shift registers

Shift registers load the data present on its inputs and then moves or shifts it to its output once in every clock cycle.

1. Serial Input Serial Output (SISO)

2. Serial Input Parallel Output (SIPO)

3. Parallel Input Serial Output (PISO)

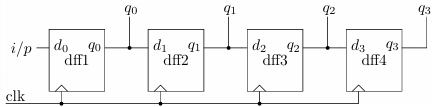
4. Parallel Input Parallel Output (PIPO)

Serial Input Parallel Output (SIPO)

SIPO is a type of shift register which is used to convert a serial data stream into a parallel data stream.

Tinput​=ntff Toutput​=tff

Tlatency​=n⋅tff​



module SIPO #(parameter WIDTH = 4)(

input wire clk, // Clock signal

input wire rst, // Asynchronous reset (active high)

input wire serial\_in, // Serial data input

output reg [WIDTH-1:0] parallel\_out // Parallel data output

);

// Internal register to hold the shift register values

reg [WIDTH-1:0] shift\_reg;

always @(posedge clk or posedge rst)

begin

if (rst) begin

shift\_reg <= 0; // Reset all bits to 0

parallel\_out <= 0; // Reset parallel output to 0

end

else begin

shift\_reg <= {shift\_reg[WIDTH-2:0], serial\_in};

// Shift left

parallel\_out <= shift\_reg; // Update parallel output

end

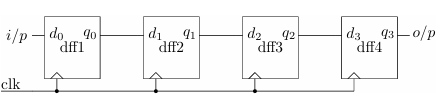
end

endmodule

Serial Input Serial Output (SISO)

SISO is a type of shift register which receives a serial stream of bits or bytes and shifts the stream serially.

Tinput= n tff Toutput= n tff  Tlatency=(n−1)⋅tff​



module SISO #(parameter WIDTH = 4)(

input wire clk, // Clock signal

input wire rst, // Asynchronous reset

input wire serial\_in, // Serial data input

output reg serial\_out // Serial data output

);

// Internal register to hold the shift register values

reg [WIDTH-1:0] shift\_reg;

always @(posedge clk or posedge rst) begin

if (rst) begin

shift\_reg <= 0; // Reset all bits to 0

serial\_out <= 0; // Reset output to 0

end else begin

serial\_out <= shift\_reg[WIDTH-1];

// Output the MSB

shift\_reg <= {shift\_reg[WIDTH-2:0], serial\_in};

// Shift left and insert new input

en

end

endmodule

**Universal Shift Register**

module UniversalShiftRegister (

input clk, // Clock signal

input rst\_n, // Active-low reset

input shift\_left, // Shift left control

input shift\_right, // Shift right control

input load, // Load control

input [3:0] parallel\_in,// 4-bit parallel input

output reg [3:0] q // 4-bit parallel output

);

always @(posedge clk or negedge rst\_n) begin

if (~rst\_n) begin

q <= 4'b0; // Reset the shift register

end else if (load) begin

q <= parallel\_in; // Load parallel data into register

end else if (shift\_left) begin

q <= q << 1; // Shift left

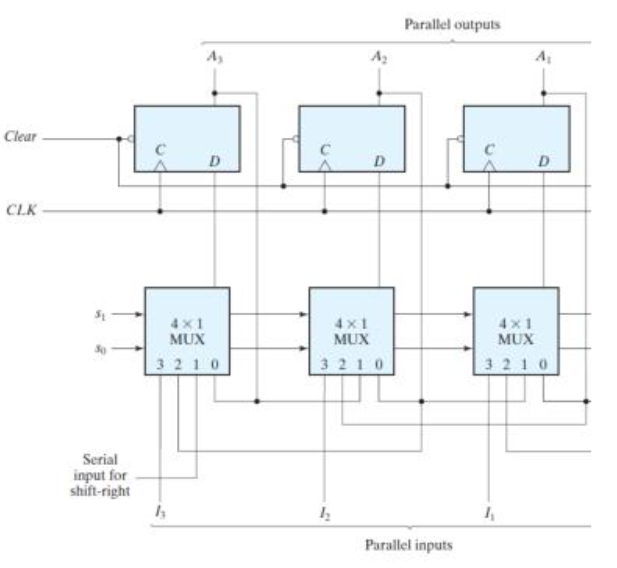
end else if (shift\_right) begin

q <= q >> 1; // Shift right

end

end

endmodule

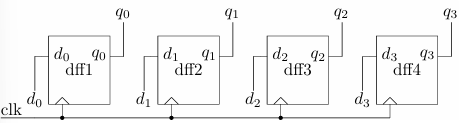


**PIPO (Parallel-In Parallel-Out)**

In PIPO a parallel data stream is input and a parallel data stream is output. PIPO is most popularly known as pipeline register or simply as register. PIPO is used for storing a data for one clock cycle or delaying a data by one clock cycle.

Tinput​=tff Toutput​=tff

Tlatency​=tff



module PIPO (

input clk, // Clock signal

input rst\_n, // Active-low reset

input load, // Load signal (to load parallel data)

input [7:0] parallel\_in,// 8-bit parallel input

output reg [7:0] parallel\_out // 8-bit parallel output

);reg [7:0] shift\_reg; // 8-bit shift register

always @(posedge clk or negedge rst\_n) begin

if (~rst\_n) begin

shift\_reg <= 8'b0; // Reset the shift register

parallel\_out <= 8'b0; // Reset parallel output

end else if (load) begin

shift\_reg <= parallel\_in; // Load parallel data

end else begin

parallel\_out <= shift\_reg;

end

end

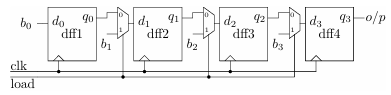
endmodule

**PISO (Parallel-In Serial-Out)**

PISO takes a parallel data stream as input and outputs a serial stream of data. In integrated circuits, due to the limitation of input and output ports parallel data stream is converted to the serial data stream.

Tinput​=tff Toutput​=n⋅tff​

tlatency​=tff​



module PISO (

input clk, // Clock signal

input rst\_n, // Active-low reset

input load, // Load signal (to load parallel data)

input [7:0] parallel\_in,// 8-bit parallel input

output reg serial\_out // Serial output );

reg [7:0] shift\_reg; // 8-bit shift register

always @(posedge clk or negedge rst\_n)

begin

if (~rst\_n) begin

shift\_reg <= 8'b0; // Reset the shift register

serial\_out <= 0; // Reset serial output

end else if (load) begin

shift\_reg <= parallel\_in; // Load parallel data

serial\_out <= shift\_reg[7]; // Output MSB initially

end

else begin

serial\_out <= shift\_reg[7]; // Output MSB of the register

shift\_reg <= shift\_reg << 1; // Shift left

end

end

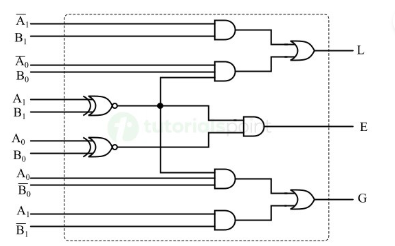
endmodule

**MAGINTUDE COMPARATOR**

A=B: E=(A0⊙B0)(A1⊙B1)

A<B: L=~A1B1+(A1⊙B1)(~A0)B0

A>B: G=A1(~B1)+(A1⊙B1)A0(~B0)



module magComp (  
input [7:0] In1,input [7:0] In2,  
output Gt,output Lt,output Eq);

reg Gt, Lt, Eq;

always @ (\*)  
begin  
 Gt <= (In1 > In2) ? 1’b1 : 1’b0;  
 Lt <= (In1 < In2) ? 1’b1 : 1’b0;  
 Eq <= (In1 == In2) ? 1’b1 : 1’b0;  
 end

endmodule

* IIT KHARAGPUR **: PROF. INDRANIL SENGUPTA** ( HARDWARE MODELING USING VERILOG)

<https://www.youtube.com/watch?v=NCrlyaXMAn8&list=PLJ5C_6qdAvBELELTSPgzYkQg3HgclQh-5>

* **ANKIT GOYAL SIR**: DIGITAL ELECTRONICS

<https://youtube.com/playlist?list=PLs5_Rtf2P2r41iuDKULDHHnIwfXyTAxBH&si=PhMhOvo8_rT1a53y>

* BOOKS:

DIGITAL ELECTRONICS: **Digital Logic and Computer Design by M. Morris Mano.**

VERILOG : **Verilog HDL - Samir Palnitkar**

Digital Design With an Introduction to the Verilog HDL, VHDL, and SystemVerilog by **M. Morris Mano and Michael D. Ciletti**