EE214 GCD Report

Shashwat Shukla Roll Number 150260025

April 11, 2017

1 Overview of the experiment

The aim of this experiment is to implement an algorithm to find the Greatest Common Divisor(GCD) of 16 numbers, each of which is taken to be 16 bits long.

Euclid's algorithm is used to find the gcd of two numbers. It has been implemented in the component gcd2. gcd2 in turn uses a 16bit divider as a component.

I have combined main and gcd into one system level entity called gcd. gcd takes in inputs sequentially and invokes gcd2 repeatedly to arrive at the final answer, the gcd of the 16 numbers.

The control path of gcd and gcd2 employ FSMs and the data path makes use of registers. I have greatly reduced the number of states in the FSMs as compared to what the pseudocode specified.

Combining main and gcd has further simplified matters considerably and led to a more compact and efficient implementation.

Note that, in the pseudocode provided, gcd2 would first check if a>b and else would swap the two so that the divider would get it's dividend to be larger than the divisor. But this can be avoided as in the next cycle of gcd2, a and b automatically get swapped, due to how Euclid's algorithm works. Hence this takes one extra cycle, but has bypassed the use of the comparator and associated mux, simplifying things.

Various other components have been used: 16bit registers, 16bit MUX, D-flipflops, comparators, subtractor etc.

The next section contains state diagrams for gcd and gcd2.

This is followed by the code for all the components that I have written. All the other components that I have used (including the divider) were provided by Prof. Madhav Desai.

The results from GHDL, RTL and Scanchain testing can be found in the last section titled Observations.

2 State transition diagrams

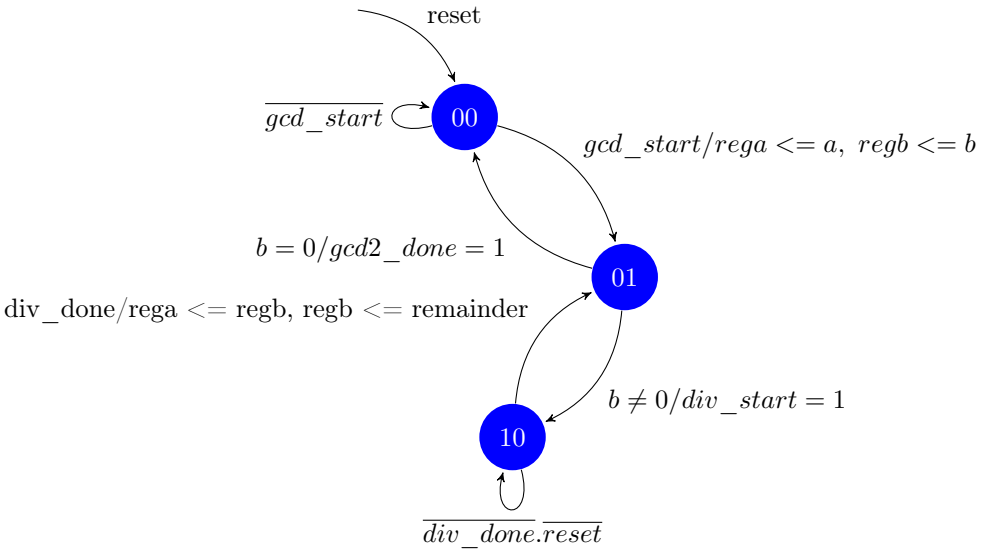

Figure 1: State diagram for gcd2

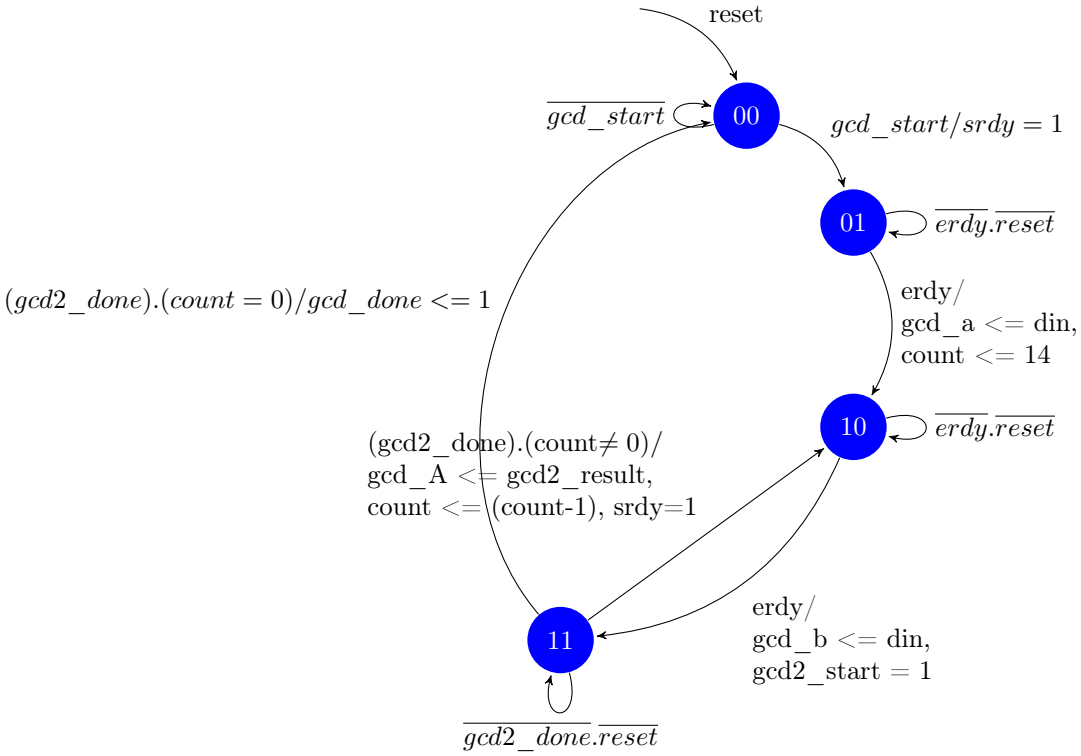

Figure 2: State diagram for gcd

3 Code for various components

3.1 gcd.vhd

```
library std;
use std.textio.all;
library std;
use std.standard.all;
library ieee;
use ieee.std_logic_1164.all;
library work;
use work.EE224_Components.all;
use work.GCDCOMPONENTS.all;

entity gcd is
        port(din: in std_ulogic_vector(15 downto 0);
                dout: out std_ulogic_vector(15 downto 0);
                start: in std_ulogic; done: out std_ulogic;
                erdy: in std_ulogic; srdy: out std_ulogic;
                clk: in std_ulogic; reset: in std_ulogic);
end entity;

architecture gcd16 of gcd is

component gcd2 is
        port(clk,reset,gcd2_start: in std_ulogic; gcd2_done: out std_ulogic;
                a,b: in std_ulogic_vector(15 downto 0); o: out std_ulogic_vector(15 downto
end component;

component register16 is
        port (clk,enable : in std_ulogic;
                input_vector : in std_ulogic_vector(15 downto 0);
                output_vector : out std_ulogic_vector(15 downto 0));
end component;

component MUX16 is
        port (a,b: in std_ulogic_vector(15 downto 0);
                e: in std_ulogic ;
                o: out std_ulogic_vector(15 downto 0));
end component;

component unsigned_comparator is
  generic (
    nbits : integer);
  port (
    a      : in  std_ulogic_vector(nbits-1 downto 0);
```

```
    b      : in  std_ulogic_vector(nbits-1 downto 0);
    a_lt_b : out std_ulogic;
    a_eq_b : out std_ulogic;
    a_gt_b : out std_ulogic);
end component;

signal s2,s1,ns2,ns1: std_ulogic;
signal sa,sb,sc,sd: std_ulogic; --States of the FSM

signal rega, regb: std_ulogic_vector(15 downto 0); --Input to the registers
signal ena, enb: std_ulogic; --Enable signals for the registers
signal a,b,o: std_ulogic_vector(15 downto 0); --Inputs and outputs for gcd2
signal gcd2_start, gcd2_done: std_ulogic;
signal count, countInput,countUpdate: std_ulogic_vector(15 downto 0); --Counter
signal low,equal,high: std_ulogic; --Comparator signals

begin

sa <= (not s2) and (not s1);
sb <= (not s2) and (s1);
sc <= (s2) and (not s1);
sd <= (s2) and (s1);

ns2 <= ((sb and erdy) or sc or (sd and gcd2_done and (not equal)) or (sd and (not gcd2_done
ns1 <= ((sa and start) or (sb and (not erdy)) or (sc and erdy) or (sd and (not gcd2_done))

gc: gcd2 port map (clk,reset,gcd2_start,gcd2_done,a,b,o);

ena <= (sb and erdy) or (sd and gcd2_done and (not equal));
enb <= (sc and erdy);

mux1: MUX16 port map(din,o,sb,rega);
mux2: MUX16 port map ("1111111111111100",countUpdate,sb,countInput);

regb <= din;

countUpdate(15 downto 1) <= count(14 downto 0);
countUpdate(0) <= '0';

reg1: register16 port map(clk,ena,rega,a);
reg2: register16 port map(clk,enb,regb,b);
reg3: register16 port map (clk, ena, countInput, count);

comp: unsigned_comparator generic map (nbits => 16 ) port map (count,"0000000000000000",low

gcd2_start <= sd;
```

```
srdy <= sb or sc;
done <= (sd and gcd2_done and equal);
dout <= o;

d2: DFF port map (d => ns2, clk => clk, q => s2);
d1: DFF port map (d => ns1, clk => clk, q => s1);

end gcd16;
```

3.2 gcd2.vhd

```
library std;
use std.textio.all;
library std;
use std.standard.all;
library ieee;
use ieee.std_logic_1164.all;
library work;
use work.EE224_Components.all;
use work.GCDCOMPONENTS.all;

entity gcd2 is
        port(clk,reset,gcd2_start: in std_ulogic; gcd2_done: out std_ulogic;
                a,b: in std_ulogic_vector(15 downto 0); o: out std_ulogic_vector(15 downto
end entity;

architecture twogcd of gcd2 is

component Divider16 is
   port ( divisor: in std_ulogic_vector(15 downto 0);
    dividend: in std_ulogic_vector(15 downto 0);
    quotient: out std_ulogic_vector(15 downto 0); remainder: out std_ulogic_vector(15 downt
    div_start: in std_ulogic; div_done: out std_ulogic;
    clk: in std_ulogic; reset: in std_ulogic);
end component;

component register16 is
        port (clk,enable : in std_ulogic;
                  input_vector : in std_ulogic_vector(15 downto 0);
                  output_vector : out std_ulogic_vector(15 downto 0));
end component;

component MUX16 is
        port (a,b: in std_ulogic_vector(15 downto 0);
                e: in std_ulogic;
```

```
                o: out std_ulogic_vector(15 downto 0));
end component;

component unsigned_comparator is
  generic (
    nbits : integer);
  port (
    a      : in  std_ulogic_vector(nbits-1 downto 0);
    b      : in  std_ulogic_vector(nbits-1 downto 0);
    a_lt_b : out std_ulogic;
    a_eq_b : out std_ulogic;
    a_gt_b : out std_ulogic);
end component;

signal s1,s2,ns1,ns2: std_ulogic;
signal l,m,n: std_ulogic; --States of the FSM

signal divisor,dividend: std_ulogic_vector(15 downto 0); --Inputs to the divider
signal quotient,remainder: std_ulogic_vector(15 downto 0); --Outputs of the divider
signal div_start, div_done: std_ulogic;

signal rega, regb: std_ulogic_vector(15 downto 0); --Input to the two registers
signal en: std_ulogic;

signal low,high,check: std_ulogic;
constant zero: std_ulogic_vector := "0000000000000000";

begin

l <= (not s2) and (not s1);
m <= (not s2) and (s1);
n <= (s2) and (not s1);

ns2 <= ((n and (not div_done)) or (m and (not check))) and (not reset);
ns1 <= ((gcd2_start and l) or (n and div_done)) and (not reset);

muxa: MUX16 port map(a,divisor,l,rega);
muxb: MUX16 port map(b,remainder,l,regb);

en <= (l and gcd2_start) or (n and div_done);

reg1: register16 port map(clk,en,rega,dividend); --For the dividend
reg2: register16 port map(clk,en,regb,divisor);  --For the divisor

checkdiv: unsigned_comparator generic map (nbits => 16 ) port map(divisor,zero,low,check,hi
```

```
div_start <= m and (not check);

divide: Divider16 port map(divisor,dividend,quotient,remainder,div_start,div_done,clk,'0');

o <= dividend;
gcd2_done <= m and check;

d2: DFF port map (d => ns2, clk => clk, q => s2);
d1: DFF port map (d => ns1, clk => clk, q => s1);

end twogcd;
```

3.3 register16.vhd

```
library std;
use std.textio.all;
library std;
use std.standard.all;
library ieee;
use ieee.std_logic_1164.all;
library work;
use work.EE224_Components.all;

entity register16 is
        port (clk,enable : in std_ulogic;
                input_vector : in std_ulogic_vector(15 downto 0);
                output_vector : out std_ulogic_vector(15 downto 0));
end entity;

architecture behave of register16 is

component bitreg is
port (clk, enable, d : in std_ulogic; o : out std_ulogic);
end component;

begin

g1      :bitreg port map (clk,enable,input_vector(0),output_vector(0));
g2      :bitreg port map (clk,enable,input_vector(1),output_vector(1));
g3      :bitreg port map (clk,enable,input_vector(2),output_vector(2));
g4      :bitreg port map (clk,enable,input_vector(3),output_vector(3));
g5      :bitreg port map (clk,enable,input_vector(4),output_vector(4));
g6      :bitreg port map (clk,enable,input_vector(5),output_vector(5));
g7      :bitreg port map (clk,enable,input_vector(6),output_vector(6));
g8      :bitreg port map (clk,enable,input_vector(7),output_vector(7));
g9      :bitreg port map (clk,enable,input_vector(8),output_vector(8));
```

```
g10     :bitreg port map (clk,enable,input_vector(9),output_vector(9));
g11     :bitreg port map (clk,enable,input_vector(10),output_vector(10));
g12     :bitreg port map (clk,enable,input_vector(11),output_vector(11));
g13     :bitreg port map (clk,enable,input_vector(12),output_vector(12));
g14     :bitreg port map (clk,enable,input_vector(13),output_vector(13));
g15     :bitreg port map (clk,enable,input_vector(14),output_vector(14));
g16     :bitreg port map (clk,enable,input_vector(15),output_vector(15));

end behave;
```

3.4 bitreg.vhd

```
library std;
use std.textio.all;
library std;
use std.standard.all;
library ieee;
use ieee.std_logic_1164.all;
library work;
use work.EE224_Components.all;

entity bitreg is
        port (clk, enable, d : in std_ulogic; o : out std_ulogic);
end entity;

architecture store of bitreg is

        component MUX is
                port (a,b,e : in std_ulogic ; o: out std_ulogic);
        end component MUX;

signal data,t :std_ulogic;

begin
        g1      :MUX port map (d,data,enable,t);
        g2      :DFF port map (d => t, clk => clk, q => data);
        o <= data;
end store;
```

3.5 MUX16.vhd

```
library std;
use std.textio.all;
library std;
use std.standard.all;
library ieee;
use ieee.std_logic_1164.all;
library work;
use work.EE224_Components.all;

entity MUX16 is
        port (a,b: in std_ulogic_vector(15 downto 0);
        e: in std_ulogic ; o: out std_ulogic_vector(15 downto 0));
end entity;

architecture behave of MUX16 is

component MUX is
        port (a,b,e : in std_ulogic ; o: out std_ulogic);
end component MUX;

begin

g15 : MUX port map (a(15),b(15),e,o(15));
g14 : MUX port map (a(14),b(14),e,o(14));
g13 : MUX port map (a(13),b(13),e,o(13));
g12 : MUX port map (a(12),b(12),e,o(12));
g11 : MUX port map (a(11),b(11),e,o(11));
g10 : MUX port map (a(10),b(10),e,o(10));
g9  : MUX port map (a(9),b(9),e,o(9));
g8  : MUX port map (a(8),b(8),e,o(8));
g7  : MUX port map (a(7),b(7),e,o(7));
g6  : MUX port map (a(6),b(6),e,o(6));
g5  : MUX port map (a(5),b(5),e,o(5));
g4  : MUX port map (a(4),b(4),e,o(4));
g3  : MUX port map (a(3),b(3),e,o(3));
g2  : MUX port map (a(2),b(2),e,o(2));
g1  : MUX port map (a(1),b(1),e,o(1));
g0  : MUX port map (a(0),b(0),e,o(0));

end behave;
```

3.6 MUX.vhd

```
library std;
use std.textio.all;
library std;
use std.standard.all;
library ieee;
use ieee.std_logic_1164.all;
library work;
use work.EE224_Components.all;

entity MUX is
        port (a,b,e : in std_ulogic ; o: out std_ulogic);
end entity MUX;

architecture behave of MUX is
signal s1,s2 : std_ulogic ;

begin

s1 <= a and e;
s2 <= b and (not e);
o <= s1 or s2;

end behave;
```

4 Observations

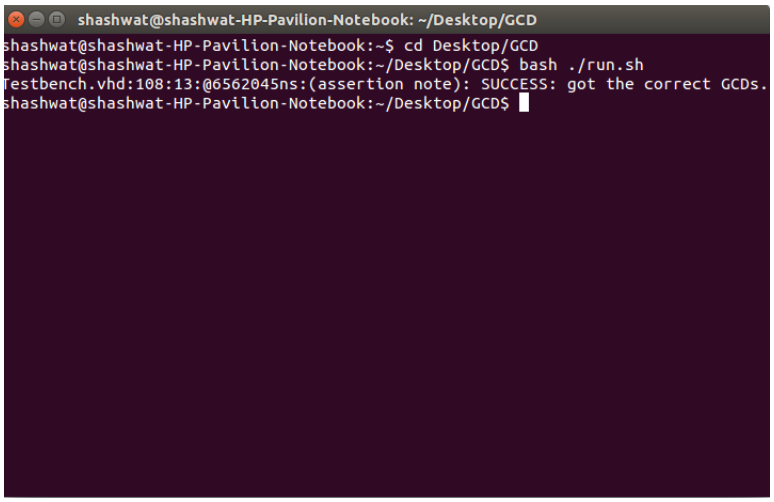

Figure 3: GHDL simulation

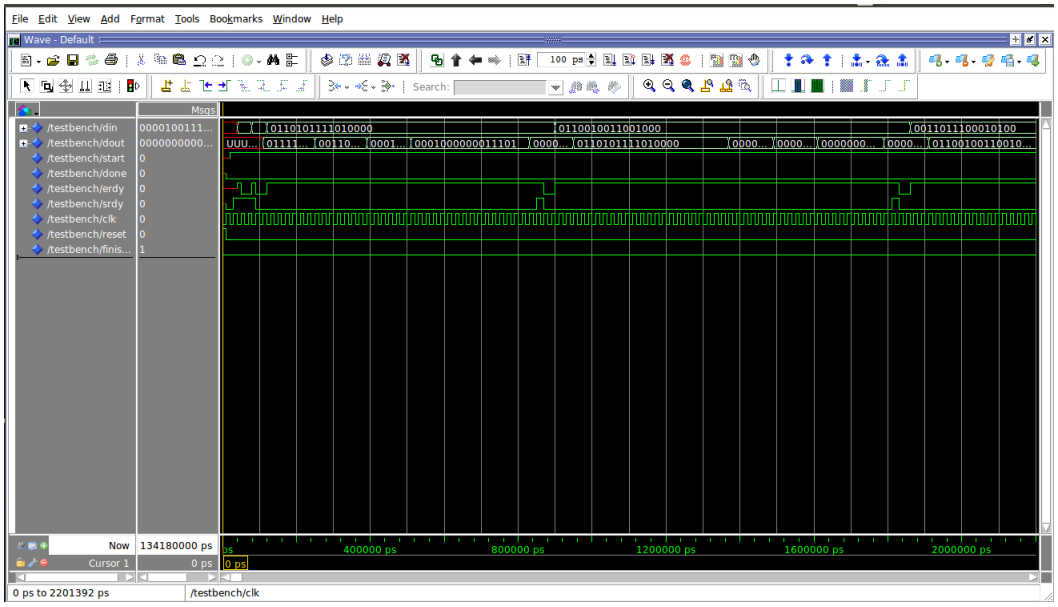

Figure 4: RTL waveforms

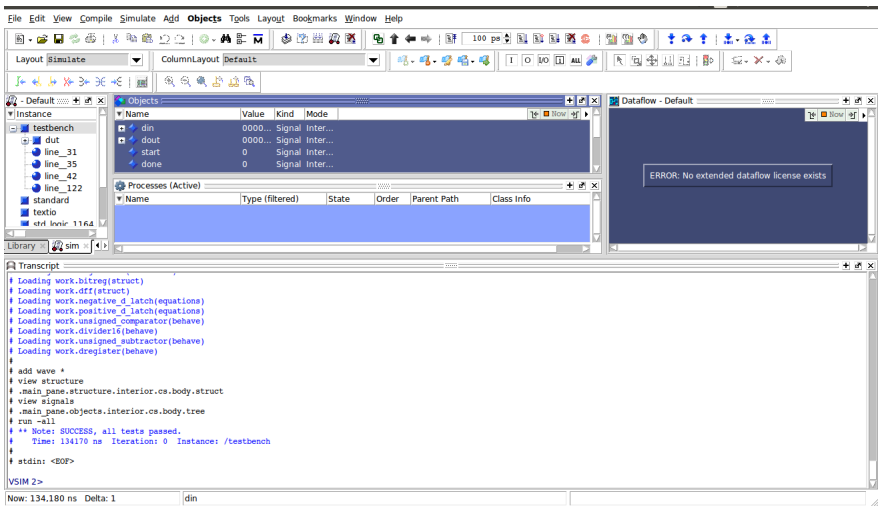

Figure 5: RTL simulation

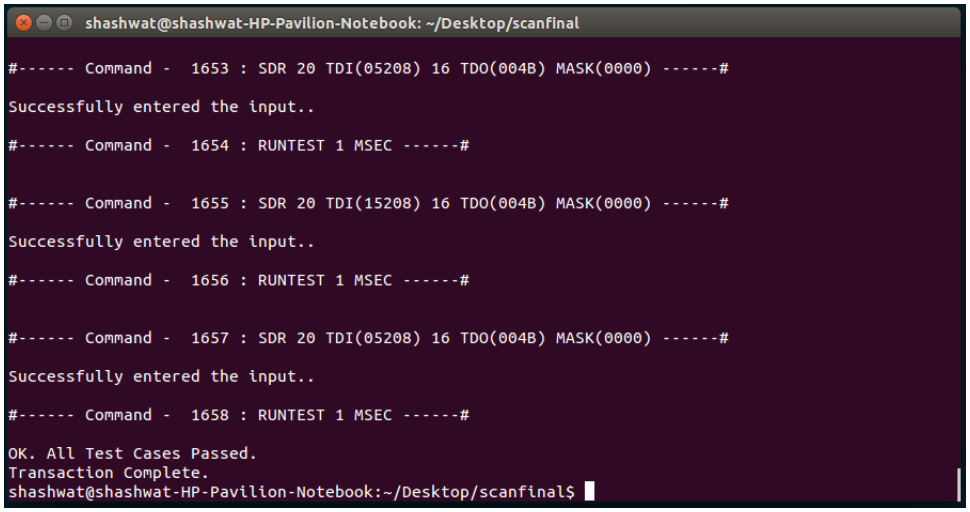

Figure 6: Successful completion of scan chain