parity.v

```
001: `timescale 1ns / 1ns
002:
003: module parity(
004:     input clk,
005:     input rst_n,
006:     input [31:0] din,
007:     output reg [31:0] dout,
008:     output [3:0] parity
009: );
010: //Assumption taken, even parity
011:
012:
013: wire [7:0] Q0 = din[7:0];
014: wire [7:0] Q1 = din[15:8];
015: wire [7:0] Q2 = din[23:16];
016: wire [7:0] Q3 = din[31:24];
017: reg[2:0] onesQ0;
018: reg[2:0] onesQ1;
019: reg[2:0] onesQ2;
020: reg[2:0] onesQ3;
021:
022: assign parity[0] = onesQ0[0];
023: assign parity[1] = onesQ1[0];
024: assign parity[2] = onesQ2[0];
025: assign parity[3] = onesQ3[0];
026:
027:
028: always @(posedge clk or negedge rst_n) begin
029:   if (!rst_n) begin
030:     dout <= 0;
031:     onesQ0 = 3'b0;
032:     onesQ1 = 3'b0;
033:     onesQ2 = 3'b0;
034:     onesQ3 = 3'b0;
035:   end
036:   else begin
037:     dout <= din;
038:     onesQ0 <= Q0[7] + Q0[6] + Q0[5] + Q0[4] + Q0[3] + Q0[2] + Q0[1] + Q0[0];
039:     onesQ1 <= Q1[7] + Q1[6] + Q1[5] + Q1[4] + Q1[3] + Q1[2] + Q1[1] + Q1[0];
040:     onesQ2 <= Q2[7] + Q2[6] + Q2[5] + Q2[4] + Q2[3] + Q2[2] + Q2[1] + Q2[0];
041:     onesQ3 <= Q3[7] + Q3[6] + Q3[5] + Q3[4] + Q3[3] + Q3[2] + Q3[1] + Q3[0];
042:
043:   end
044: end
045:
046: endmodule
```

Your version is good, just a bit long.

myversion.v

```
`timescale 1ns / 1ps

module parity(
    input clk,
    input rst_n,
    input [31:0] din,
    output reg [31:0] dout,
    output reg [3:0] parity
);

wire p0 = ^din[7:0];
wire p1 = ^din[15:8];
wire p2 = ^din[23:16];
wire p3 = ^din[31:24];


always @(posedge clk or negedge rst_n) begin
  if (!rst_n) begin
    dout <= 0;
    parity <= 0;
  end else begin
    dout <= din;
    parity <= {p3,p2,p1,p0};
  end
end

endmodule
```

tb.v : dont do includes. Includes are bad as a rule. Includes can be used for shared across the board parameters and functions.

for simulation invoke: **iverilog -o tb.vvp -g2012 tb.v parity.v p**