Chapter 12
Multi Cycle Paths

By default, each path is timed for a single cycle, i.e., data launched at any edge of the clock should be captured by the next flop at the next rising edge of the clock on the destination flop. Figure 12.1 shows this relationship.

However, sometimes a designer might need to provide some additional cycles before the data is to be captured. Figure 12.2 shows this scenario of an additional cycle. The paths which get additional cycles are called multi cycle paths.

12.1 SDC Command for Multi Cycle Paths

The SDC command for declaring a path as multi cycle is:

```
set_multicycle_path     [-setup]
                        [-hold]
                        [-rise] [-fall]
                        [-start] [-end]
                        [-from from_list]
                        [-to to_list] [-through through_list]
                        [-rise_from rise_from_list]
                        [-rise_to rise_to_list]
                        [-rise_through rise_through_list]
                        [-fall_from fall_from_list]
                        [-fall_to fall_to_list]
                        [-fall_through fall_through_list]
                        path_multiplier
                        [-comment comment_string]
```

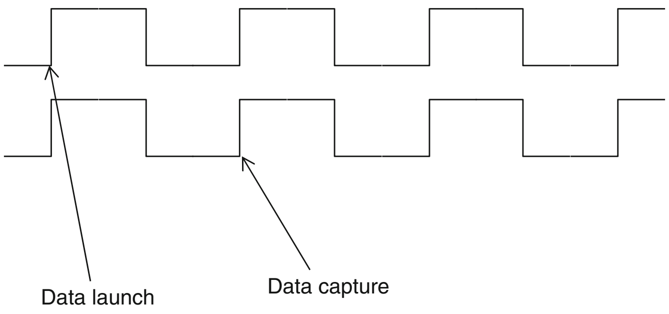

Fig. 12.1 Default setup timing relationship

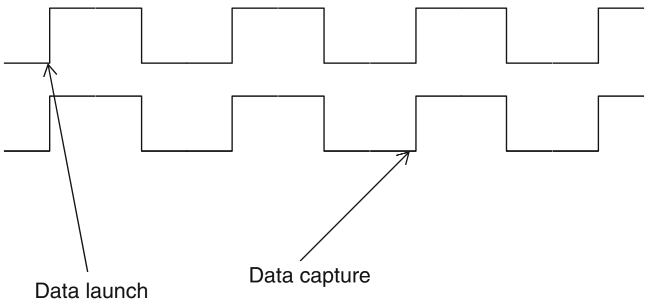

Fig. 12.2 Multi cycle of 2

Between all these switches, the command specifies:

– The exact path(s) which are to be treated as multi cycle
– The transitions within the paths which are to be treated as multi cycle
– Whether the multi cycle relationship is for setup or for hold
– Whether the additional cycle(s) are in terms of launch clock or capturing clock
– The number of cycles
– Any additional textual annotation to explain the context/justification for the multi cycle nature

12.2 Path and Transition Specification

Path and transition specification (options: *-rise, -fall, -from, -to, -through, -rise_from, -rise_to, -rise_through, -fall_from, -fall_to, -fall_through*) for a multi cycle specification is exactly same as that explained in Chap. 11. These options and path specifications are not being repeated here.

12.3 Setup/Hold Specification

For the purpose of this section, we will assume that both the launching device and the capturing device are triggered by the same clock. The implication of different clock frequencies will be discussed in Sect. 12.4. A clock waveform is depicted in Fig. 12.3.

Timing analysis will assume that the launching flop will launch the data at edge *A*. For setup analysis, it will consider that the data will be captured at the edge *B*. So, setup relation is analyzed between edges *A* and *B*. Use of *-setup* switch causes the capturing edge for setup analysis to be moved further to the right – away from *A* to *C*, *D,* etc., depending upon the number of cycles specified.

Let us assume that the capturing edge for setup has been moved to edge *D*, through *-setup* switch. For the purpose of hold analysis, the timing analysis tool considers the immediately preceding edge at the capture flop (when launch and capture clocks are the same). Thus, hold analysis will be considered using the edge *C* for capture. Use of *-hold* switch causes the capturing edge for hold analysis to be moved towards left, to either *B* or *A* – depending upon the number of cycles specified. The general practice is to restore the hold check back to edge *A*. If the hold check is not brought back to edge *A*, there might be buffers inserted in the path to ensure some delay. These buffers will take up silicon area as well as power.

We have assumed above, the setup edge was moved to 3 cycles (so that it reached edge *D*). The hold edge automatically moved to edge *C* (the immediately preceding edge). Now, in order to bring it back, it has to be moved back by 2 cycles. This can be achieved through use of *-hold* switch.

In order to move the hold check edge back to *A*, we have moved it by 2 cycles. It has now come back to the same edge as launch edge, i.e., at *0*th edge (with launch edge being considered Origin). It should be noted that we are now talking about two different numbers. A hold multiplier number *2* which specifies the number of edges by which the hold edge needs to move towards left. This is the number that is specified in the *set_multicycle_path* with -hold. And, another is number *0*, which specifies the actual edge number, where the check is happening. These two numbers are often a source of confusion during conversation. When you are talking about hold edge – specially in the context of multi cycle path – make sure that all the people involved understand, whether the number being mentioned is the "hold multiplier" the number by which the edge will move towards left, or the edge number, where the check will be performed. For the waveform shown in Fig. 12.3 (assuming setup number of 3), edge *A* corresponds to a hold multiplier number of *2*, edge *B* corresponds to a hold multiplier number of *1*, and edge *C* corresponds to a hold multiplier number of *0*, in the context of the *set_multicycle_path* definition.

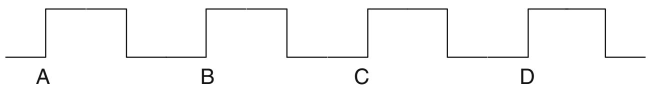

Fig. 12.3 Clock waveform

-setup switch specifies the period to which (not "by which") the setup capture edge will move to the right. Thus, a specification of *N* means move to *N*th period. This is different from move by *N* cycles. The *-hold* switch on the other hand specifies the period by which (not "to which") the hold capture edge will move to the left. If setup edge was moved to *N*th edge, the hold edge is automatically moved to *N–1*th edge. In order to restore it back to its original location, the hold check needs to be moved backwards by *N–1* cycles, so that it goes back to "*0*" edge.

Thus, *-setup* switch will move the capturing edge for setup. Simultaneously, it also moves the capturing edge for hold. After that, another *set_multicycle_path* might be needed with *-hold* switch to restore the hold check back to the original edges. Multi cycle path specifications are usually found in pairs of *-setup* and *-hold*. If the *-hold* specification is not given, the hold edge remains where it had got moved due to setup edge movement.

12.4 Shift Amount

The path multiplier specifies the number of clock cycles for the multi cycle path specification. If the launching device and the capturing devices are triggered by the same clock (or different clocks but with the same frequency), and the command specifies a multi cycle relationship, it is not important as to whether the number of cycles mentioned is for the start clock or for the end clock. However, if the start and the end clocks are different, then, it is important to specify whether the number of cycles specified is in terms of start clock or the end clock. For the purpose of our understanding, let us assume the clocks to be synchronous (Though the same approach can be extended to asynchronous clocks, however, asynchronous clocks are usually not timed).

Let start clock have a period of *10ns* and end clock have a period of *20ns*. For performing setup checks, the timing analysis tool identifies pairs of launch edge and the next capture edge. For all such pairs determined, the timing analysis tool figures out which of these pairs gives the minimal time for the data to travel. Figure 12.4 shows the waveforms for these clocks.

For the given example waveforms, the setup check would be made for the combination, launch at *B* and capture at *N*. So, for setup to be met, data path has to be within *10ns*.

Similarly, determine which are the worst case hold combinations, and use that launch/capture combination for hold check. The edge combinations used for hold check may not have any relation with the edges used for setup check.

For the given waveform, the hold check would be made for the combination: launch at *A* and capture at *M* (this is equivalent to launch at *C* and capture at *N*).

Within 1 cycle of destination clock, two data can get launched, which will mean losing one data. Within 1 cycle of destination clock, only one data should be launched – to avoid data loss. For a capture at *N*, the data could be launched either at *A* or at *B*. There is no advantage of launching at *B*, since the capture is still at *N*.

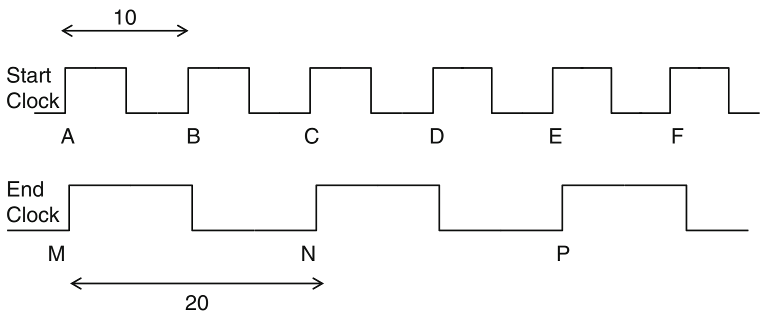

Fig. 12.4 Start and end clocks have different period

We might as well launch the data at *A* and provide 2 cycles (of source clock) for the data to reach the destination. Thus typically, for setup, we want to move the launch edge back to *A*. So, we declare a path of 2 cycles in terms of source clock. This can be specified through use of *-start* switch. This results in the launch edge being moved to the previous triggering edge of the launch clock, namely, launch at *A* and capture at *N*. So, setup gets another *10ns*. Note that the number "2" represents the number of cycles to be allowed for setup. It is different from the number of cycles by which the check got moved. The check got moved by 1 cycle, since the original setup check allowed for 1 cycle.

Now, for determining the hold check, the two pairs of edges are determined:

1. Launch at *B* and capture at *N* (hold launch edge one later than the setup launch edge).
2. Launch at *A* and capture at *M* (hold capture edge one earlier than the setup capture edge).

Out of these two combinations, the first one is more restrictive (higher requirement for hold check). Hence, the hold check moves to launch at *B* and capture at *N*. This means a minimum delay of *10ns* (*B* to *N*). The aim was to allow additional time, if needed, not to force a higher delay. So, we would want to restore the hold checks to default positions (viz., launch at *C* and capture at *N*). Thus, we need to move the launch edge towards right by 1 cycle of the start clock. This can be achieved by specifying *-start* switch with *-hold*. Now, the hold edges are *C, N* combination, which is same as *A, M* combination. Thus, the hold check edges have been restored to the original conditions. Note that in the case of hold multiplier, the number "1" represents the number of cycles by which the check got moved.

Let us consider another example. This time, the start clock has a period of *20ns* and the end clock has a period of *10ns*. Figure 12.5 shows the corresponding waveform:

For this combination of start and end clocks, the default setup check would be made at launch edge *M* and capture edge *B*. And, the hold check would be made at launch edge *M* and capture edge *A*.

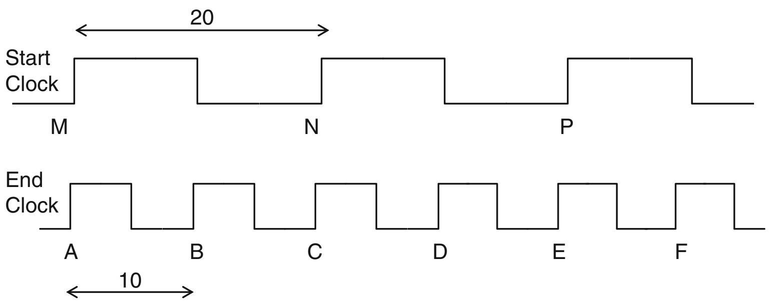

Fig. 12.5 Start clock slower than end clock

The data launched at *M* does not get changed for 2 cycles in terms of destination clock. So, multiple edges of destination clock would capture the same data (launched at *M*). We might as well disable the capture at all the edges of destination and do the capture only once for each launch. And, for a data launched at *M*, we would rather do the capture at *C*, thereby allowing more time for the signal to reach the destination. So, typically for setup, we declare a path of 2 cycles in terms of destination clock. This can be specified through use of *-end* switch. This results in the capture edge being moved to the next triggering edge of the destination clock, namely, launch at *M* and capture at *C*. Thus, setup gets another *10ns*.

This will cause the hold check to be moved to launch at *M* and capture at *B* (hold capture edge being 1 clock edge before the setup capture edge). If we want to restore the hold checks to default positions (viz., launch at *M* and capture at *A*), we need to move the capture edge towards left by 1 cycle of the end clock. This can be achieved by specifying *-end* switch with *-hold*.

So, effectively, for setup checks, *-end* means move the capture edge to the right, and *-start* means move the launch edge to the left. *-start* with *-hold* causes the launch edge to move to the right, and *-end* switch with *-hold* causes the capture edge to move to the left.

Stated alternately, *-start* moves the launch clock edge and *-end* moves the capture clock edge. With multi cycle, these edges move in a direction so as to make the checks less stringent. The number of cycles moved is always in terms of the edge that is moving. So, if launch edge is moving, the number of cycles is in terms of the launch clock.

Few more observations apparent from these two examples are:

1. In order to restore the hold edge back to the original location, the hold multiplier is 1 less than the setup multiplier.
2. For synchronous clocks with different frequencies, the setup number is equal to the ratio of the time period of the two clocks.
3. The multiplier is specified in terms of the period of the faster clock (smaller time period).

There should be an exact match in the number of data launched and data captured. For a one-to-one transmittal of data, that would mean one launch and one capture per one period of the slower clock. The remaining edges of the faster clock should be disabled, so that they neither capture nor launch additional data.

And the launch/capture edges can be so chosen that they provide maximum time for the data to travel. This will result in the observation 2 and 3 above.

These are just thumb rules. Each multi cycle path should be analyzed on its own for the right value of the multiplier, and the right clock period to be used.

The comment option can be used to specify a text annotation – mostly to mention the reasoning behind the multi cycle specification.

12.5 Example Multi Cycle Specification

Let us consider a few example situations of multi cycle specification. In the previous section, we already saw examples of synchronous data transfers between a fast to slow and a slow to fast clocks.

12.5.1 FSM-Based Data Transfer

Let us consider a circuit as shown in Fig. 12.6.

When the *data* is generated by *Cs*, the same clock also generates an *enable* signal. This signal goes through an FSM, and then the target capturing device is enabled to capture the data. Let us assume that the *enable* signal takes *N* cycles within the FSM, before the target device is ready to capture the data. In such a situation, there is no need for the actual *data* to rush to the target device immediately. It can take time up till *N* cycles (of destination clock) to reach there. Hence, this path needs to be constrained as:

set_multicycle_path -from Cs -through F1/Q -to Cd -setup N -end
set_multicycle_path -from Cs -through F1/Q -to Cd -hold N-1 -end

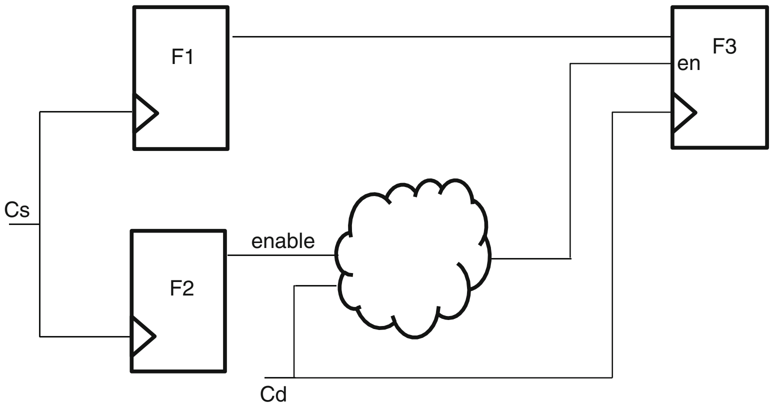

Fig. 12.6 FSM-based data transfer

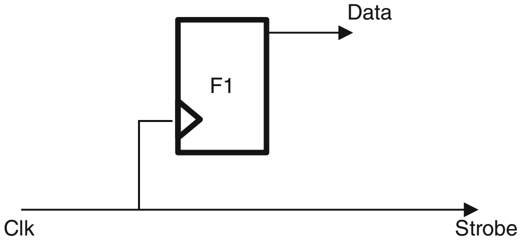

Fig. 12.7 Simple realization of source synchronous interface

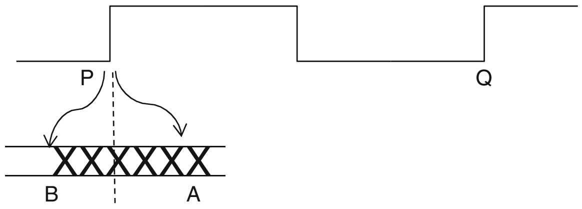

Fig. 12.8 Source synchronous interface – corresponding waveform

Note the use of *-through*. This is needed, so that only the *F1* to *F3* path are covered. The *enable* signal might need its own multi cycle specification, if it takes more than a cycle.

Paths using Handshake, Acknowledge, etc., for data transfer are examples of such situations which might use multi cycle. Another example is when a data bus is crossing a clock domain and instead of trying to synchronize the whole bus, an enable signal is synchronized. If the control FSM is going to take more than 1 cycle before enabling the capture device, the data can take those many cycles in reaching the destination device.

12.5.2 Source Synchronous Interface

In a source synchronous interface, as a data is presented at the output, the clock is also sent out. Both data and the clock lines can be routed on the board to have similar delays. Thus, the receiving device need not worry about the delays through the board trace. Whenever there is a clock signal, the receiving device knows that the actual data is also available around the same time. Figure 12.7 shows a typical realization of a source synchronous interface, and Fig. 12.8 shows the corresponding waveform.

In a system synchronous interface (in which all the signals are synchronized to system clocks), the output delay on the *Data* pin would be specified with respect to *Clk*. However, in the source synchronous interface, the reference is with respect to the *Strobe*. Thus, a timing relation specified wrt *Strobe* would cover *Data* pin also. The *Strobe* itself might be specified through *create_generated_clock* using *Clk* as the master.

Usually, *Data* launched on *Clk* edge corresponding to edge *P* of *Strobe* would be timed for setup edge *Q*. However, in this interface, *Data* should be timed with respect to the edge *P* itself. Thus, the setup edge needs to be moved. This can be done through the command:

set_multicycle_path -from [get_clocks Clk] -to [get_clocks Strobe] -setup 0

Notice that the setup multiplier number is specified as *0*, which moves the setup edge towards left – to the same edge as the launch edge. The start point is the flop *F1* triggered by *Clk* and the end point is the port *Data*, constrained with respect to *Strobe*.

When the setup edge moved back to point *P*, the hold edge needs to be restored back to its original location. This can be done through the command:

set_multicycle_path -from [get_clocks Clk] -to [get_clocks Strobe] -hold -1

The same path can be specified as *-to [get_ports Data],* instead of *-from [get_clocks Clk] -to [get_clocks Strobe].* Notice the negative value of the hold multiplier.

The *Data* cannot be available at exactly the same time as the *Strobe* edge. Let us assume that the duration *AB* indicates the time during which the *Data* is expected to change. That means, till *B*, the old *Data* would be available, and the new *Data* should be available by time *A*. This has to be modeled through appropriate values on *set_output_delay*.

Let us assume the duration *P* to *A* is *1.5ns*. So, this can be specified as:

set_output_delay -clock [get_clocks Strobe] -max -1.5 [get_ports Data]

Note the negative value of the delay. This indicates that the data availability is after the reference edge.

Let us assume the duration *B* to *P* is *1ns*. So, this can be specified as:

set_output_delay -clock [get_clocks Strobe] -min 1.0 [get_ports Data]

Notice that the min value is larger than the max value. Some tools do not support this. Make sure that your tool allows this!

The same timing effect can also be specified without *set_multicycle_path*. Let us assume the clock period to be *10ns*. Without the *set_multicycle_path*, the default edge for setup check would be launch at *P* and capture at *Q*. The need for data to be available at *A* can also be looked as a requirement that the data should be available *8.5ns* before the next edge *Q*. This can be specified as:

set_output_delay -clock [get_clocks Strobe] -max 8.5 [get_ports Data]

The hold check would still be made for capture at *P*. So, min delay remains the same, namely, *set_output_delay -clock [get_clocks Strobe] -min 1.0 [get_ports Data].*

This example showed how the same path can be specified in multiple ways. And also, the same timing effect can be achieved through different ways.

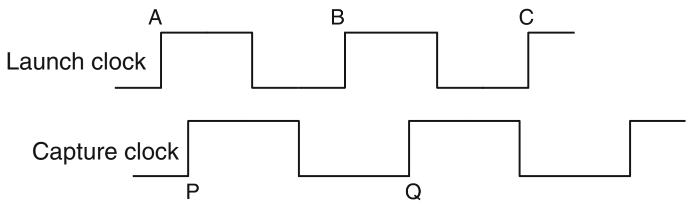

Fig. 12.9 Asynchronous clocks

12.5.3 Reset

In many ASIC designs, the master reset signal remains asserted for several cycles. So, the assertion of these signals can be declared as multi cycle paths. Assuming an active low asynchronous reset kept asserted for 3 cycles, the command would be:

set_multicycle_path -fall -from reset_n -setup 3
set_multicycle_path -fall -from reset_n -hold 2

12.5.4 Asynchronous Clocks

In Chap. 7, we have seen that asynchronous clock domain crossings are declared as *set_clock_groups -asynchronous*. The timer effectively disables such paths from any timing analysis. Thus, a path crossing clock domain can be allowed to have any amount of delay. Many designers want to put some kind of upper limit on the delay that such paths might have. This can be achieved through the commands:

set_multicycle_path -from [get_clocks C1] -to [get_clocks C2] -setup 2
set_multicycle_path -from [get_clocks C1] -to [get_clocks C2] -hold 1

When we specify a setup of 2 cycles, effectively the delay on the asynchronous path gets capped to 1 cycle (and not 2 cycles). Because the path is asynchronous, an edge pair will be considered where the launch and the capture edges are close. Thus, 1 cycle is effectively lost because of the close edges. Figure 12.9 explains this.

The default launch – capture combination is edges *A* and *P* which are very close. With a setup multi cycle of *2* (in terms of end clock), the capture edge moves to *Q*. The data path then gets the time from *A* to *Q*, which is almost one clock (of destination clock).

12.5.5 Large Data Path Macros

Some data paths have huge adders, multipliers, or other data path elements. Or, they might have deep levels of logic. Or, they might have a high setup requirement for the capturing device (say, a memory), or the launching device might have a high *Clk-to-output* delay (e.g., a memory). Or, the path might be on a clock which has very high frequency. In many such cases, it might be difficult for the data to meet the timing requirements of a single cycle. In such cases, the path might have to be declared as multi cycle.

12.5.6 Multimode

In Chap. 15, when we discuss multimode, we will also see one more example situation of *set_multicycle_path* command.

12.6 Conclusion

Multi cycle path provides additional relaxation to the specified paths. While specifying multi cycle paths, you should be careful to ensure:

– Unintended paths do not become multi cycle.
– The amount of additional time allowed is in line with what you had intended.

If a path is under-constrained (i.e., multi cycle specification allows a wider range for the signals to arrive) than what designed for, the device might not operate at the desired frequency.

When we move the setup edge through a multi cycle path specification, the hold edge also moves. You should check if the hold edge needs to be restored back to its original location. In most cases, it should be restored back. If you do not restore the hold edge, the design might have additional buffers in the data path, in order to increase delay to meet the increased hold requirement. This would cause wasted silicon area as well as power.

This chapter discussed multi cycle paths only in the context of timing. However, there are implications on the functional design (RTL) also, to ensure:

– Data is not lost.
– Glitches are not captured.