Interfaces Between Clock Domains

This lecture covers two topics related to the design of interfaces between logic circuits that use different clocks: metastability and glitches.
After this lecture you should be able to: identify circuits where metastable behaviour is possible; add synchronizer flip-flops to reduce the probability of metastability; compute the mean time between metastable outputs; identify combinational circuits that might contain hazards; and modify these circuits to remove hazards by registering outputs.

Metastability

Introduction

The proper operation of a clocked flip-flop depends on the input being stable for a certain period of time before and after the clock edge. If the setup and hold time requirements are met, the correct output will appear at a valid output level (between V_{OL} and V_{OH}) at the flip-flop output after a maximum delay of t_{CO} (the clock-to-output delay). However, if these setup and hold time requirements are not met then the output of the flip-flop may take *much* longer than t_{CO} to reach a valid logic level. This is called *metastable* behaviour or *metastability.*

An invalid logic level at the output of the flip-flop may be interpreted by some logic gates as a '1' and by others as a '0'. This leads to unpredictable and usually incorrect behaviour of the circuit.

In the synchronous circuits we have studied thus far we have been able to prevent metastability by clocking all flip-flops from the same clock and ensuring that the maximum propagation delay of any combinational logic path is less than the clock period minus the flip-flop setup time and clock-to-output delay.

However, when inputs to a synchronous circuit are not synchronized to the clock, it is *impossible* to ensure that the setup and hold times will be met. This will eventually lead to the incorrect behaviour of the device. It is important to realize that all practical logic circuits will eventually fail due to metastability. However, the designer should try to ensure that these failures happen very infrequently (e.g. once per 10^3 or 10^6 years of operation) so that other causes of failure predominate.

Computing MTBF

The average time between metastable outputs (mean time between failures or 'MTBF') is given by the formula:

$$\text{MTBF} = \frac{e^{C_2 t_{MET}}}{C_1 f_{clk} f_{data}}$$

where C_1 and C_2 are constants that depend on the technology used to build the flip-flop, t_{met} is the duration of the metastable output, and f_{clk} and f_{data} are the frequencies of the synchronous clock and the asynchronous input respectively.

Let's compute the MTBF assuming we used the lab FPGA board's internal oscillator as a clock to register the PC-104 bus IOR* signal. Since the clock and the signal being registered are coming from different oscillators the input is asynchronous. The clock frequency, f_{clk} is 25.175 MHz. The exact frequency of IOR* will depend on the program being executed, but let's assume a value of $f_{CLK2}/4 = 8.333/4 = 2.08$ MHz. For the Altera Flex10K family $C_1 = 1 \times 10^{-13}$ and $C_2 = 1.3 \times 10^{-10}$. For correct operation of our circuit the settling time of the flip-flop output, the metastable time t_{MET}, must be less than the clock period minus the maximum propagation delays through the combinational logic elements minus the setup times of the other flip-flops in the circuit. The setup time of the -4 speed grade 10K20 input flip-flops, t_{IOSU}, is is 3.2 ns, thus $t_{MET} = t_{clk} - t_{PD} - t_{IOSU}$. If we assume t_{PD} is, for example, 30 ns, $t_{MET} = 39.7 - 30 - 3.2 = 6.5ns$ and the MTBF is:

$$\text{MTBF} = \frac{e^{1.3\times10^{10}\times6.5\times10^{-9}}}{1\times10^{-13}\times25.175\times10^{6}\times2.08\times10^{6}}\text{s}$$

which about 10^{33} seconds (a very long time).

Reducing Metastability

The simplest approach is to slow down the clock since this provides a longer time for the output of the flip-flop to reach a stable output value. Because the MTBF increases exponentially with t_{MET} a small reduction in clock frequency will often be enough to increase the MTBF to an acceptable value. However, in other cases this approach will be unacceptable because the resulting clock rate will be too slow.

Another approach is to use flip-flops with shorter setup and hold times (and correspondingly smaller C_1 and larger C_2 values). Whenever possible, these "metastable-hardened" flip-flops should be used on asynchronous inputs.

If this does not result in the desired degree of reliability it is possible to use two or more flip-flops in series. In this case the output of the second flip-flop will only be metastable if *both* flip-flop outputs were metastable. The disadvantage of this approach is that the input will now be delayed by one to two clock periods (instead of zero to one clock periods).

Input Synchronizers

An asynchronous input should not be used directly in a synchronous circuit because different propagation delays within the circuit may cause some flip-flops to register one value and others to register the opposite value. This will almost certainly lead to incorrect behaviour.

Even when such problems are avoided, such a circuit is more likely to violate of setup and hold times and there will be a higher likelihood of metastability.

Such problems can be avoided by registering each asynchronous input using a single metastable-hardened flip-flop and using the output of this flip-flop output to drive the rest of the logic. This results in a delay of up to 1 clock period in before the circuit can respond to the changed input. Usually this is an acceptable trade-off for improved reliability.

Exercise: Draw the schematic of an input synchronizer.

Glitches and Hazards

Glitches are short temporary changes in outputs that are caused by different propagation delays in a circuit. There are two reasons why glitches are undesirable.

The first set of problems is related to noise and power. Since glitches are short pulses much of their energy is at high frequencies and this power couples easily onto adjacent conductors. This induces noise into other circuits and reduces their noise immunity. Glitches also cause power supply current spikes which result in voltage transients on the power supply lines. Another problem with glitches is that in CMOS logic families current consumption is proportional to the number of gate output changes and glitches lead to increased current consumption.

The second set of problems arises when the digital output of one circuit is used as a clock in another circuit (e.g. to drive a counter or register). In this case glitches cause undesired clock edges (similar to the switch bounce problems). In synchronous (single-clock) circuits these glitches are not a problem.

Exercise: Why not?

Glitches can be reduced by modifying the design of the combinational logic. However, this usually introduces additional logic. Glitches on signals that are confined to short paths within a circuit or inside a chip are usually tolerated. However, when outputs are brought off a chip, board or system (e.g. onto a bus) it is good practice to eliminate glitches.

The simplest way to eliminate glitches is to use a registered output signal. The output of a flip-flop changes only once, on the clock edge, and thus eliminates any glitches on its input. There are two ways to register outputs. Often it is possible to use register outputs directly such as when an output is already in a data register or when the signals are state machine state registers. The second method is to pass the signal through an additional flip-flop before it is output. The disadvantage of this method is that the output will be delayed by one clock period.