```
module RSFF(output reg q, output reg qn, input s, input r,input
clk, input reset);
reg [1:0]sr;
always @(posedge clk or posedge reset)
      begin
            sr={s,r};
            if(reset==0)
                  begin
                        case (sr)
                              2'd1:q=1'b0;
                              2'd2:q=1'b1;
                              2'd3:q=1'b1;
                        default: begin end
                        endcase
                  end
            else
                  q=1'b0;
      qn=~q;
      end
endmodule

module DFF(input d, input clk, input reset, output q, output qn);
RSFF s(q, qn, d, ~d, clk, reset);
endmodule

module Ripple_Counter(input clk, input reset, output [3:0] out);
wire [3:0] q;
wire [3:0] qn;

DFF dff0 (qn[0], clk, reset, q[0], qn[0]);
DFF dff1 (qn[1], qn[0], reset, q[1], qn[1]);
DFF dff2 (qn[2], qn[1], reset, q[2], qn[2]);
DFF dff3 (qn[3], qn[2], reset, q[3], qn[3]);

assign out = q;
endmodule

module MEM1(output reg [7:0] dout, output reg p, input [2:0]
addrs);

always @(*)
begin
      case(addrs)
            3'b000 : begin dout = 8'h1f;p = 1;end
            3'b001 : begin dout = 8'h31;p = 1;end
            3'b010 : begin dout = 8'h53;p = 1;end
            3'b011 : begin dout = 8'h75;p = 1;end
            3'b100 : begin dout = 8'h97;p = 1;end
            3'b101 : begin dout = 8'hb9;p = 1;end
            3'b110 : begin dout = 8'hdb;p = 1;end
            3'b111 : begin dout = 8'hfd;p = 1;end
            default: dout = 8'bxxxx_xxxx;
```

```
      endcase
end

endmodule

module MEM2(output reg [7:0] dout, output reg p, input [2:0]
addrs);

always @(*)
begin
      case(addrs)
            3'b000 : begin dout = 8'h00;p = 0;end
            3'b001 : begin dout = 8'h22;p = 0;end
            3'b010 : begin dout = 8'h44;p = 0;end
            3'b011 : begin dout = 8'h66;p = 0;end
            3'b100 : begin dout = 8'h88;p = 0;end
            3'b101 : begin dout = 8'haa;p = 0;end
            3'b110 : begin dout = 8'hcc;p = 0;end
            3'b111 : begin dout = 8'hee;p = 0;end
            default: dout = 8'bxxxx_xxxx;
      endcase
end
endmodule

module Mux2To1(out, sel, in1, in2);
  input in1, in2, sel;
  output  out;
  wire  not_sel, a1, a2;
  not g1(not_sel, sel);
  and g2(a1, in1, not_sel);
  and g3(a2, in2, sel);
  or  g4(out, a1, a2);
endmodule


module Mux16To8(out, sel, q1, q2);
      input  [7:0] q1, q2;
      input sel;
      output [7:0] out;
      genvar  j;
  generate  for(j = 0; j < 8; j = j + 1)
    begin:  mux_loop
      Mux2To1 Mux(out[j], sel, q1[j], q2[j]);
    end
  endgenerate
endmodule

module Fetch_Data(output [7:0] dout, output p, input [2:0] addrs,
input sel);
wire [7:0] dout1,dout2;
wire p1,p2;
MEM1 m1(dout1, p1, addrs[2:0]);
```

```
MEM2 m2(dout2, p2, addrs[2:0]);
Mux16To8 mux8(dout, sel, dout1, dout2);
Mux2To1 mux2(p, sel, p1, p2);
endmodule

module Parity_Checker(input [7:0]d, input p, output pout, output
match);
assign pout = d[7] ^ d[6] ^ d[5] ^ d[4] ^ d[3] ^ d[2] ^ d[1] ^
d[0];
assign match = (p==pout)?1:0;

endmodule

module Design(input clk, input reset, output [3:0]
addr_sel,output [7:0] dout, output p, output pout, output match);
Ripple_Counter R(clk,reset,addr_sel);
Fetch_Data d(dout, p, addr_sel[2:0],addr_sel[3]);
Parity_Checker c(dout,p,pout, match);
endmodule


module tb_Ripple_Counter;
reg clk;
reg reset;
wire [3:0] addr_sel;
wire [7:0] dout;
wire p,pout,match;

Design d(clk,reset,addr_sel,dout,p,pout,match);

initial begin
      reset = 1;
      clk = 1'b0;
      #0.5 reset = 0;

end
always
      #0.5 clk = ~clk;

initial begin
      #0 $monitor($time," Address=%4b, data=%8b, parity=%b
detectedParity=%b match=%b",addr_sel,dout,p,pout, match);
      #31 $finish;
end

initial
    begin
        $dumpfile("test.vcd");
       $dumpvars();
       #32 $finish;
    end
```

```
endmodule
```

****

MARKING SCHEME

Ripple Counter (11 M)
- SRFF (5M)
- DFF using SR (2M)
- Ripple_Counter (4M)

Memory Design (4M)
- MEM1 (2M)
- MEM2 (2M)

Data Fetch (10M)
- MUX2To1 (3M)
- MUX16To8 (3M)
- Fetch_Data (4M)

Parity Checker (4M)
Overall Design (3M)
TESTBENCH (3M)
FINAL O/P (5M)