HDL Code and Timing/Simulation Diagram.

a. Memristor.v

```
module Memristor
    # (
     parameter N=4)
     (
       input reg [N-1:0] neg_inp,
       input reg [N-1:0] memristor_inp,
       output wire [N-1:0] memristor_out
      );
assign memristor_out = (~neg_inp) | memristor_inp;

endmodule
```

b. Parallel Min_Max.v

```
module ParallelMinMax
 # (
  parameter N=4
   )
(
  input reg clk, rst,
  input reg [N-1:0] a,
  input reg [N-1:0] b,
  output wire [N-1:0] min,
  output wire [N-1:0] max
);
reg [N-1:0] C_and;
reg [N-1:0] C_or;
always@(posedge clk)
begin
     if (rst)
           begin
           C_and <= '0;
           C_or <= '0;
     end
     else begin
         //min value calculation by AND operation
           C_and[0] = a[0] & b [0];
```

```
            C_and[1] = a[1] & b [1];
            C_and[2] = a[2] & b [2];
            C_and[3] = a[3] & b [3];
          // max value calculation by OR operation
            C_or[0] = a[0] | b [0];
            C_or[1] = a[1] | b [1];
            C_or[2] = a[2] | b [2];
            C_or[3] = a[3] | b [3];
        end
end

assign min = C_and;
assign max = C_or;

endmodule
```

c. Min_Max_Memristor.v

```
module Min_Max_memristor
 #(
  parameter N=4,
  parameter M=4
 )
 (
   input clk, rst,
   input [3:0] a,
   input [3:0] b,
   output reg [3:0] min,
   output reg [3:0] max
   );

wire [3:0] c;
wire [3:0] c1;
wire [3:0] c2;
wire [3:0] c3;
wire [3:0] c4;
reg [3:0] zero = 4'b0000;

always @ (posedge clk or posedge rst)
begin
    if (rst) begin
      min<= 0;
      max<= 0;
    end
    else begin
```

```
        max<= c1;
        min<= c4;
      end
    end
      Memristor OR1(
            .neg_inp(a),
            .memristor_inp(zero),
            .memristor_out(c));

      Memristor OR2(
            .neg_inp(c),
            .memristor_inp(b),
            .memristor_out(c1));

      Memristor AND1(
            .neg_inp(a),
            .memristor_inp(zero),
            .memristor_out(c2));

      Memristor AND2(
            .neg_inp(b),
            .memristor_inp(c2),
            .memristor_out(c3));

      Memristor AND3(
            .neg_inp(c3),
            .memristor_inp(zero),
            .memristor_out(c4));


    Endmodule
```

b. Sorter_tb

```
`timescale 1ns/1ps

module Sorter_TB();
 parameter N=4;    // individual input width
 parameter M=5;    // number of inputs
 reg [3:0] A, B, C, D;
 wire [3:0] P, Q, R, S;
 reg        clk;
 reg        rst;

 initial clk = 1;
 always #10 clk = ~clk;
```

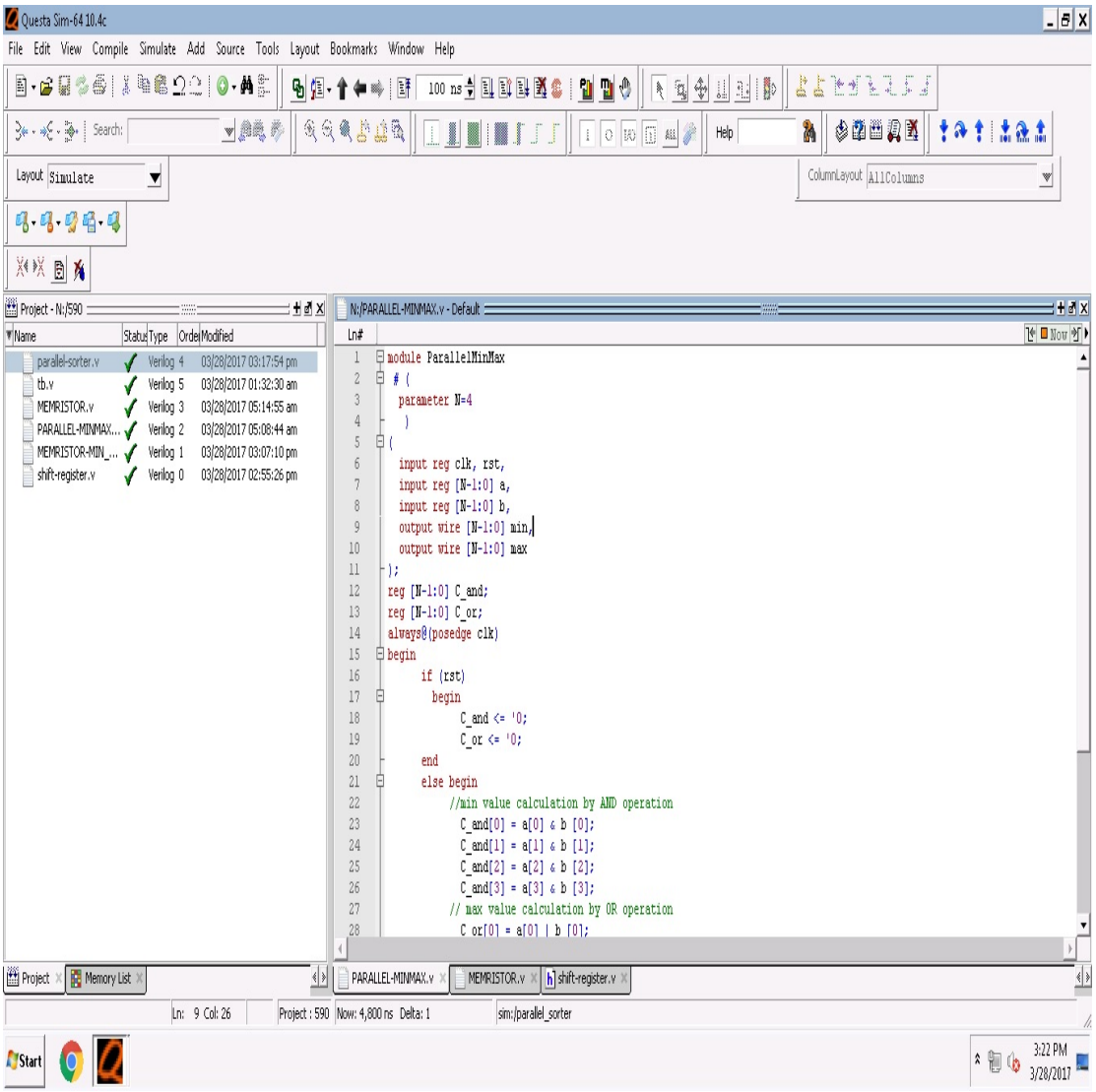
```
  parallel_sorter #(
    .N      (N),
    .M      (M)
  ) sort    (
    .clk    (clk),
    .rst  (rst),
    .A      (A),
    .B      (B),
    .C      (C),
    .D      (D),
    .P      (P),
    .Q      (Q),
    .R      (R),
    .S      (S)
  );

  initial begin
    rst = 1'b1;
    #5  rst = ~rst;
    $display("\n");
    $display("time\trst\tA\tB\tC\tD\t\t\P\tQ\tR\tS");
    $display("=============================================");
    $monitor("%2d\t%0d\t%p\t%p\t%p\t%p\t%p\t%p\t%p\t%p\t%p", $time, rst, A, B, C, D,
P, Q, R, S);
    //$display("%2d\t%0d\t%p\t%p", $time, rst, X, Y);
   @(negedge clk); A = 4'h15; B = 4'h0; C = 4'h7; D = 4'h3 ;
    //$display("%2d\t%0d\t%p\t%p", $time, rst, X, Y);
    //@(negedge clk) X = '{M{0}};
/*
    //$display("%2d\t%0d\t%p\t%p", $time, rst, X, Y);
    #10  X = '{8,7,7,5,1};
    //$display("%2d\t%0d\t%p\t%p", $time, rst, X, Y);
    #10  X = '{4,1,2,3,8};
    //$display("%2d\t%0d\t%p\t%p", $time, rst, X, Y);
    #10  X = '{1,8,3,2,8};
    //$display("%2d\t%0d\t%p\t%p", $time, rst, X, Y);
    #10  X = '{0,0,1,1,8};
    //$display("%2d\t%0d\t%p\t%p", $time, rst, X, Y);
    #10  X = '{8,8,8,8,8};
    //$display("%2d\t%0d\t%p\t%p", $time, rst, X, Y);
*/
    #1000
    $display("=============================================");
    $display("\n\n");
```

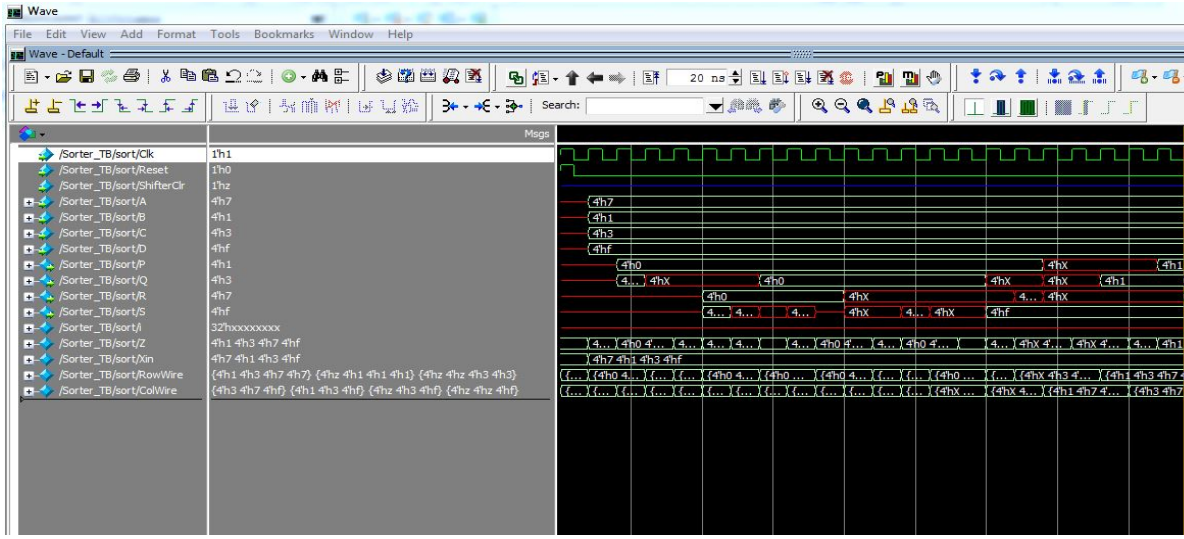
```
  $stop;
 end
endmodule
```

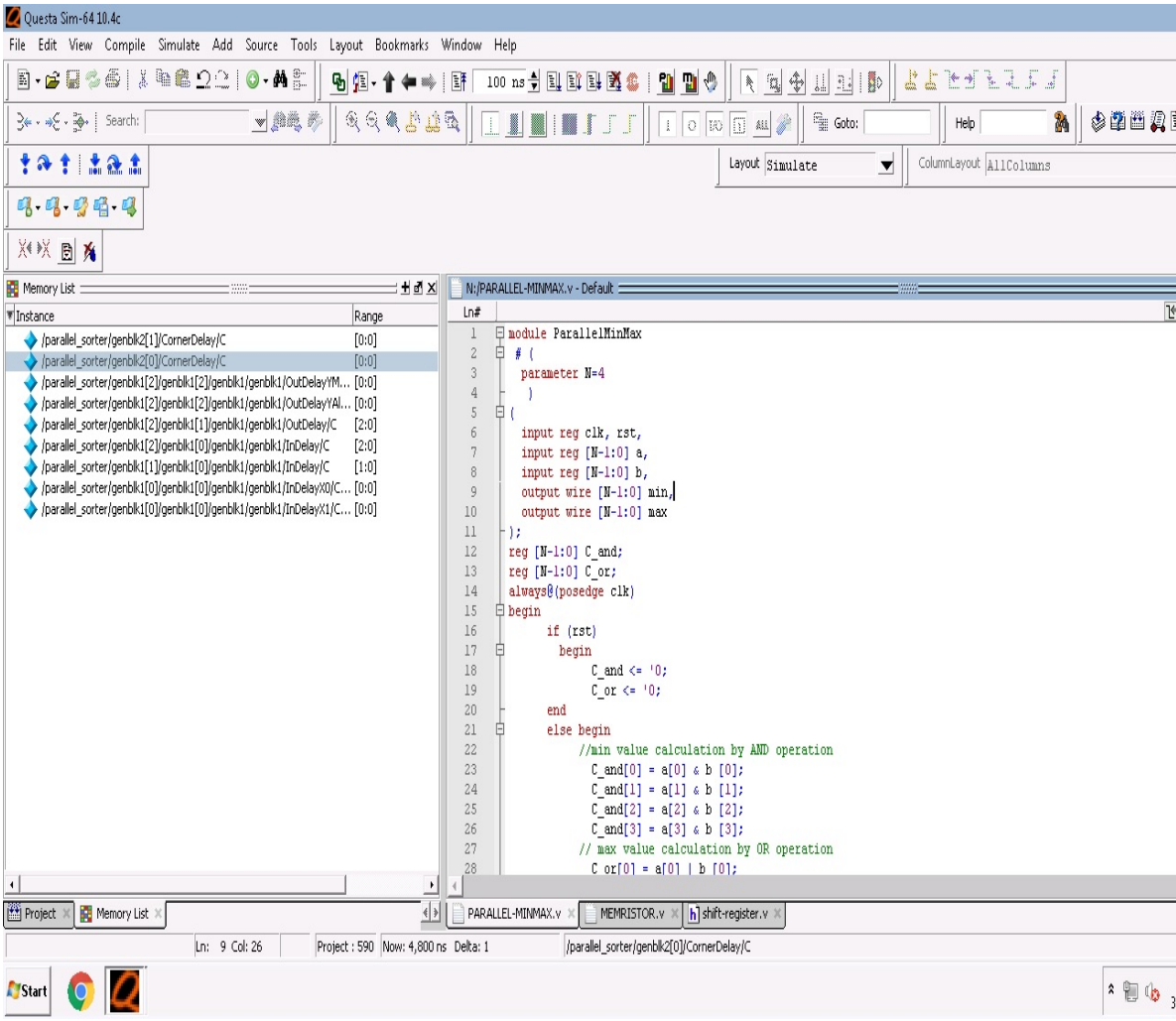
Shift_register.v

```
   module shiftregister
  #(
  parameter N=4, // width
  parameter M=4 // depth
 )(
   input reg Clk,Clr,
   input reg [N-1:0] SRI,
   output wire [N-1:0] SRO
   );

integer i;
reg [N-1:0] C [M-1:0];

always@(posedge Clk) begin
if (Clr)
begin
        for (i=0;i<M;i=i+1)
        begin
                C[i] <= 0;
        end
end
else
begin
  for (i=0;i<M-1;i=i+1)
        begin
        C[i+1] <= C [i];
        C[0] <= SRI;
        end
end
end

assign SRO = C [M-1];
endmodule
```

Timing Diagrams




Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Search:
Msgs
/Sorter_TB/sort/Clk
/Sorter_TB/sort/Reset
/Sorter_TB/sort/ShifterClr
/Sorter_TB/sort/A
/Sorter_TB/sort/B
/Sorter_TB/sort/C
/Sorter_TB/sort/D
/Sorter_TB/sort/P
/Sorter_TB/sort/Q
/Sorter_TB/sort/R
/Sorter_TB/sort/S
/Sorter_TB/sort/i
/Sorter_TB/sort/Z
/Sorter_TB/sort/Xin
/Sorter_TB/sort/RowWire
/Sorter_TB/sort/ColWire


Questa Sim-64 10.4c
File Edit View Compile Simulate Add Source Tools Layout Bookmarks Window Help
Memory List
Instance
Range
N:/PARALLEL-MINMAX.v - Default
module ParallelMinMax
(
parameter N=4
)
(
input reg clk, rst,
input reg [N-1:0] a,
input reg [N-1:0] b,
output wire [N-1:0] min,
output wire [N-1:0] max
);
reg [N-1:0] C_and;
reg [N-1:0] C_or;
always@(posedge clk)
begin
if (rst)
begin
C_and <= '0;
C_or <= '0;
end
else begin
//min value calculation by AND operation
C_and[0] = a[0] & b [0];
C_and[1] = a[1] & b [1];
C_and[2] = a[2] & b [2];
C_and[3] = a[3] & b [3];
// max value calculation by OR operation
C_or[0] = a[0] | b [0];
Project
Memory List
PARALLEL-MINMAX.v
MEMRISTOR.v
shift-register.v
Start