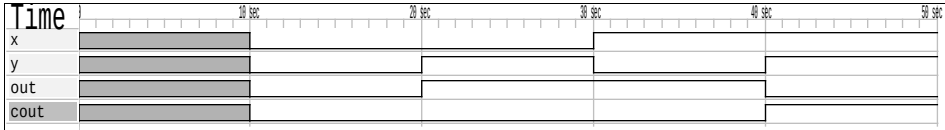
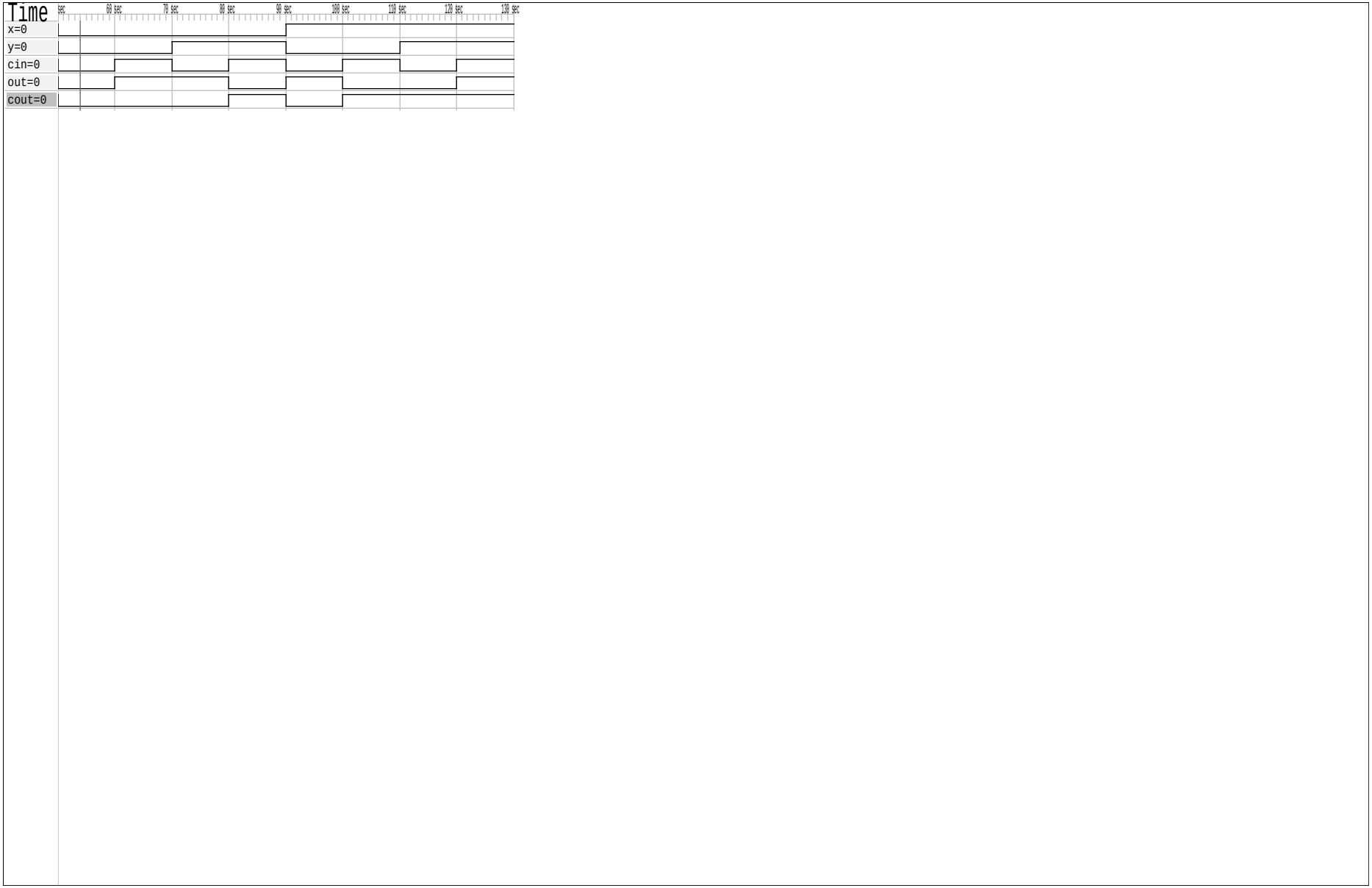
```
==> half_adder.v <==

module half_adder (
    output out,
    output cout,
    input  x,
    input  y
);
  xor (out, x, y);
  and (cout, x, y);
endmodule

==> full_adder.v <==
module full_adder (
    output out,
    output cout,
    input  x,
    input  y,
    input  cin
);
  wire med1, med2, med3;
  and (med1, x, y);
  xor (med2, x, y);
  and (med3, med2, cin);
  or (cout, med1, med3);
  xor (out, med2, cin);
endmodule

==> ripple_carry_adder.v <==
module ripple_carry_adder (
    output [3:0] out,
    output cout,
    input [3:0] in1,
    input [3:0] in2
);
  wire c0, c1, c2;
  full_adder U0 (
      .out(out[0]),
      .x(in1[0]),
      .y(in2[0]),
      .cout(c0),
      .cin(1'b0)
  );
  full_adder U1 (
      .out(out[1]),
      .x(in1[1]),
      .y(in2[1]),
      .cout(c1),
      .cin(c0)
  );
  full_adder U2 (
      .out(out[2]),
      .x(in1[2]),
      .y(in2[2]),
      .cout(c2),
      .cin(c1)
  );
  full_adder U3 (
      .out(out[3]),
      .x(in1[3]),
      .y(in2[3]),
      .cout(cout),
      .cin(c2)
  );
endmodule

```

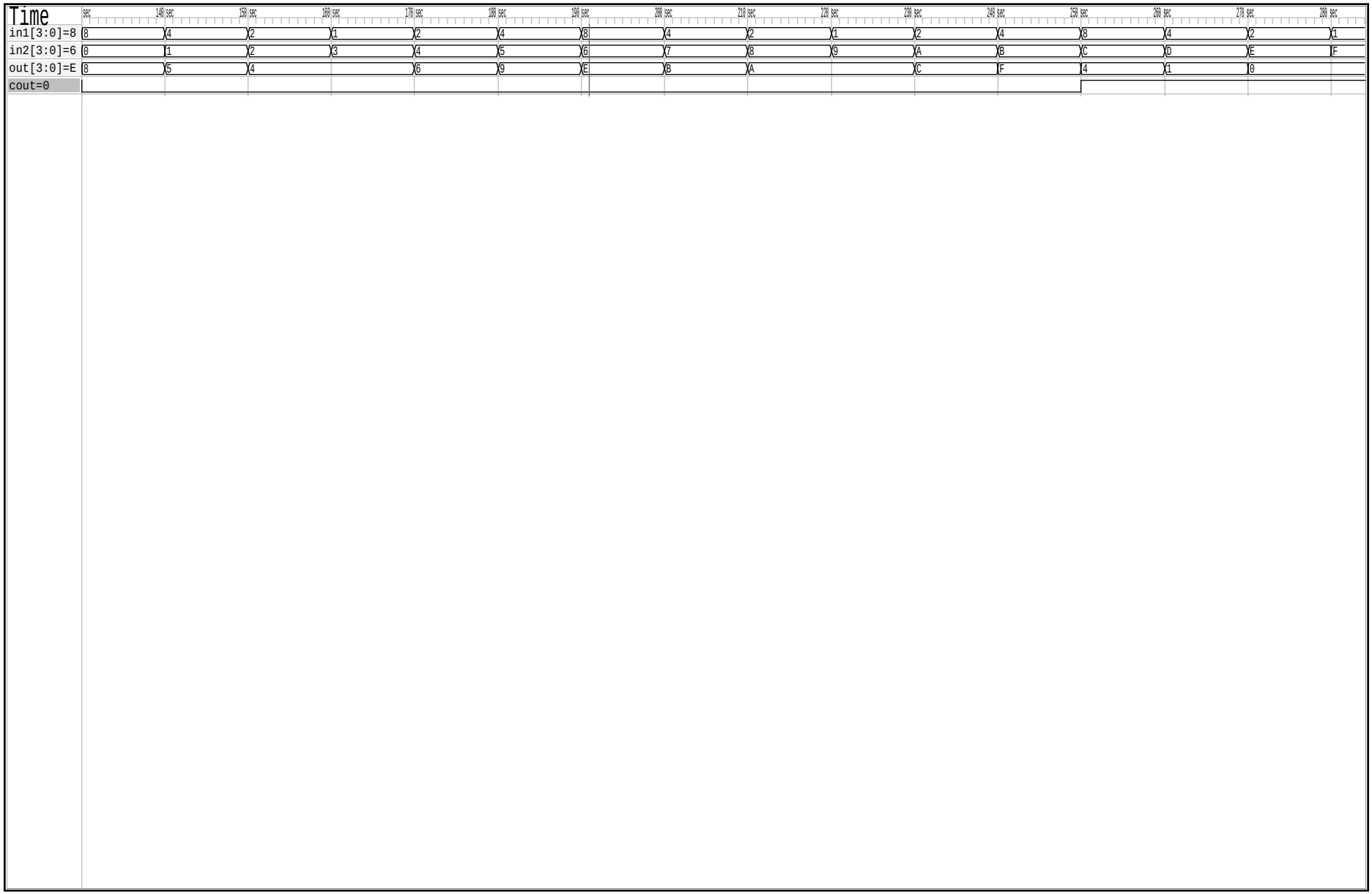
```
==> bcd_adder.v <==
module bcd_adder (
    output [3:0] out,
    output cout,
    input [3:0] x,
    input [3:0] y
);
  wire [3:0] med_out;
  wire med_cout1, med_cout2;

  ripple_carry_adder U0 (
      .out (med_out),
      .in1 (x),
      .in2 (y),
      .cout(med_cout1)
  );

  wire [3:0] offset;
  wire med1;
  or (med1, med_out[1], med_out[2]);
  and (offset[1], med_out[3], med1);
  and (offset[2], med_out[3], med1);
  and (offset[0], 1'b0, 1'b0);
  and (offset[3], 1'b0, 1'b0);

  ripple_carry_adder U1 (
      .out (out),
      .in1 (med_out),
      .in2 (offset),
      .cout(med_cout2)
  );
  or (cout, med_cout1, med_cout2);


endmodule

==> adders_tb.v <==
module adders_tb ();
  reg half_in1, half_in2;
  reg full_in1, full_in2, full_cin;
  reg [3:0] ripple_in1;
  reg [3:0] ripple_in2;
  reg [3:0] bcd_in1;
  reg [3:0] bcd_in2;

  wire half_out;
  wire half_cout;
  wire full_out;
  wire full_cout;
  wire [3:0] ripple_out;
  wire ripple_cout;
  wire [3:0] bcd_out;
  wire bcd_cout;

  initial begin
    $display("Half Adder:");
    $monitor("%b %b %b %b", half_in1, half_in2, half_out, half_cout);

    #10{half_in1, half_in2} = 2'b00;
    #10{half_in1, half_in2} = 2'b01;
    #10{half_in1, half_in2} = 2'b10;
    #10{half_in1, half_in2} = 2'b11;

    $display("Full Adder:");
    $monitor("%b %b %b %b %b", full_in1, full_in2, full_cin, full_out, full_cout);

    #10{full_in1, full_in2, full_cin} = 3'b000;
```

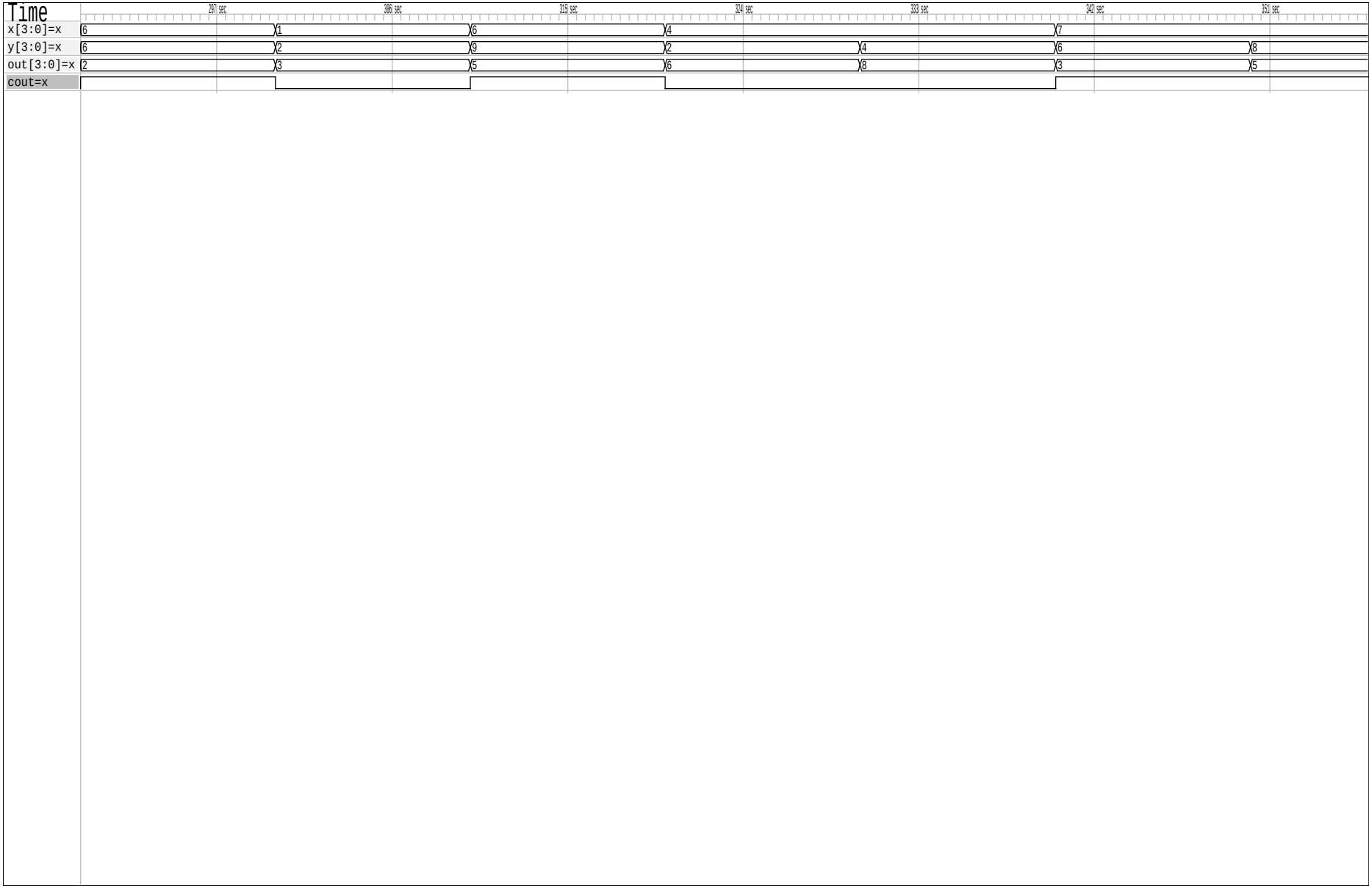
```
    #10{full_in1, full_in2, full_cin} = 3'b001;
    #10{full_in1, full_in2, full_cin} = 3'b010;
    #10{full_in1, full_in2, full_cin} = 3'b011;
    #10{full_in1, full_in2, full_cin} = 3'b100;
    #10{full_in1, full_in2, full_cin} = 3'b101;
    #10{full_in1, full_in2, full_cin} = 3'b110;
    #10{full_in1, full_in2, full_cin} = 3'b111;

    $display("Ripple Adder:");
    $monitor("%4b %4b %4b %b", ripple_in1, ripple_in2, ripple_out, ripple_cout);

    #10{ripple_in1, ripple_in2} = 8'b10000000;
    #10{ripple_in1, ripple_in2} = 8'b01000001;
    #10{ripple_in1, ripple_in2} = 8'b00100010;
    #10{ripple_in1, ripple_in2} = 8'b00010011;
    #10{ripple_in1, ripple_in2} = 8'b00100100;
    #10{ripple_in1, ripple_in2} = 8'b01000101;
    #10{ripple_in1, ripple_in2} = 8'b10000110;
    #10{ripple_in1, ripple_in2} = 8'b01000111;
    #10{ripple_in1, ripple_in2} = 8'b00101000;
    #10{ripple_in1, ripple_in2} = 8'b00011001;
    #10{ripple_in1, ripple_in2} = 8'b00101010;
    #10{ripple_in1, ripple_in2} = 8'b01001011;
    #10{ripple_in1, ripple_in2} = 8'b10001100;
    #10{ripple_in1, ripple_in2} = 8'b01001101;
    #10{ripple_in1, ripple_in2} = 8'b00101110;
    #10{ripple_in1, ripple_in2} = 8'b00011111;

    $display("BCD Adder:");
    $monitor("%4b %4b %4b %b", bcd_in1, bcd_in2, bcd_out, bcd_cout);
    #10{bcd_in1, bcd_in2} = 8'b01100110;
    #10{bcd_in1, bcd_in2} = 8'b00010010;
    #10{bcd_in1, bcd_in2} = 8'b01101001;
    #10{bcd_in1, bcd_in2} = 8'b01000010;
    #10{bcd_in1, bcd_in2} = 8'b01000100;
    #10{bcd_in1, bcd_in2} = 8'b01110110;
    #10{bcd_in1, bcd_in2} = 8'b01111000;
    #10 $finish();

end


half_adder U0 (
    .out(half_out),
    .cout(half_cout),
    .x(half_in1),
    .y(half_in2)
);
full_adder U1 (
    .out(full_out),
    .cout(full_cout),
    .x(full_in1),
    .y(full_in2),
    .cin(full_cin)
);
ripple_carry_adder U2 (
    .out (ripple_out),
    .cout(ripple_cout),
    .in1 (ripple_in1),
    .in2 (ripple_in2)
);
bcd_adder U3 (
    .out(bcd_out),
    .cout(bcd_cout),
    .x(bcd_in1),
    .y(bcd_in2)
);
```

```
201
202   initial begin
203     $dumpfile("adders.vcd");
204     $dumpvars(0, adders_tb);
205   end
206 endmodule
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)
```


Time
10 sec
20 sec
30 sec
40 sec
50 sec
x
y
out
cout


Time
sec
60 sec
70 sec
80 sec
90 sec
100 sec
110 sec
120 sec
130 sec
x=0
y=0
cin=0
out=0
cout=0


Time
sec
140 sec
150 sec
160 sec
170 sec
180 sec
190 sec
200 sec
210 sec
220 sec
230 sec
240 sec
250 sec
260 sec
270 sec
280 sec
in1[3:0]=8
8 4 2 1 2 4 8 4 2 1 2 4 8 4 2 1
in2[3:0]=6
0 1 2 3 4 5 6 7 8 9 A B C D E F
out[3:0]=E
8 5 4 6 9 E B A C F 4 1 0
cout=0


Time
297 sec
306 sec
315 sec
324 sec
333 sec
342 sec
351 sec
x[3:0]=x
y[3:0]=x
out[3:0]=x
cout=x
6
1
6
4
7
6
2
9
2
4
6
8
2
3
5
6
8
3
5