Synchronizer techniques for multi-clock domain SoCs & FPGAs

Tejas DaveAmit Jain,Divyanshu Jain, - September 30, 2014

This week we will look at standard synchronization techniques for multi-clock domain SoCs. Let us begin with the most common and simple option.

Conventional two flip-flop synchronizer

In general, a conventional two flip-flop synchronizer is used for synchronizing a single bit level signal. As shown in **Figure 1** and **Figure 2**, flip flop A and B1 are operating in asynchronous clock domain. There is probability that while sampling the input B1-d by flip flop B1 in CLK_B clock domain, output B1-q may go into metastable state. But during the one clock cycle period of CLK_B clock, output B1-q may settle to some stable value. Output of flop B2 can go to metastable if B1 does not settle to stable value during one clock cycle, but probability for B2 to be metastable for a complete destination clock cycle is very close to zero.

A greater number of flop stages may be used if frequency is too high as it will help in reducing the probability of synchronizer output to remain in metastable state.

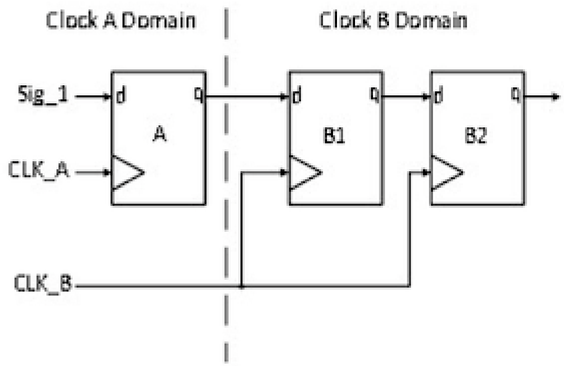

Figure 1 - Conventional 2FF synchronizer

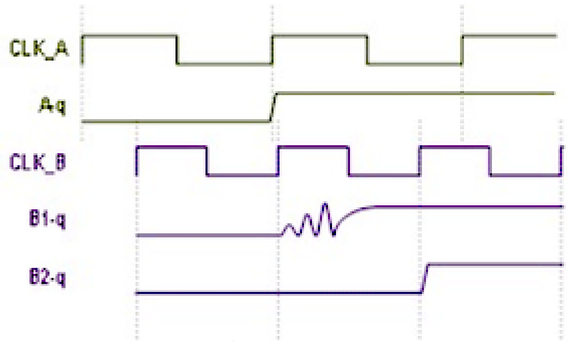

Figure 2 Timing for Conventional 2FF synchronizer

Toggle synchronizer

Toggle synchronizer is used to synchronize a pulse generating in source clock domain to destination clock domain. A pulse cannot be synchronized directly using 2 FF synchronizer. While synchronizing from fast clock domain to slow clock domain using 2 FF synchronizer, the pulse can be skipped which can cause the loss of pulse detection & hence subsequent circuit which depends upon it, may not function properly. Diagram in **Figure 3** and **Figure 4** shows toggle synchronizer implementation and Timing diagram.

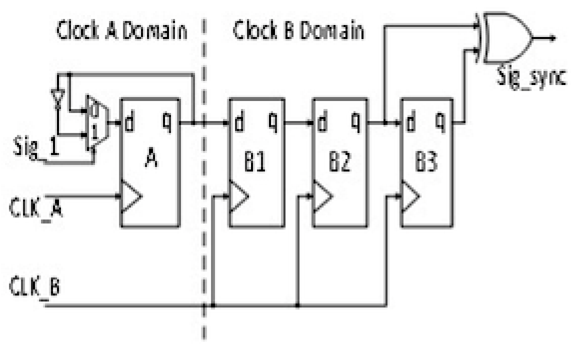

Figure 3 Toggle synchronizer

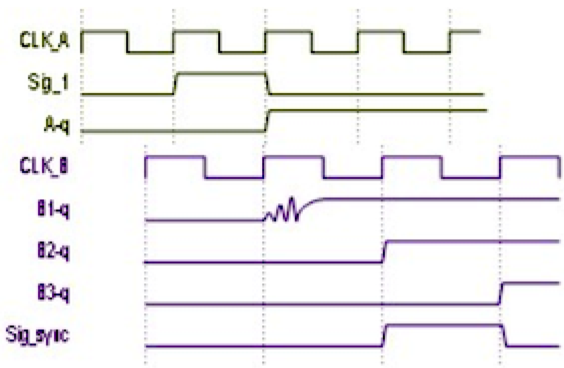

Figure 4 Timing for Toggle synchronizer

Handshake based pulse synchronizer

In handshake based pulse synchronizer, as shown in **Figure 5** and **Figure 6**, synchronization of a pulse generated into source clock domain is guaranteed into destination clock domain by providing an acknowledgement. There is one restriction in pulse synchronizer that back to back (one clock gap) pulses cannot be handled. To make sure the next generated pulse in source clock domain gets definitely transferred and synchronized in the destination clock domain, the handshake based pulse synchronizer generates a “Busy” signal by ORing A1 and A3 flip-flop outputs. Thus the logic generating the pulse shall not generate another pulse till the busy signal is asserted.

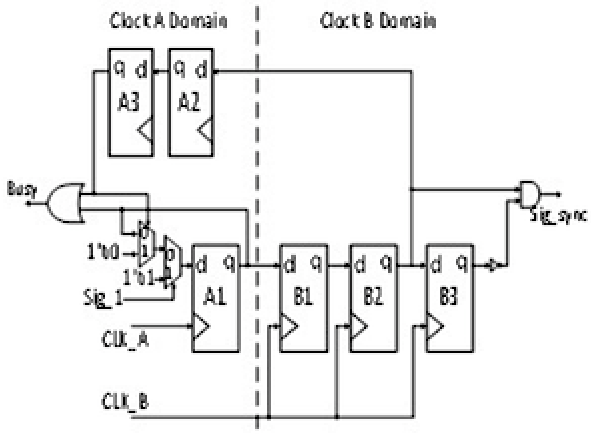

Figure 5 Handshake based pulse synchronizer

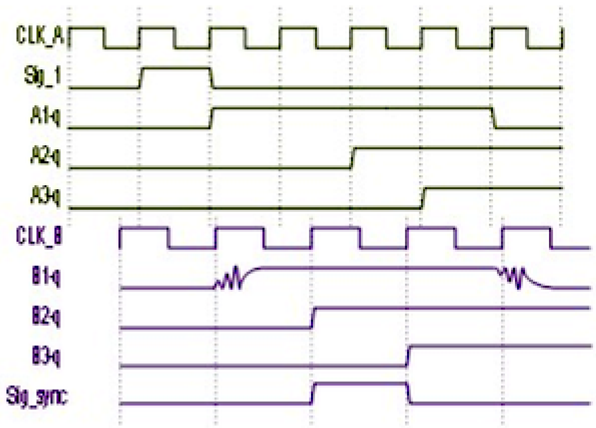

Figure 6 Timing for handshake based pulse synchronizer

Gray encoding for multi bits signal

When multi bit signals are synchronized with 2 flip flop synchronizer, each bit is synchronized using separate 2-FF synchronizer. Metastability can cause a flip flop to settle down either to a true value or a false value. So output of every synchronizer may not settle to correct value at same clock. This causes data incoherency. In order to synchronize multi bit signal using 2 flip flop synchronizer method, only a single bit change must be guaranteed at a particular clock cycle. This can be achieved by gray encoding. So, for example, in asynchronous FIFO design, when we synchronize read pointer value after converting to gray value in write clock domain using 2-FF synchronizer, there is possibility of metastability. As there is only one bit change in the gray encoding so even if there is metastability when clock crossing, the gray counter value will be previous value. For example, read pointer (gray counter) value is changing from 0110 to 0111 and synchronized with write clock then due to metastability (if it occurs) possibility is read pointer still remains 0110. Now, suppose earlier FIFO Full Flag was high at read gray counter value 0110, then FIFO Full will remain high for 1 more clock cycle, but this won't cause an issue, because in next clock cycle the read pointer value will become 0111 and FIFO full flag will get deasserted. If instead of gray counter binary counter is taken from one clock domain to another through two flip flop synchronizer then the multi bit change could cause unpredicted recovery of different bits after metastability (e.g. value change from "1001" to "1010"). The recovered read or write pointer value could be erroneous causing wrong Flag (FIFO full or FIFO empty) generation. **Figure 7** and **Figure 8** show how Binary to gray conversion can help to resolve the issue.

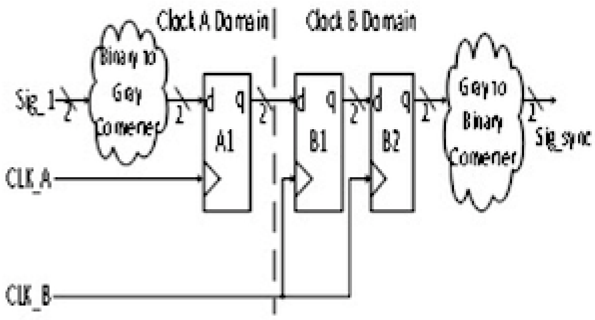

Figure 7 Gray encoding for multi bit signal

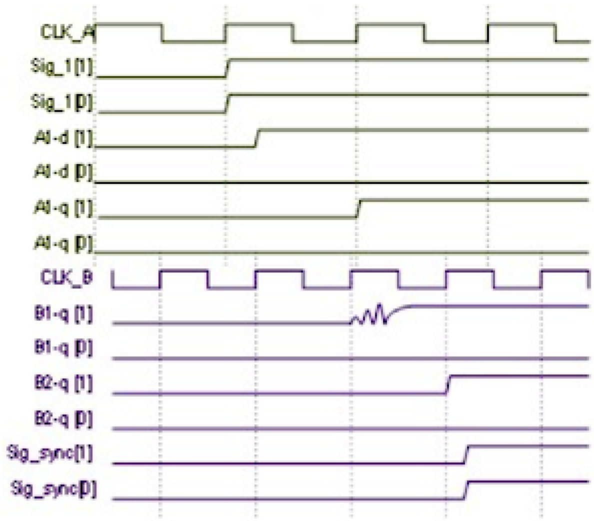

Figure 8 Timing for gray encoding for multi bit signal

Recirculation mux synchronization

For isolated data and where multiple bits can transit at the same time, Recirculation mux synchronization technique shown in **Figure 9** and **Figure 10** is used. In order to synchronize data, a control pulse is generated in source clock domain when data is available at source flop. Control Pulse is then synchronized using 2 flip flop synchronizer or pulse synchronizer (Toggle or Handshake) depending on clock ratio between source and destination domain. Synchronized control pulse is used to sample the data on the bus in destination domain. Data should be stable until it is sampled in destination clock domain.

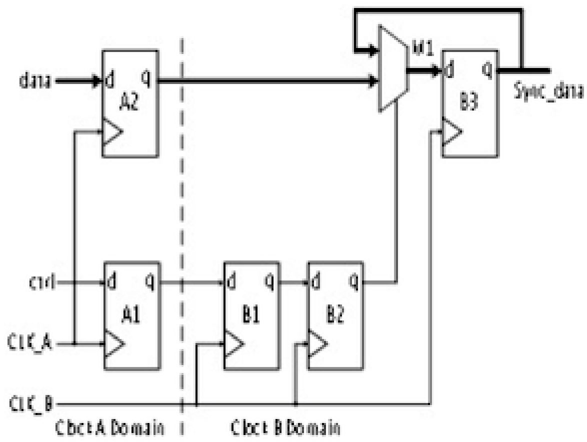

Figure 9 Recirculation mux synchronizer

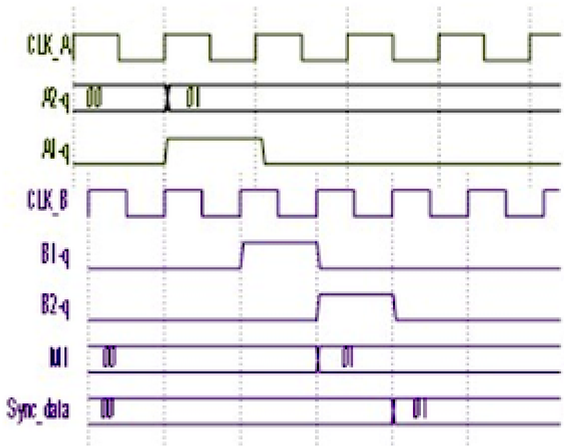

Figure 10 Timing for Recirculation mux synchronizer

Handshake synchronization

In this synchronization scheme request and acknowledge mechanism is used to guarantee the sampling of correct data into destination clock domain irrespective of clock ratio between source clock and destination clock. This technique is mainly used to synchronize vector signal which is not changing continuously or very frequently. As shown in **Figure 12**, data should remain stable on the bus until synchronized Acknowledge signal (A2-q) is received from destination side and it (A2-q) goes low. Diagram in **Figure 11** shows this implementation and **Figure 12** shows timing for handshake synchronizer.

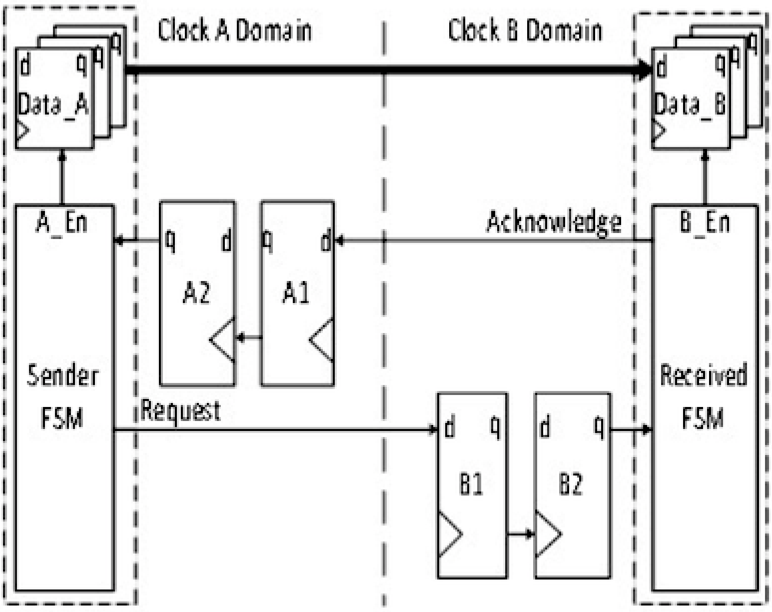

Figure 11 Handshake synchronizer

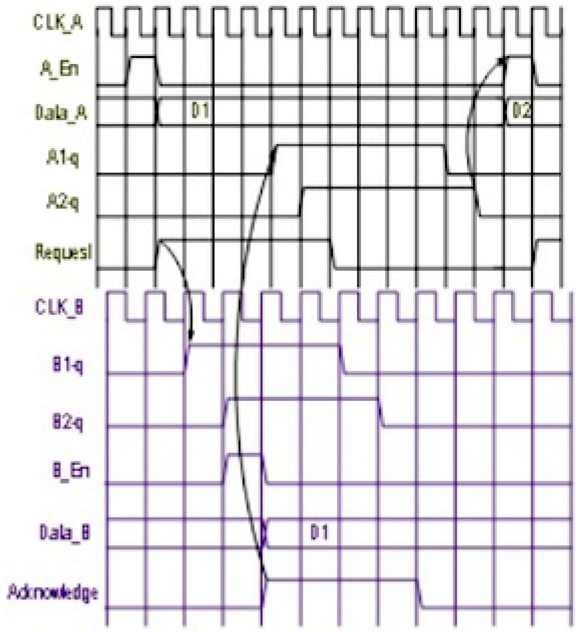

Figure 12 Timing for Handshake Synchronizer

Asynchronous FIFO synchronization

FIFO is best way to synchronize continuously changing vector data between two asynchronous clock domains. Asynchronous FIFO synchronizer offers solution for transferring vector signal across clock domain without risking metastability and coherency problems.

In Asynchronous FIFO design, FIFO provides full synchronization independent of clock frequency. As shown in **Figure 13**, read and write pointers are synchronized to write and read clock domains respectively after performing binary to gray conversion.

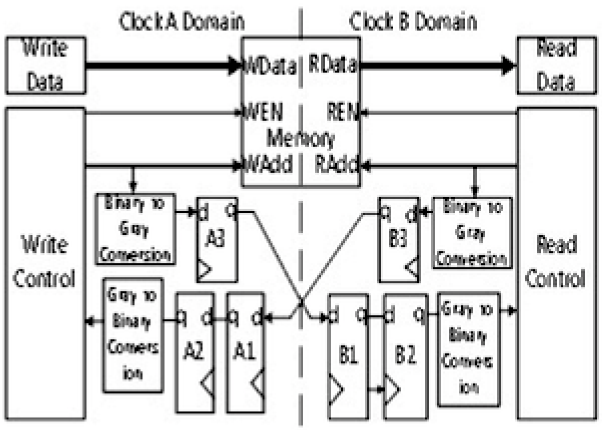

Figure 13 FIFO synchronizer

Next week we will continue with [verification techniques for multi-clock domain SoCs](#).

The authors are engineers at [eInfochips](#).

Also see:

- Efficient analysis of CDC violations in a million gate SoC, part 1
- Efficient analysis of CDC violations in a million gate SoC, part 2