

```
1 ==> half_adder.v <==
2
3 module half_adder (
4     output out,
5     output cout,
6     input  x,
7     input  y
8 );
9     xor (out, x, y);
10    and (cout, x, y);
11 endmodule
12
13 ==> full_adder.v <==
14 module full_adder (
15     output out,
16     output cout,
17     input  x,
18     input  y,
19     input  cin
20 );
21     wire med1, med2, med3;
22     and (med1, x, y);
23     xor (med2, x, y);
24     and (med3, med2, cin);
25     or  (cout, med1, med3);
26     xor (out, med2, cin);
27 endmodule
28
29 ==> ripple_carry_adder.v <==
30 module ripple_carry_adder (
31     output [3:0] out,
32     output cout,
33     input  [3:0] in1,
34     input  [3:0] in2
35 );
36     wire c0, c1, c2;
37     full_adder U0 (
38         .out(out[0]),
39         .x(in1[0]),
40         .y(in2[0]),
41         .cout(c0),
42         .cin(1'b0)
43     );
44     full_adder U1 (
45         .out(out[1]),
46         .x(in1[1]),
47         .y(in2[1]),
48         .cout(c1),
49         .cin(c0)
50     );
51     full_adder U2 (
52         .out(out[2]),
53         .x(in1[2]),
54         .y(in2[2]),
55         .cout(c2),
56         .cin(c1)
57     );
58     full_adder U3 (
59         .out(out[3]),
60         .x(in1[3]),
61         .y(in2[3]),
62         .cout(cout),
63         .cin(c2)
64     );
65 endmodule
66
```

```

67 ==> bcd_adder.v <==
68 module bcd_adder (
69     output [3:0] out,
70     output cout,
71     input [3:0] x,
72     input [3:0] y
73 );
74     wire [3:0] med_out;
75     wire med_cout1, med_cout2;
76
77     ripple_carry_adder U0 (
78         .out (med_out),
79         .in1 (x),
80         .in2 (y),
81         .cout(med_cout1)
82     );
83
84     wire [3:0] offset;
85     wire med1;
86     or (med1, med_out[1], med_out[2]);
87     and (offset[1], med_out[3], med1);
88     and (offset[2], med_out[3], med1);
89     and (offset[0], 1'b0, 1'b0);
90     and (offset[3], 1'b0, 1'b0);
91
92     ripple_carry_adder U1 (
93         .out (out),
94         .in1 (med_out),
95         .in2 (offset),
96         .cout(med_cout2)
97     );
98     or (cout, med_cout1, med_cout2);
99
100
101 endmodule
102
103 ==> adders_tb.v <==
104 module adders_tb ();
105     reg half_in1, half_in2;
106     reg full_in1, full_in2, full_cin;
107     reg [3:0] ripple_in1;
108     reg [3:0] ripple_in2;
109     reg [3:0] bcd_in1;
110     reg [3:0] bcd_in2;
111
112     wire half_out;
113     wire half_cout;
114     wire full_out;
115     wire full_cout;
116     wire [3:0] ripple_out;
117     wire ripple_cout;
118     wire [3:0] bcd_out;
119     wire bcd_cout;
120
121     initial begin
122         $display("Half Adder:");
123         $monitor("%b %b %b %b", half_in1, half_in2, half_out, half_cout);
124
125         #10{half_in1, half_in2} = 2'b00;
126         #10{half_in1, half_in2} = 2'b01;
127         #10{half_in1, half_in2} = 2'b10;
128         #10{half_in1, half_in2} = 2'b11;
129
130         $display("Full Adder:");
131         $monitor("%b %b %b %b %b", full_in1, full_in2, full_cin, full_out, full_cout);
132
133         #10{full_in1, full_in2, full_cin} = 3'b000;

```

```

134 #10{full_in1, full_in2, full_cin} = 3'b001;
135 #10{full_in1, full_in2, full_cin} = 3'b010;
136 #10{full_in1, full_in2, full_cin} = 3'b011;
137 #10{full_in1, full_in2, full_cin} = 3'b100;
138 #10{full_in1, full_in2, full_cin} = 3'b101;
139 #10{full_in1, full_in2, full_cin} = 3'b110;
140 #10{full_in1, full_in2, full_cin} = 3'b111;
141
142 $display("Ripple Adder:");
143 $monitor("%4b %4b %4b %b", ripple_in1, ripple_in2, ripple_out, ripple_cout);
144
145 #10{ripple_in1, ripple_in2} = 8'b10000000;
146 #10{ripple_in1, ripple_in2} = 8'b01000001;
147 #10{ripple_in1, ripple_in2} = 8'b00100010;
148 #10{ripple_in1, ripple_in2} = 8'b00010011;
149 #10{ripple_in1, ripple_in2} = 8'b00100100;
150 #10{ripple_in1, ripple_in2} = 8'b01000101;
151 #10{ripple_in1, ripple_in2} = 8'b10000110;
152 #10{ripple_in1, ripple_in2} = 8'b01000111;
153 #10{ripple_in1, ripple_in2} = 8'b00101000;
154 #10{ripple_in1, ripple_in2} = 8'b00011001;
155 #10{ripple_in1, ripple_in2} = 8'b00101010;
156 #10{ripple_in1, ripple_in2} = 8'b01001011;
157 #10{ripple_in1, ripple_in2} = 8'b10001100;
158 #10{ripple_in1, ripple_in2} = 8'b01001101;
159 #10{ripple_in1, ripple_in2} = 8'b00101110;
160 #10{ripple_in1, ripple_in2} = 8'b00011111;
161
162 $display("BCD Adder:");
163 $monitor("%4b %4b %4b %b", bcd_in1, bcd_in2, bcd_out, bcd_cout);
164 #10{bcd_in1, bcd_in2} = 8'b01100110;
165 #10{bcd_in1, bcd_in2} = 8'b00010010;
166 #10{bcd_in1, bcd_in2} = 8'b01101001;
167 #10{bcd_in1, bcd_in2} = 8'b01000010;
168 #10{bcd_in1, bcd_in2} = 8'b01000100;
169 #10{bcd_in1, bcd_in2} = 8'b01110110;
170 #10{bcd_in1, bcd_in2} = 8'b01111000;
171 #10 $finish();
172
173 end
174
175
176 half_adder U0 (
177     .out(half_out),
178     .cout(half_cout),
179     .x(half_in1),
180     .y(half_in2)
181 );
182 full_adder U1 (
183     .out(full_out),
184     .cout(full_cout),
185     .x(full_in1),
186     .y(full_in2),
187     .cin(full_cin)
188 );
189 ripple_carry_adder U2 (
190     .out (ripple_out),
191     .cout(ripple_cout),
192     .in1 (ripple_in1),
193     .in2 (ripple_in2)
194 );
195 bcd_adder U3 (
196     .out(bcd_out),
197     .cout(bcd_cout),
198     .x(bcd_in1),
199     .y(bcd_in2)
200 );

```

```
201     initial begin
202         $dumpfile("adders.vcd");
203         $dumpvars(0, adders_tb);
204     end
205 endmodule
206
207
208 ==> output.txt <==
209 Half Adder:
210 VCD info: dumpfile adders.vcd opened for output.
211 x x x x
212 0 0 0 0
213 0 1 1 0
214 1 0 1 0
215 Full Adder:
216 x x x x x
217 0 0 0 0 0
218 0 0 1 1 0
219 0 1 0 1 0
220 0 1 1 0 1
221 1 0 0 1 0
222 1 0 1 0 1
223 1 1 0 0 1
224 Ripple Adder:
225 xxxx xxxx xxxx x
226 1000 0000 1000 0
227 0100 0001 0101 0
228 0010 0010 0100 0
229 0001 0011 0100 0
230 0010 0100 0110 0
231 0100 0101 1001 0
232 1000 0110 1110 0
233 0100 0111 1011 0
234 0010 1000 1010 0
235 0001 1001 1010 0
236 0010 1010 1100 0
237 0100 1011 1111 0
238 1000 1100 0100 1
239 0100 1101 0001 1
240 0010 1110 0000 1
241 BCD Adder:
242 xxxx xxxx xxxx x
243 0110 0110 0010 1
244 0001 0010 0011 0
245 0110 1001 0101 1
246 0100 0010 0110 0
247 0100 0100 1000 0
248 0111 0110 0011 1
249 0111 1000 0101 1
250 adders_tb.v:68: $finish called at 360 (1s)
```