Asynchronous FIFO Verilog Project

AYUSH YADAV

1 Introduction

A First In, First Out (FIFO) memory is a type of buffer or queue that stores data in the order they are written and allows for reading in the same order. An asynchronous FIFO is used to transfer data between two clock domains that may have different clock frequencies. This document describes the implementation of an asynchronous FIFO using Verilog.

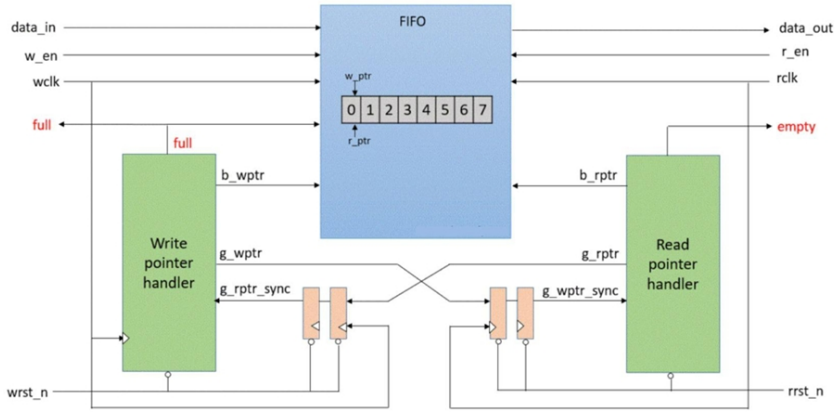

Figure 1: Asynchronous FIFO

2 Verilog Code

The following Verilog code implements an asynchronous FIFO with parameterizable data width and address width.

```
module async_fifo #(parameter DATA_WIDTH = 8, parameter
    ADDR_WIDTH = 4)
(
    input wire wr_clk,          // Write clock
    input wire rd_clk,          // Read clock
    input wire rst,             // Reset signal
    input wire wr_en,           // Write enable
```

```
    input wire rd_en,           // Read enable
    input wire [DATA_WIDTH-1:0] din, // Data input
    output reg [DATA_WIDTH-1:0] dout, // Data output
    output reg full,            // FIFO full flag
    output reg empty            // FIFO empty flag
);

    localparam DEPTH = 1 << ADDR_WIDTH; // FIFO depth

    // FIFO memory
    reg [DATA_WIDTH-1:0] mem [0:DEPTH-1];

    // Write and read pointers
    reg [ADDR_WIDTH:0] wr_ptr = 0;
    reg [ADDR_WIDTH:0] rd_ptr = 0;

    // Gray-coded pointers for synchronization
    reg [ADDR_WIDTH:0] wr_ptr_gray = 0;
    reg [ADDR_WIDTH:0] rd_ptr_gray = 0;
    reg [ADDR_WIDTH:0] wr_ptr_gray_sync1 = 0;
    reg [ADDR_WIDTH:0] wr_ptr_gray_sync2 = 0;
    reg [ADDR_WIDTH:0] rd_ptr_gray_sync1 = 0;
    reg [ADDR_WIDTH:0] rd_ptr_gray_sync2 = 0;

    // Write operation
    always @(posedge wr_clk or posedge rst) begin
        if (rst) begin
            wr_ptr <= 0;
            wr_ptr_gray <= 0;
            full <= 0;
        end else if (wr_en && !full) begin
            mem[wr_ptr[ADDR_WIDTH-1:0]] <= din;
            wr_ptr <= wr_ptr + 1;
            wr_ptr_gray <= (wr_ptr >> 1) ^ wr_ptr; //
                Convert to Gray code
        end
    end

    // Read operation
    always @(posedge rd_clk or posedge rst) begin
        if (rst) begin
            rd_ptr <= 0;
            rd_ptr_gray <= 0;
            empty <= 1;
        end else if (rd_en && !empty) begin
            dout <= mem[rd_ptr[ADDR_WIDTH-1:0]];
            rd_ptr <= rd_ptr + 1;
            rd_ptr_gray <= (rd_ptr >> 1) ^ rd_ptr; //
                Convert to Gray code
        end
```

```
55      end
56
57      // Synchronize write pointer to read clock domain
58      always @(posedge rd_clk or posedge rst) begin
59          if (rst) begin
60              wr_ptr_gray_sync1 <= 0;
61              wr_ptr_gray_sync2 <= 0;
62          end else begin
63              wr_ptr_gray_sync1 <= wr_ptr_gray;
64              wr_ptr_gray_sync2 <= wr_ptr_gray_sync1;
65          end
66      end
67
68      // Synchronize read pointer to write clock domain
69      always @(posedge wr_clk or posedge rst) begin
70          if (rst) begin
71              rd_ptr_gray_sync1 <= 0;
72              rd_ptr_gray_sync2 <= 0;
73          end else begin
74              rd_ptr_gray_sync1 <= rd_ptr_gray;
75              rd_ptr_gray_sync2 <= rd_ptr_gray_sync1;
76          end
77      end
78
79      // Full and empty flag logic
80      always @(*) begin
81          full = (wr_ptr_gray == {~rd_ptr_gray_sync2[
                    ADDR_WIDTH:ADDR_WIDTH-1], rd_ptr_gray_sync2[
                    ADDR_WIDTH-2:0]});
82          empty = (rd_ptr_gray == wr_ptr_gray_sync2);
83      end
84
85  endmodule
```

3 Explanation

3.1 Parameters

- **DATA_WIDTH**: The width of the data to be stored in the FIFO.
- **ADDR_WIDTH**: The width of the address pointers, which determines the depth of the FIFO.

3.2 Ports

- **wr_clk**: Write clock signal.
- **rd_clk**: Read clock signal.

- **rst**: Reset signal.
- **wr_en**: Write enable signal.
- **rd_en**: Read enable signal.
- **din**: Data input.
- **dout**: Data output.
- **full**: FIFO full flag.
- **empty**: FIFO empty flag.

3.3 Internal Signals

- **DEPTH**: The depth of the FIFO, calculated as 2^{ADDR_WIDTH}.
- **mem**: The FIFO memory array.
- **wr_ptr**: Write pointer.
- **rd_ptr**: Read pointer.
- **wr_ptr_gray**: Gray-coded write pointer.
- **rd_ptr_gray**: Gray-coded read pointer.
- **wr_ptr_gray_sync1**: First-stage synchronization register for the write pointer in the read clock domain.
- **wr_ptr_gray_sync2**: Second-stage synchronization register for the write pointer in the read clock domain.
- **rd_ptr_gray_sync1**: First-stage synchronization register for the read pointer in the write clock domain.
- **rd_ptr_gray_sync2**: Second-stage synchronization register for the read pointer in the write clock domain.

3.4 Operation

The FIFO operates with separate write and read clocks. The write operation writes data to the FIFO memory if the write enable signal is high and the FIFO is not full. The read operation reads data from the FIFO memory if the read enable signal is high and the FIFO is not empty. The full and empty flags are updated accordingly. Synchronization between the two clock domains is achieved using Gray-coded pointers and two-stage synchronization registers.

3.4.1 Write Operation

The write operation occurs on the positive edge of the write clock (‘wr_clk‘) or when the reset signal (‘rst‘) is asserted. During a write:

- If ‘rst‘ is high, both the write pointer (‘wr_ptr‘) and the Gray-coded write pointer (‘wr_ptr_gray‘) are reset to 0, and the ‘full‘ flag is cleared.
- If ‘wr_en‘ is high and the FIFO is not full (‘full‘ is low), the data input (‘din‘) is written to the memory at the position indicated by the lower bits of ‘wr_ptr‘, and the write pointer is incremented.
- The Gray-coded write pointer is updated to reflect the new position of the write pointer.

3.4.2 Read Operation

The read operation occurs on the positive edge of the read clock (‘rd_clk‘) or when the reset signal (‘rst‘) is asserted. During a read:

- If ‘rst‘ is high, both the read pointer (‘rd_ptr‘) and the Gray-coded read pointer (‘rd_ptr_gray‘) are reset to 0, and the ‘empty‘ flag is set.
- If ‘rd_en‘ is high and the FIFO is not empty (‘empty‘ is low), the data at the position indicated by the lower bits of ‘rd_ptr‘ is read from the memory and assigned to the data output (‘dout‘), and the read pointer is incremented.
- The Gray-coded read pointer is updated to reflect the new position of the read pointer.

3.4.3 Pointer Synchronization

To ensure data integrity across different clock domains, the Gray-coded write pointer is synchronized to the read clock domain, and the Gray-coded read pointer is synchronized to the write clock domain using two-stage synchronization registers.

3.4.4 Full and Empty Flag Logic

The ‘full‘ flag is asserted when the Gray-coded write pointer equals the Gray-coded read pointer in the read clock domain, with the most significant bit inverted. The ‘empty‘ flag is asserted when the Gray-coded read pointer equals the Gray-coded write pointer in the write clock domain.