Arria 10 CLKUSR Guideline for Transceiver Calibration

Every Arria 10 FPGA device has a dedicated CLKUSR pin. CLKUSR is not new to Altera FPGAs. For example, Stratix V has a dedicated CLKUSR pin for FPGA configuration and initialization. It can also be used to synchronize multiple Stratix V FPGAs. Arria 10 adds a new feature to CLKUSR. When transceivers are used in an application, the CLKUSR provides a dedicated clock source for Arria 10 transceiver calibration. This application note focuses on how to use CLKUSR for transceiver calibration.

According to the Arria 10 datasheet, the maximum clock frequency supported by CLKUSR on transceiver calibration is 125MHz. In order for all transceivers to properly calibrate, Arria 10 transceivers have the following clock requirements:

- CLKUSR must be running and stable at the start of FPGA programming
- Transceiver reference clock driving PLLs (ATXPLL, fPLL, CDR/CMU PLL) must be running and stable at the start of FPGA programming

Transceiver calibration can start before the FPGA enters User Mode. Depending on the number of transceiver channels used, the transceiver calibration process can extend after user mode. System designers or board designers need to make sure both CLKUSR and reference clocks are present when FPGA programming starts. There are special use cases as below when CLKUSR or the transceiver reference clock is not ready before the FPGA enters User Mode:

- Missing CLKUSR
- Missing reference clock
- CLKUSR comes after reference clock
- Reference clock comes after CLKUSR

This application note will cover these special scenarios. When a system can't provide a clock source to CLKUSR, this paper will explain how to bring the FPGA fabric clock to CLKUSR for transceiver calibration. Please use this application note in conjunction with the following literatures:

Arria® 10 GX, GT, and SX Device Family Pin Connection Guidelines

Arria 10 device datasheet

Arria 10 Transceiver PHY Overview

Special Scenarios

1. Missing CLKUSR at start of FPGA programming

Power up calibration won't be processed without CLKUSR. If cal_busy outputs are observed to be high all the time without being triggered low after the FPGA has entered user mode, it signals a stalled CLKUSR that has not toggled once since power up. The transceiver toolkit can't find any linked transceiver channels when CLKUSR has been flat since system power up.

2. Missing transceiver reference clock

When the transceiver reference clock is missing, calibration won't start if the system contains a PCIe application. The cal_busy outputs will remain high until the PCIe reference clock arrives.

If the user application does not have a PCIe module, all channels will be calibrated without a reference clock as long as CLKUSR is active. At this time, when using the transceiver toolkit to monitor channels, CDR won't show lock and BER will be about 10E-2. The calibration result won't be accurate without a reference clock. It requires a user recalibration when the reference clock is alive and stable.

3. CLKUSR comes after transceiver reference clock

Power up calibration won't start without CLKUSR even if the reference clock is active and stable. When CLKUSR comes late, CLKUSR is expected to be stable and glitch free when it startssince power up calibration starts right after CLKUSR toggles. If the reference clock is stable when CLKUSR starts to toggle, the calibration result is the same as normal power up calibration. If the reference clock is not stable when CLKUSR starts to toggle, calibration may not deliver the best result, especially PLL calibration. It requires a user recalibration.

4. Transceiver reference clock comes after CLKUSR

If the user application has a PCIe channel and the reference clock comes late after CLKUSR, calibration won't start until reference clock is active, so the reference clock must be stable and glitch free when it starts. If the user application does not have a PCIe module, all active channels will be calibrated without the reference clock. If the reference clock is late and not stable at the calibration stage, it is recommended to gate CLKUSR until the reference clock is stable and locked so the calibration result will be the same as normal power up calibration.

Bring clock from FPGA Fabric to CLKUSR

CLKUSR is a configurable bidirectional GPIO pin which provides a dedicated clock source for the Arria 10 Nios calibration engine and Aux calibration circuits. In a normal application which uses transceivers, the user design does not have to define a CLKUSR input. Quartus can automatically figure out a system clock for transceiver calibration and place CLKUSR to the right pin location. The user only needs to provide a stable clock source to CLKUSR from the board without defining anything for CLKUSR in design.

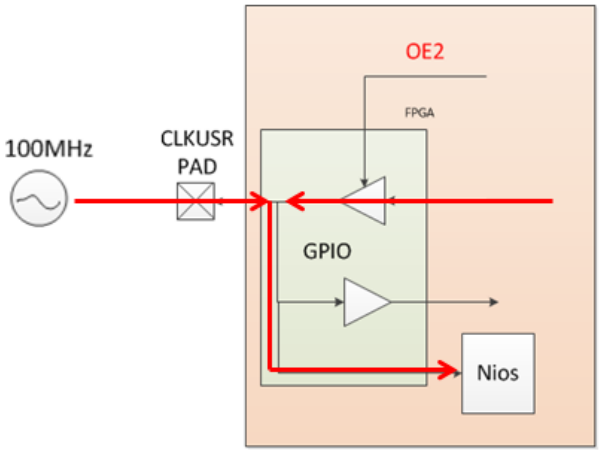

Figure 1. CLKUSR is a configurable GPIO pin which directly drive Nios calibration engine

Since CLKUSR is also a regular bidirectional GPIO pin, the user can define an input, or an output, or a bidirectional inout in the user design, and assign it to the dedicated CLKUSR pin location. The nature of CLKUSR allows a clock to be routed to Nios from the FPGA fabric. It can be used for applications which have no additional clock source from the board to drive CLKUSR. This application note gives two examples on how to configure CLKUSR as a bidirectional GPIO and bring a fabric clock for calibration:

- A design using a register and an IOPLL
- A design using a clock control and an IOPLL

As shown in Figure 1, if the user system provides a 100MHz clock from the board and also assigns CLKUSR as an output GPIO pin with a clock coming from fabric, the calibration clock would be multi-driven by two clock sources. The user design should avoid multi-driven clock for Nios calibration.

1. Use a register and an IOPLL to provide clock to CLKUSR

The following items need to be added in the user design in order to bring a fabric clock to CLKUSR using a register and an IOPLL:

a) Generate a GPIO IP which has the external pad set as bidirectional and assigned to the CLKUSR location
b) Generate an IOPLL IP which has the input connected to a transceiver reference clock and the output set below 250MHz
c) Use a register to divide IOPLL clock output by 2 and send to the GPIO input
d) No external clock source driving the CLKUSR pad

Figure 2 gives an example on how to use a register and an IOPLL to provide a clock source to CLKUSR. The external 100MHz clock source can come from the transceiver reference clock. An IOPLL uses this clock to generate a 200MHz clock which feeds a register that divides it down to a 100MHz calibration clock. The OE2 switch must be open to let the 100MHz clock passto Nios.

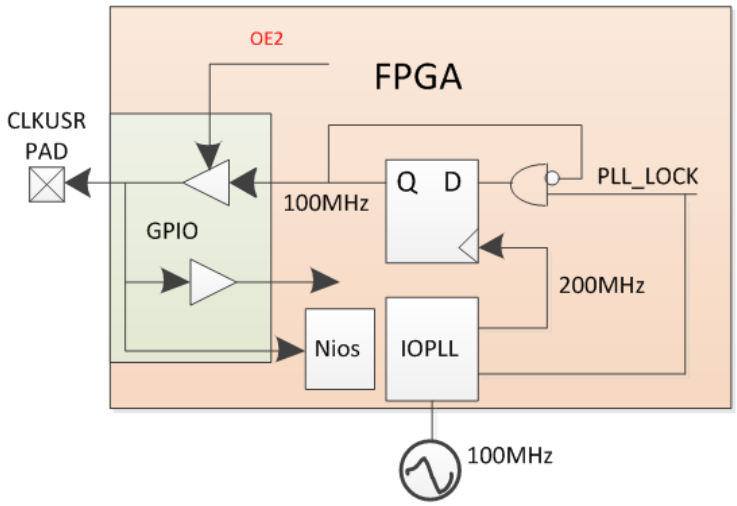

Figure 2. Use a register and an IOPLL to generate a clock for CLKUSR

2. Use a clock control and an IOPLL to provide clock to CLKUSR

The following items need to be added in the user design in order to bring a fabric clock to CLKUSR using a clock mux and an IOPLL:

a) Generate a GPIO IP which has the external pad set as bidirectional and assigned to the CLKUSR location
b) Generate an IOPLL IP which has the input connected to a transceiver reference clock and output set below 125MHz
c) Generate a clock control megafunction block to receive the clock from IOPLL and send to the GPIO input. Use IOPLL lock to gate the clock until locked
d) No external clock source driving the CLKUSR pad

Figure 3 gives an example on how to use a clock control block and an IOPLL to provide a clock source to CLKUSR. The external 100MHz clock source can come from the transceiver reference clock. It is recommended to use the IOPLL pll_lock signal to block the clock until the IOPLL is locked. The OE2 switch must be open to let the 100MHz clock pass to Nios.

The function of IOPLLs, registers, and clock control blocks won't be active until after the FPGA is in user mode. So bringing the FPGA fabric clock to CLKUSR can only be used for applications which have no

transceiver calibration requirement to start before user mode. PCIe autonomous and Cvp applications require calibration to start before user mode, so they can't use fabric clock for calibration.

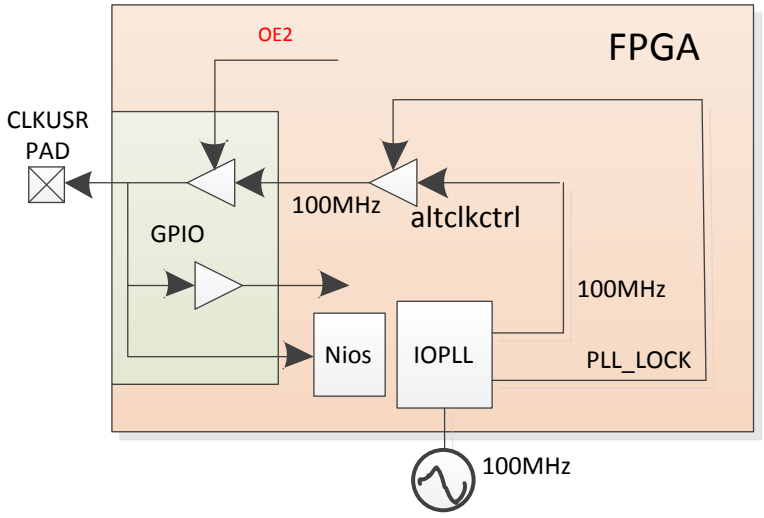

Figure 3. Use a clock mux and an IOPLL to generate a clock for CLKUSR

Since Quartus II 15.1 B128, the CLKUSR pin is reserved whenever there are transceivers in the design or certain IOs are used that require PMA AUX for calibration. CLKUSR is not allowed to be assigned by the user application. If the user tries to add location constraints to it, Quartus will generate an error. The .pin file will have the CLKUSR pin properly named when reserved. For example, it will contain something as follows:

```
~ALTERA_CLKUSR~ / RESERVED_INPUT : BD32 : input : 1.8 V : : 2A : N
```

The two design examples which bring the fabric clock source to CLKUSR internally require a bi-directional pin being defined and assigned to CLKUSR by the user application. There are two ways to bypass the Quartus reservation on CLKUSR from erroring out during compilation. One is to use quartus.ini, the other is to use QSF assignment.

Use quartus.ini , add the below line in quartus.ini:

arria10_reserve_CLKUSR_pin_for_transciever_calibration = false

Use QSF assignment, add the below line in QSF file.

set_global_assignment -name AUTO_RESERVE_CLKUSR_FOR_CALIBRATION OFF