```
// File: bin2bcd.v
// John Hubbard, 24 Oct 2014
// hw5 assignment for Verilog 0764 class: Problem 4.4.7: Binary to BCD converter
//

// Structural Verilog version, as required by the problem statement:

// Attribution: this implementation is lifted right out of the classic
// shift-and-add-3 algorithm, which I would not have easily figured out on my
// own. There was a sample implementation, and this is essentially the same
// as that. Given more time, I would have loved to solve this from first
// principles, but I'm at the 8 or 10 hour point already with the overall
// homework assignment, so this will have to do.
//
// Anyway, it was interesting getting the test bench set up, and debugging.
// It does work perfectly now. --john hubbard, 25 Oct 2014 / 12:17am
//
module bin2bcd(bin, bcdHundreds, bcdTens, bcdOnes);
    input [7:0]bin;
    output [1:0]bcdHundreds;
    output [3:0]bcdTens;
    output [3:0]bcdOnes;

    wire [3:0] c1, c2, c3, c4, c5, c6, c7;
    wire [3:0] d1, d2, d3, d4, d5, d6, d7;

    assign d1 = {1'b0, bin[7:5]};
    assign d2 = {c1[2:0], bin[4]};
    assign d3 = {c2[2:0], bin[3]};
    assign d4 = {c3[2:0], bin[2]};
    assign d5 = {c4[2:0], bin[1]};
    assign d6 = {1'b0, c1[3], c2[3], c3[3]};
    assign d7 = {c6[2:0], c4[3]};

    shift_add3 m1(d1, c1);
    shift_add3 m2(d2, c2);
    shift_add3 m3(d3, c3);
    shift_add3 m4(d4, c4);
    shift_add3 m5(d5, c5);
    shift_add3 m6(d6, c6);
    shift_add3 m7(d7, c7);

    assign bcdHundreds = {c6[3], c7[3]};
    assign bcdTens = {c7[2:0], c5[3]};
    assign bcdOnes = {c5[2:0], bin[0]};
endmodule

// It's getting really late, so this helper module is in Behavioral Verilog:
module shift_add3(in, result);
    input [3:0] in;
    output [3:0] result;
    reg [3:0] result;

    always@(*)
```

```
        case (in)
            4'b0000: result = 4'b0000;
            4'b0001: result = 4'b0001;
            4'b0010: result = 4'b0010;
            4'b0011: result = 4'b0011;

            4'b0100: result = 4'b0100;
            4'b0101: result = 4'b1000;
            4'b0110: result = 4'b1001;

            4'b0111: result = 4'b1010;
            4'b1000: result = 4'b1011;
            4'b1001: result = 4'b1100;
            default: result = 4'b0000;
        endcase
endmodule
```