ELP305: DESIGN AND SYSTEM LABORATORY

EXPERIMENT 6: Design and implement GCD calculator

GROUP NUMBER: 22
EXPERIMENT DONE ON: 20/04/2018
SUBMITTED ON: 22/04/2018

SUBMITTED BY:
DRASHTI KHATSURIYA(2015MT10598)
CHARVI NAHAR(2015MT10595)
AVINASH KUMAR(2015MT10319)

Objective: To learn basics of ASIC design and prototyping

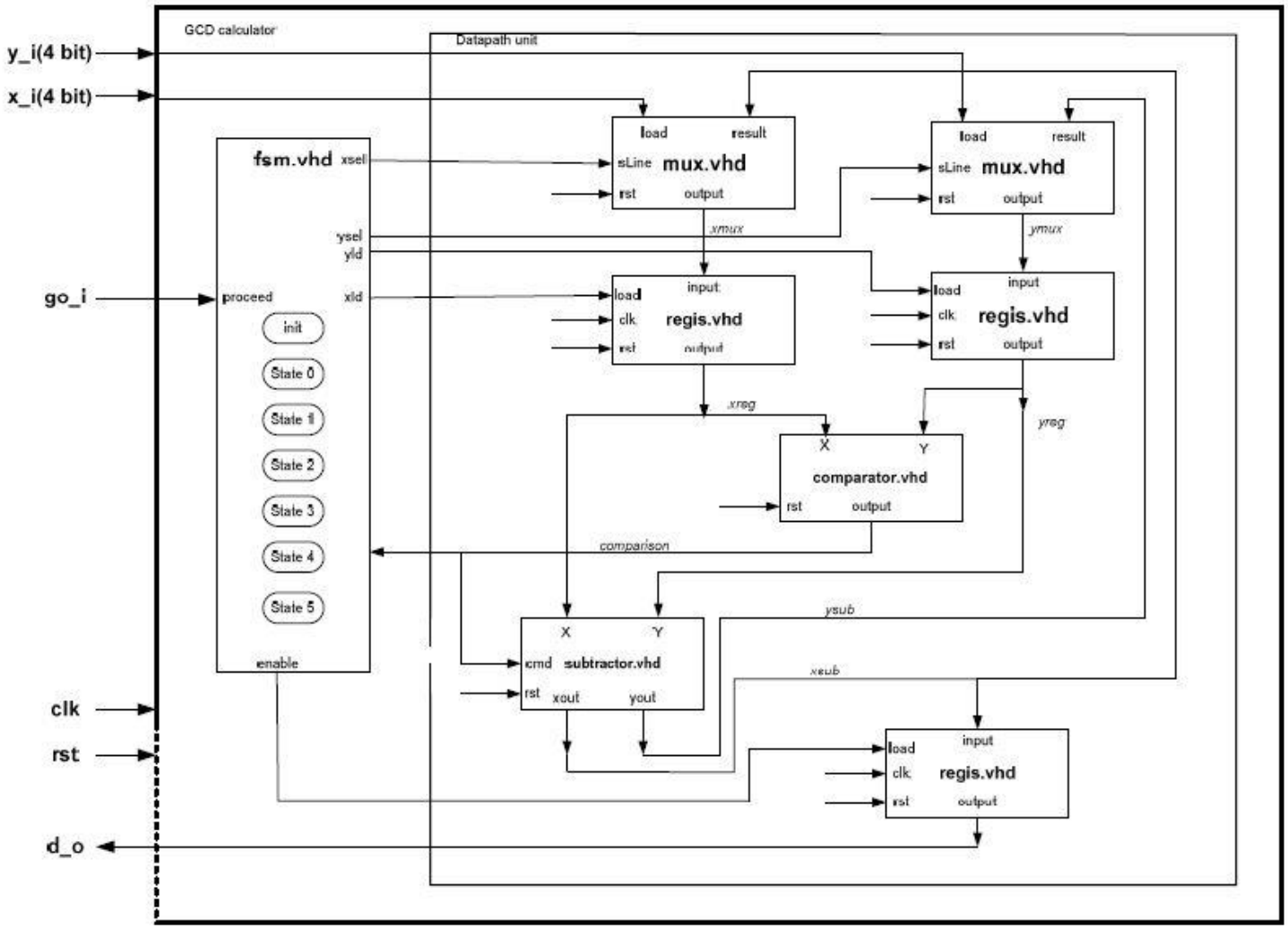
Problem Statement: Design and implement the greatest common divisor of two numbers. Design the data path of a gcd processor and construct a Moore-type state table defining the control unit of the gcd processor, an excitation table, logic equations and a complete logic circuit that uses D flip-flop and NAND gates only. Code the design in VHDL and implement on a FPGA board.

Components required: Xilinx Software

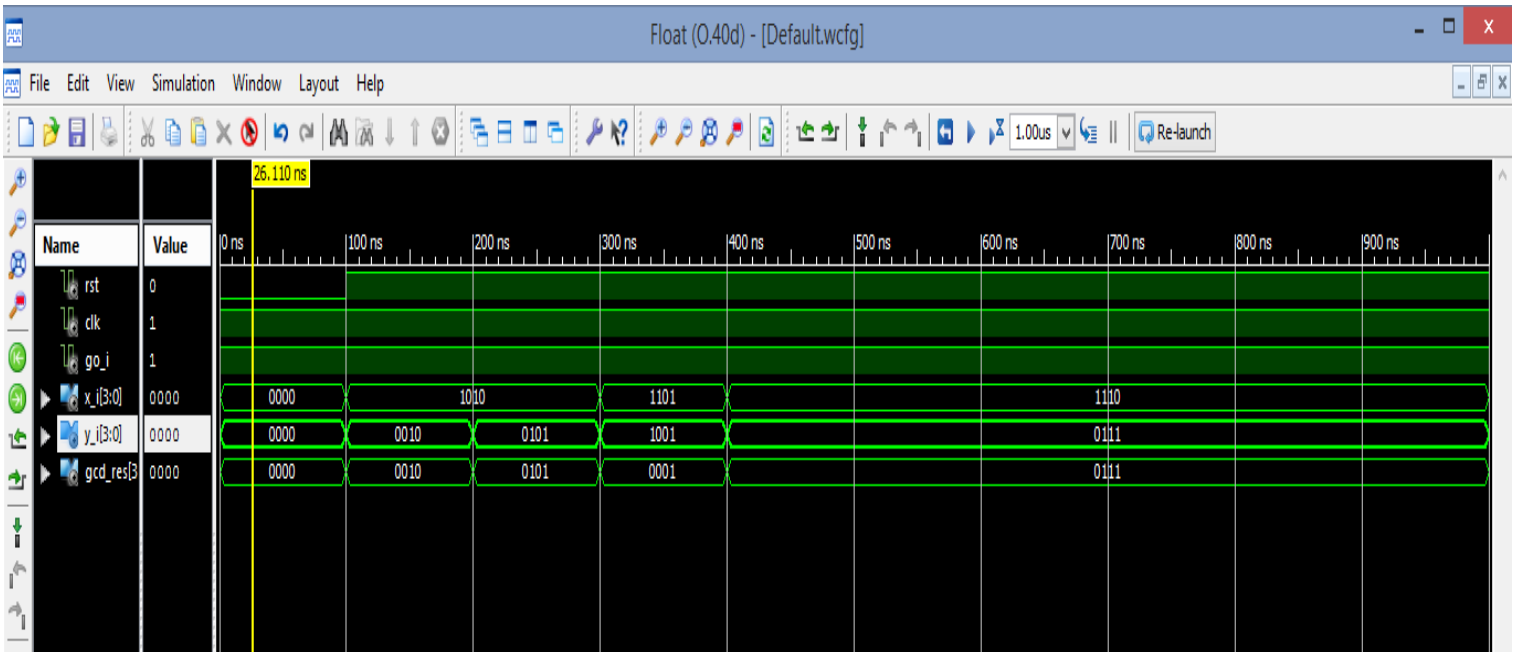
Challenges Faced:

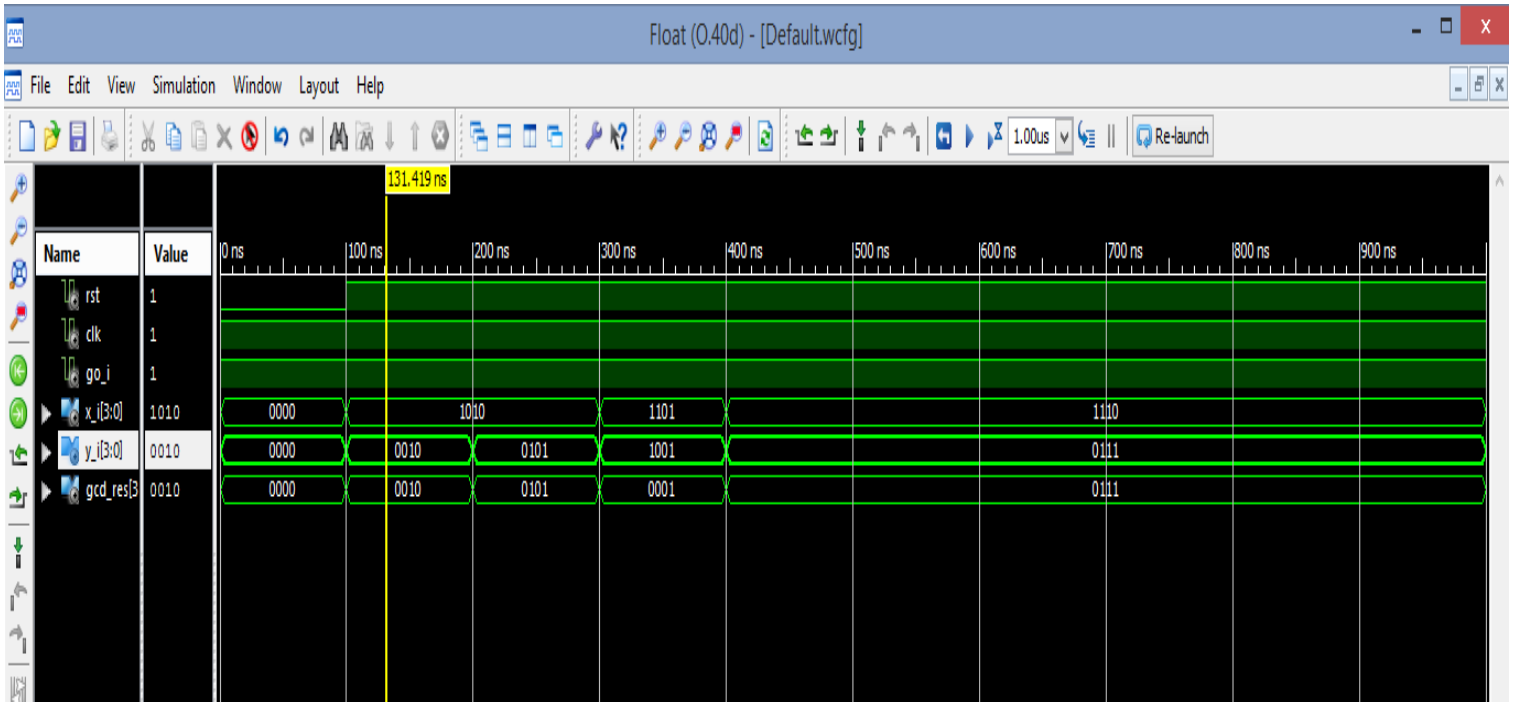
Using simple while loop for the Euclidean division algorithm to implement the GCD calculator was easy but implementing using different components like register, subtractor, multiplexer, comparator posed difficulty.

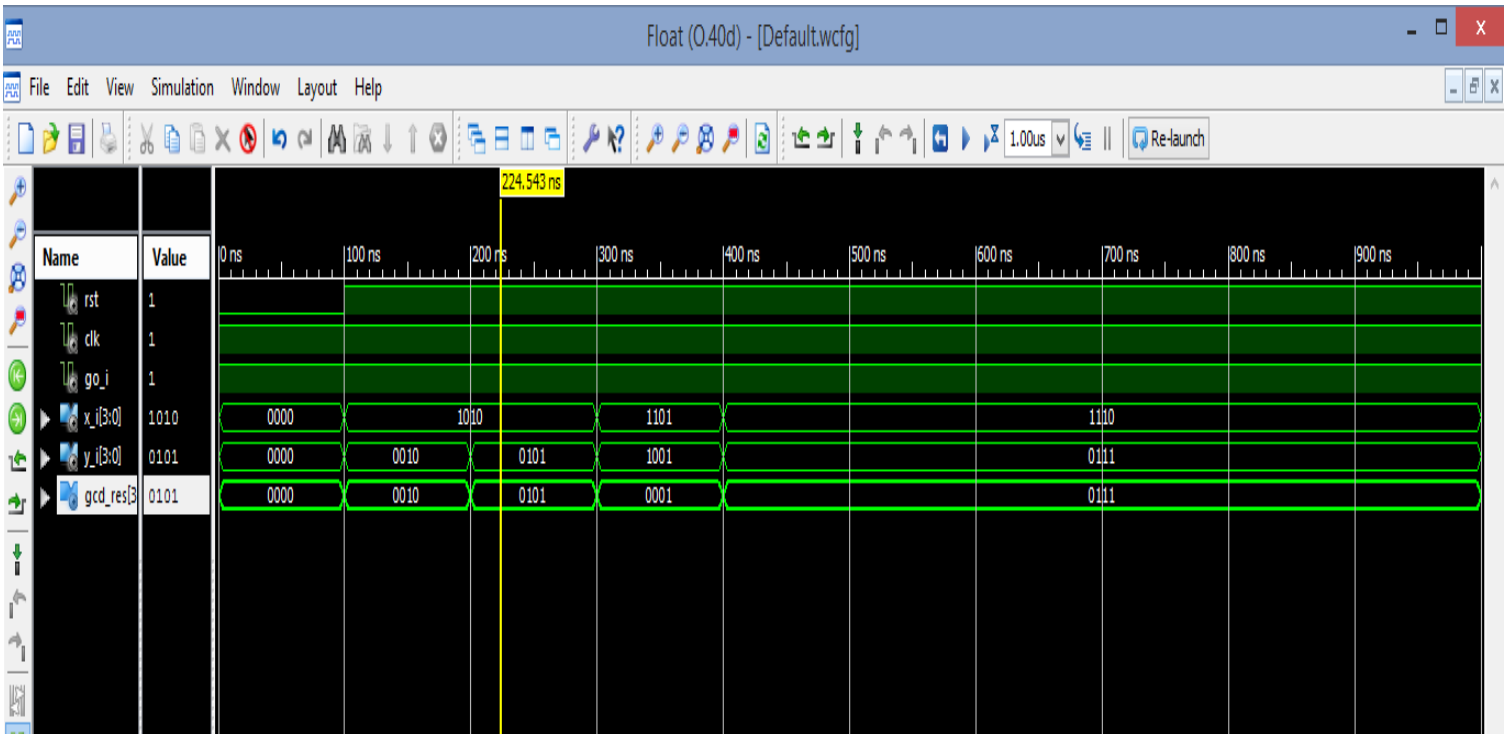
Block Diagram:

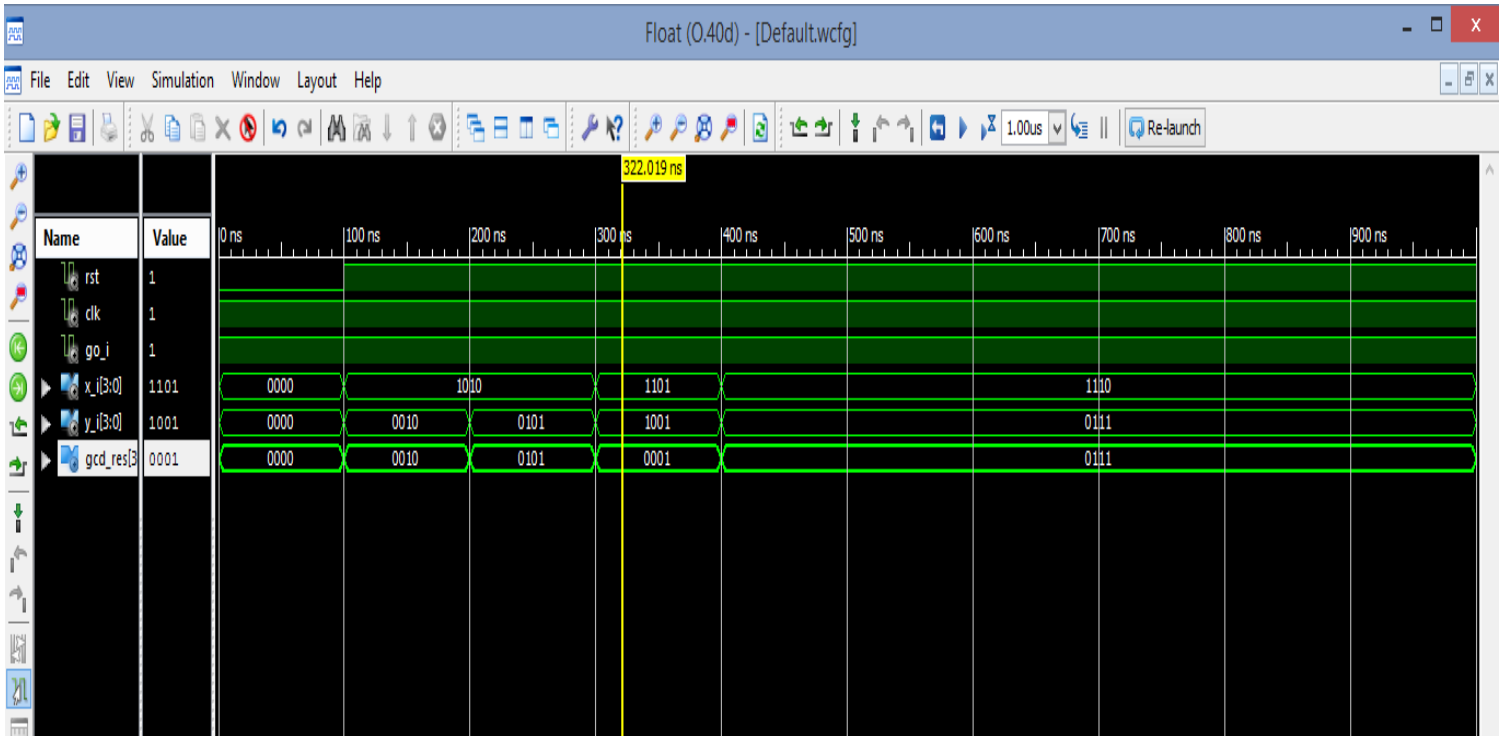


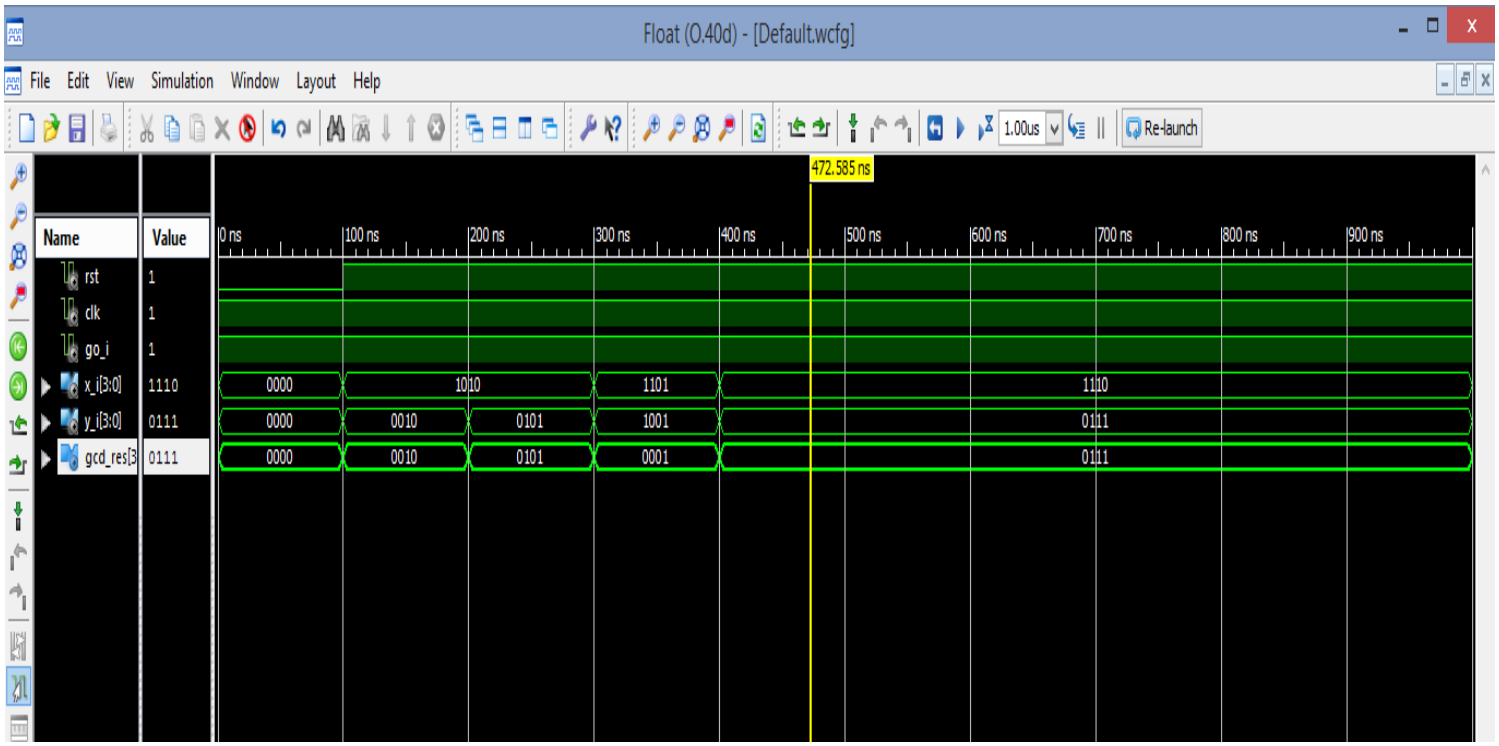
Observations:



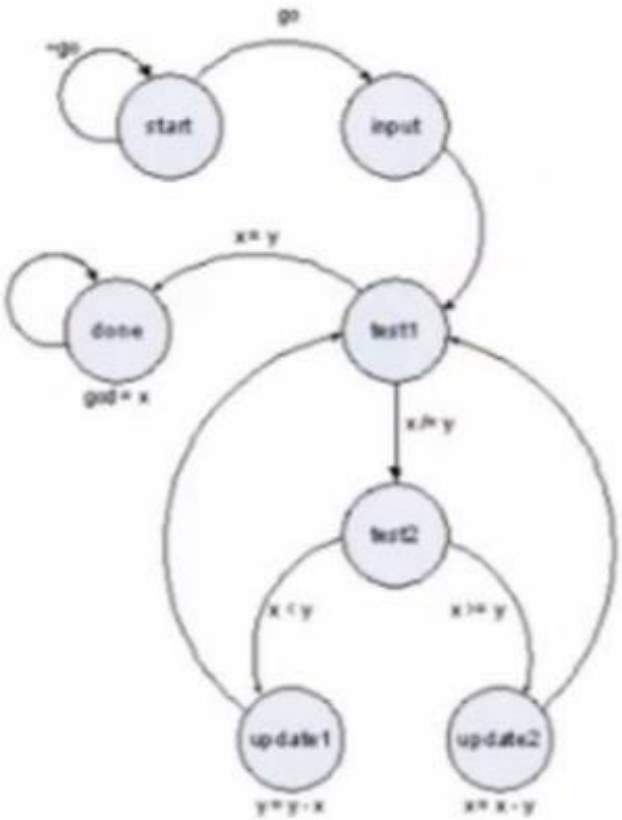

Float (0.40d) - [Default.wcfg]
File Edit View Simulation Window Layout Help
Re-launch
131.419 ns
Name
Value
rst
clk
go_i
x_i[3:0]
y_i[3:0]
gcd_res[3
1
1
1
1010
0010
0010
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
800 ns
900 ns
0000
1010
1101
1110
0000
0010
0101
1001
0111
0000
0010
0101
0001
0111


Float (0.40d) - [Default.wcfg]
File Edit View Simulation Window Layout Help
Re-launch
224.543 ns
Name
Value
rst
clk
go_i
x_i[3:0]
y_i[3:0]
gcd_res[3
1
1
1
1010
0101
0101
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
800 ns
900 ns
0000
1010
1101
1110
0000
0010
0101
1001
0111
0000
0010
0101
0001
0111


Float (0.40d) - [Default.wcfg]
File Edit View Simulation Window Layout Help
1.00us
Re-launch
322.019 ns
Name
Value
rst
1
clk
1
go_i
1
x_i[3:0]
1101
y_i[3:0]
1001
gcd_res[3
0001
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
800 ns
900 ns
0000
1010
1101
1110
0000
0010
0101
1001
0111
0000
0010
0101
0001
0111


Float (0.40d) - [Default.wcfg]
File Edit View Simulation Window Layout Help
1.00us
Re-launch
472.585 ns
Name
Value
rst
1
clk
1
go_i
1
x_i[3:0]
1110
y_i[3:0]
0111
gcd_res[3
0111
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
800 ns
900 ns
0000
1010
1101
1110
0000
0010
0101
1001
0111
0000
0010
0101
0001
0111

State Diagram for code:



In this state diagram--
Start is: init
Input is: s0
Test1 is: s1
Update1 is: S2
Update2 is: s3
Done is: s4

Discussion and Result:

The design of the GCD Calculator is divided into two parts: a controller and a datapath. The controller is a Finite State Machine which issues commands to the datapath based on the current state and external inputs. The datapath contains a net list of functional units like multiplexers, resistors, subtractors and a comparator and hence this design is structural. The controller steps through the GCD algorithm. If x=y, the GCD computing is finished and we go to the final state and assert the data output line. The datapath does the actual GCD computation. It has the following components:

- Mux
- Input register
- Comparator
- Subtractor
- Output register

The code implemented on the above description computes the GCD of X and Y.

References:

https://www.xilinx.com/itp/xilinx10/books/docs/qst/qst.pdf
http://esd.cs.ucr.edu/labs/tutorial/

Appendix (code):

Main file:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;


entity gcd is
	port( rst,clk,go_i: in std_logic;
			x_i,y_i: in std_logic_vector(3 downto 0);
			gcd_res: out std_logic_vector(3 downto 0)
			);
end gcd;

architecture gcd_arc of gcd is

component fsm is
	port( rst,clk,proceed: in std_logic;
			comparison: in std_logic_vector(1 downto 0);
			enable,xsel,ysel,xld,yld: out std_logic
			);
end component;

component mux is
	port( rst,sLine: in std_logic;
```

```
                load,result: in std_logic_vector(3 downto 0);
                output: out std_logic_vector(3 downto 0)
                );
end component;

component comparator is
        port( rst: in std_logic;
                x,y: in std_logic_vector(3 downto 0);
                output: out std_logic_vector(1 downto 0)
                );
end component;

component subtractor is
        port( rst: in std_logic;
                cmd: in std_logic_vector(1 downto 0);
                x,y: in std_logic_vector(3 downto 0);
                xout,yout: out std_logic_vector(3 downto 0)
                );
end component;

component regis is
        port( rst,clk,load: in std_logic;
                input: in std_logic_vector(3 downto 0);
                output: out std_logic_vector(3 downto 0)
                );
end component;


signal xld,yld,xsel,ysel,enable: std_logic;
signal comparison: std_logic_vector(1 downto 0);
signal result: std_logic_vector(3 downto 0);

signal xsub,ysub,xmux,ymux,xreg,yreg: std_logic_vector(3 downto 0);

begin
        --- doing structure modeling here by using portmap
        --- FSM controller
        TOFSM: fsm port map(rst,clk,go_i,comparison,enable,xsel,ysel,xld,yld);

        ---Datapath
        X_MUX: mux port map(rst,xsel,x_i,xsub,xmux);
        Y_MUX: mux port map(rst,ysel,y_i,ysub,ymux);
        X_REG: regis port map(rst,clk,xld,xmux,xreg);
```

```
        Y_REG: regis port map(rst,clk,yld,ymux,yreg);
        U_COMP: comparator port map(rst,xreg,yreg,comparison);
        X_SUB: subtractor port map(rst,comparison,xreg,yreg,xsub,ysub);
        OUT_REG: regis port map(rst,clk,enable,xsub,result);

        gcd_res <= result;

end gcd_arc;
```

Component:Mux

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;


entity mux is
        port( rst,sLine: in std_logic;
                        load, result: in std_logic_vector(3 downto 0);
                        output: out std_logic_vector(3 downto 0)
                        );
end mux;

architecture mux_arc of mux is
begin
        process(rst,sLine,load,result)
        begin
                if(rst = '1') then
                        output <= "0000";      ---do nothing
                elsif sLine = '0' then
                        output <= load;                         ---load inputs
                else
                        output <= result;      ---load result
                end if;
        end process;
end mux_arc;
```

Component:Register

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
```

```
entity regis is
	port(	rst,clk,load: in std_logic;
			input: in std_logic_vector(3 downto 0);
			output: out std_logic_vector(3 downto 0)
			);
end regis;

architecture regis_arc of regis is
begin
	process(rst,clk,load,input)
	begin
		if(rst = '1') then
			output<= "0000";
		elsif(clk'event and clk = '1')then
			if(load = '1') then
				output<= input;
			end if;
		end if;
	end process;
end regis_arc;
```

Component:subtractor

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity subtractor is
	port( rst: in std_logic;
			cmd: in std_logic_vector(1 downto 0);
			x,y: in std_logic_vector(3 downto 0);
			xout,yout: out std_logic_vector(3 downto 0)
			);
end subtractor;

architecture subtractor_arc of subtractor is
begin
	process(rst,cmd,x,y)
	begin
		if(rst = '1' or cmd = "00") then
			xout<= "0000";
			yout<= "0000";
```

```
		elsif(cmd = “10”) then
			xout<= (x-y);
			yout<= y;
		elsif(cmd=”01”) then
			xout<=x;
			yout<=(y-x);
		else
			xout<= x;
			yout<= y;
		end if;
	end process;
end subtractor_arc;
```

Component:fsm

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity fsm is
	port( rst,clk,proceed: in std_logic;
			comparison: in std_logic_vector(1 downto 0);
			enable,xsel,ysel,xld,yld: out std_logic
			);
end fsm;

architecture fsm_arc of fsm is
	type states is (init,s0,s1,s2,s3,s4);
	signal nstate,cstate: states;

begin
	process(rst,clk)
	begin
		if(rst = ‘1’) then
			cstate <= nstate;
		end if;
	end process;

	process(proceed,comparison,cstate)
	begin
		case cstate is

		when init => if(proceed = ‘0’) then
```

```
                                        nstate <= init;
                        else
                                        nstate <= s0;
                        end if;

when s0 => enable <= '0';
                        xsel <= '0';
                        ysel <= '0';
                        xld <= '1';
                        yld <= '1';
                        nstate <= s1;

when s1 => xld <= '0';
                        yld <= '0';
                        if(comparison = "10") then
                                        nstate <= s2;
                        elsif(comparison = "01") then
                                        nstate <= s3;
                        elsif(comparison = "11") then
                                        nstate <= s4;
                        end if;

when s2 => enable <= '0';
                        xsel <= '1';
                        ysel <= '0';
                        xld <= '1';
                        yld <= '0';
                        nstate <= s1;

when s3 => enable <= '0';
                        xsel <= '0';
                        ysel <= '1';
                        xld <= '0';
                        yld <= '1';
                        nstate <= s1;

when s4 => enable <= '1';
                        xsel <= '1';
                        ysel <= '1';
                        xld <= '1';
                        yld <= '1';
                        nstate <= s0;
```

```
                when others => nstate <= s0;

                end case;
        end process;

end fsm_arc;
```

Component:Comparator

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity comparator is
        port( rst: in std_logic;
            x, y: in std_logic_vector(3 downto 0);
            output: out std_logic_vector(1 downto 0)
            );
end comparator;

architecture comparator_arc of comparator is
begin
    process(x,y,rst)
    begin
        if(rst = '1') then
                output <= "00";
        elsif(x>y) then
                output<= "10";
        elsif(x<y) then
                output<= "01";
        else
                output<="11";
        end if;
    end process;
end comparator_arc;
```

****