```
1  //=============================================================
2  // lcdIp: simple 8-bit HD44780 write engine (instruction/data)
3  // RS=0 for instructions, RS=1 for data. R/W is always 0.
4  //=============================================================
5  module lcdIp (
6      input  wire       clk,
7      input  wire [1:0] userOp,         // 00=INSTR, 01=DATA
8      input  wire       send,
9      input  wire       reset,
10     input  wire [7:0] inputCommand,
11     output reg  [7:0] lcd_data,
12     output reg        lcd_rs,
13     output reg        lcd_rw,
14     output reg        lcd_e,
15     output reg        busy,
16     output reg        systemReady
17 );
18     localparam OP_INSTR = 2'b00;
19     localparam OP_DATA  = 2'b01;
20
21     typedef enum logic [2:0] { INIT_WAIT, IDLE, LOAD, SETUP, E_HIGH, E_LOW, WAIT_DONE }
   state_t;
22     state_t st;
23
24     reg [7:0] cmd_latched;
25     reg [1:0] op_latched;
26     reg [19:0] t; // enough for ~1.5ms at 50MHz
27
28     // timing @50MHz
29     localparam integer T_INIT   = 750_000;  // ~15ms after power-up
30     localparam integer T_SETUP  = 4;        // >=80ns setup (4 cycles @20ns)
31     localparam integer T_E_PW   = 1000;      // 2 µs E high
32     localparam integer T_E_LOW  = 1000;      // 2 µs guard before wait
33     localparam integer T_CMD    = 2_500;    // 50 µs (>37 µs)
34     localparam integer T_CLEAR  = 100_000;  // 2 ms (>1.52 ms)
35
36     wire is_clear_or_home = (op_latched==OP_INSTR) && ((cmd_latched==8'h01) ||
   (cmd_latched==8'h02));
37
38     // FSM
39     always @(posedge clk or posedge reset) begin
40         if (reset) begin
41             st <= INIT_WAIT; t <= T_INIT;
42             lcd_data <= 8'h00;
43             lcd_rs   <= 1'b0;
44             lcd_rw   <= 1'b0;
45             lcd_e    <= 1'b0;
46             busy     <= 1'b1;
47             systemReady <= 1'b0;
48             cmd_latched <= 8'h00;
```

```
49              op_latched  <= OP_INSTR;
50          end else begin
51              // defaults
52              lcd_rw <= 1'b0; // always write
53              systemReady <= (st==IDLE);
54
55              case (st)
56                  INIT_WAIT: begin
57                      busy <= 1'b1; lcd_e <= 1'b0;
58                      if (t!=0) t <= t-1;
59                      else begin st <= IDLE; busy <= 1'b0; end
60                  end
61
62                  IDLE: begin
63                      busy <= 1'b0; lcd_e <= 1'b0;
64                      if (send) begin
65                          cmd_latched <= inputCommand;
66                          op_latched  <= userOp;
67                          st <= LOAD;
68                      end
69                  end
70
71                  LOAD: begin
72                      // Place RS and DATA and then wait a few cycles before E↑
73                      busy     <= 1'b1;
74                      lcd_rs   <= (op_latched==OP_DATA);
75                      lcd_data <= cmd_latched;
76                      t        <= T_SETUP;
77                      st       <= SETUP;
78                  end
79
80                  SETUP: begin
81                      // Meet HD44780 address/data setup time before toggling E
82                      if (t!=0) t <= t-1;
83                      else begin
84                          t  <= T_E_PW;
85                          st <= E_HIGH;
86                      end
87                  end
88
89                  E_HIGH: begin
90                      lcd_e <= 1'b1;
91                      if (t!=0) t <= t-1;
92                      else begin
93                          t  <= T_E_LOW;
94                          st <= E_LOW; // falling edge latches into LCD
95                      end
96                  end
97
98                  E_LOW: begin
```

```
                    lcd_e <= 1'b0;
                    if (t!=0) t <= t-1;
                    else begin
                        t  <= is_clear_or_home ? T_CLEAR : T_CMD;
                        st <= WAIT_DONE;
                    end
                end

                WAIT_DONE: begin
                    if (t!=0) t <= t-1;
                    else begin
                        st <= IDLE; busy <= 1'b0;
                    end
                end
            endcase
        end
    end
endmodule
```