Design of 16-bit, 128-depth Dual Clock Asynchronous FIFO

Jaydev Bapat - 23m1173

Introduction

In electronic circuits, First In First Out Buffers (FIFOs) are frequently used for buffering and flow management between various hardware components or between hardware and software. The main components of a FIFO are storage, control logic, and a set of read and write pointers. Typically, an SRAM with two ports—one for writing and the other for reading—is utilized.

The read and write ports of an asynchronous FIFO buffer run at distinct (unrelated) clock rates. The phrase "clock domain crossing" refers to the process of passing data from one clock domain to another, where it must be written using one clock and read using another.

Meta-stability

Meta-stability is a condition that can occur in digital circuits when setup or hold time violations occur. If a signal is operating at a logic level that cannot be represented as a 0 (low) or a 1 (high), it is said to be in a meta-stable condition. Such signals have the potential to destroy the circuit if they spread.

Timing checks are used when using a single clock to make sure that setup and hold time requirements are fulfilled, ensuring that meta-stability will never happen. It is more difficult to transition between clock domains, though. A signal that passes all timing tests in one clock domain may break the setup or hold time requirements of the other clock, assuming the two clock domains are unrelated. This will definitely cause meta-stable signals to arise, thereby producing complications.

To safely pass a signal from one clock domain to another, synchronizers must be used. A synchronizer is just a flip-flop running on the required clock.

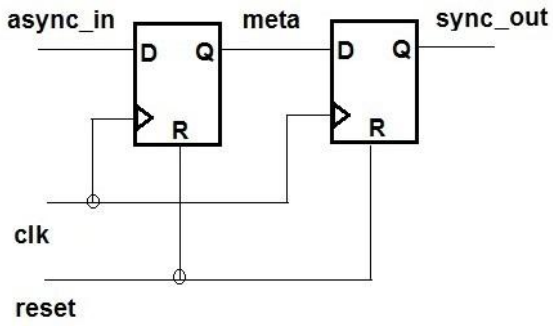

Fig - 1 : Synchronizer for CDC

This sort of synchronizer works only for single bit data change. If the data has multiple bits and all are potentially changing, then the synchronizer doesn't work because different bits can get resolved in different clock cycles.

Because the read and write pointers in an asynchronous FIFO architecture are operating on different clocks, they must be synchronized; that is, the read pointer must be synchronized with the write clock and the write pointer with the read clock. Gray-code counters are employed for synchronization because they guarantee that only one bit will change when pointers are incremented, making the aforementioned technique effective.

*(signal names in the figures throughout the document and that of in the codes are different)

Asynchronous FIFO Design

Problem specifications :

1. Write clock frequency is 100MHz
2. Read clock frequency is 25MHz
3. Both clocks have 50% duty cycle
4. Data width is 16 bits
5. Maximum burst size is 160

It is assumed that data will be written once at every write clock edge and read once at every read clock edge.

Calculation of FIFO Depth:

One data item's writing time is equal to 1/100MHz, or 10ns.

160 × 10 ns = 1600 ns is the amount of time needed to write all the data in the burst.

One data item's reading time is equal to 1/25MHz, or 40ns.

One burst data point will be read every 40 ns. Therefore, 160 data items can be written into the FIFO in 1600 ns, and 1600/40 = 40 data items can be read from the FIFO in the same amount of time.

The FIFO, which is 160 – 40 = 120, must be used to store the remaining data pieces. Thus, 120 data pieces, the minimum depth, must be able to be stored in the FIFO. The depth must be a power of two since the gray-code counter approach is employed to make synchronization easier. The best option is to choose depth = 128

The Asynchronous FIFO consists of the following building blocks:

1. A memory of depth 128 and data width 16-bits.
2. A dual-flop synchronizer to synchronize read address with write clock.
3. A dual-flop synchronizer to synchronize write address with read clock.
4. Read pointer management unit which generates empty condition.
5. Write pointer management unit which generates full condition.

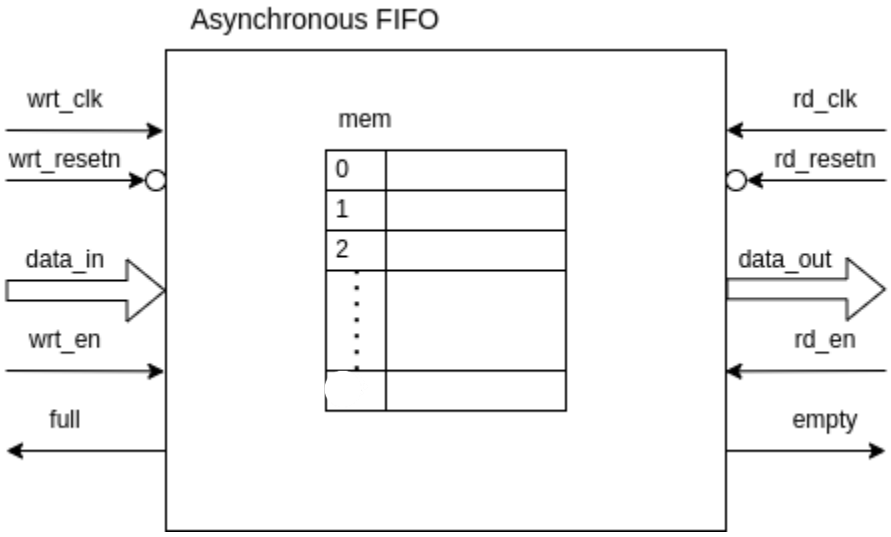

Figure 2: Asynchronous FIFO Module

The data in the module above is the data that will be written into the FIFO at the pace of the wrt clk signal. It arrives in bursts of no more than 160 at a time. The write pointer address is increased using the signal wrt en, which indicates when incoming data must be written into the FIFO.

Likewise, data out refers to the data that has to be read at the pace of the third clk signal from the FIFO. The read pointer address is increased using the signal rd en, which signals when the data stored in the FIFO needs to be read.

Empty and Full Conditions:

Six bits are needed to address the memory because the FIFO's depth of 128 is selected. However, it will be impossible to discern between full and empty states if only six bits are employed. This is due to the fact that read and write pointers must be equal for FIFO to be full and empty.

For this reason, an additional bit is employed as an indicator. The write address has wrapped around and caught up to the read pointer, indicating that the FIFO is full, if the additional bit in the read and write pointers differs. The FIFO is clearly empty if the extra bit is the same for both.

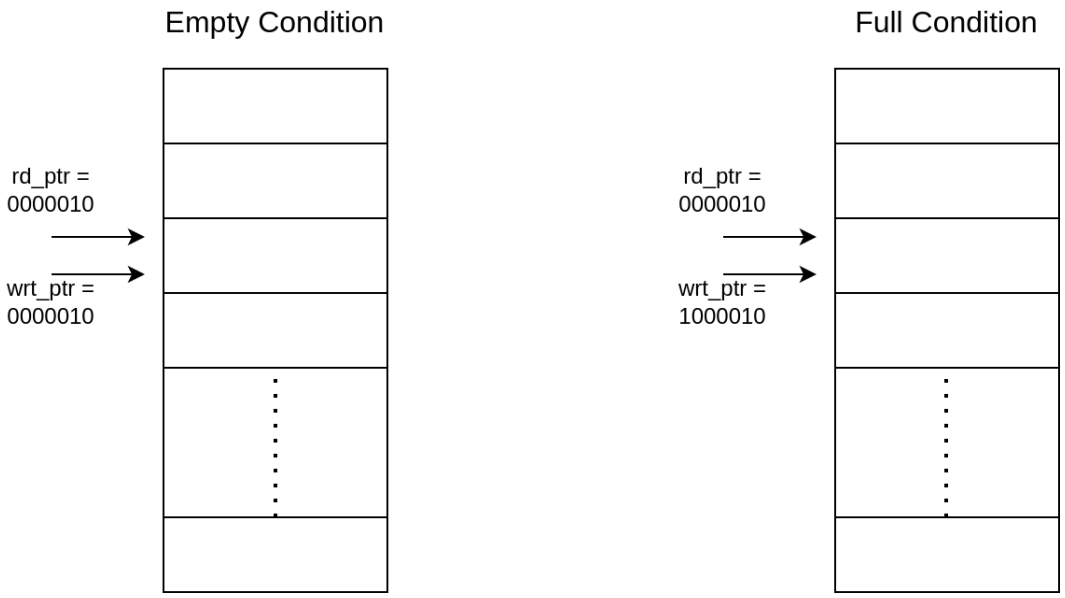

Figure 3 : Empty and Full condition illustration

General expressions for empty and full conditions, if binary coded pointers were used are as follows.

Empty condition : rd ptr[7:0] == wrt ptr[7:0]

Full condition : (rd ptr[6:0] == wrt ptr[6:0]) && (rd ptr[7] != wrt ptr[7])

The creation of full and empty situations must be done using a gray-code counter since binary code is used for memory access while a gray-code counter is required for synchronization with the opposite clock domain.
Keep in mind that the wrapping around check in a gray counter differs from that in a binary counter. Finding patterns is not too difficult. While the remaining bits must be the same, the read pointer's two MSBs must complement the write pointer's two MSBs.

Gray-Code Counter Design:
As previously stated, a gray-code counter and a binary code counter are required for synchronization and addressing, respectively. As a result, this system made use of two registers, an incrementer, and a converter from binary to gray code.
Both the binary and gray code registers have a width of 8.
Every time an enable signal is received and the FIFO is neither full nor empty, the binary register is increased. This binary counter's seven LSB bits are utilized directly to address the FIFO.
The converted value of the increased binary code value is entered into the gray register. This gray counter's seven LSB bits are utilized to create full and empty states as well as to synchronize with the opposite clock domain.

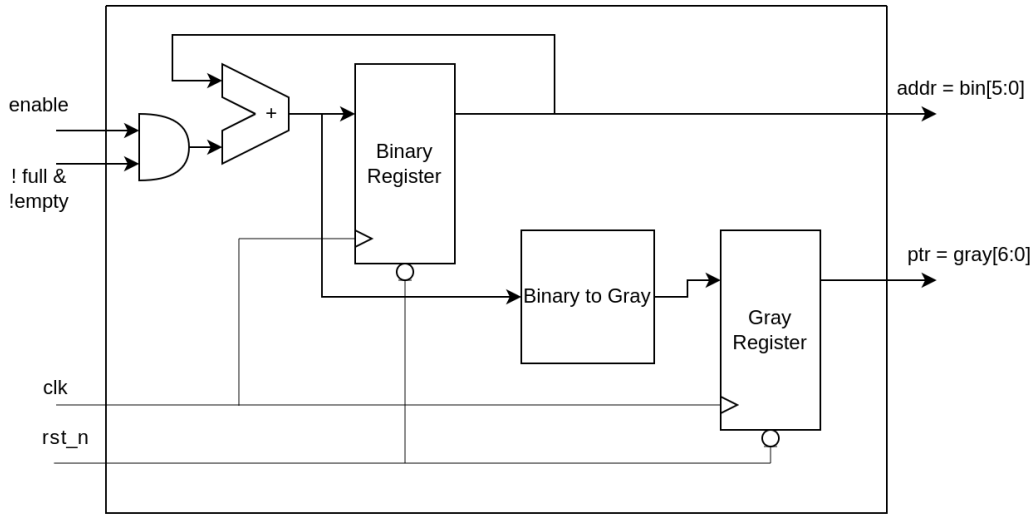

Figure 4 : Binary and Gray pointer circuit

Note that the same structure is used for both read pointer and write pointer.

Hence, 4 pointer registers are used in the entire FIFO design.
The rd ptr and wrt ptr (i.e gray pointers) are passed through dual-flop synchronizers that are clocked with wrt clk and rd clk respectively (opposite
clock domains).

Generating Empty Condition:
The FIFO is empty when the synchronized read pointer (with respect to write clock) is equal to the write pointer. Note that gray code pointers are used here.
empty = (rd ptr[7:0] == wrt2rd ptr[7:0])
Generating Full Condition:
The FIFO is full when the synchronized write pointer (with respect to read clock) has wrapped around and reached the read pointer.
full = (rd ptr[7:0] == {!wrt2rd ptr[7:6], wrt2rd ptr[5:0]})

Example :
Binary read pointer = 0010101 -> Gray code = 0011111
Binary write pointer = 1010101 -> Gray code = 1111111
This should result in Full
In gray codes of pointers, two MSBs are complement of each other and rest all bits are same.

Verilog Modeling and Verification

Verilog is used to model the Asynchronous FIFO previously discussed. Intel Quartus Lite (18.1) is used to synthesize the FIFO, which is then implemented on the MAX-10 FPGA. Below is the synthesized FIFO in RTL perspective.

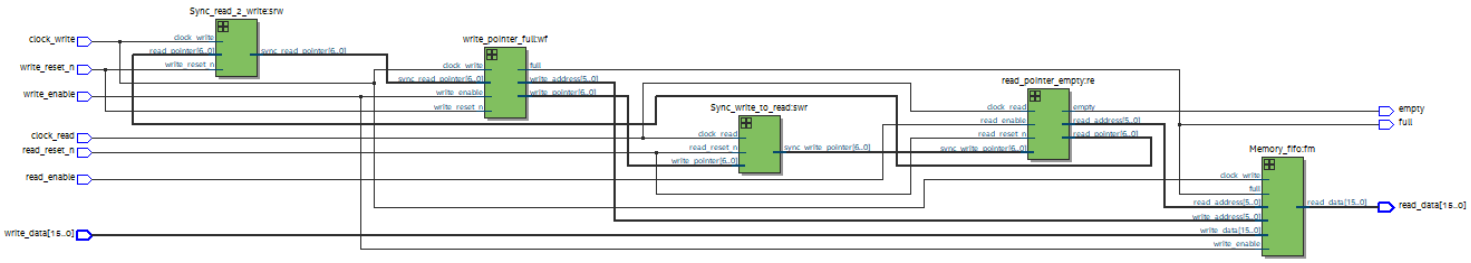

Figure 5 : RTL view of Asynchronous FIFO in Intel Quartus

The Verilog testbench is used to verify that the FIFO is operating as intended. Both of the points are initially reset. Following then, data is read out at a frequency of 25 MHz and written into the FIFO in 100 MHz bursts.
The outcome of the ModelSim Altera gate level simulation is displayed below.

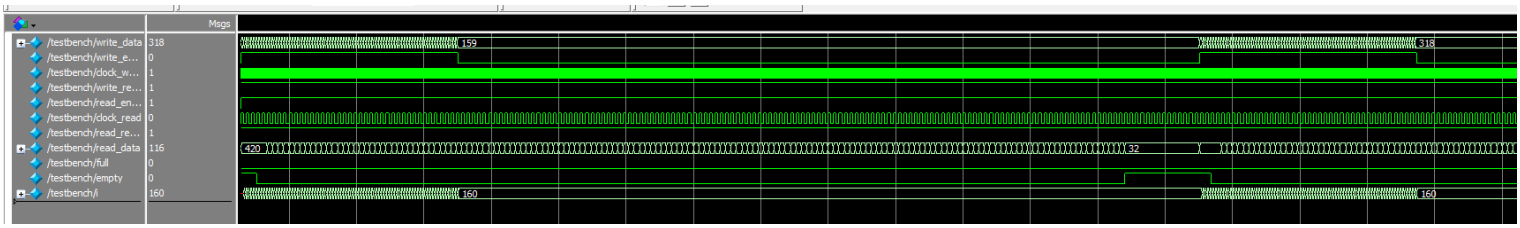

Figure 6: Gate Level Simulation in ModelSim Altera

Note that the FIFO is empty when all the data has been read out, but is never full because the depth is 128, but it only needed a depth of 40 to function properly for the given specifications.
This FIFO should ideally be able handle a write data burst of up-to 80 (if data is also continuously being read). But due to synchronizer delays, it can handle a write data burst of 74 before becoming full.