7.4. Automatic gated-clock conversion

Modern FPGA synthesis tools perform this gated-clock conversion process automatically without us having to change the RTL, however, we may need to guide the synthesis tools appropriately to perform the gated-clock conversion. It should be noted that some tools are more capable than others in this task.

Here are some of the guidelines to make the synthesis tools convert the gated clocks successfully:

- Identify the base clocks and define them to the synthesis tool by adding frequency or period constraints.
- Do not define the downstream gated clocks as clocks. Remove any period or frequency constraints on the gated clocks, which may have been specified during the SoC flow.
- Set any necessary controls in the synthesis tools to enable gated-clock conversion.
- Identify any black boxes in the design which are driven by gated clocks. To fix gated clocks that drive black boxes, the clock and clock-enable signal inputs to the black boxes must be identified. Synthesis tool specific directives should be used to identify them.
- If there are combinatorial loops in the clock-gating logic then the combinatorial loops should be broken. This can be done by inserting a feed-through black box, which is a black box with one input and one output and is placed in the combinatorial loop paths as shown in Figure 1. We can then create a separate netlist for the black box with the output simply connected to input. And the created netlist for the black box must then be added to the design during place and route.
When all the above guidelines are followed then the synthesis tools can automatically convert all the convertible gated clocks.

The gated clock is convertible when all of the following conditions are met.

- For certain combinations of the gating signals, the gated-clock output must be capable of being disabled.
- For the remaining combinations of the gating signals, the gated-clock output should equal either the base clock or its inverted value.
- The gated clock is derived based on only one base clock.
In order to illustrate these guidelines, Figure 2 gives some examples of simple convertible and non-convertible gates.

FPGA synthesis tools report all the converted and non-converted sequential components in its log files. The tools also list the reasons why the conversion did not happen for the non-converted sequential components. It is always advisable to look at these reports to make sure that the gated-clock conversion had happened for all the necessary sequential components.

### 7.4.1. Handling non-convertible gating logic

For an SoC design to work reliably on an FPGA-based prototype, all the gated clocks in the design should be converted. If the gated clock is derived based on multiple clocks, or the gating logic is complex, then synthesis tools cannot do the gated-clock conversion. However, these scenarios are sometimes common in SoC designs which can lead to many setup and hold-time violations. Here are some of the ways in which these scenarios can be handled. Use all these methods collectively as applicable.

- If there are no paths between the sequential elements driven by the base clocks and the unconverted gated clock, then the latter will not create any cross-domain timing violations. However, their routing in the FPGA may need to be carefully controlled to avoid the races described above.

An intermediate node in the design can be identified and defined as a base clock such that the gating logic present, driven by that node, is convertible. Usually, SoC designs will have a clock-generation logic block with complicated logic to generate
a glitch free, fail-safe and error-free clock. This clock will be created based on switching between many different clocks. And this generated clock will be used as the base clock for the rest of the blocks in the design with individual gating logic. Defining the clock on the output of the clock-generation logic block will make sure that all the gated clocks created, based on this clock, will be converted by the synthesis tool as shown in Figure 3.

**Figure 3: Handling complex clock gating**

- If there are valid timing paths between one base clock and its complex gated clock, then try to manually balance the clock paths between these paths. It can be balanced by introducing feed-through LUTs, clock buffers, PLLs and digital clock managers in one of the clock paths.
If there are still some gated clocks which are not converted, and there are huge valid timing violations, then try to run all the sequential elements in the FPGA at very high frequency – around 10x the fastest clock in the design. Insert rising-edge detectors with respect to the faster clock for all the gated clocks in the design. This rising-edge detector can be designed by double registering (say clk_reg1 and clk_reg2) the gated-clock signals using the faster clock and then forming a logic to detect the change from LOW to HIGH (NOT(clk_reg2) AND clk_reg1) as shown in Figure 4. If the original clock drives FFs which operate on a negative edge also, then negative-edge detector circuits will also be required.

In this way, the whole of the FPGA is driven by a single faster clock source as shown in Figure 5. This clock will use the dedicated global routing resources in the FPGA and therefore the associated clock skew will be very minimal and the timing can be easily met.
7.4.2. Clock gating summary

Clock gating is common in SoC designs and gated clocks should be handled with care to successfully prototype the SoC designs on FPGA. Contemporary FPGA synthesis tools automatically take care of most of these gated clocks when properly constrained. By following the guidelines in this chapter, SoC designs with complex clock gating can also be handled and successfully prototyped in FPGAs.