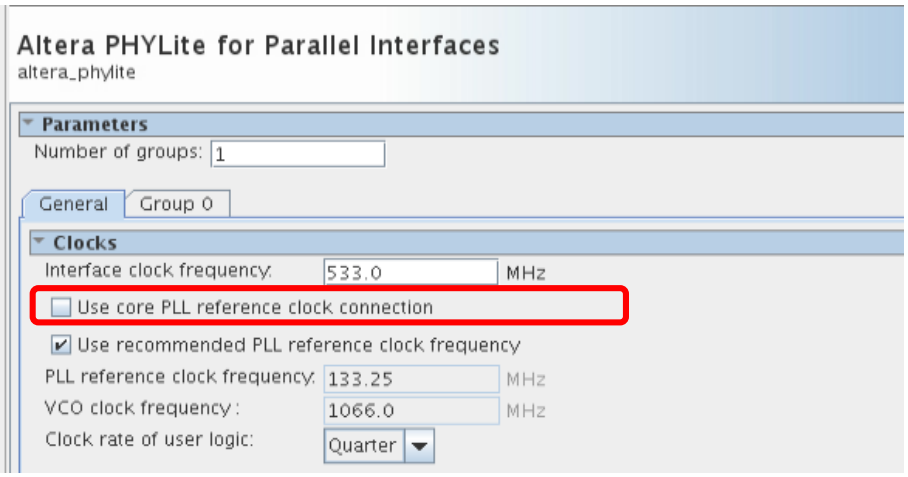
Applying Clock Uncertainty for Cascaded PLL or non-dedicated clock pin for Arria 10 Altera PHYLite IP

Starting with Quartus Prime software version 17.0, the "Use core PLL reference clock connection" is no longer visible in the Arria® 10 Altera PHYLite IP parameter editor. The recommended way is to use dedicated clock pin to connect it to PHYLite IOPLL reference clock. If your design requires to use this feature, please read the following guidelines carefully and abide to them.

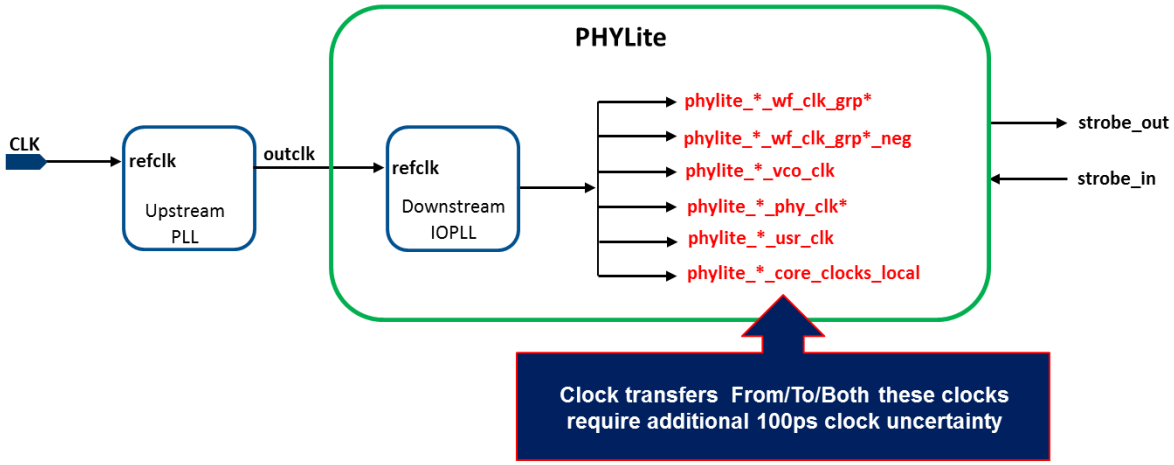
To enable the "Use core PLL reference clock connection" (Quartus Prime software version 17.0 and later), please add the below INI in the quartus.ini file.

ip_altera_phylite_en_pll_core_ref_ck = on



When connecting PLL reference clock from a PLL output or a non-dedicated clock pin, additional jitter will be introduced.

For design implemented with Arria® 10 Altera PHYLite version 17.0 and prior, this jitter can be compensated by adding 100ps clock uncertainty constraint at the **output clocks of the downstream PLL**



in the design.

Following are the steps to apply the clock uncertainty in the PHYLite generated SDC file. You will need to make the following changes every time you regenerate the Altera PHYLite IP.

1. Add 100ps clock uncertainty on the write path:

```
if {[llength $write_clocks]>0} {
        # We don't need derive_clock_uncertainty numbers because we have include FLS JITTER in set_output_delay
        #old:  set_clock_uncertainty -to [get_clocks $write_clocks] 0.0
        #new constraint:
         set_clock_uncertainty -add -from [get_clocks ${inst}_wf_clk_grp_${i_grp_idx}_*] -to [get_clocks $write_clocks] 0.1
      }
```

2. No change on the read_clocks

```
if {[llength $read_clocks]>0} {

    # We don't need derive_clock_uncertainty numbers because we have include FLS JITTER in set_input_delay

    set_clock_uncertainty -to [get_clocks $read_clocks] 0.0

 }
```

3. Add 100ps clock uncertainty on c2p/p2c transfers (phy_clk <->usr_clock)

```
if {$i_phy_clock > $same_tile_index} {
            # C2P/P2C where the periphery tile != CPA tile.
            # For these transfers the SDC explicitly overrides the clock uncertainty values.
            # Therefore, when overconstraining we must not use the "-add" option.
            set add_to_derived ""
            set c2p_su      [expr {$p2c_c2p_multi_tile_clock_uncertainty + [lindex $periphery_overconstraints 0] + [lindex $periphery_clock_uncertainty 0]} + 0.1]
            set c2p_h       [expr {$p2c_c2p_multi_tile_clock_uncertainty + [lindex $periphery_overconstraints 1] + [lindex $periphery_clock_uncertainty 1]} + 0.1]
            set p2c_su      [expr {$p2c_c2p_multi_tile_clock_uncertainty + [lindex $periphery_overconstraints 2] + [lindex $periphery_clock_uncertainty 2]} + 0.1]
            set p2c_h       [expr {$p2c_c2p_multi_tile_clock_uncertainty + [lindex $periphery_overconstraints 3] + [lindex $periphery_clock_uncertainty 3]} + 0.1]
} else {
 # C2P/P2C where the periphery tile == CPA tile
            # For these transfers it is safe to use the -add option since we rely on
            # derive_clock_uncertainty for the base value.
```

```
        set add_to_derived "-add"
        set c2p_su      [expr [lindex $periphery_overconstraints 0] + [lindex
$periphery_clock_uncertainty 0] + 0.1]
        set c2p_h       [expr [lindex $periphery_overconstraints 1] + [lindex
$periphery_clock_uncertainty 1] + 0.1]
        set p2c_su      [expr [lindex $periphery_overconstraints 2] + [lindex
$periphery_clock_uncertainty 2] + 0.1]
        set p2c_h       [expr [lindex $periphery_overconstraints 3] + [lindex
$periphery_clock_uncertainty 3] + 0.1]
    }
```

4. Add 100ps cu within core transfer (usr_clock/extra core clock <->usr_clock/extra core clock)

```
    set c2c_same_su      [expr [lindex $core_overconstraints 0] + [lindex $core_clock_uncertainty 0]
+ 0.1]

    set c2c_same_h       [expr [lindex $core_overconstraints 1] + [lindex $core_clock_uncertainty 1]
+ 0.1]

    set c2c_diff_su      [expr [lindex $core_overconstraints 2] + [lindex $core_clock_uncertainty 2]
+ 0.1]

    set c2c_diff_h       [expr [lindex $core_overconstraints 3] + [lindex $core_clock_uncertainty 3]
+ 0.1]

 foreach src_core_clock_local $core_clocks_local {

     if {$src_core_clock_local != ""} {

       foreach dst_core_clock_local $core_clocks_local {

         if {$dst_core_clock_local != ""} {

           if {$src_core_clock_local == $dst_core_clock_local} {

             # Same clock network transfers

             set_clock_uncertainty -from $src_core_clock_local -to $dst_core_clock_local -setup -add
$c2c_same_su

             set_clock_uncertainty -from $src_core_clock_local -to $dst_core_clock_local -hold -
enable_same_physical_edge -add $c2c_same_h

           } else {

             # Transfers between different core clock networks

             set_clock_uncertainty -from $src_core_clock_local -to $dst_core_clock_local -setup -add
$c2c_diff_su
```

set_clock_uncertainty -from $src_core_clock_local -to $dst_core_clock_local -hold -add $c2c_diff_h

}

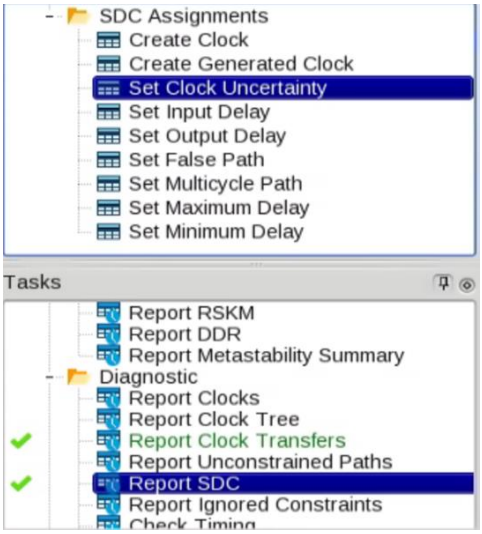
}

}

}

}

5. For user logic that involves user_created_clock and PHYLite output clock, user will need to add the extra 100ps clock uncertainty on that clock transfer path in the user sdc file.
6. Report SDC and check in the SDC Assignments>Set Clock Uncertainty report, make sure the extra 100ps is added into the affected clock transfer paths



7. Retime or recompile the design and ensure timing closure
8. Perform rigorous hardware testing to ensure design is working properly before going into production