Multicycles Exception Between Two Synchronous Clock Domains

set_multicycle_path **path_multiplier** [**-setup**|**-hold**] [**-start**|**-end**] -from <Start-Point> -through <ThroughPoint> -to <EndPoint>

Default path_multiplier: Setup 1, Hold 0.

Setup: Regarding to EndClock by Default. (-end is the default)
Hold: Regarding to StartClock by Default. (-start is the default)

In **single clock domain** design there is no meaning to the **-start** and **-end** flags. !!!

Note: In case your setup path_multiplier is X, use the path_multiplier of X-1 for hold. In any case check out the report of your multicycles and the report exceptions (Plus the report_exceptions -ignore) results.

Table 1: Clock Edges in M.C.

RED = Default	StartClock	EndClock
Setup	<----	---->
Hold	---->	<----

Table 1 describes how the check points for setup and hold timing are moved back-ward of forward depend on the -start/-end flags and the path_multiplier.

Multicycle In Single Clock Domain

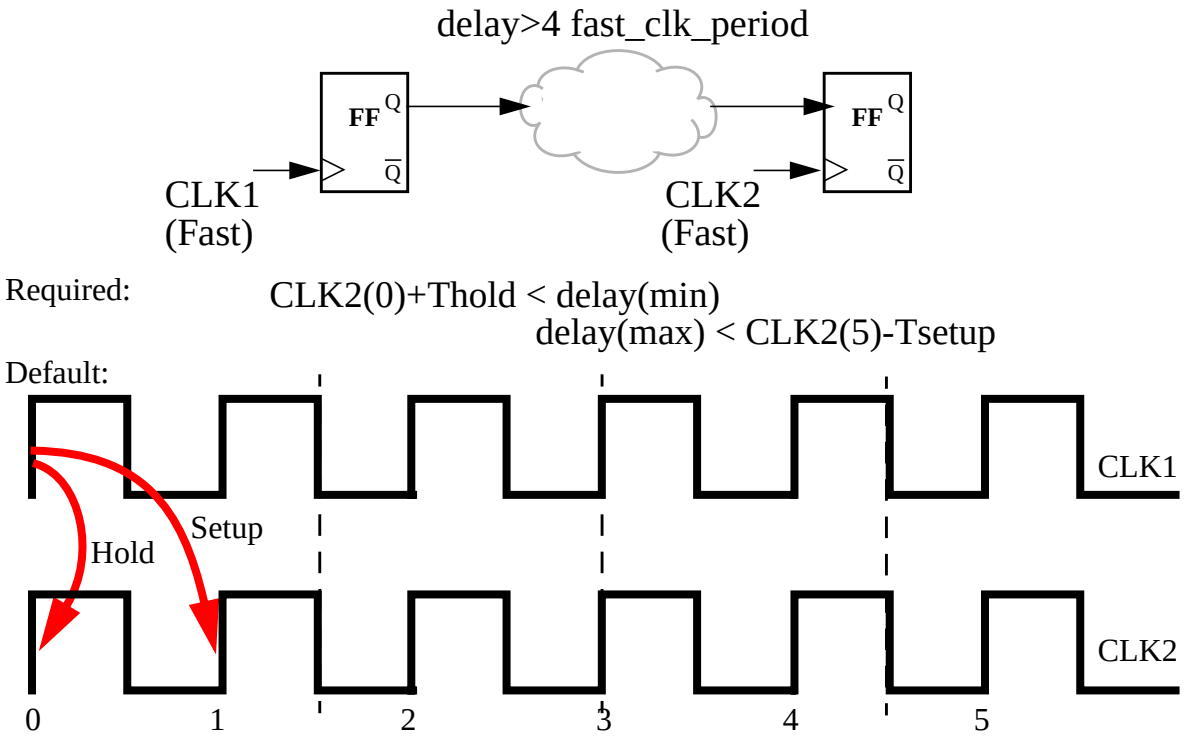


Figure 0-1. Single Clock Domains Design

0.1 SETUP PATH_MULTIPLIER 5, HOLD REMAINS DEFAULT

- set_multicycle_path -setup 5 -from CLK1 -to CLK2

without hold multiplier.

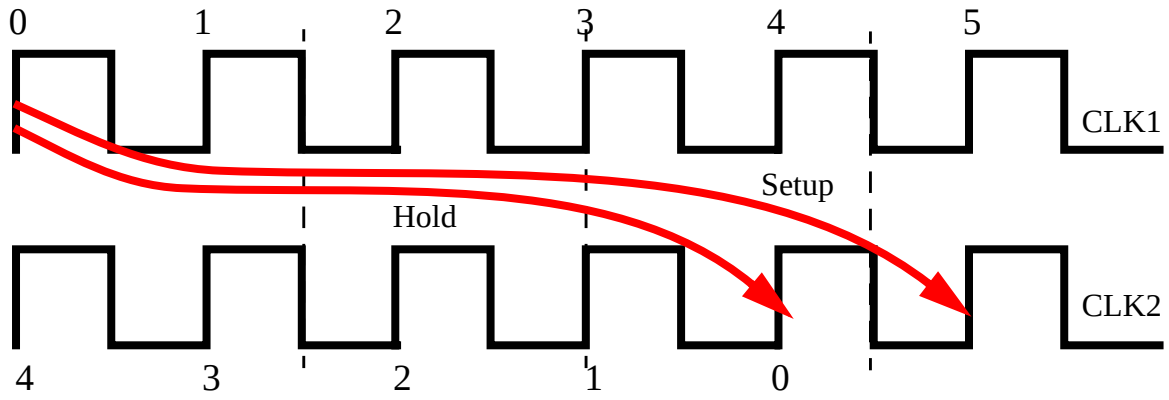


Figure 0-2. Setup 5, Hold default.

0.2 SETUP PATH_MULTIPLIER 5, HOLD PATH_MULTIPLIER 4

- set_multicycle_path -setup 5 -from CLK1 -to CLK2
- set_multicycle_path -hold 4 -from CLK1 -to CLK2

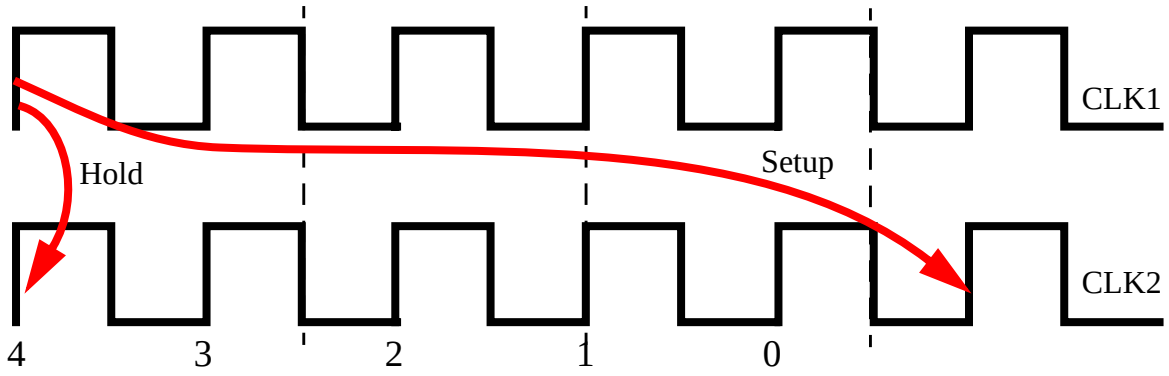


Figure 0-3. Setup 5, Hold 4.

Examples for SLOW-to-FAST path.

Figure 0-4 - Figure 0-5 show the timing analysis in case the StartClock, CLK1, is slow clock and the EndClock, CLK2, is fast clock.

The RED text is the default and can be removed.

The GREEN text is the user override

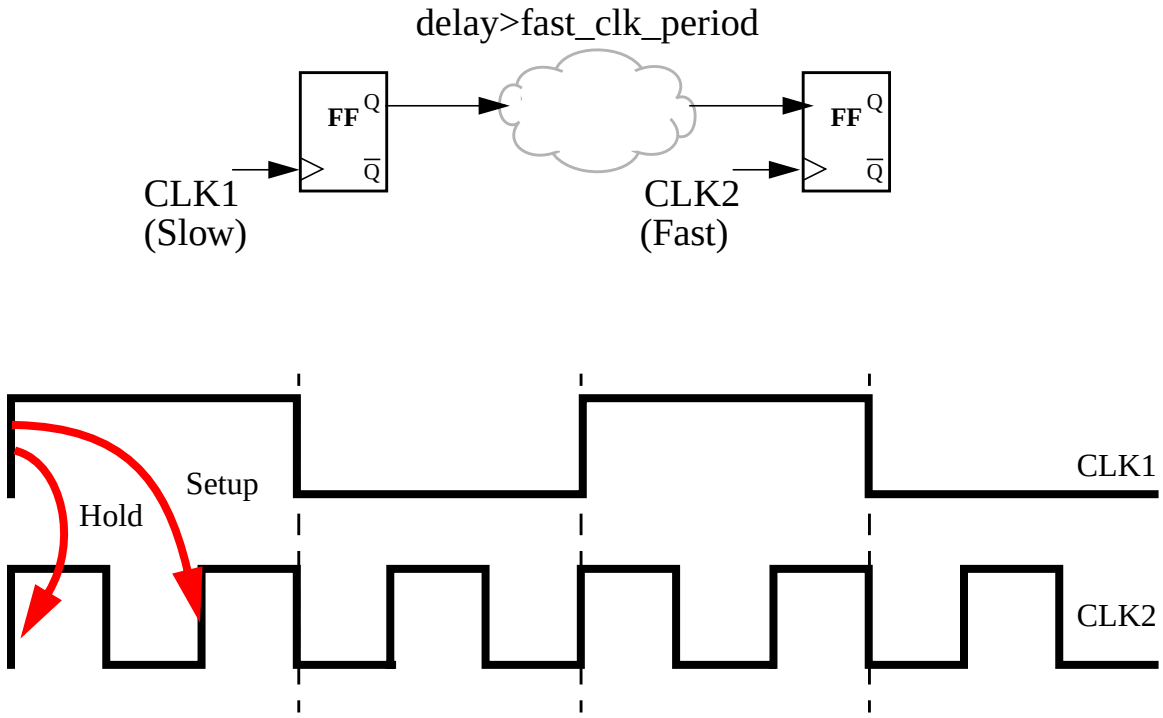


Figure 0-4. Two Clock Domains Design

0.3 DEFAULT

Figure 0-5 shows the default scenario:

- set_multicycle_path 1 -setup -end -from CLK1 -to CLK2
- set_multicycle_path 0 -hold -start -from CLK1 -to CLK2

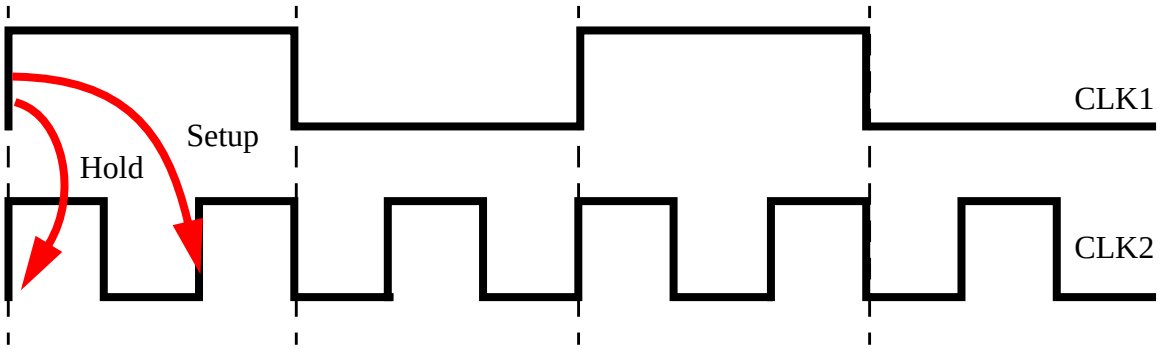


Figure 0-5. Default Scenario. Setup 1 (-end) , Hold 0 (-start)

0.4 SETUP PATH_MULTIPLIER 2, HOLD REMAINS DEFAULT

Figure 0-6 shows the case of path_multiplier 2 for setup and not defined path_multiplier for hold. (The user is not writing the multicycle for hold)

In this case even though the user is not writing the multicycle for the hold, the hold check is updated upon the setup check !!!

- set_multicycle_path 2 -setup -end -from CLK1 -to CLK2

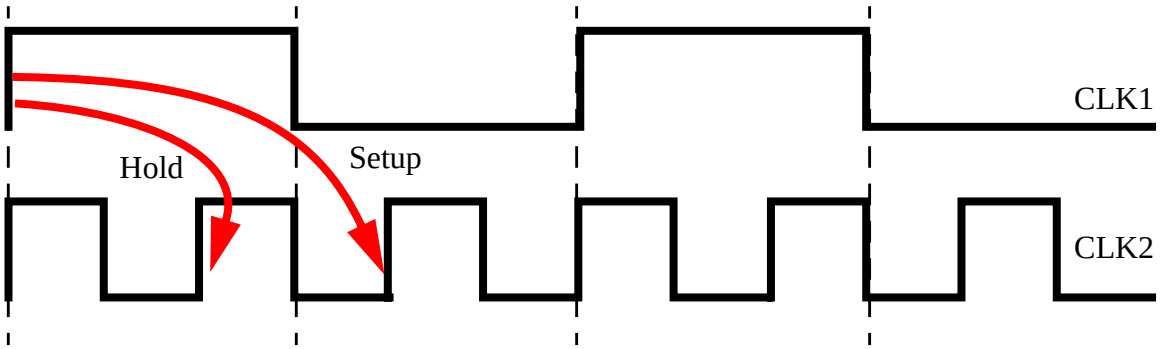


Figure 0-6. Setup 2 (-end) , Default Hold.

0.5 SETUP PATH_MULTIPLIER 2, HOLD PATH_MULTIPLIER 1

Figure 0-7 shows the case where the user sets the setup path_multiplier to 2 and the hold path_multiplier to 1. (Using the default **-start**) In this case the hold check is moved one clock forward relative to StartClock, CLK1.

- set_multicycle_path 2 -setup -end -from CLK1 -to CLK2
- set_multicycle_path 1 -hold -start -from CLK1 -to CLK2

Note: This check is incorrect !!! Jump to the next example.

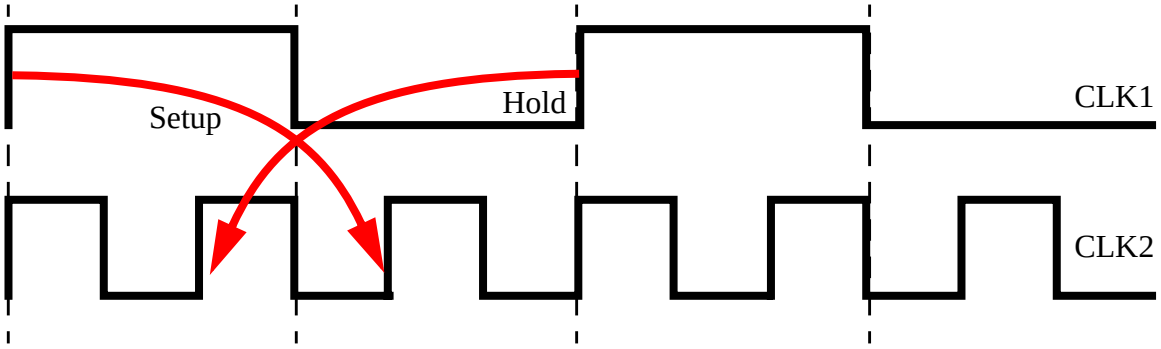


Figure 0-7. Setup 2 (-end) , Hold 1 (-start)

0.6 SETUP PATH_MULTIPLIER 2, HOLD PATH_MULTIPLIER 1

Figure 0-8 the case where the user sets the setup path_multiplier to 2 and the hold path_multiplier to 1 relative to the EndClock, CLK2 (Using the **-end** flag). In this case the hold check was moved one clock backward relative to EndClock, CLK2.

- set_multicycle_path 2 -setup -end -from CLK1 -to CLK2
- set_multicycle_path 1 -hold **-end** -from CLK1 -to CLK2

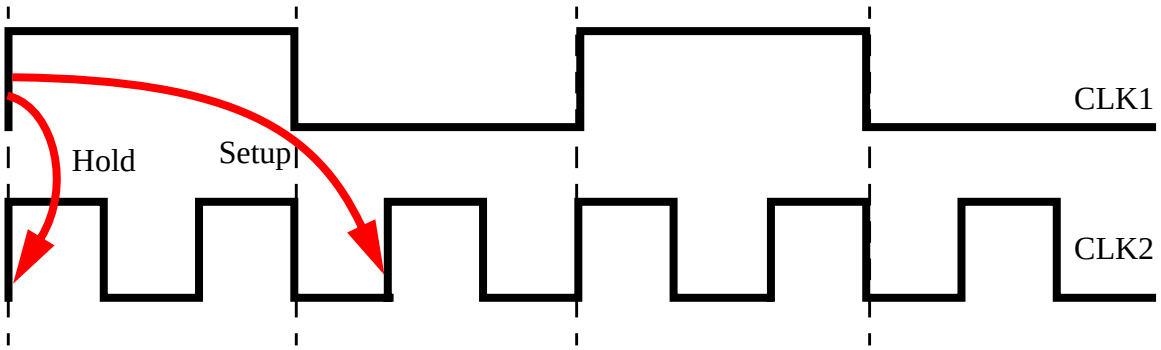


Figure 0-8. Setup 2 (-end) , Hold 1 (-end)

This is the required check. !!!

In General:

set_multicycle_path X -setup -end -from CLK1 -to CLK2

set_multicycle_path X-1 -hold -end -from CLK1 -to CLK2

RED=Default

GREEN=Override

Examples of FAST-to-SLOW path.

Figure 0-9 - Figure 0-12 show the timing analysis in case the StartClock, CLK1, is fast clock and the EndClock, CLK2, is slow clock.

The RED text is the default and can be removed.

The GREEN text is the user override

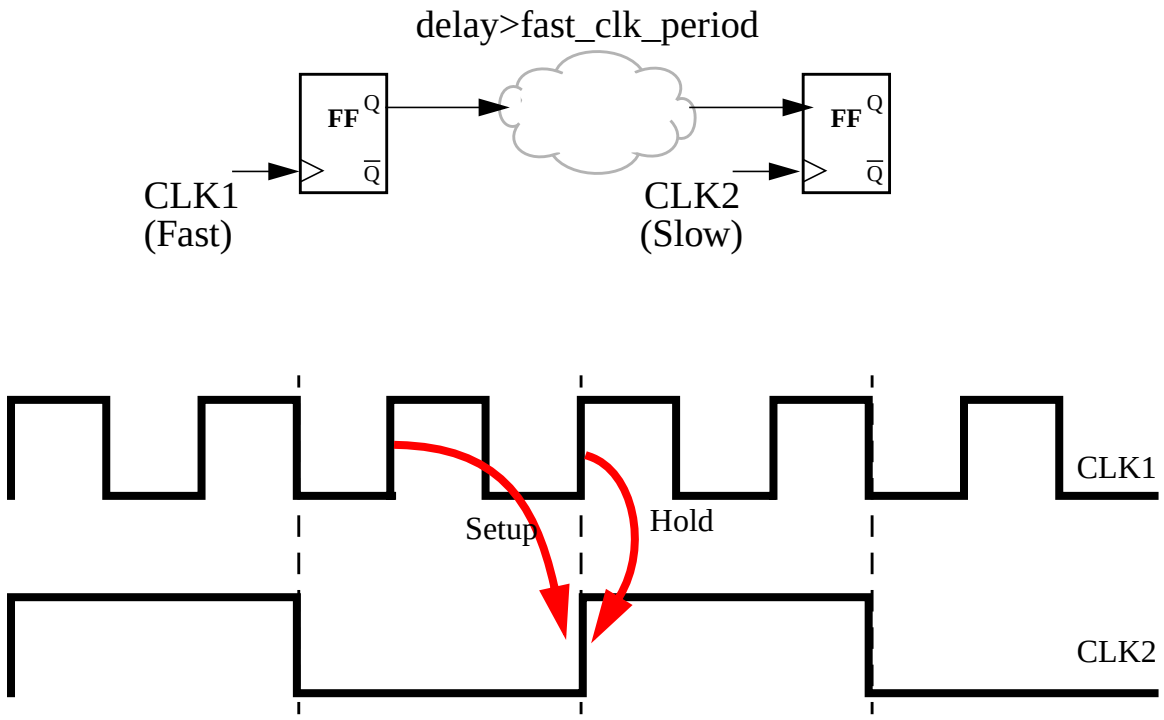


Figure 0-9. Two Clock Domains Design

0.7 DEFAULT

Figure 0-10 shows the default scenario:

- set_multicycle_path 1 -setup -end -from CLK1 -to CLK2
- set_multicycle_path 0 -hold -start -from CLK1 -to CLK2

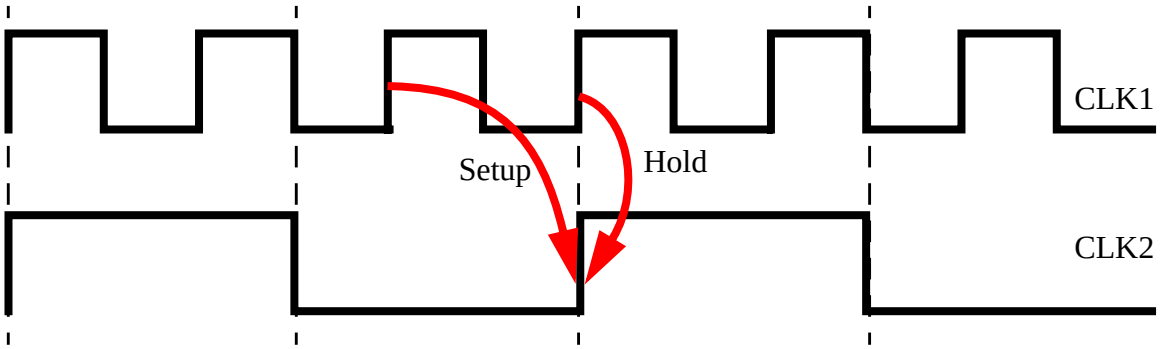


Figure 0-10. Default Scenario. Setup 1 (-end) , Hold 0 (-start)

0.8 SETUP PATH_MULTIPLIER 2, HOLD REMAINS DEFAULT

Figure 0-11 shows the case of path_multiplier 2 for setup and not defined path_multiplier for hold. (The user is not writing the multicycle for hold)

In this case even though the user is not writing the multicycle for the hold, the hold check is updated upon the setup check !!!

- set_multicycle_path 2 -setup -start -from CLK1 -to CLK2

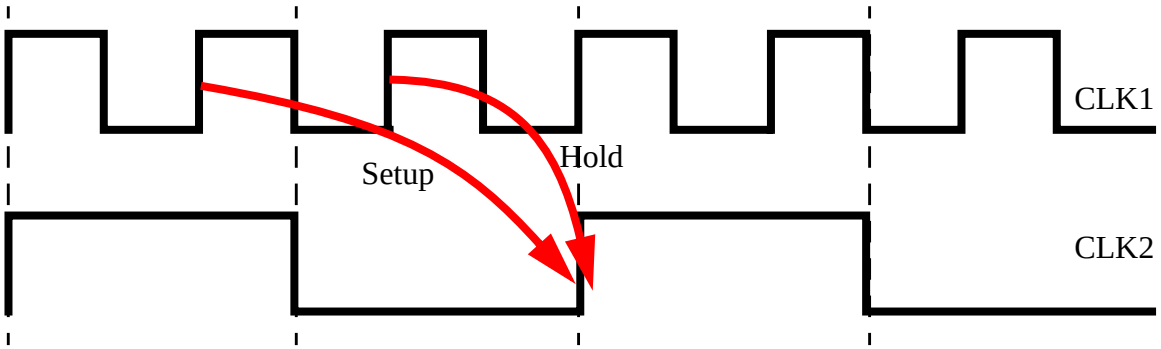


Figure 0-11. Setup 2 (-start) , Default Hold.

0.9 SETUP PATH_MULTIPLIER 2, HOLD PATH_MULTIPLIER 1

Figure 0-12 shows the case where the user sets the setup path_multiplier to 2 relative to the StartClock (Using the override **-start** flag), and the hold path_multiplier to 1 relative to the StartClock (Using the default **-start** flag). In this case the hold check was moved one clock backward relative to StartClock, CLK1.

- set_multicycle_path 2 -setup -start -from CLK1 -to CLK2
- set_multicycle_path 1 -hold -start -from CLK1 -to CLK2

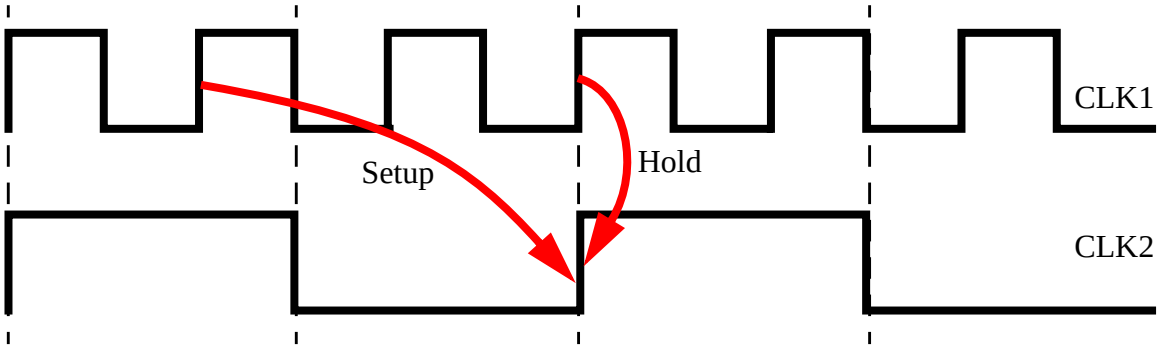


Figure 0-12. Setup 2 (-start) , Hold 1 (-start)

This is the required check !!!

In General:

set_multicycle_path X -setup -start -from CLK1 -to CLK2

set_multicycle_path X-1 -hold -start -from CLK1 -to CLK2

RED=Default

GREEN=Override

Solvnet Articles Regarding to Multicycles Setting

- **set_multicycle_path and hold checks**

  Question:

  I have a path that is set as multicycle path for the setup check. For some reason, PrimeTime seems to be treating it as a multicycle path for hold time checking as well. I'm using:

  set_multicycle_path -setup 7 -to [whatever]

  Why are the hold time checks multicycle?

  Answer:

  By default, if you specify 'set_multicycle_path -setup X', PrimeTime and Design Compiler assume the datapath could change during any clock before clock edge number X. To deal with this situation, PrimeTime and Design Compiler implicitly add 'set_multicycle_path -hold 0 -to [whatever]'. This positions the hold check one clock cycle before the setup check, effectively constraining the path delay to be between X-1 and X clock cycles, or in equation form:

  X-1 cycles + T_hold < path delay (min)
  path delay (max) < X cycles - T_setup

  So by default the tools assume you want the path buffered up so that the minimum change is > X-1 cycles.

  This may not be the desired behavior. You can move the hold check back towards the start of the multicycle period by specifying:

  set_multicycle_path -hold X-1 -to [whatever]

  In the above example, add

  set_multicycle_path -hold 6 -to [whatever]

  to the constraints and the hold check should occur on the desired edge. Note that moving this check back requires the designer to handle possible metastability. If the endpoint is a multi-bit signal, then you may need to generate register-enabling logic to avoid clocking data before all of it is valid.

- **Setting a Multicycle Path in a Multifrequency Design**

  Question:

  In my design, I have a crossing domain path from FF1 (controlled by 50-MHz clock A) to FF2 (controlled by 100-MHz clock B), on which I need to set a multicycle path 2. I used the following constraints:

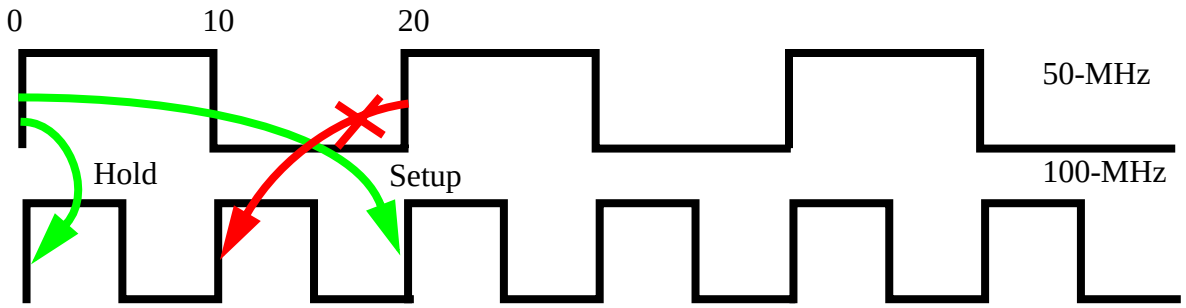
  Set_multicycle_path 2 -from FF1/CK -to FF2/D -setup
  Set_multicycle_path 1 -from FF1/CK -to FF2/D -hold

  However, report_timing shows that only the setup timing check works, with a timing window from waveform edge 0 to edge 20. The hold timing check is performed from edge 20 to edge 10 instead of from edge 0 to edge 0. This hold timing check result is too optimistic.

  What can I do?

  Answer:

  To set a multicycle maximum path, you can either move the end clock forward or the start clock backward. To set a multicycle minimum path, you can either move the end clock backward or the start clock forward. You control the movement by using the -end and -start switches. By default, set_multicycle_path moves the end clock for the max path and the start clock for the min path. For a single clock domain path, there is no need to specify the -end or -start option (it has no effect). But for a crossing domain path, moving the start clock forward one cycle is not equal to moving the end clock backward one cycle. The different options give totally different timing windows. So in the case of your hold timing check, you need to use the -end option to tell the tool to move the end clock backward one cycle.



- **How to use -setup and -hold options for set_multicycle_path**

  Question:

  How do I specify a multicycle path with a setup constraint of 2 cycles? I tried ’set_multicycle_path 2 -setup -hold -from A’ but this does not seem to give me what I want.

  Answer:

  The ’2 -setup -hold’ syntax tells Design Compiler to ensure that the data is available after 2 clocks, but makes the hold constraint very loose, which you might not want.

  Also, if you specify a ’set_multicycle_path -setup X’, the default behavior is for Design Compiler to assume that your want the path to be X cycles long and no less. For example, if X=2, you are saying that the path has 2 cycles to meet setup, and you don’t want any of the data to be valid after only 1 cycle. (That is, data launched on one edge cannot be captured by the following edge.) **So, Design Compiler inserts buffers to insure that the data will not be valid after only 1 cycle.**

  If you do not care if the path is faster than your ’-setup X’ number of cycles, you can move the hold check. Here is an example:

  ```
  set_multicycle_path -setup 2 -from DFF1/CLK
  set_multicycle_path -hold 1 -from DFF1/CLK
  ```

  This set of commands says that you do not care how fast the path starting at DFF1/CLK is, as long as it meets the 2 cycle setup. The hold check will work the same as it does without a multicycle path. (Data launched on one edge cannot be captured on the same edge.)

  For paths with more then 2 cycles, increase the -hold amount similarly:

  ```
  set_multicycle_path -setup 3 -from DFF1/CLK
  set_multicycle_path -hold 2 -from DFF1/CLK
  ```