

```

1  /*
2  Non-restoring 32-bit division with quotient placed in Z[31:0] and remainder places in
3  Z[63:32].
4  Operations takes 32 cycles to complete.
5  */
6  module DIV(
7
8      input [31:0] Q, // Dividend
9      input [31:0] M, // Divisor
10     input clk, resetn,
11
12     output [31:0] quotient,
13     output [31:0] remainder
14
15 );
16 reg [63:0] AQ_reg;
17 integer count;
18 wire [31:0] M_signed = (M[31]) ? -M : M;
19
20 always @(posedge clk) begin
21
22     if (~resetn) begin
23         AQ_reg = 64'b0;
24         count = 0;
25
26     end
27
28     else if (count == 0) begin
29         count = count + 1;
30
31         AQ_reg = {32'b0, Q};
32
33     end
34
35     else if (count >= 1 && count <= 32) begin
36         count = count + 1;
37         AQ_reg = AQ_reg << 1;
38         if (AQ_reg[63] == 1'b0) begin
39             AQ_reg[63:32] = AQ_reg[63:32] - M_signed;
40         end
41         else begin
42             AQ_reg[63:32] = AQ_reg[63:32] + M_signed;
43         end
44         AQ_reg[0] = (AQ_reg[63] == 1'b0) ? 1'b1 : 1'b0;
45     end
46
47     else if (AQ_reg[63] == 1) begin
48         AQ_reg[63:32] = AQ_reg[63:32] + M_signed;
49     end
50
51 end
52
53 assign quotient = (M[31]) ? -AQ_reg[31:0] : AQ_reg[31:0];
54 assign remainder = AQ_reg[63:32];
55
56 endmodule

```

```
74
75 module DIV_tb;
76
77     // Declare inputs as reg type
78     reg [31:0] Q;
79     reg [31:0] M;
80     reg clk, resetn;
81
82     // Declare outputs as wire type
83     wire [31:0] quotient;
84     wire [31:0] remainder;
85
86     // Instantiate the DIV module
87     DIV uut (
88         .Q(Q),
89         .M(M),
90         .clk(clk),
91         .resetn(resetn),
92         .quotient(quotient),
93         .remainder(remainder)
94     );
95
96     // Clock generation
97     always begin
98         clk = 0;
99         forever #5 clk = ~clk;
100     end
101
102     initial begin
103
104         resetn = 0;
105
106         @ (posedge clk)
107
108         resetn = 1;
109         Q = 32'd38;
110         M = 32'd6;
111
112         #340;
113
114         @ (posedge clk)
115
116         resetn = 0;
117
118         @ (posedge clk)
119
120         resetn = 1;
121         Q = 32'd100;
122         M = 32'd25;
123         #340;
124
125         @ (posedge clk)
126
127         resetn = 0;
128
129         @ (posedge clk)
130
131         resetn = 1;
132         Q = {0,{31{1'b1}}};
133         M = {0,{31{1'b1}}};
134         #340;
135
136         @ (posedge clk)
137
138         resetn = 0;
139
140         @ (posedge clk)
141
142         resetn = 1;
143         Q = {0,{31{1'b1}}};
144         M = 32'b1;
145         #340;
146
147         @ (posedge clk)
```

```
148
149     resetn = 0;
150
151     @ (posedge clk)
152
153     resetn = 1;
154     Q = 32'b1;
155     M = 32'd50;
156     #340;
157
158     @ (posedge clk)
159
160     resetn = 0;
161
162     @ (posedge clk)
163
164     $stop;
165 end
166
167 endmodule
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
```