Control Path Code:

```
`timescale 1ns / 1ps


module control(
    input clk,
    input reset,
    input eq_flag,
    input if_flag,
     output reg done,
    output x_load,
    output y_load,
    output x_sel,
    output y_sel,
    output gcd_load
    );


      reg [3:0]present_state, next_state;
      reg x_sel, y_sel, x_load, y_load, gcd_load;
      parameter start = 1,
                        in = 2,
                        test = 3,
                        test2 = 4,
                        update1 = 5,
                        update2 = 6,
                        stop = 7;
      always@(posedge clk or reset)
      begin
      if(reset)
          begin
```

```
     present_state <= start;
     next_state <= start;
     done <= 0;

     end
else
     case(next_state)
      start: begin
                    next_state <= in;

                    end
    test: begin
                    x_sel  <=  1'b0;
                    y_sel <= 1'b0;
                    x_load <= 1'b0;
                    y_load <= 1'b0;
                    next_state <= test2;
                end
      test2: begin
                    x_sel <= 1'b0;
                    y_sel <= 1'b0;
                    x_load <= 1'b0;
                    y_load <= 1'b0;
                    if( eq_flag == 1'b1)
                          next_state <= stop;
                    else if( if_flag == 1'b1)
                          next_state <= update1;
                    else
                          next_state <= update2;
```

```
                    end
        update2: begin      // y = y - x;
                            x_load <= 1'b1;
                            y_load <= 1'b0;
                            x_sel <= 1'b0;
                            next_state <= test;
                     end
        update1: begin
                            y_load <= 1'b1;
                            x_load <= 1'b0;
                            y_sel <= 1'b0;
                       next_state <= test;
                      end
        in: begin
                  x_sel <= 1'b1; y_sel <= 1'b1;
                  x_load <= 1'b1; y_load <= 1'b1;
                  if( eq_flag == 1'b1)
                        next_state <= stop;
                  else
                        next_state <= test;
            end
      stop: begin
            gcd_load <= 1'b1;
            done <= 1;
            next_state <= stop;
            end
   endcase
end
```

```
endmodule
```

Data Path Code:

```
`timescale 1ns / 1ps


module data(
    input [31:0] num1, num2,
    output [31:0] gcd_out,
      output reg DONE,
      input clk, reset
    );


      wire xsel, ysel, xload, yload, gcdload, eqflag, ifflag;
      wire done;


       datapath d1( .clk(clk), .reset(reset), .x_sel(xsel),
.y_sel(ysel), .x_load(xload),
       .y_load(yload), .gcd_load(gcdload),
       .a(num1), .b(num2), .eq_flag(eqflag), .if_flag(ifflag),
    .gcd(gcd_out));


       control c1( .clk(clk),
    .reset(reset),
    .eq_flag(eqflag),
    .if_flag(ifflag),
      .done(done),
    .x_load(xload),
    .y_load(yload),
```

```
    .x_sel(xsel),
    .y_sel(ysel),
    .gcd_load(gcdload));

     always @(posedge clk) begin
          DONE <= done;
     end
endmodule
```

GCD Code:

```
`timescale 1ns / 1ps

module gcd(
    input [31:0] num1, num2,
    output [31:0] gcd_out,
     output reg DONE,
     input clk, reset
    );

     wire xsel, ysel, xload, yload, gcdload, eqflag, ifflag;
     wire done;

      datapath d1( .clk(clk), .reset(reset), .x_sel(xsel),
.y_sel(ysel), .x_load(xload),
      .y_load(yload), .gcd_load(gcdload),
      .a(num1), .b(num2), .eq_flag(eqflag), .if_flag(ifflag),
    .gcd(gcd_out));

      control c1( .clk(clk),
    .reset(reset),
```

```
    .eq_flag(eqflag),
    .if_flag(ifflag),
      .done(done),
    .x_load(xload),
    .y_load(yload),
    .x_sel(xsel),
    .y_sel(ysel),
    .gcd_load(gcdload));

      always @(posedge clk) begin
            DONE <= done;
      end
endmodule
```

MUX Code:

```
module mux(
    input sel,
    input [31:0] in0, in1,
       output [31:0] mux_out
    );

assign mux_out = (sel==0) ? in0 : in1;

endmodule
```

Reg Code:

```
`timescale 1ns / 1ps
module reg_file(
    input clk,
```

```
    input reset,
    input load,
    input [31:0] data,
    output [31:0] out
    );

     reg [31:0] out;
     always@(posedge clk)
     begin
     if(reset == 1'b1) begin
          out <= 0;
     end else if(load == 1'b1) begin
          out <= data;
     end else begin
          out <= out;
     end
     end

endmodule
```

GCD TB Code:

```
module tb_gcd1;

     // Inputs
     reg [31:0] num1;
     reg [31:0] num2;
     reg clk;
```

```
	reg reset;

	// Outputs
	wire [31:0] gcd_out;
	wire done;

	// Instantiate the Unit Under Test (UUT)
	gcd uut (
		.num1(num1),
		.num2(num2),
		.gcd_out(gcd_out),
		.DONE(done),
		.clk(clk),
		.reset(reset)
	);
always
	   #10 clk = ~clk;
initial
		begin
		#1000
		$finish;
		end


	initial
	begin
		num1 = 32'd0;
		num2 = 32'd0;
		clk = 0;
```

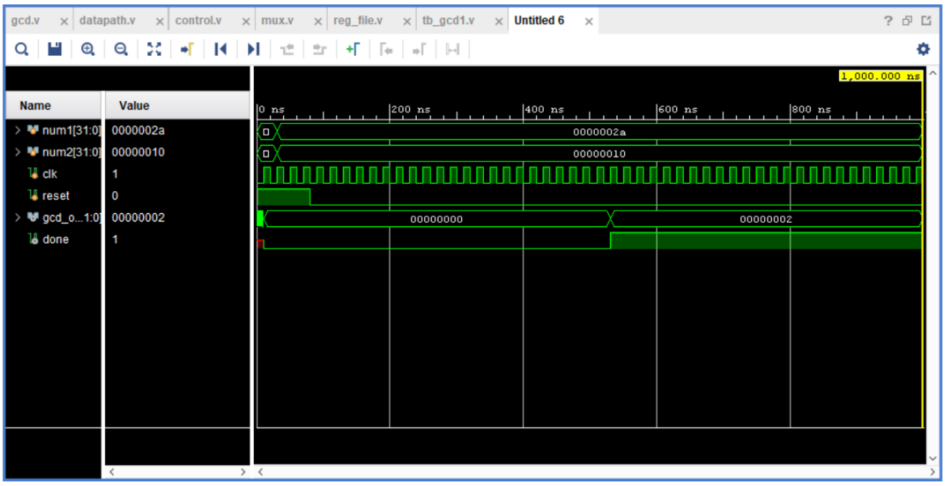
```
            reset = 1;
          end

    initial
    begin
    #30
    reset = 0;
    end

    initial
    begin
    #50
    num1 = 32'd42;
    num2 = 32'd16;
    end

endmodule
```

Output:


gcd.v
datapath.v
control.v
mux.v
reg_file.v
tb_gcd1.v
Untitled 6
Name
Value
num1[31:0]
0000002a
num2[31:0]
00000010
clk
1
reset
0
gcd_o...1:0]
00000002
done
1
1,000.000 ns
0 ns
200 ns
400 ns
600 ns
800 ns
0000002a
00000010
00000000
00000002