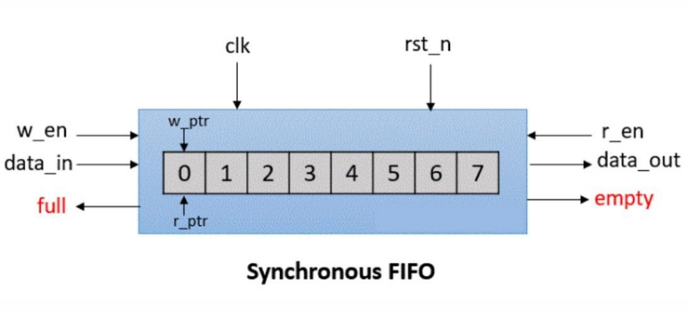
Synchronous FIFO Verilog Project

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. This document describes the implementation of a synchronous FIFO using Verilog.



2 Verilog Code

The following Verilog code implements a synchronous FIFO with parameterizable data width and address width.

```
module synchronous_fifo #(parameter DATA_WIDTH = 8,
    parameter ADDR_WIDTH = 4)
(
    input wire clk,              // Clock signal
    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];

    // Read and write pointers
    reg [ADDR_WIDTH-1:0] rd_ptr = 0;
    reg [ADDR_WIDTH-1:0] wr_ptr = 0;

    // Count of items in the FIFO
    reg [ADDR_WIDTH:0] fifo_count = 0;

    // Write operation
    always @(posedge clk or posedge rst) begin
        if (rst) begin
            wr_ptr <= 0;
            fifo_count <= 0;
            full <= 0;
            empty <= 1;
        end else if (wr_en && !full) begin
            mem[wr_ptr] <= din;
            wr_ptr <= wr_ptr + 1;
            fifo_count <= fifo_count + 1;
            if (fifo_count == DEPTH-1)
                full <= 1;
            empty <= 0;
        end
    end

    // Read operation
    always @(posedge clk or posedge rst) begin
        if (rst) begin
            rd_ptr <= 0;
            fifo_count <= 0;
            empty <= 1;
            full <= 0;
        end else if (rd_en && !empty) begin
            dout <= mem[rd_ptr];
            rd_ptr <= rd_ptr + 1;
            fifo_count <= fifo_count - 1;
            if (fifo_count == 1)
                empty <= 1;
            full <= 0;
        end
    end

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

- **clk**: The clock signal.
- **rst**: The 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.
- **rd_ptr**: Read pointer.
- **wr_ptr**: Write pointer.
- **fifo_count**: The count of items in the FIFO.

3.4 Operation

The FIFO operates on the rising edge of the clock signal. 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.