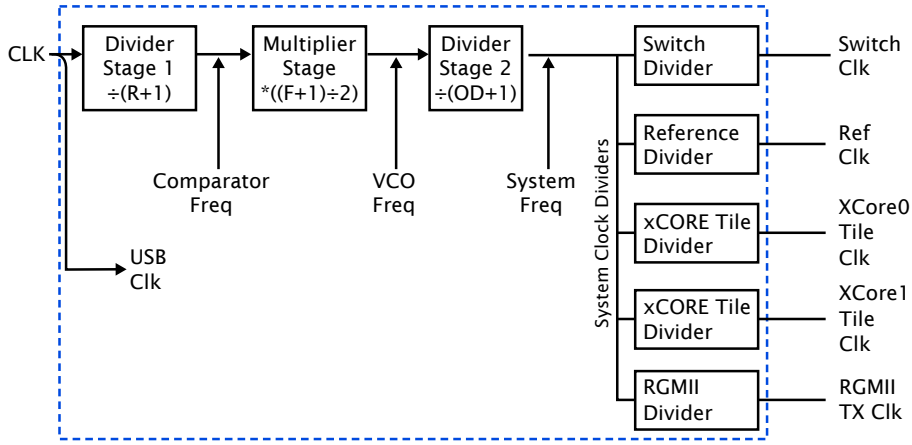


**Figure 1:** PLL and Clock Dividers

The PLL's initial settings are determined by the state of any mode pins on the xCORE-200 device. The standard configuration allows a 25MHz external clock to be used to operate the xCORE tiles and the switch at 400MHz, and the reference


Publication Date: 2016/10/3
Document Number: XM010761A

clock at 100MHz. In many applications this configuration will be selected, requiring no reprogramming of the PLL or dividers. If the application requires a different input frequency or system frequency then the PLL must be reprogrammed. The xTIMEcomposer tools can be used to reprogram the PLL automatically by specifying the application's configuration in the XN file.

## 2 Constraints

There are a number of constraints on the frequencies of clocks at different points on the xCORE-200 devices. These constraints must be met for the initial boot sequence, and if the PLL is reprogrammed, for the reprogrammed values too.

**Figure 2:** Clock Frequency Constraints

| Clock | Constraint |
|---|---|
| CLK | 4.22–100.0 MHz |
| VCO frequency | 260–1300 MHz |
| System clock | Maximum operating frequency—see device datasheet |
| Switch clock | System clock maximum |
| Reference clock | System clock maximum |
| xCORE Tile tiles clock | System clock maximum |
| RGMII clock | System clock maximum |
| USB clock | 12 or 24 MHz |

## 3 PLL Settings

There are three dividers within the PLL. R divides the input clock down. The next divider divides the output of the voltage controlled oscillator (VCO) stage down to the same frequency as the output of the R divider. Therefore this divider sets the multiplication factor (F) of the PLL. The OD divider divides the output clock of the VCO.

There is a constraint on the frequency of the clock at the comparator—the output of R. There is another constraint placed on the output of the VCO.

## 4 Configuring the xCORE-200 Device

Some packages have mode pins that are used to determine the initial PLL settings used after reset. This configuration must be such that all of the constraints are met for the input clock driven onto CLK.

**Figure 3:** Mode Pins and Boot Configuration

| CLK Range (MHz) | Mode 1 | Mode 0 | XCore Clock (MHz) | Multiplier | OD | F | R |
|---|---|---|---|---|---|---|---|
| 3.25-10 MHz | 0 | 0 | 130-400 MHz | 40 | 1 | 159 | 0 |
| 9-25 MHz | 1 | 1 | 144-400 MHz | 16 | 1 | 63 | 0 |
| 25-50 MHz | 1 | 0 | 167-400 MHz | 8 | 1 | 31 | 0 |
| 50-100 MHz | 0 | 1 | 196-400 MHz | 4 | 1 | 15 | 0 |

If a different PLL configuration is required from that used to boot the application, the new settings should be written to the PLL_CTRL register. The PLL_CTRL register comprises five fields (R, F, OD, LOCKN, RESETN), detailed in §5. That register contains a bit to instruct the PLL to hold the chip in reset, and a bit to pause the chip whilst the PLL is not locked.

Small changes to either R or F that result in a frequency change of no more than +/- 20% can be made by writing a new value to the register with a '1' in the RESETN bit and a '1' in the LOCKN bit (0xCnnn nnnn). The PLL will gradually adjust to the new values without either the need for a reset or a lock.

Larger changes to R or F, or changes that require both R and F to be modified, or changes to OD can be made by writing a '0' in the RESETN bit and a '0' in the LOCKN bit (0x0nnn nnnn).

If you choose to reset the device (by setting the RESETN bit low), the boot code should read the value of the PLL_CTRL register and compare it to the reconfigured value. If there is a difference, then this is the first time the boot code has executed and the new PLL settings should be written to PLL_CTRL, causing a reset. The second time the boot code executes, the value read back from the PLL_CTRL register will be the reconfigured value and the boot process can continue.

The easiest way to reprogram the PLL is to specify the application's frequency requirements in the XN file and use the xTIMEcomposer tools to reprogram the PLL—see §8.1.

# 5 Frequency Control Registers

To access the frequency control registers packets of data must be constructed and communicated to the Switch through a channel end. Global PLL settings are controlled through registers in the Node Configuration control registers. From C or XC, use the *write_node_config_reg()* and *read_node_config_reg()* functions defined in *xs1.h*. The bits that can be controlled are shown in Figure 4.

Settings on an individual tile basis are controlled through registers in the Tile Configuration control registes. From C or XC, use the *write_tile_config_reg()* and *read_tile_config_reg()* functions defined in *xs1.h*. The bits that can be controlled are shown in Figure 5.

# 6 Example PLL Configurations

## 6.1 Standard Configuration: 25MHz Oscillator

Use **MODE[1:0] = 11**, ie, leave any mode pins Not Connected. The PLL will configure to the standard 400MHz, with the xCORE tile and SSwitch running at 400MHz, with a 100MHz reference clock.

| Register | Bitfield | Reset | Description |
|---|---|---|---|
| XS1_SSWITCH_PLL_CTL_NUM | [6:0] | Mode Pins | R; PLL input divider stage = R+1 |
| XS1_SSWITCH_PLL_CTL_NUM | [20:8] | Mode Pins | F; Multiplier stage of the PLL = (F+1)/2 |
| XS1_SSWITCH_PLL_CTL_NUM | [25:23] | Mode Pins | OD; PLL output divider stage = OD+1 |
| XS1_SSWITCH_PLL_CTL_NUM | 30 | N/A | LOCKN; '0' will force a wait for PLL lock |
| XS1_SSWITCH_PLL_CTL_NUM | 31 | N/A | RESETN; '0' will force reset on PLL change |
| XS1_SSWITCH_CLK_DIVIDER_NUM | [15:0] | 0 | System switch clock divider = SSDIV+1. Reset value produces 400MHz for a 400MHz system clock |
| XS1_SSWITCH_REF_CLK_DIVIDER_NUM | [15:0] | 3 | Reference clock divider = REFDIV+1. Reset value produces 100MHz for a 400MHz system clock. |

**Figure 4:** Node Configuration Registers

| Register | Bitfield | Reset | Description |
|---|---|---|---|
| XS1_PSWITCH_PLL_CLK_DIVIDER_NUM | [15:0] | 0 | xCORE Tile clock divider = XCDIV+1. Reset value produces 400MHz for an 400MHz system clock |

**Figure 5:** Tile Configuration control registers

## 6.2 24MHz Oscillator

Use **MODE[1:0] = 11**, ie, leave any mode pins Not Connected. For the initial boot, the system clock will be 100.0MHz, with the xCORE tile also running at 384 MHz. The following are required: **R** = 0, **F** = 124, **OD** = 2. Write **0xC1007C00** to the PLL Settings register in the Node Configuration to bring the PLL output up to 500MHz, with code similar to the following:

```
#define PLL_500MHz 0x01007C00

...
  unsigned pllCtrlReadData;
  read_node_config_reg(tile[0], XS1_SSWITCH_PLL_CTL_NUM, pllCtrlReadData);
  if (pllCtrlReadData != PLL_500MHz) {
    write_node_config_reg(tile[0], XS1_SSWITCH_PLL_CTL_NUM, PLL_500MHz);
  }
...
```

# 7 Example System Clock Divider Configurations

## 7.1 133MHz Reference Clock

To adjust the Reference Clock to 133MHz with a 400MHz System Clock, set **REFDIV** to **2** using the following code:

```
write_node_config_reg(tile[0], XS1_SSWITCH_REF_CLK_DIVIDER_NUM, 0x02);
```

This will adjust all timers and clock-blocks to run at 133.3MHz, and allow ports to be configured at 66MHz, 33MHz and so on.

## 7.2 Slow Switch Clock

For applications where only a single xCORE-200 device is used, the SSwitch is only used for configuration purposes. Once the system is configured, the SSwitch clock can be substantially reduced to save on dynamic power. 1MHz is a good option for a low power SSwitch clock because the SSwitch power is dominated by the static power at this frequency.

To reduce the SSwitch clock to 1MHz with a system clock of 400MHz, set **SSDIV** to **399** using the following code:

```
write_node_config_reg(tile[0], XS1_SSWITCH_CLK_DIVIDER_NUM, 399);
```

## 7.3 xCORE Tile Clock 200MHz

If your application does not need to run the xCORE tile at full speed to work, dynamic power can be saved by running the tile at a slower rate. For this to work, each tile has its own clock divider that is enabled by setting a bit in the Processor Status Configuration

To run a tile at 200MHz from a system frequency of 400MHz, set **XCDIV** to **1** and enable the clock divider for this processor by writing **0x10** to **XCORE_CTRL0**:

```
write_tile_config_reg(tile[0], XS1_PSWITCH_PLL_CLK_DIVIDER_NUM, 1);
setps(XS1_PS_XCORE_CTRL0,0x10);
```

# 8 Configuring the Clock System Through the XN File

The PLL and the reference clock frequency can be programmed automatically for an application by using the xTIMEcomposer tools. The application's input oscillator frequency, system frequency and reference frequency can be specified in the XN file. When the application code is written to a flash device with XFLASH, the code to reprogram the PLL to the desired system and reference frequencies will be added. When run with XRUN or XGDB the PLL is reprogrammed via JTAG.

The frequency control attributes should be added to the Node node within the XN file. Frequencies should be specified with their unit of MHz, kHz or Hz, (for

| Attribute | Description | Default Value |
|---|---|---|
| Oscillator | Input frequency on the CLK pin. If this attribute is specified, the system frequency and the reference frequency are programmed using their specified (or default) values. If this attribute is not specified, the boot configuration for the system and reference frequencies are used for the application. | Uses boot configuration |
| SystemFrequency | The desired system frequency. The *Oscillator* attribute must be specified if this attribute is specified. | 400MHz |
| ReferenceFrequency | The desired reference frequency. The *Oscillator* attribute must be specified if this attribute is specified. | 100MHz |

**Figure 6:** XN File Frequency Control Attributes

example 500MHz, 24576kHz or 6745800Hz). If the frequency control attributes are not specified in the XN file, then the xTIMEcomposer tools will not modify the frequency control registers.

If the target frequency specified in the XN file for either the system or reference frequency cannot be met exactly for the application's input frequency, a frequency close to the target frequency will be selected by the tools and a warning will be issued. XFLASH always issues the warning when it occurs, as does XGDB. XRUN only issues the warning if it has been run with the `--verbose` switch. XGDB issues the warning when the `connect` command is issued. Within xTIMEcomposer Studio, the XFLASH warning is issued to the *Console*, but the XGDB or XRUN warning is not available to the user.

## 8.1 Example XN file using Frequency Control Attributes

```
<?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">
  <Type>Board</Type>
  <Declarations>
    <Declaration>tileref tile[1]</Declaration>
  </Declarations>
  <Packages>
    <Package id="0" Type="XS2-UEnA-512-TQ128">
      <Nodes>
        <Node Id="0" InPackageId="0" Type="XS2-L16A-512" Oscillator="20MHz"
SystemFrequency="500MHz">
          <Tile Number="0" Reference="tile[0]"/>
          <Tile Number="1" Reference="tile[1]"/>
        </Node>
      </Nodes>
    </Package>
  </Packages>
  <JTAGChain>
    <JTAGDevice NodeId="0"/>
  </JTAGChain>
</Network>
```

for more information on XN files see the xTIMEcomposer user guide.

# 9 Document History

| Date | Release | Comment |
|---|---|---|
| 2016-09-30 | 1.0 | First release |