Data path and Control Path Example with Verilog

GCD Algorithm

EUCLID'S GCD

```
a = read();
b = read();
while(a != b)
    if(a < b)
        b = b - a
    else if (a > b)
        a = a - b
    endif
endwhile
Print a
```

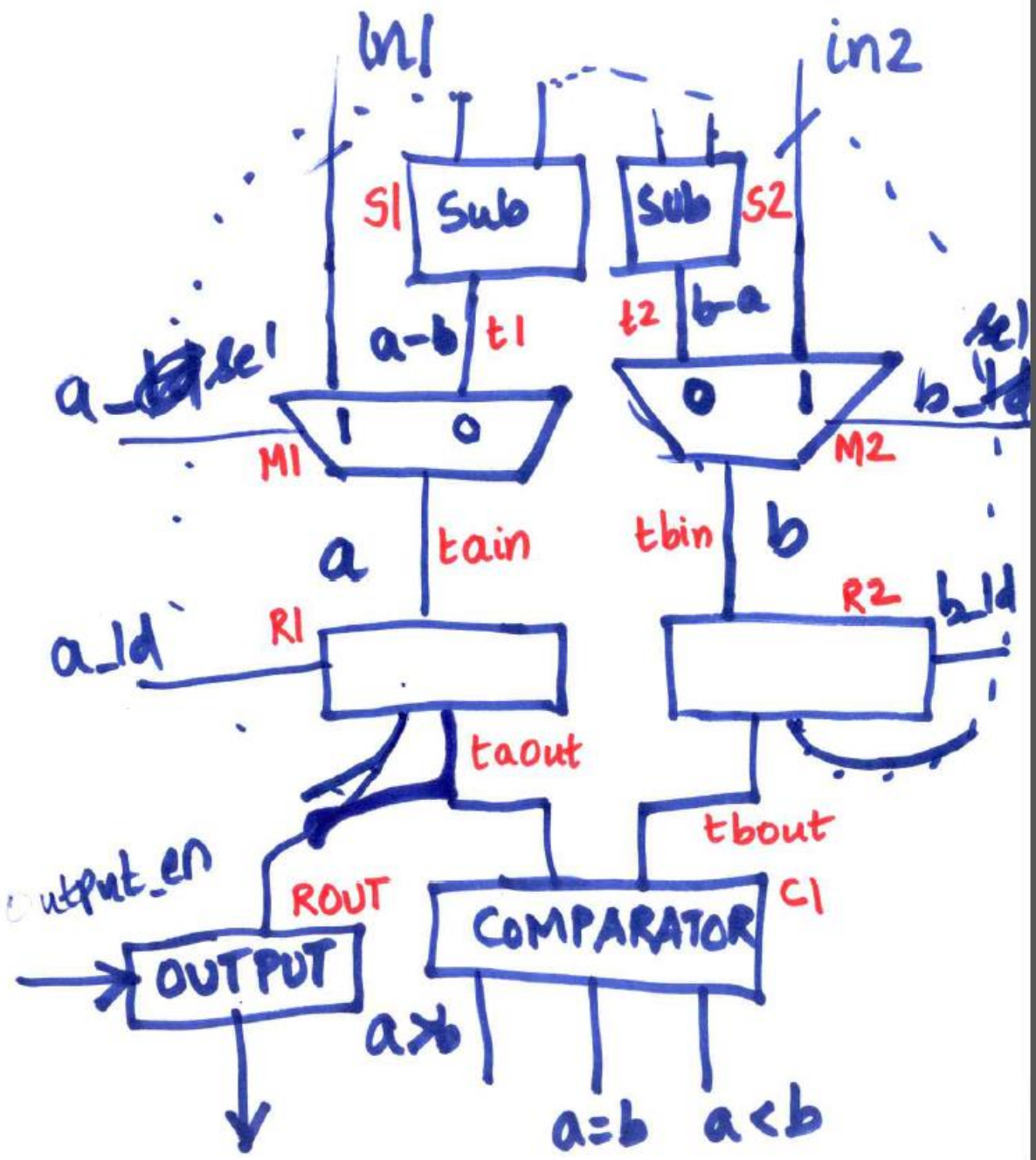
	a	b
EX1	42	16
	26	16
	10	16
	10	6
	4	6
	4	2
	2	2

a	b
60	45
15	45
15	30
15	15

( )

Datapath

```
module datapath (clk, rst, in1, in2, a_sel, b_sel, a_ld, b_ld,
a_gt_b, a_eq_b, a_lt_b, output_en, out);
input clk, rst;
input a_sel, b_sel, a_ld, b_ld;
output a_gt_b, a_eq_b, a_lt_b;
input output_en;
output [7:0] out;
input [7:0] in1, in2;

wire [7:0] tain, tbin, taout, tbout, t1,t2;

subtractor S1(taout, tbout, t1);
subtractor S2(tbout, taout, t2);
mux M1(tain, a_sel, t1, in1);
mux M2(tbin, b_sel, t2, in2);
register R1(clk,rst, tain, a_ld, taout);
register R2 (clk, rst, tbin, b_ld, tbout);
register Rout (clk, rst, taout, output_en, out);
comparator C1 (taout, tbout, a_gt_b, a_eq_b, a_lt_b);
endmodule
```



Inner modules of datapath

```
module comparator (a, b, a_gt_b, a_eq_b, a_lt_b);
input [7:0] a, b;
output reg a_gt_b, a_eq_b, a_lt_b;
always @(a or b)
begin
if (a>b)
{a_gt_b, a_eq_b, a_lt_b} = 3'b100;
else if (a==b)

{a_gt_b, a_eq_b, a_lt_b} = 3'b010;
else
{a_gt_b, a_eq_b, a_lt_b} = 3'b001;
end
endmodule
```

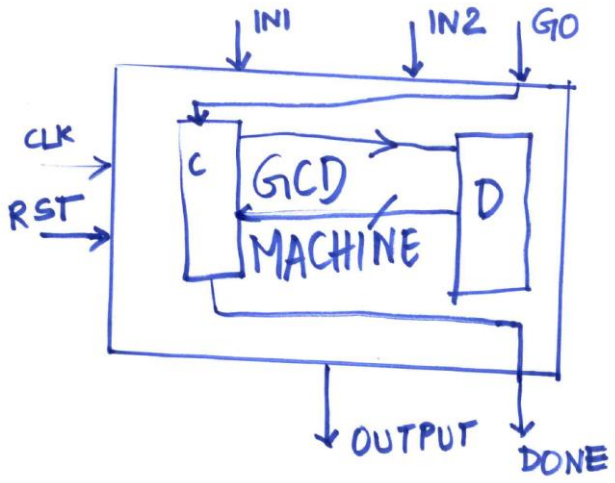
```
module mux (out, sel, in0, in1);
output reg [7:0] out;
input [7:0] in0, in1;
input sel;

always@ (sel or in0 or in1)
begin
if (sel == 0)
out = in0;
else
out =in1;
end
endmodule
```

```
module register (clk, rst, in, lden, out);
input clk, rst;
input [7:0] in ;
input lden;
output reg [7:0] out;

always@ (posedge clk)
begin
if (rst ==1)
out<= 7'b0000000;
else if (lden ==1)
out <= in ;
end
endmodule
```

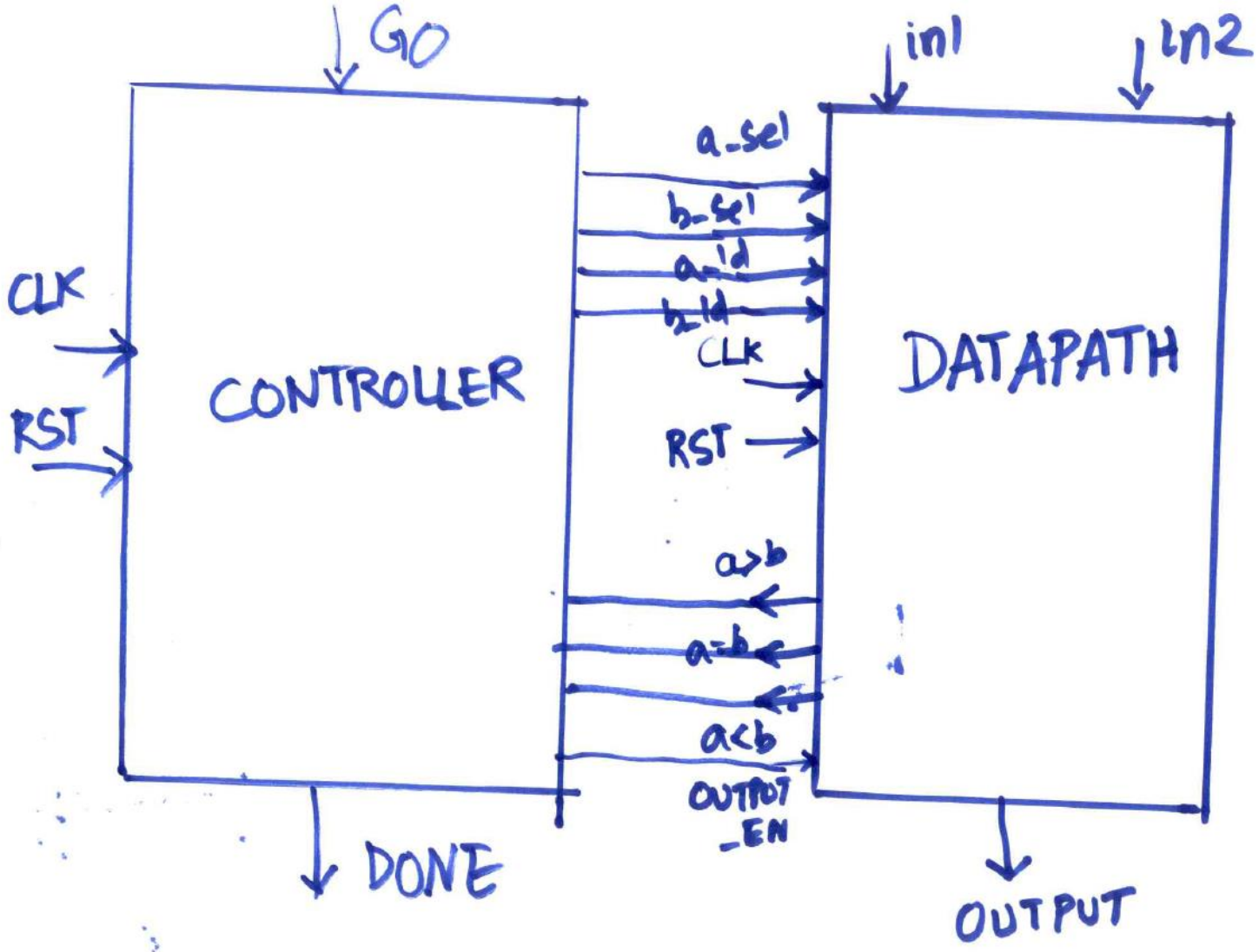
```
module subtractor (in1, in2, out);
input [7:0] in1, in2;
output reg [7:0] out;
always@(in1 or in2)
begin
out = in1- in2;
end
endmodule
```

Top Module

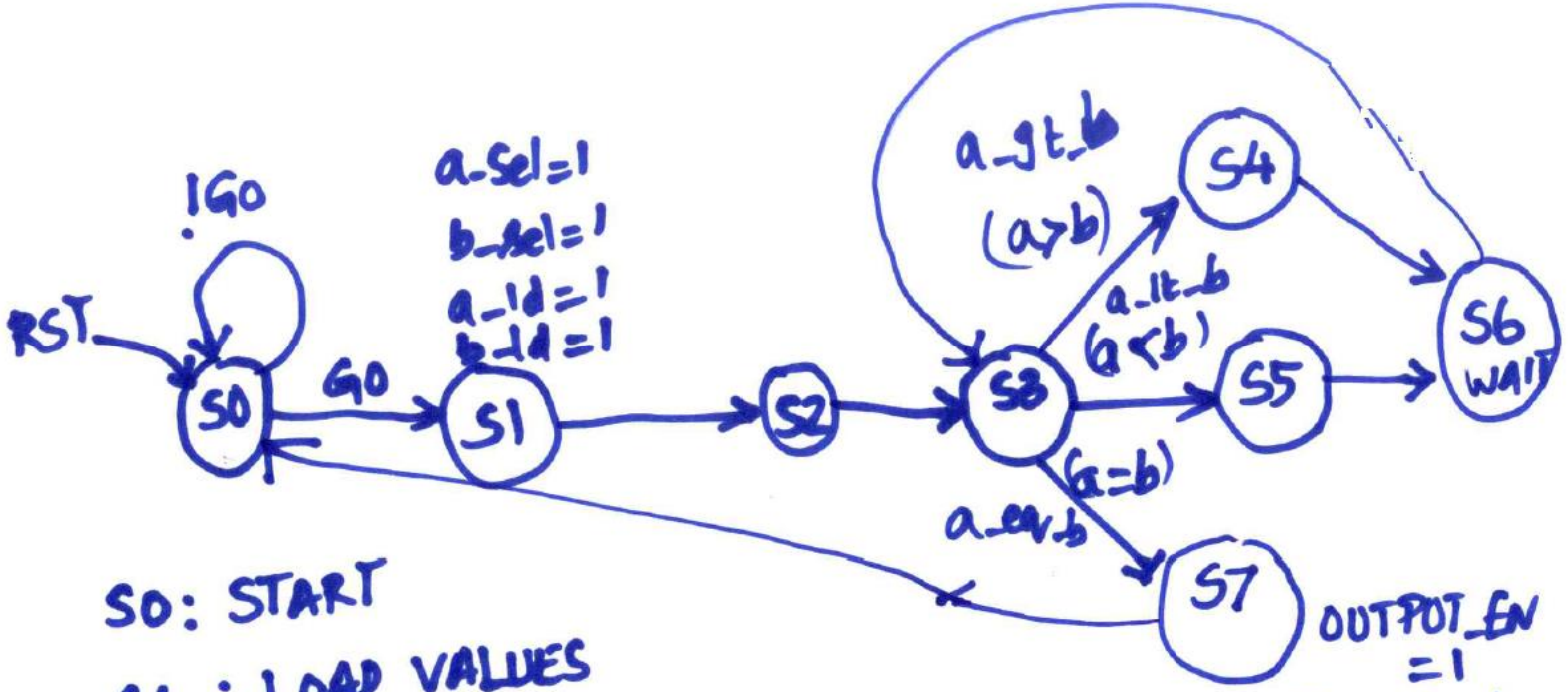
```
module gcd_machine (clk, rst, go, in1, in2, out, done);
input clk, rst, go;
input [7:0] in1, in2;
output [7:0] out;
output done;

wire a_gt_b, a_eq_b, a_lt_b;
wire a_ld, b_ld, a_sel, b_sel;
wire output_en;

controller C1 (clk, rst, go, a_gt_b, a_eq_b, a_lt_b,a_ld, a_sel,
b_ld,  b_sel, output_en, done);
datapath D1 (clk, rst, in1, in2, a_sel, b_sel, a_ld, b_ld,
a_gt_b, a_eq_b, a_lt_b, output_en, out);
endmodule
```



Data path and Control Path



GCD State Machine



S0: START

S1 : LOAD VALUES

S2: WAIT STATE(for inputs to load)

S3 : COMPARISON STATE

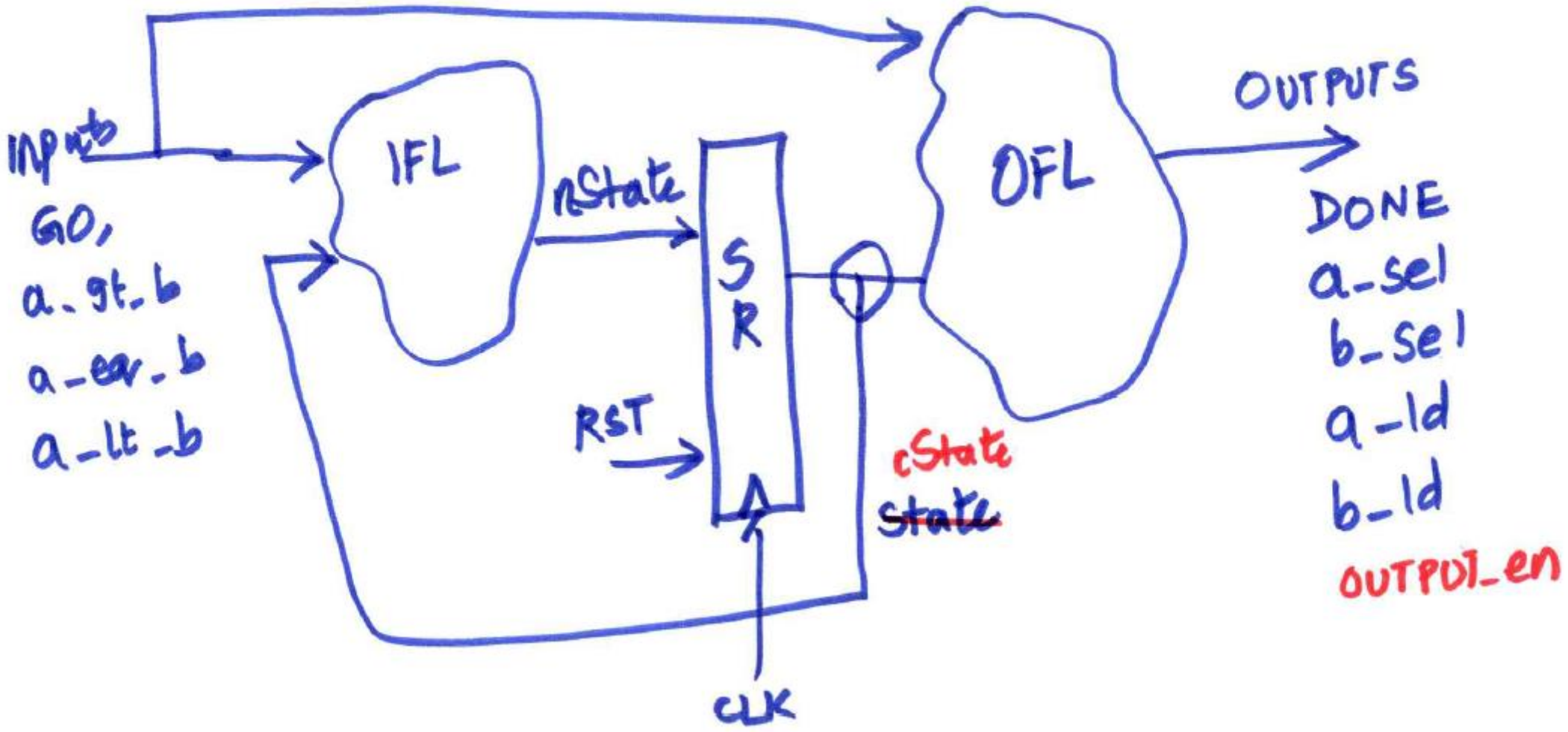
S4: a_sel=0 and a_ld=1 [a=a-b]

S5: b_sel=0 and b_ld=1 [b=b-a]

S6: WAIT

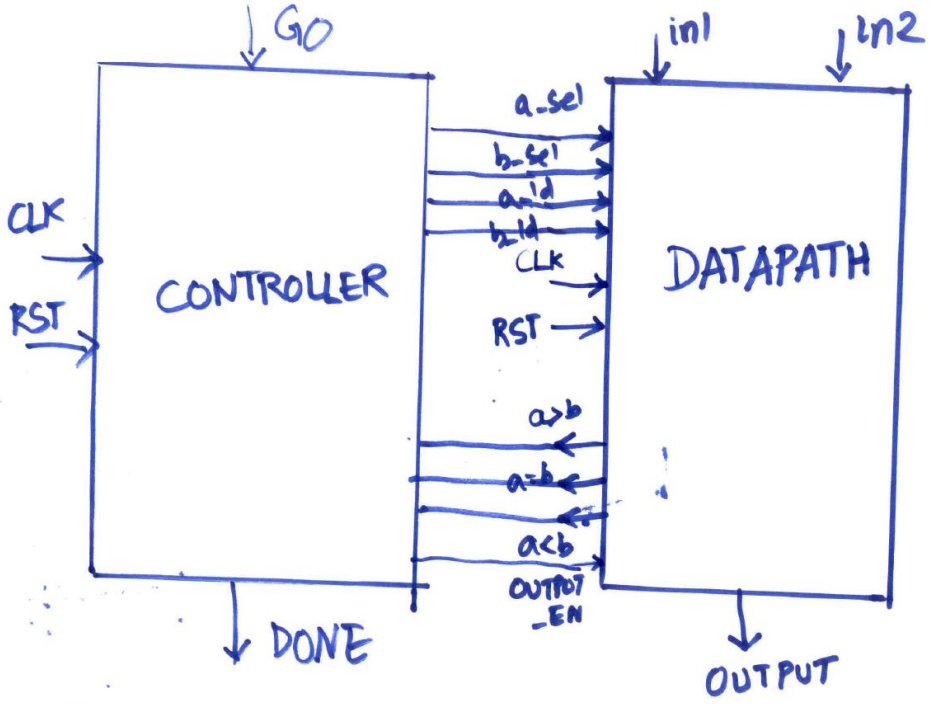
S7: DONE

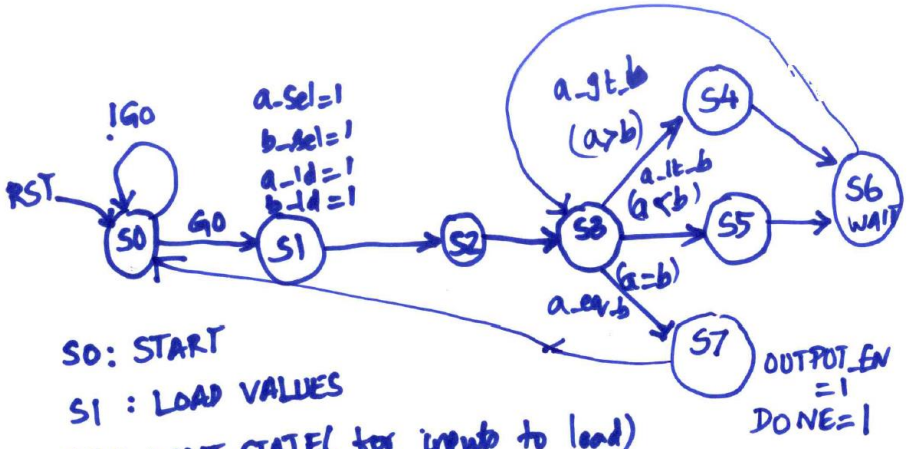
State Machine



Controller

```
module controller (clk, rst, go, a_gt_b, a_eq_b, a_lt_b,  a_ld,
a_sel, b_ld,  b_sel, output_en, done);
input clk, rst;
input go;
input a_gt_b, a_eq_b, a_lt_b;
output reg a_sel, b_sel, a_ld, b_ld, done, output_en;
reg [2:0] cState, nState;

parameter S0 = 3'b000;
parameter S1 = 3'b001;
parameter S2 = 3'b010;
parameter S3 = 3'b011;
parameter S4 = 3'b100;
parameter S5 = 3'b101;
parameter S6 = 3'b110;
parameter S7 = 3'b111;

always@(posedge rst or posedge clk)
begin
if(rst ==1)
cState <=S0;
else
cState <= nState;
end
```





S0: START

S1 : LOAD VALUES

S2: WAIT STATE(for inputs to load)

S3 : COMPARISON STATE

S4: a_sel=0 and a_ld=1 [a=a-b]

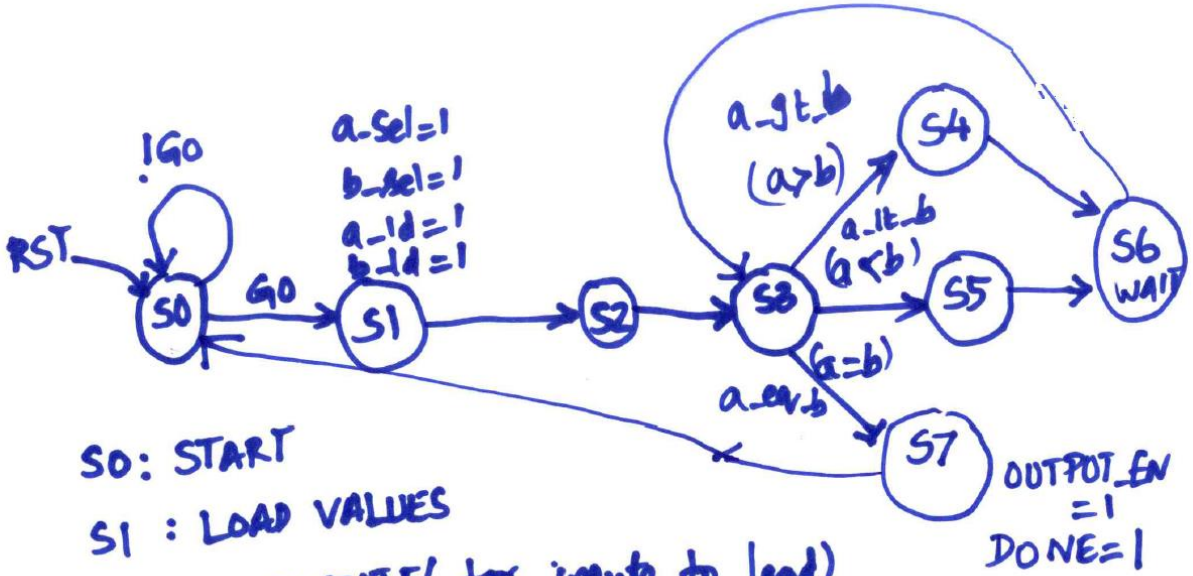
S5: b_sel=0 and b_ld=1 [b=b-a]

S6: WAIT

S7: DONE

Controller

```
always@(go or a_gt_b or a_lt_b or a_eq_b or cState)

begin
 case (cState)

S0: begin
if (go == 0) nState = S0;
else nState = S1;
end

S1: nState = S2;
S2: nState = S3;
S3 :begin
if (a_gt_b == 1)
nState = S4;
else if (a_lt_b == 1)
nState = S5 ;
else nState = S7;
end
S4: nState =S6;
S5 : nState =S6;
S6 : nState = S3;
S7 : nState = S0;
default : nState =S0;
endcase
end
```



S0: START

S1 : LOAD VALUES

S2: WAIT STATE(for inputs to load)

S3 : COMPARISON STATE

S4: a_sel=0 and a_ld=1 [a=a-b]

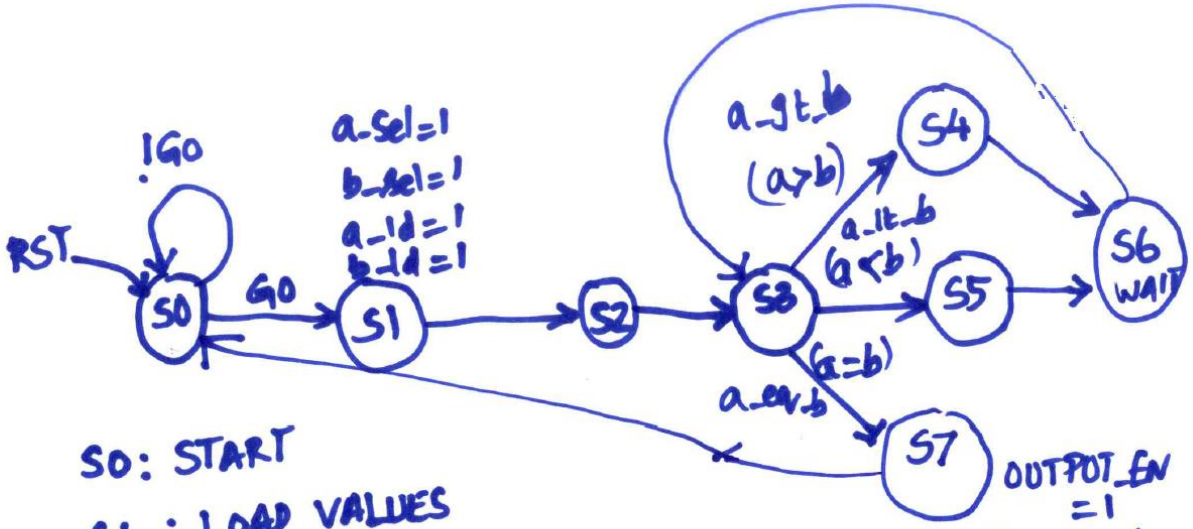
S5: b_sel=0 and b_ld=1 [b=b-a]

S6: WAIT

S7: DONE

Controller

```
always @ (go or a_gt_b or a_lt_b or a_eq_b or cState)

begin
case (cState)

S0:
begin
a_sel = 0;
b_sel = 0;
a_ld = 0;
b_ld = 0;
done = 1;
output_en =0;
end
S1:
begin
a_sel = 1;
b_sel = 1;
a_ld = 1;
b_ld = 1;
done = 0;
output_en =0;
end

S2:
begin
a_sel = 0;
b_sel = 0;
a_ld = 0;
b_ld = 0;
done = 0;
output_en =0;
end
```



S0: START

S1 : LOAD VALUES

S2: WAIT STATE(for inputs to load)

S3 : COMPARISON STATE

S4: a_sel=0 and a_ld=1 [a=a-b]

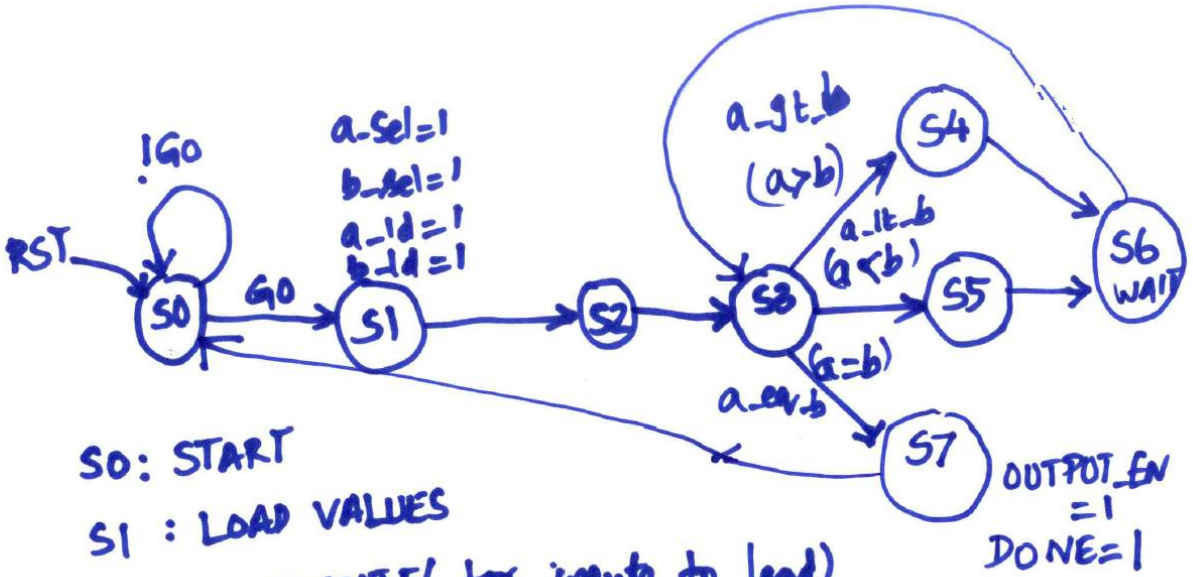
S5: b_sel=0 and b_ld=1 [b=b-a]

S6: WAIT

S7: DONE

Controller

```
S3:
begin
a_sel = 0;
b_sel = 0;
a_ld = 0;
b_ld = 0;
done = 0;
output_en =0;
end

S4:

begin
a_sel = 0;
b_sel = 0;
a_ld = 1;
b_ld = 0;
done = 0;
output_en =0;
end

S5:

begin
a_sel = 0;
b_sel = 0;
a_ld = 0;
b_ld = 1;
done = 0;
output_en =0;
end

S6:
begin
a_sel = 0;
b_sel = 0;
a_ld = 0;
b_ld = 0;
done = 0;
output_en =0;
end

S7:

begin
a_sel = 0;
b_sel = 0;
a_ld = 0;
b_ld = 0;
done = 1;
output_en =1;
end

default:

begin
a_sel = 0;
b_sel = 0;
a_ld = 0;
b_ld = 0;
done = 0;
output_en =0;
end

endcase
end

endmodule
```



S0: START

S1 : LOAD VALUES

S2: WAIT STATE(for inputs to load)

S3 : COMPARISON STATE

S4: a_sel=0 and a_ld=1 [a=a-b]

S5: b_sel=0 and b_ld=1 [b=b-a]

S6: WAIT

S7: DONE

Test Plan

TEST PLAN

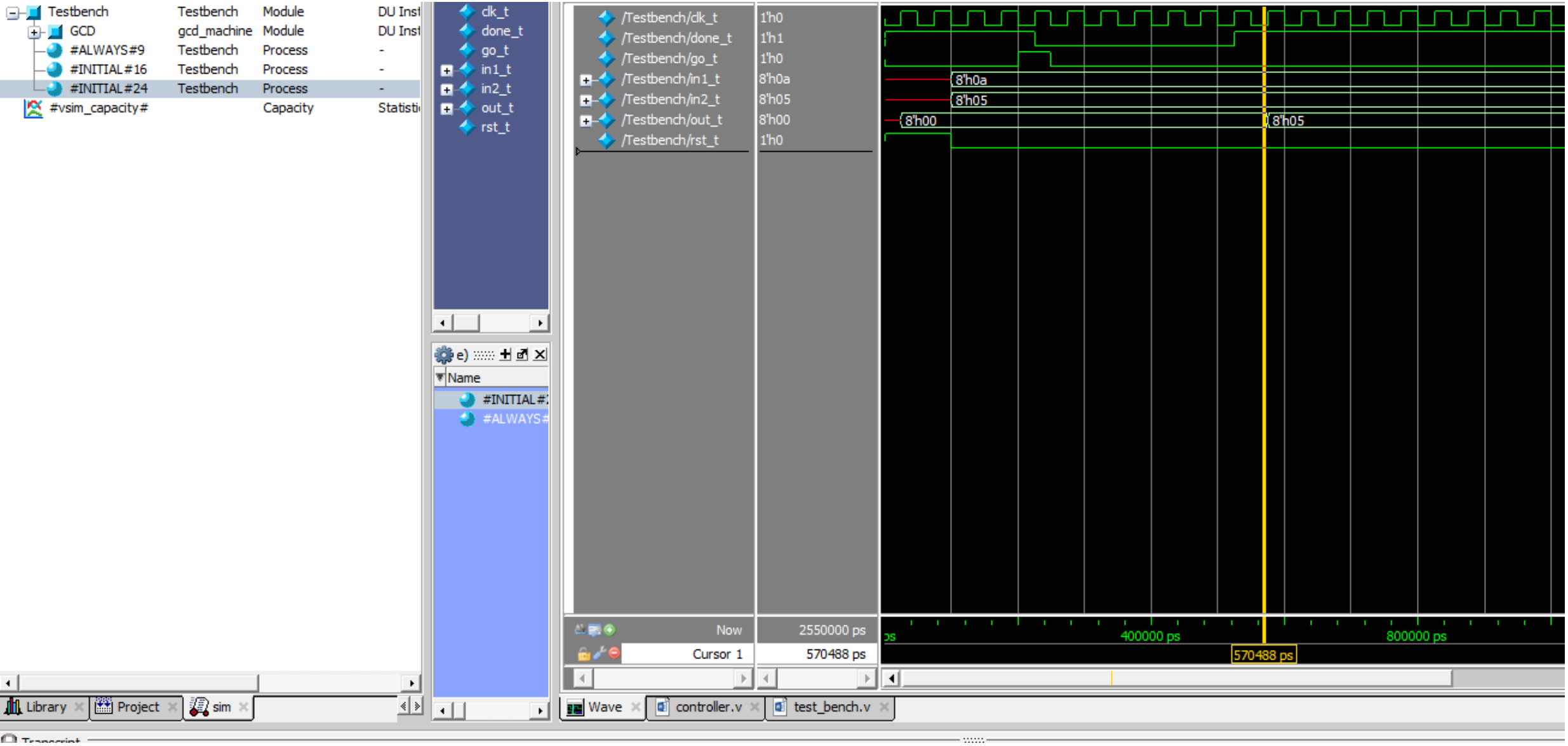
1. Reset the chip
2. Place inputs
3. Turn on "Go"
4. wait till done = 0
5. Remove go. FSM will start working
6. Wait till done = 1
7. Output is ready

Test Bench

```
`timescale 1ns/1ps
module Testbench;
reg clk_t, rst_t, go_t;
reg [7:0] in1_t;
reg [7:0] in2_t;
wire [7:0] out_t;
wire done_t;
gcd_machine GCD( clk_t, rst_t, go_t, in1_t,
in2_t, out_t, done_t);
always
begin
clk_t <=0;
#25;
clk_t <=1;
#25;
end
initial
begin
$dumpfile ("dump.vcd");
$dumpvars;
$dumpon;
#4000 $dumpoff;
end
```

```
initial
begin

rst_t <=1;
go_t <=0;
#100
rst_t <=0;

in1_t <=10;
in2_t <=5;
#100;
go_t <=1;
while (done_t ==1)
begin
#50; end
go_t <= 0;
while (done_t !==1)
begin #50; end
$monitor ("done =%b out =%b", done_t, out_t);

#2000; $finish;
end

endmodule
```

Output


Testbench Testbench Module DU Inst
GCD gcd_machine Module DU Inst
#ALWAYS#9 Testbench Process -
#INITIAL#16 Testbench Process -
#INITIAL#24 Testbench Process -
#vsim_capacity# Capacity Statisti
clk_t
done_t
go_t
in1_t
in2_t
out_t
rst_t
Name
#INITIAL#
#ALWAYS#
/Testbench/clk_t 1'h0
/Testbench/done_t 1'h1
/Testbench/go_t 1'h0
/Testbench/in1_t 8'h0a
/Testbench/in2_t 8'h05
/Testbench/out_t 8'h00
/Testbench/rst_t 1'h0
8'h0a
8'h05
8'h00
8'h05
Now 2550000 ps
Cursor 1 570488 ps
400000 ps
800000 ps
570488 ps
Library
Project
sim
Wave
controller.v
test_bench.v

As an assignment

- Change the port mapping to named ports
- Make the same GCD machine work for 32 bit input
- Change the MUX module. Write it either using case statement or assign
- Write a structural description for the 32 bit substractor such that delay in the path is minimized.