7.3. Clock gating

Clock gating is a methodology of turning off the clock for a particular block when it is not needed and is used by most SoC designs today as an effective technique to save dynamic power. In SoC designs clock gating may be done at two levels:

- **Clock RTL gating** is designed into the SoC architecture and coded as part of the RTL functionality. It stops the clocks for individual blocks when those blocks are inactive, effectively disabling all functionality of those blocks. Because large blocks of logic are not switching for many cycles it saves substantial dynamic power. The simplest and most common form of clock gating is when a logical “AND” function is used to selectively disable the clock to individual blocks by a control signal, as illustrated in Figure 1.

- During synthesis, the tools identify groups of FFs which share a common enable control signal and use them to selectively switch off the clocks to those groups of flops.

Figure 1: RTL clock gating in SoC to reduce dynamic power
Both of these clock-gating methods will eventually introduce physical gates in the clock paths which control their downstream clocks. These gates could introduce clock skew and lead to setup and hold-time violations even when mapped into the SoC, however, this is compensated for by the clock-tree synthesis and layout tools at various stages of the SoC back-end flow. Clock-tree synthesis for SoC designs balances the clock buffering, segmentation and routing between the sources and destinations to ensure timing closure, even if those paths include clock gating.

This is not possible in FPGA technology, so some other method will be required to map the SoC design if it contains a large number of gated clocks or complex clock networks.

### 7.3.1. Problems of clock gating in FPGA

As we saw in chapter 3, all FPGA devices have dedicated low-skew clock tree networks called global clocks. These are limited in number, but they can clock all sequential resources in an FPGA at frequencies of many hundreds of megahertz. Owing to diligent chip design by the FPGA vendors, the clock networks also have skew of only a few tens of picoseconds between any two destinations in the FPGA. Therefore, it is always advisable to use these global clocks when we target a design into FPGAs.

However, FPGA clock resources are not suited to creating a large number of relatively small clock domains, such as we commonly find in SoCs. On the contrary, an FPGA is better suited to implementing a small number of large synchronous clock networks which can be considered global across the device.

Global clock networks are very useful, but may not be flexible enough to represent the clocking needs of a sophisticated SoC design, especially if the clock gating is performed in the RTL. This is because physical gates are introduced into the clock paths by the clock-gating procedure and the global clock lines cannot naturally accommodate these physical gates. As a consequence, the place & route tools will be forced to use other on-chip routing resources for the clock networks with inserted gates, usually resulting in large clock skews between different paths to destination registers.

A possible exception to this happens when architecture-level clock gating is employed in the SoC, for example when using coarse-grained on-off control for clocks in order to reduce dynamic power consumption. In those cases it may be possible to partition all the loads for the gated clock into the same FPGA and drive them from the same clock driver block. The clock driver blocks in the latest FPGAs, for example, the clock management tiles (CMT) in Virtex-6 devices with their mixed-mode clock managers (MMCMs) have different controls to allow control of the clock output. Some clock-domain on-off control could be modeled using this coarse-grained capability of the CMT.
In some SoC designs there may also be paths in the design with source and destination FFs driven by different related clocks e.g., a clock and a derived gated clock created by a physical gate in the clock path, as shown in Figure 2. It is quite possible that the data from the source FF will reach the destination FF quicker/later than the gated clock, and this race condition can lead to timing violations.

### 7.3.2. Converting gated clocks

The solution to the above race condition is to separate the base clock and gating from the gated clock. Then route the separated base clock to the clock and gating to the clock enables of all the sequential elements. When the clock is to be switched “on,” the sequential elements will be enabled and when the clock is to be switched “off,” the sequential elements will be disabled. Typically, many gated clocks are derived from the same base clock, so separating the gating from the clock allows a single global clock line to be used for many gated clocks. This way the functionality is preserved and logic gates present in the clock path are moved into the datapath, which eliminates the clock skew as illustrated in Figure 2.
This process is called gated clock conversion. All the sequential elements in an FPGA have dedicated clock-enable inputs so in most of the cases, the gated clock conversion could use this and not require any extra FPGA resource. However, manually converting gated clocks to equivalent enables is a difficult and error-prone process, although it could be made a little easier if the clock gating in the SoC design were all performed at the same place in the design hierarchy, rather than scattered throughout various sub-functions.

As we saw in Error! Reference source not found. earlier, the chip support block at the top level could include all the clock generation and clock gating necessary to
drive the whole SoC. Then, during prototyping, this chip support block can be replaced with its FPGA equivalent. At the same time, we can manually replace the clock gates, either instantiated or inferred, with an enable signal which can routed throughout the device. This would then perform the role of enabling only a single edge of the global clock at each time that the original gated clock would have risen.

In most cases, manual manipulation is not possible owing to complexity, for example, if clocks are gated locally at many different always or process blocks in the RTL. In that case, and probably as the default in most design flows, automated gated-clock conversion can be employed.