Techniques to make clock switching glitch free

Techniques to make clock switching glitch free
Rafey Mahmud

6/26/2003 04:00 PM EDT

With more and more multi-frequency clocks being used in today's chips, especially in the communications field, it is often necessary to switch the source of a clock line while the chip is running. This is usually implemented by multiplexing two different frequency clock sources in hardware and controlling the multiplexer select line by internal logic.

The two clock frequencies could be totally unrelated to each other or they may be multiples of each other. In either case, there is a chance of generating a glitch on the clock line at the time of the switch. A glitch on the clock line is hazardous to the whole system, as it could be interpreted as a capture clock edge by some registers while missed by others.

In this article, two different methods of avoiding a glitch at the output clock line of a switch are presented. The first method is used when clocks are multiples of each other, while the second deals with clocks totally unrelated to each other.

The problem with on-the-fly clock switching

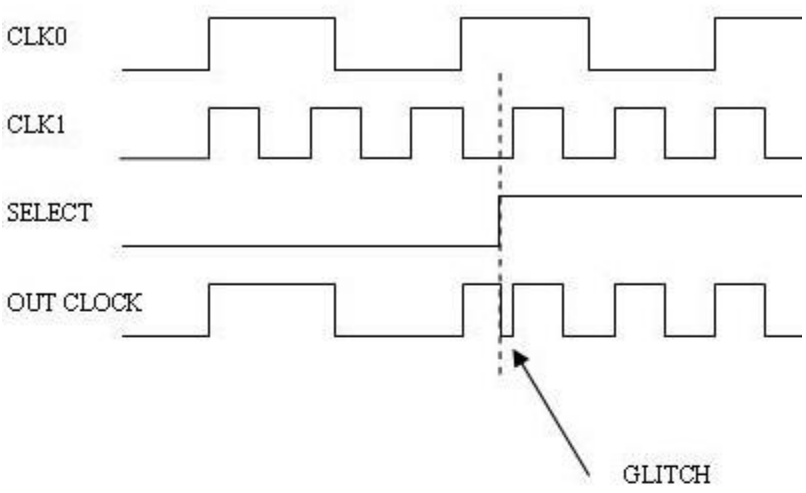
Figure 1 shows a simple implementation of a clock switch, using an AND-OR type multiplexer logic.

The multiplexer has one control signal, named SELECT, which either propagates CLK0 to the output when set to "zero" or propagates CLK1 to the output when set to "one." A glitch may be caused due to immediate switching of the output from Current Clock source to the Next Clock source, when the SELECT value changes. Current Clock is the clock source currently selected while Next Clock is the clock source corresponding to the new SELECT value.

The timing diagram in Figure 1 shows how a glitch is generated at the output, OUT CLOCK, when the SELECT control signal changes. The problem with this kind of switch is that the switch control signal can change any time with respect to the source clocks, thus creating a potential for chopping the output clock or creating a glitch at the output.

The select control signal is most likely generated by a register driven by either of the two source clocks, which means that either it has a known timing relationship to both clocks, if both clocks are multiples of each other, or it may be asynchronous to at least one clock, if source clocks are not related in any way.

Switching during either clock's high state needs to be avoided without having any idea about the frequencies or phase relationship of these clocks. Fixed delay can be used to induce the gap between the start and stop time of the two source clocks, but only if a fixed relationship exists between the two clock sources. It cannot be used where either the input frequencies are not known, or the clocks are not related.



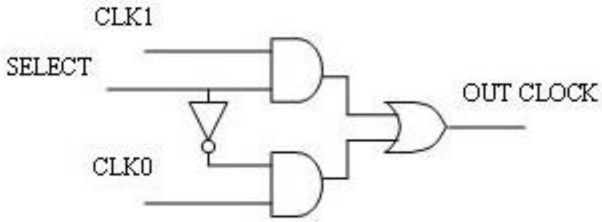


Figure 1 -- Clock switching multiplexer

Glitch protection for related clock sources

A solution to prevent glitch at the output of a clock switch where source clocks are multiples of each other is presented in Figure 2. A negative edge triggered D flip-flop is inserted in the selection path for each of the clock sources. Registering the selection control at negative edge of the clock, along with enabling the selection only after other clock is de-selected first, provides excellent protection against glitches at the output.

Registering the select signal at negative edge of the clock guarantees that no changes occur at the output while either of the clocks is at high level, thus protecting against chopping the output clock. Feedback from one clock's selection to the other enables the switch to wait for de-selection of the Current Clock before starting the propagation of the Next Clock, avoiding any glitches.

The figure 2 timing diagram shows how the transition of the SELECT signal from 0 to 1 first stops propagation of CLK0 to the output at the proceeding falling edge of CLK0, then starts the propagation of CLK1 to the output at following negative edge of CLK1.

There are three timing paths in this circuit that need special consideration — the SELECT control signal to either one of the two negative edge triggered flip flops, the output of DFF0 to input of DFF1, and the output of DFF1 to the input of DFF0. If the signal on any of these three paths changes at the same time as the capturing edge of the destination flip flop's clock, there is a small chance that the output of that register may become meta-stable, meaning it may go to a state between an ideal "one" and an ideal "zero."

A meta-stable state can be interpreted differently by the clock multiplexer and the enable feedback of the other flip flop. Therefore, it is required that capturing edges of both flip flops and the launch edge of the SELECT signal should be set apart from each other to avoid any asynchronous interfacing. This can be easily accomplished by using proper multi-cycle hold constraints or minimum delay constraints, as the timing relationship is known between the two clocks.

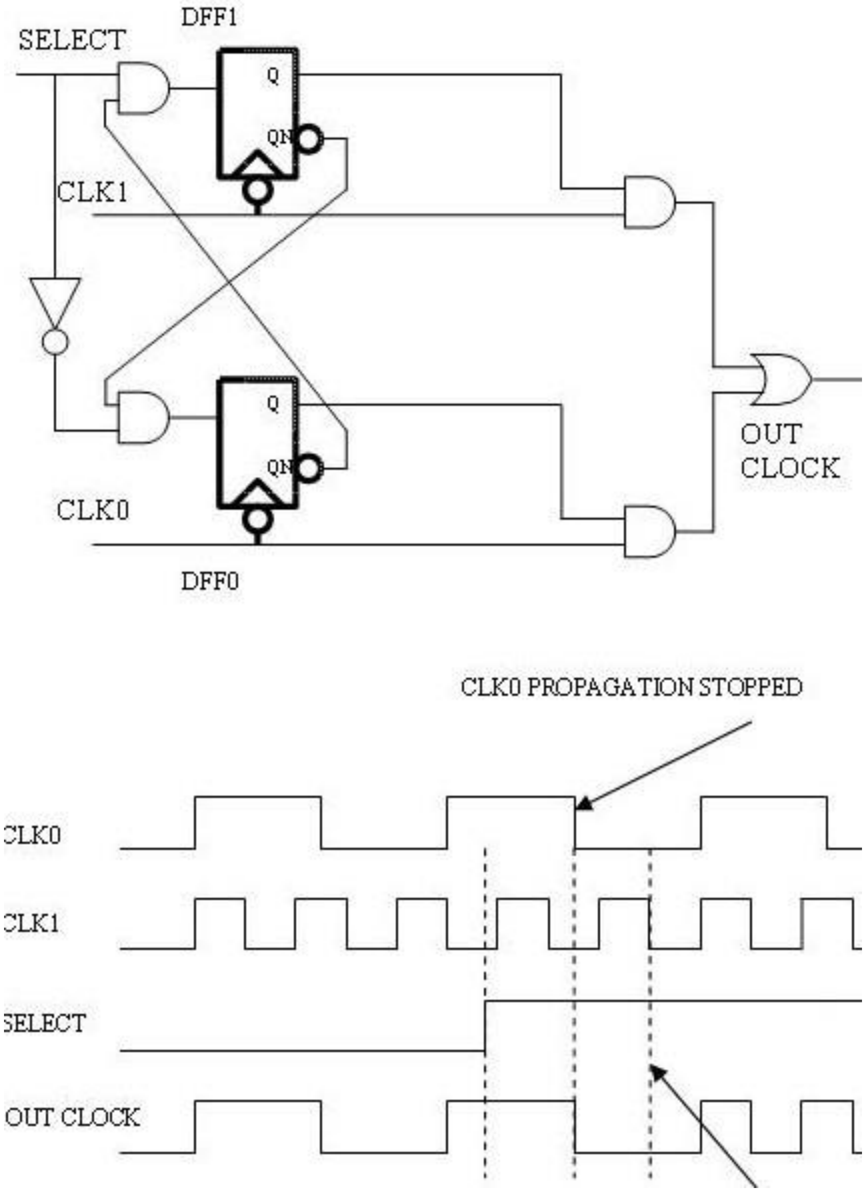


Figure2 -- Glitch-free clock switching for related clocks

Fault tolerance

At chip startup time, both flip flops DFF0 and DFF1 should be reset to the "zero" state so that neither one of the clocks is propagated initially. By starting both flip flops in "zero" state, fault tolerance is built into the clock switch.

Let's say that one of the clocks was not toggling due to a fault at startup time. If the flip flop associated with the faulty clock had started up in "one" state, it would prevent the selection of other clock as the Next Clock, and its own state is not changeable due to lack of a running clock. By starting both flip flops in "zero" state, even if one of the source clocks is not running, there is still the ability to propagate the other good clock to the output of the switch.

Glitch protection for unrelated clock sources

The previous method of avoiding a glitch at the output of a clock switch requires the two clock sources to be multiples of each other, such that user can avoid signals to be asynchronous with either one of the clock domains. There is no mechanism to handle asynchronous signals in that implementation.

This leads to the second method of implementing the clock switch with synchronizer circuits to avoid potential meta-stability caused by asynchronous signals. The source of asynchronous behavior could either the be SELECT signal or the feedback from one clock domain to the other, when the two clock sources are totally unrelated to each other.

As shown in Figure 3, protection is provided against meta-stability by adding one extra stage of positive edge triggered flip flop for each of the clock sources. The positive edge triggered flip flop in each of the selection paths, along with the existing negative edge triggered flip flop, guards against potential meta-stability, which may be caused by asynchronous SELECT signal or asynchronous feedback from one clock domain to the other.

A synchronizer is simply two stages of flip flops, where the first stage helps stabilize data by latching it and later passing it on to the next stage to be interpreted by rest of the circuit.

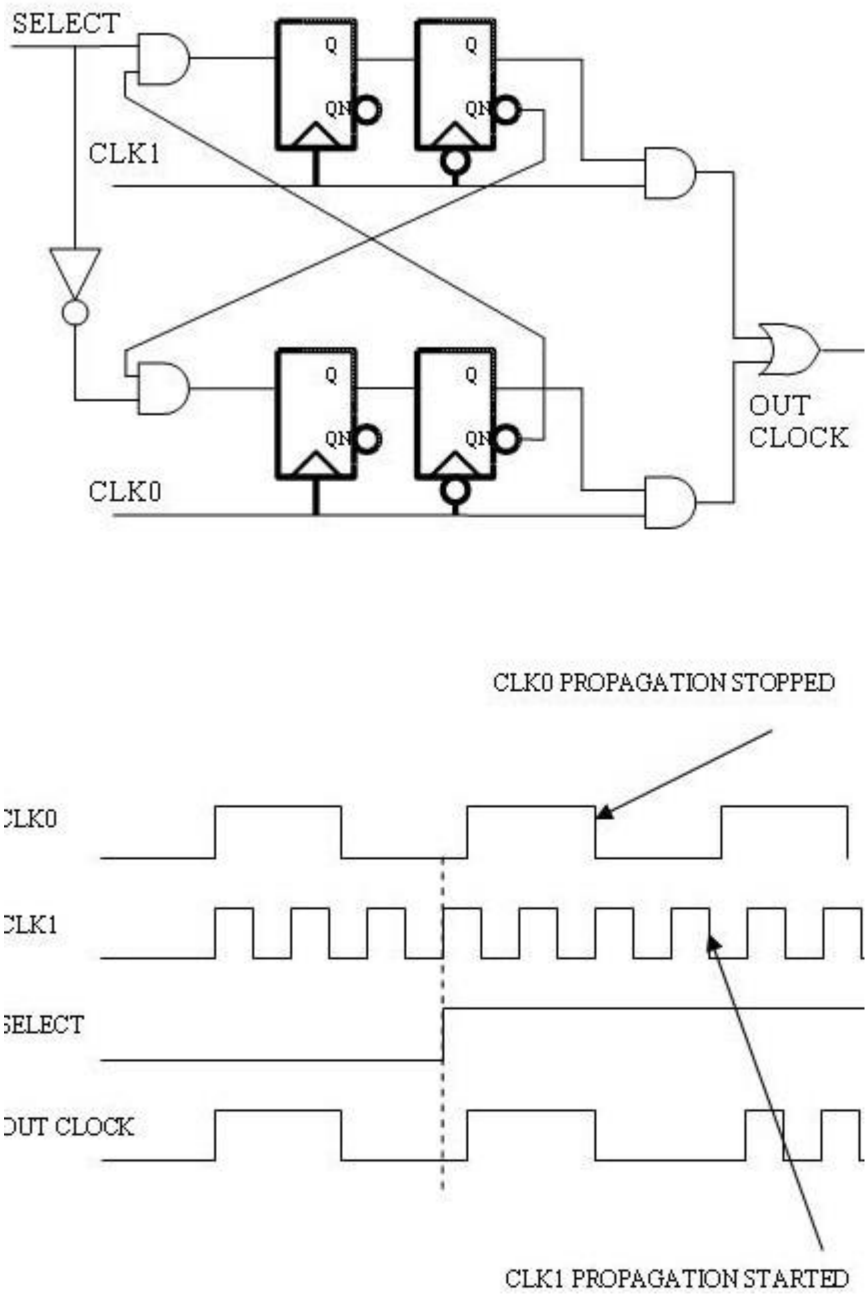


Figure 3 -- Glitch-free clock switching for unrelated clocks

Conclusion

The hazard of generating a glitch on a clock line while switching between clock sources can be avoided with very little overhead by using the design techniques presented in this article. These techniques are fully scalable and can be extended to a clock switch for more than two clocks. For multiple clock sources, the select signal for each clock source will be enabled by feedback from all the other sources.