```
1  // unsigned multiplier (non fast)
2
3  module mul(clk, reset, start, word1, word2, product, ready);
4          input clk, reset, start;
5          input [3:0] word1, word2;
6          output [7:0] product;
7          output ready;
8
9          wire  m0, load, shift, add;
10
11         datapath u1 (clk, reset, load, shift, add, word1, word2, product, m0);
12         controller u2 (clk, reset, start, m0, load, shift, add, ready);
13
14 endmodule
15
16 module datapath (clk, reset, load, shift, add, word1, word2, product, m0);
17         input clk, reset, load, shift, add;
18         input [3:0] word1, word2;
19         output [7:0] product;
20         output m0;
21
22         reg [7:0] product;
23         reg carry;
24         reg [3:0] multiplicand;
25         wire[4:0] sum;                  // 5-bit
26
27         assign m0 = product[0];
28         assign sum = product[7:4] + multiplicand;
29
30         always @ (posedge clk or posedge reset) begin
31                 if (reset) begin multiplicand <= 0; product <= 0; end
32         else if (load) begin
33                         multiplicand <= word1;
34                         product <= {4'b0, word2};       // lower 4-bit = multiplier
35                 end
36                 else if (shift)                 // shift-right
37                         begin product <= {carry, product[7:1]}; carry <= 0; end
38                 else if (add)   // add & shift right
39                         {carry, product[7:4]} <= sum;
40         end
41 endmodule
42
43 module controller (clk, reset, start, m0, load, shift, add, ready);
44         input clk, reset, start, m0;
45         output load, shift, add, ready;
46
47         reg [3:0] state;
48         localparam S0=0, S1=8, S2=9, S3=10, S4=11, S5=12, S6=13, S7=14, S8=15;
49
50         always @ (posedge clk or posedge reset)         // State transitions (with register)
51                 if (reset) state <= S0;
52                 else
53                         case (state)
54                          S0: if (start) begin state <= S1; end
55                          S1: if (m0) state <= S2; else state <= S3;
56                          S2: state <= S3;
57                          S3: if (m0) state <= S4; else state <= S5;
58                          S4: state <= S5;
59                          S5: if (m0) state <= S6; else state <= S7;
60                          S6: state <= S7;
61                          S7: if (m0) state <= S8; else state <= S0;
62                          S8: state <= S0;
63                         endcase
64
65  // control output logic
66         assign load = (state==S0) & start;
67         assign shift = (state==S1||state==S3||state==S5||state==S7) & ~m0 |
           (state==S2||state==S4||state==S6||state==S8);
```

```
68          assign add = (state==S1||state==S3||state==S5||state==S7) & m0;
69          assign ready = (state==S0) & ~reset;
70  endmodule
71
```