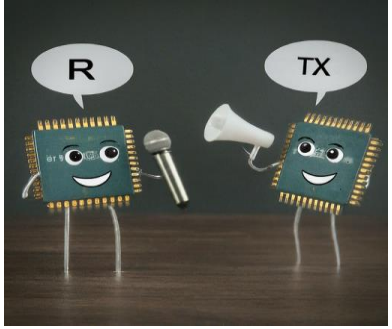
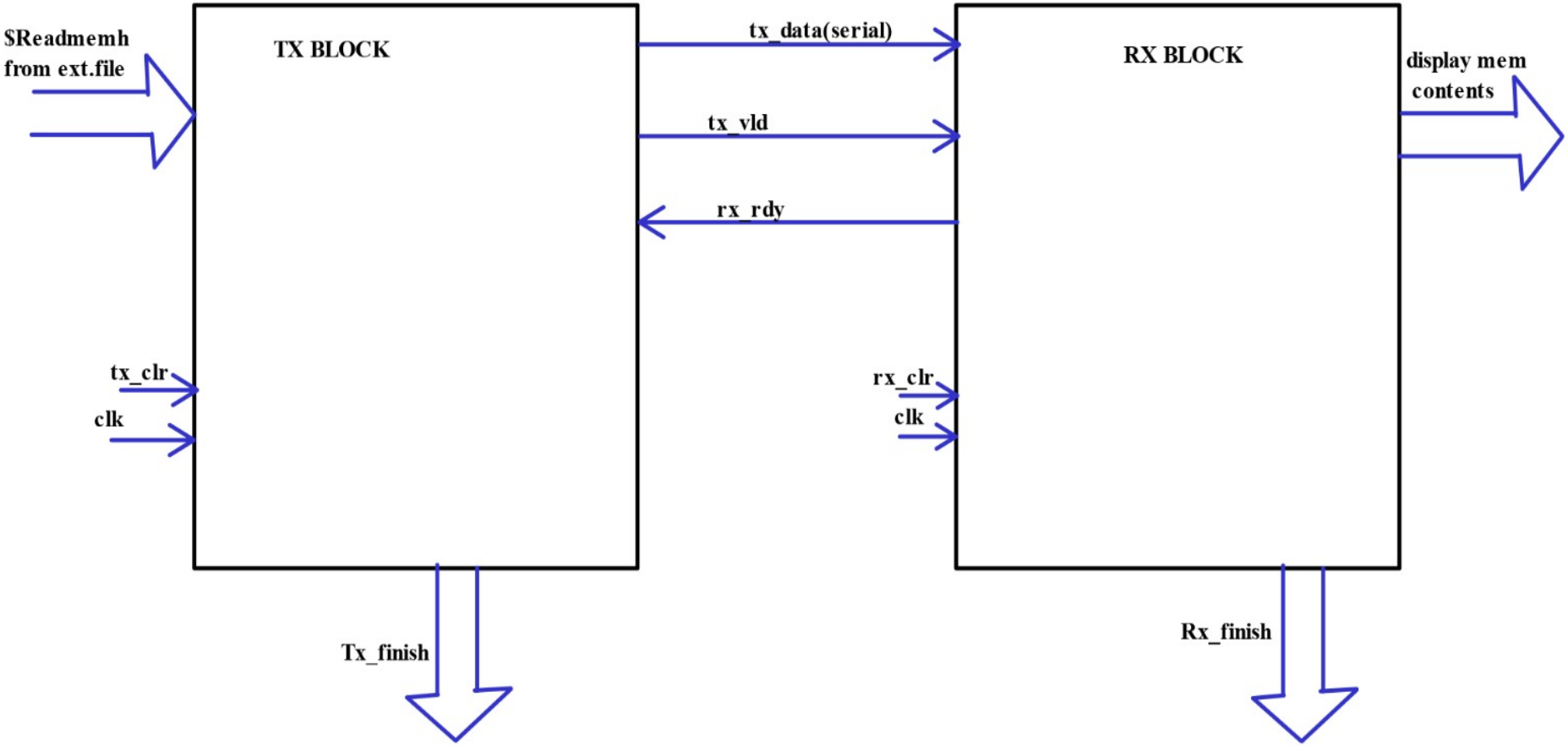
פרוייקטון תכן ספרתיות מתקדם

תבנית הגשה מסכמת

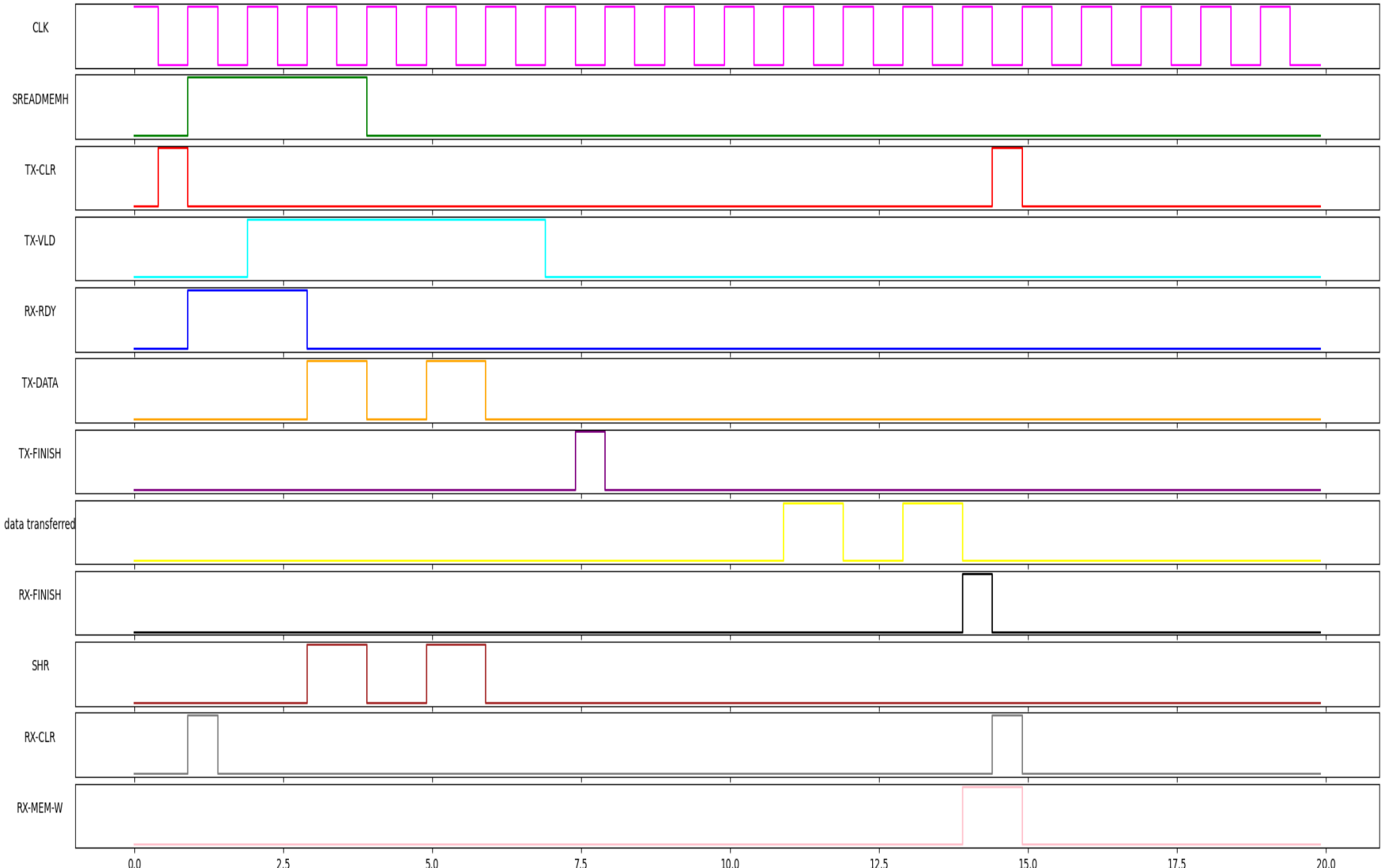

TOP

דיאגרמת בלוקים של TOP



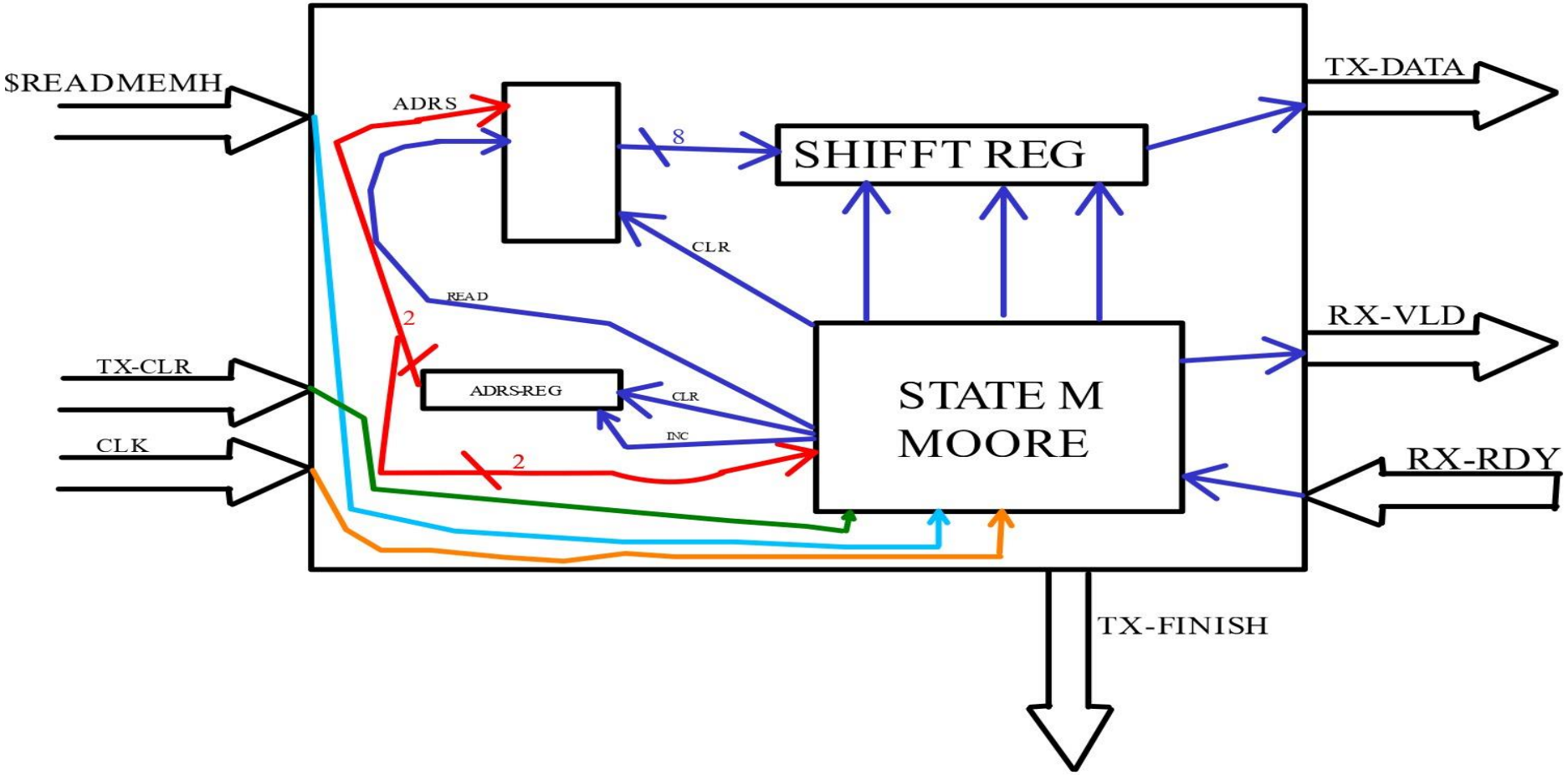


דיאגרמת גלים של TOP

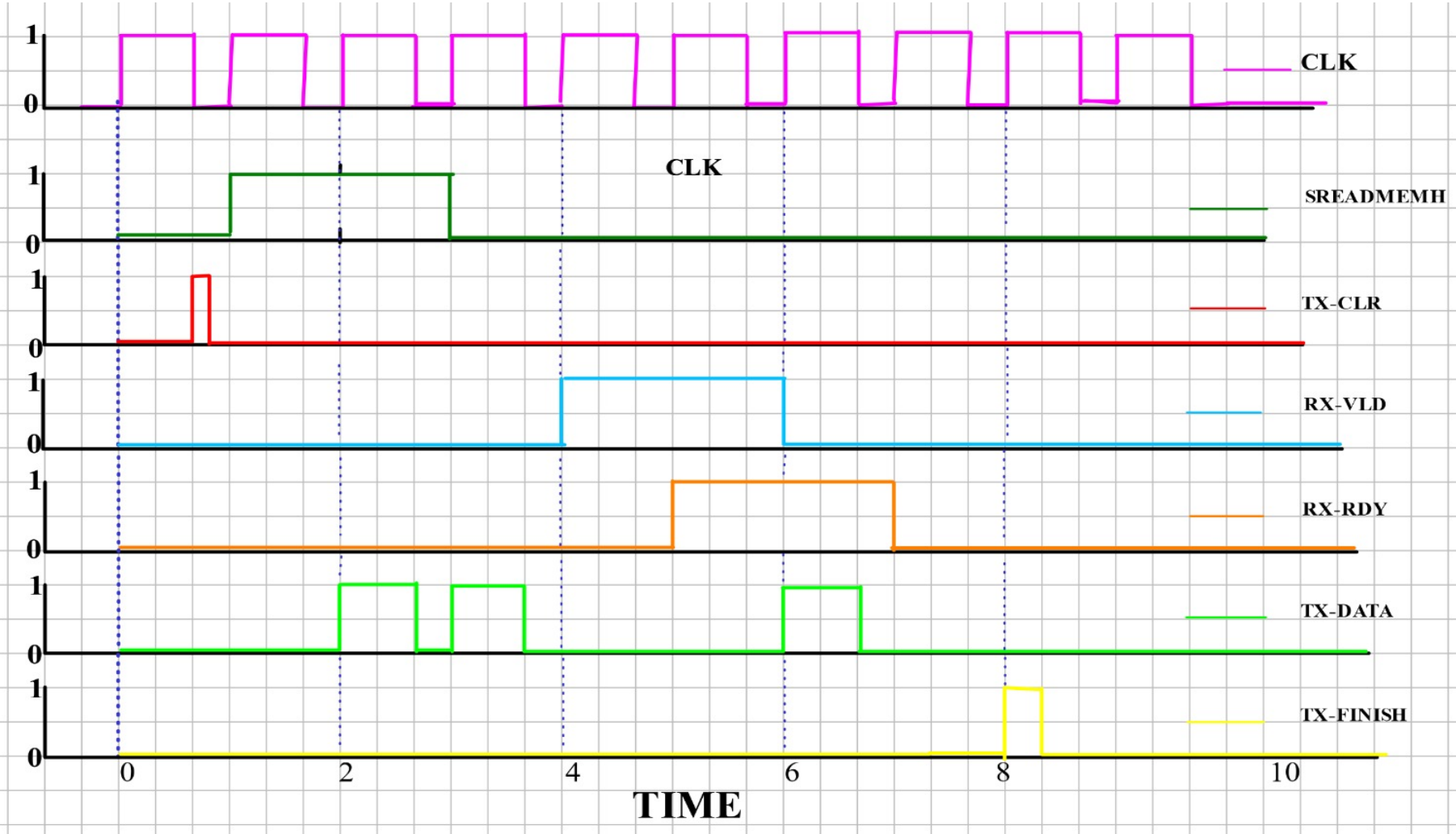



TX

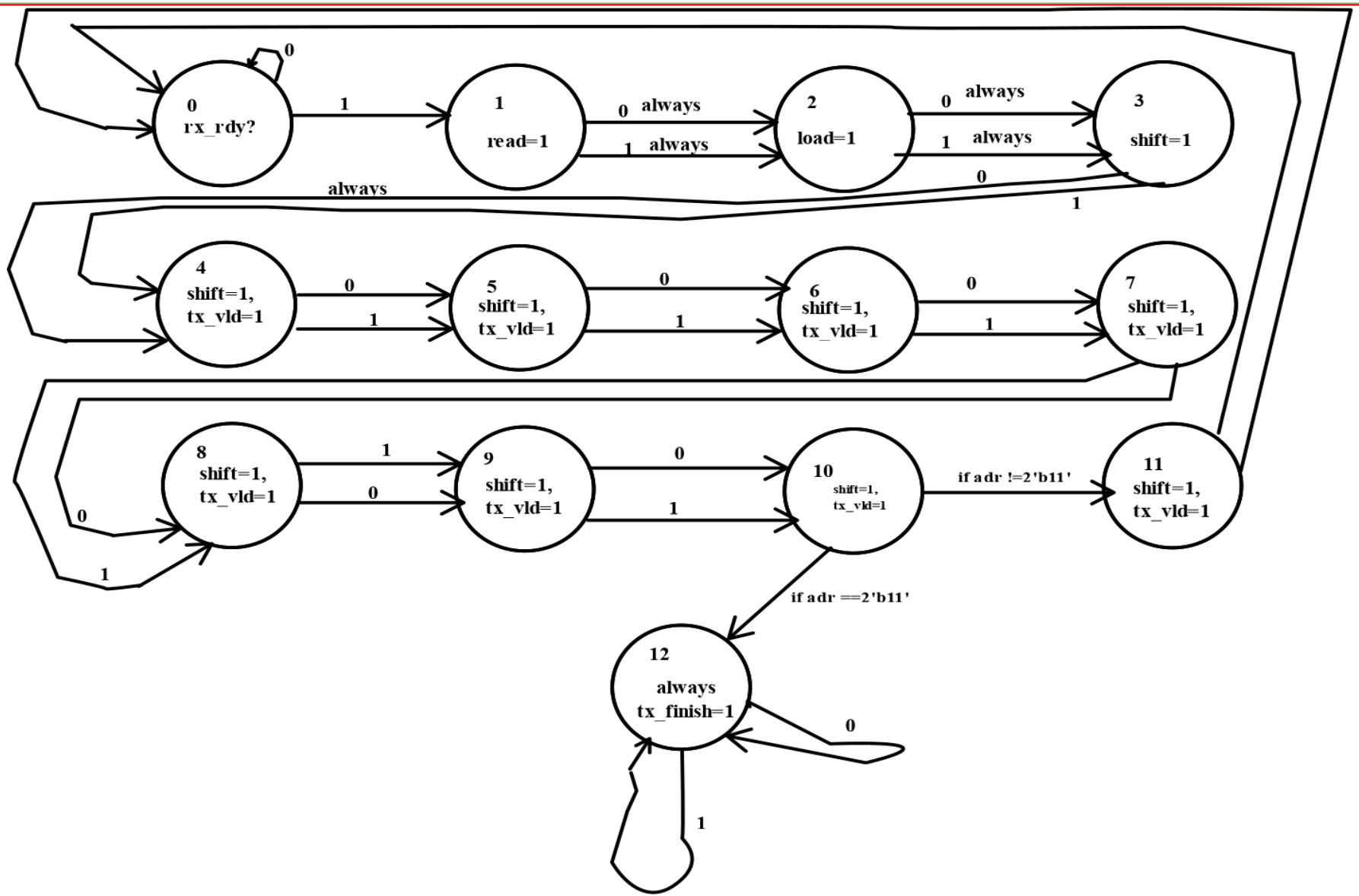
מבנה פנימי יחידת שידור TX



דיגאגרמת גלים של TX



TX SM bubble diagram



TX_block code

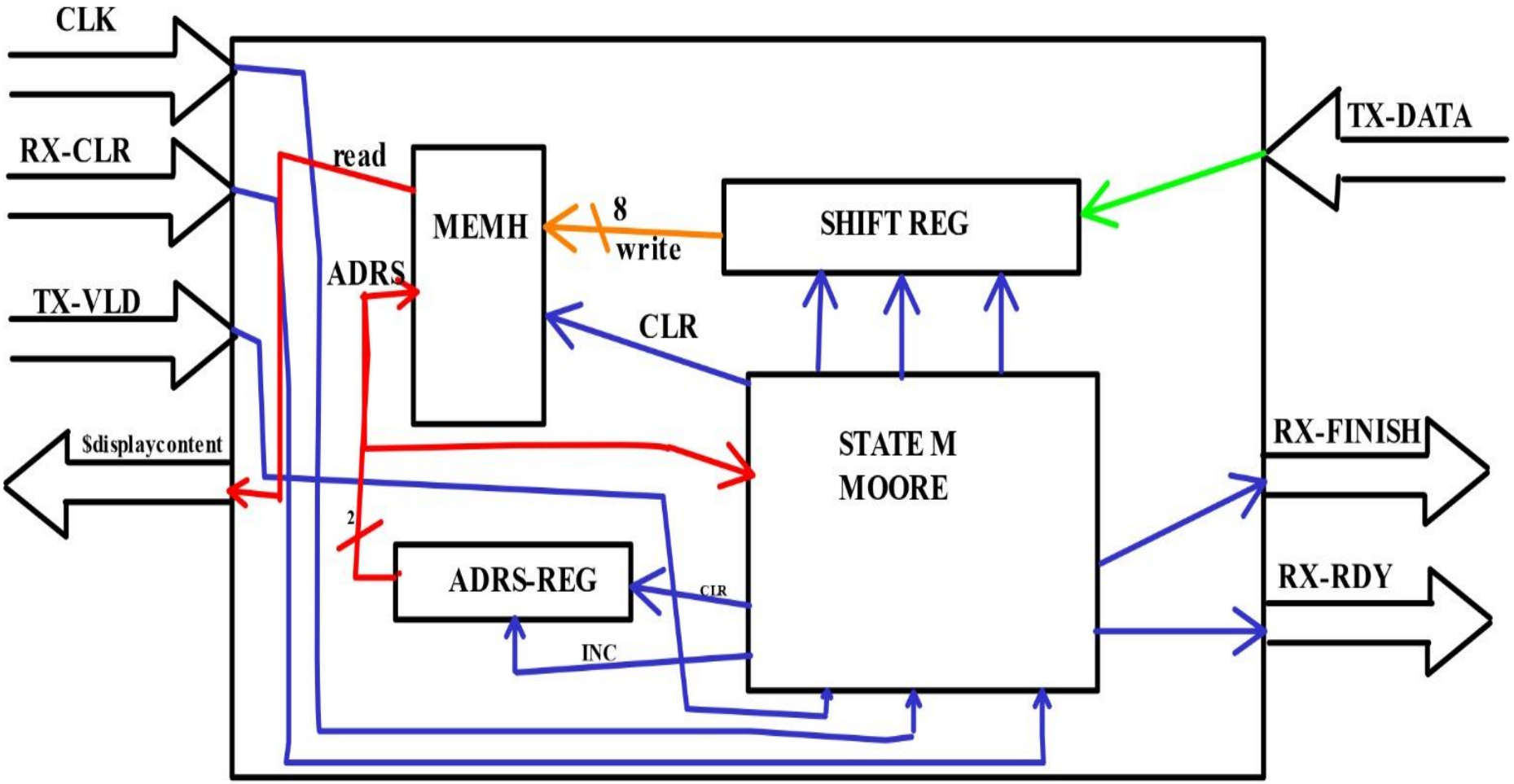
```
`timescale 1ns/1ns
module TX (
    output wire tx_data, // TX data output
    output wire tx_vld, // TX data valid
    output wire tx_finish, // TX finished signal
    input wire clk, // Clock signal
    input wire clr, // Clear signal
    input wire rx_ready // RX ready signal
);

// Internal wires
wire read; // Read enable for the ROM
wire inc; // Increment address for the address register
wire load; // Load enable for the shift register
wire shift; // Shift enable for the shift register
wire [1:0] adr; // 2-bit address for ROM
wire [7:0] data; // Data output from ROM

// Instantiate the state machine for TX
SM_TX state_machine_TX (.read(read), .inc(inc), .load(load), .shift(shift), .tx_vld(tx_vld),
    .tx_finish(tx_finish), .clk(clk), .clr(clr), .rx_ready(rx_ready), .adr(adr));

// Instantiate the ROM module
ROM memory_TX (.addr(adr), .read(read), .clk(clk), .q(data));

// Instantiate the shift register for TX
shift_reg_TX shift_reg_TX (.clk(clk), .clr(clr), .load(load), .shift(shift), .Data(data), .tx_data(tx_data));

// Instantiate the address register for TX
address_reg address_reg_TX (.clk(clk), .clr(clr), .inc(inc), .adrs(adr));

always @(posedge clk or posedge clr) begin
    if (clr) begin
        $display("Time %0t ns: clr is set", $time);
    end
    if (tx_vld) begin
        $display("Time %0t ns: TX data valid - TX_data: %b", $time, tx_data);
    end
    if (address_reg_TX.inc) begin
        $display("Time %0t ns: Address incremented - Address: %0d", $time, address_reg_TX.adrs);
    end
end

endmodule
```

TB_tx code

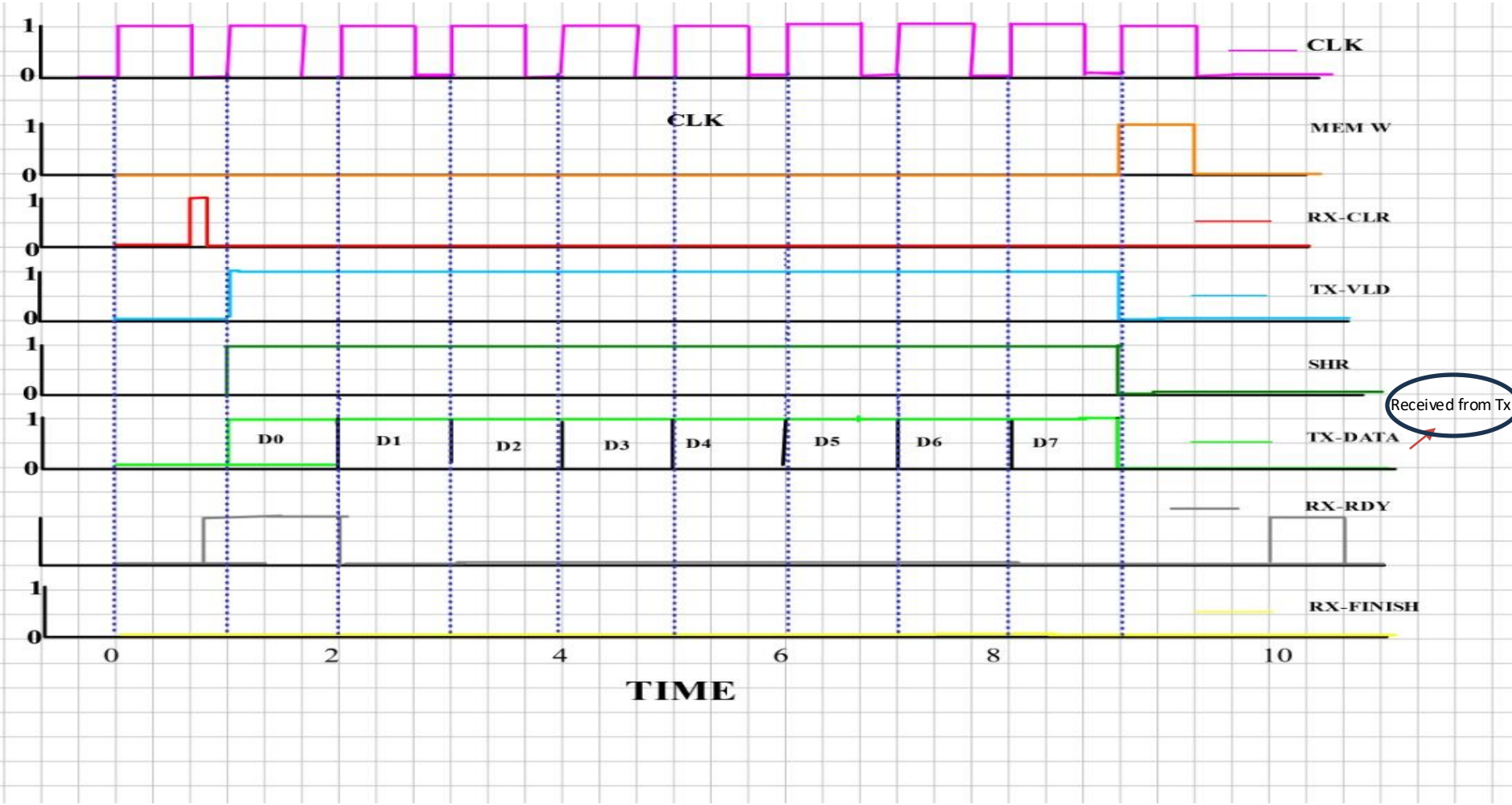
```
`timescale 1ns/1ns // Define the time unit and time precision
module tb_tx ();

// Declare wires and registers for connecting to the DUT (Device Under Test)
wire tx_data_tb; // TX data output from DUT
wire tx_vld_tb; // TX valid output from DUT
wire tx_finish_tb; // TX finish output from DUT
reg clk_tb; // Clock signal for the testbench
reg clr_tb; // Clear signal for the testbench
reg rx_ready_tb; // RX ready signal to DUT

// Instantiate the DUT
TX DUT (.clk(clk_tb),.clr(clr_tb),.rx_ready(rx_ready_tb),.tx_finish(tx_finish_tb),.tx_vld(tx_vld_tb),.tx_data(tx_data_tb));

// Initial block to initialize the signals
initial begin
    clk_tb = 0; // Initialize clock to 0
    clr_tb = 0; // Initialize clear to 0
end

// Generate a clock signal with a period of 10ns (frequency of 100MHz)
always #5 clk_tb = ~clk_tb; // Toggle clock every 5ns

// Initial block to drive the testbench signals
initial begin
    clk_tb = 0; // Initialize clock to 0
    clr_tb = 0; // Initialize clear to 0
    rx_ready_tb = 0; // Initialize RX ready to 0

    repeat (2) @(posedge clk_tb); // Wait for 2 positive edges of the clock
    clr_tb = 1; // Set clear to 1
    repeat (2) @(posedge clk_tb); // Wait for 2 positive edges of the clock
    rx_ready_tb = 1; // Set RX ready to 1
end
endmodule
```

RX

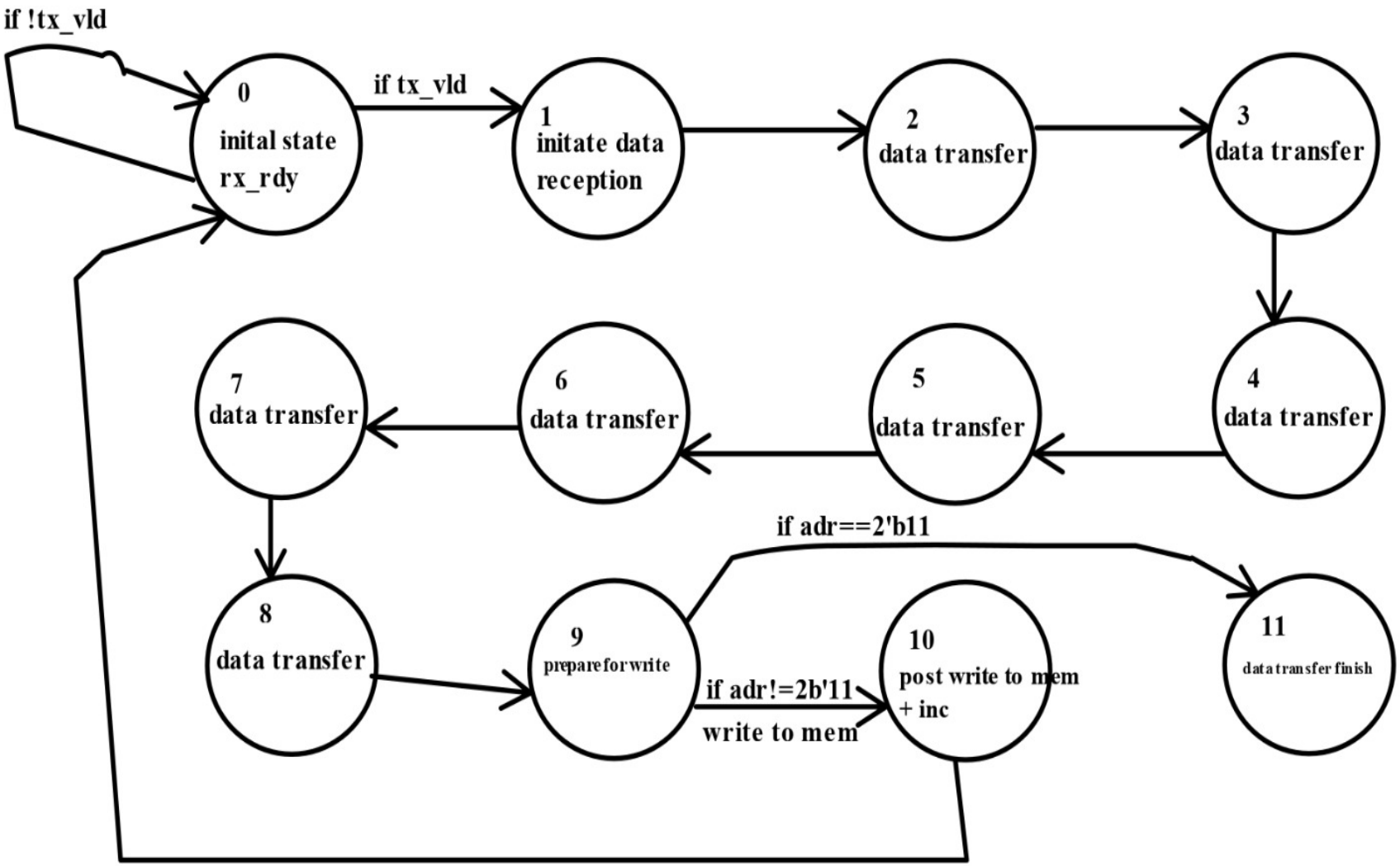
מבנה פנימי יחידת קליטה RX



דיגאגרמת גלים של RX



RX SM bubble diagram



RX_block code

```
`timescale 1ns/1ns
module RX (
    output wire rx_ready, // RX is ready to receive data
    output wire rx_finish, // RX has finished receiving data
    input wire tx_vld, // TX data is valid
    input wire clk, // Clock signal
    input wire clr, // Clear signal
    input wire tx_data // TX data input
);

// Internal wires
wire write; // Write enable for the RAM
wire inc; // Increment address for the address register
wire [1:0] adr; // 2-bit address for RAM
wire [7:0] data; // Data to be written to RAM
wire [7:0] q; // Data output from RAM

// Instantiate the state machine for RX
SM_RX state_machine_RX (.clk(clk), .clr(clr), .Tx_vld(tx_vld), .adr(adr),
    .Rx_ready(rx_ready), .write(write), .inc(inc), .Rx_finish(rx_finish));

// Instantiate the RAM module
RAM memory_RX (.addr(adr), .data(data), .we(write), .clk(clk), .q(q));

// Instantiate the shift register for RX
shift_reg_RX shiftreg_RX (.clk(clk), .clr(clr), .shift(tx_vld), .tx_data(tx_data), .shr(data));

// Instantiate the address register for RX
address_reg address_reg_RX (.clk(clk), .clr(clr), .inc(inc), .adrs(adr));

always @(posedge clk or posedge clr) begin
    if (rx_ready) begin
        $display("Time %0t ns: RX is ready", $time);
    end
end

endmodule
```

TB_rx code

```
`timescale 1ns/1ns // Define the time unit and time precision
module tb_RX ();

// Declare wires and registers for connecting to the DUT (Device Under Test)
wire rx_ready_tb; // RX ready signal from DUT
wire rx_finish_tb; // RX finish signal from DUT
reg tx_vld_tb; // TX valid signal to DUT
reg clk_tb; // Clock signal for the testbench
reg clr_tb; // Clear signal for the testbench
reg tx_data_tb; // TX data signal to DUT

// Instantiate the DUT
RX DUT (.clk(clk_tb),.clr(clr_tb),.rx_ready(rx_ready_tb), .rx_finish(rx_finish_tb),.tx_vld(tx_vld_tb), .tx_data(tx_data_tb));

// Initial block to initialize the signals
initial begin
    clk_tb = 0; // Initialize clock to 0
    clr_tb = 0; // Initialize clear to 0
    tx_vld_tb = 0; // Initialize TX valid to 0
    tx_data_tb = 0; // Initialize TX data to 0
end

// Generate a clock signal with a period of 10ns (frequency of 100MHz)
always #5 clk_tb = ~clk_tb; // Toggle clock every 5ns

// Initial block to drive the testbench signals
initial begin
    clr_tb = 0; // Initialize clear to 0
    tx_vld_tb = 0; // Initialize TX valid to 0
    tx_data_tb = 0; // Initialize TX data to 0

    repeat (2) @(posedge clk_tb); // Wait for 2 positive edges of the clock
    clr_tb = 1; // Set clear to 1
    repeat (2) @(posedge clk_tb); // Wait for 2 positive edges of the clock

    // Set tx_vld_tb and tx_data_tb as needed for the test
    tx_vld_tb = 1; // Example assignment: TX valid signal to 1
    tx_data_tb = 1; // Example assignment: TX data signal to 1
end
endmodule
```

TB

סביבת בדיקה של שני הבלוקים יחד

TB_rx_tx code

```
`timescale 1ns/1ns // Define the time unit and time precision
module TX_RX_tb (); // Testbench module for TX_RX_MAIN

// Declare wires to connect to the DUT (Device Under Test) outputs
wire tx_finish_tb;
wire rx_finish_tb;

// Declare registers to drive the DUT inputs
reg clk_tb;
reg clr_tb;

// Instantiate the DUT and connect the wires and registers to its ports
TX_RX_MAIN DUT (
    .RX_finish(rx_finish_tb),
    .TX_finish(tx_finish_tb),
    .clk(clk_tb),
    .clr(clr_tb)
);

// Initial block to initialize the signals
initial begin
    clk_tb = 0; // Initialize clock to 0
    clr_tb = 0; // Initialize clear to 0
end

// Generate a clock signal with a period of 10ns (frequency of 100MHz)
always #5 clk_tb = ~clk_tb; // Toggle clock every 5ns

// Initial block to drive the clear signal
initial begin
    clk_tb = 0; // Initialize clock to 0
    clr_tb = 0; // Initialize clear to 0

    repeat (2) @(posedge clk_tb); // Wait for 2 positive edges of the clock
    clr_tb = 1; // Set clear to 1
end

endmodule
```

Tx_rx_main

```
`timescale 1ns/1ns
module TX_RX_MAIN (
    output wire RX_finish, // RX finish signal output
    output wire TX_finish, // TX finish signal output
    input wire clk, // Clock signal
    input wire clr // Clear signal
);

// Internal wires to connect TX and RX modules
wire TX_data; // Data output from TX to RX
wire TX_vld; // TX valid signal from TX to RX
wire RX_rdy; // RX ready signal from RX to TX

// Instantiate the TX module
TX TX_1 (
    .tx_data(TX_data),
    .tx_vld(TX_vld),
    .tx_finish(TX_finish),
    .clk(clk),
    .clr(clr),
    .rx_ready(RX_rdy)
);

// Instantiate the RX module
RX RX_1 (
    .rx_ready(RX_rdy),
    .rx_finish(RX_finish),
    .tx_vld(TX_vld),
    .clk(clk),
    .clr(clr),
    .tx_data(TX_data)
);

// Variables for logging
integer i;
integer file;

// Monitor events
always @(posedge clk or posedge clr) begin
    if (clr) begin
        $display("Time %0t ns: clr is set", $time);
    end else begin
        if (RX_rdy) begin
            $display("Time %0t ns: RX is ready", $time);
        end
        if (TX_vld) begin
            $display("Time %0t ns: TX data valid - TX_data: %b", $time, TX_data);
        end
        if (TX_1.address_reg_TX.inc) begin
            $display("Time %0t ns: Address incremented - Address: %0d", $time, TX_1.address_reg_TX.adrs);
        end
    end
end

// Write RX RAM content to a file when RX_finish is set
always @(posedge RX_finish) begin
    file = $fopen("RX_RAM_content.txt", "w");
    $display("Time %0t ns: RX finished", $time);
    for (i = 0; i < 4; i = i + 1) begin
        $fwrite(file, "RX_RAM[%0d] = %0h\n", i, RX_1.memory_RX.ram[i]);
    end
    $fclose(file);
end

endmodule
```

Monitor output log

```
# Time 5 ns: RX is ready
# Time 5 ns: RX is ready
# Time 15 ns: RX is ready
# Time 15 ns: RX is ready
# Time 15 ns: clr is set
# Time 15 ns: RX is ready
# Time 15 ns: clr is set
# Time 25 ns: clr is set
# Time 25 ns: RX is ready
# Time 25 ns: clr is set
# Time 35 ns: clr is set
# Time 35 ns: RX is ready
# Time 35 ns: clr is set
# Time 45 ns: clr is set
# Time 45 ns: RX is ready
# Time 45 ns: clr is set
# Time 55 ns: clr is set
# Time 55 ns: TX data valid - TX_data: 1
# Time 55 ns: RX is ready
# Time 55 ns: clr is set
# Time 65 ns: clr is set
# Time 65 ns: TX data valid - TX_data: 1
# Time 65 ns: clr is set
# Time 75 ns: clr is set
# Time 75 ns: TX data valid - TX_data: 1
# Time 75 ns: clr is set
# Time 85 ns: clr is set
# Time 85 ns: TX data valid - TX_data: 1
# Time 85 ns: clr is set
# Time 95 ns: clr is set
# Time 95 ns: TX data valid - TX_data: 1
# Time 95 ns: clr is set
# Time 105 ns: clr is set
# Time 105 ns: TX data valid - TX_data: 1
# Time 105 ns: clr is set
# Time 115 ns: clr is set
# Time 115 ns: TX data valid - TX_data: 1
# Time 115 ns: clr is set
# Time 125 ns: clr is set
# Time 125 ns: TX data valid - TX_data: 1
# Time 125 ns: clr is set
# Time 135 ns: clr is set
# Time 135 ns: Address incremented - Address: 0
# Time 135 ns: clr is set
```

```
# Time 135 ns: Address incremented - Address: 0
# Time 135 ns: clr is set
# Time 145 ns: clr is set
# Time 145 ns: clr is set
# Time 155 ns: clr is set
# Time 155 ns: clr is set
# Time 165 ns: clr is set
# Time 165 ns: RX is ready
# Time 165 ns: clr is set
# Time 175 ns: clr is set
# Time 175 ns: RX is ready
# Time 175 ns: clr is set
# Time 185 ns: clr is set
# Time 185 ns: RX is ready
# Time 185 ns: clr is set
# Time 195 ns: clr is set
# Time 195 ns: TX data valid - TX_data: 0
# Time 195 ns: RX is ready
# Time 195 ns: clr is set
# Time 205 ns: clr is set
# Time 205 ns: TX data valid - TX_data: 0
# Time 205 ns: clr is set
# Time 215 ns: clr is set
# Time 215 ns: TX data valid - TX_data: 0
# Time 215 ns: clr is set
# Time 225 ns: clr is set
# Time 225 ns: TX data valid - TX_data: 0
# Time 225 ns: clr is set
# Time 235 ns: clr is set
# Time 235 ns: TX data valid - TX_data: 0
# Time 235 ns: clr is set
# Time 245 ns: clr is set
# Time 245 ns: TX data valid - TX_data: 0
# Time 245 ns: clr is set
# Time 255 ns: clr is set
# Time 255 ns: TX data valid - TX_data: 0
# Time 255 ns: clr is set
# Time 265 ns: clr is set
# Time 265 ns: TX data valid - TX_data: 0
# Time 265 ns: clr is set
# Time 275 ns: clr is set
# Time 275 ns: Address incremented - Address: 1
# Time 275 ns: clr is set
# Time 285 ns: clr is set
```

```
# Time 285 ns: clr is set
# Time 295 ns: clr is set
# Time 295 ns: clr is set
# Time 305 ns: clr is set
# Time 305 ns: RX is ready
# Time 305 ns: clr is set
# Time 315 ns: clr is set
# Time 315 ns: RX is ready
# Time 315 ns: clr is set
# Time 325 ns: clr is set
# Time 325 ns: RX is ready
# Time 325 ns: clr is set
# Time 335 ns: clr is set
# Time 335 ns: TX data valid - TX_data: 1
# Time 335 ns: RX is ready
# Time 335 ns: clr is set
# Time 345 ns: clr is set
# Time 345 ns: TX data valid - TX_data: 1
# Time 345 ns: clr is set
# Time 355 ns: clr is set
# Time 355 ns: TX data valid - TX_data: 1
# Time 355 ns: clr is set
# Time 365 ns: clr is set
# Time 365 ns: TX data valid - TX_data: 1
# Time 365 ns: clr is set
# Time 375 ns: clr is set
# Time 375 ns: TX data valid - TX_data: 1
# Time 375 ns: clr is set
# Time 385 ns: clr is set
# Time 385 ns: TX data valid - TX_data: 1
# Time 385 ns: clr is set
# Time 395 ns: clr is set
# Time 395 ns: TX data valid - TX_data: 1
# Time 395 ns: clr is set
# Time 405 ns: clr is set
# Time 405 ns: TX data valid - TX_data: 1
# Time 405 ns: clr is set
# Time 415 ns: clr is set
# Time 415 ns: Address incremented - Address: 2
# Time 415 ns: clr is set
# Time 425 ns: clr is set
# Time 425 ns: clr is set
```

דיאגרמת גלים של כל סביבת העבודה