

---

## EE-518 Digital IC LAB

---

### [ Experiment 8 : ]

**Design and implement GCD (Greatest Common Divisor) computation for two given numbers. Draw finite state machine and data path. Write the Verilog code and include output waveforms. ]**



Submitted by, Pushpendra Gurjar

Roll No. - 234102414

MTech.-VLSI & Nanoelectronics , EEE [ 2023-2025 ]

# Contents

<b>1</b>	<b>AIM</b>	<b>2</b>
<b>2</b>	<b>Objective</b>	<b>2</b>
<b>3</b>	<b>Theory</b>	<b>2</b>
3.1	Greatest Common divisor (GCD) . . . . .	2
<b>4</b>	<b>Verilog Code</b>	<b>3</b>
4.1	Code . . . . .	3
4.2	Test Bench . . . . .	4
4.3	RTL schemetic . . . . .	4
4.4	Functionality . . . . .	5
4.5	LUTs,Flip-Flops,Power,Delays Observations . . . . .	5
<b>5</b>	<b>Observation</b>	<b>6</b>
<b>6</b>	<b>Results</b>	<b>6</b>

# 1 AIM

Design and implement GCD (Greatest Common Divisor) computation for two given numbers. Draw finite state machine and data path. Write the Verilog code and include output waveforms.

## 2 Objective

- a) Design a GCD (Greatest Common Divisor) computation for two 8-bit numbers (x,y) and verify its functionality in Xilinx Vivado.
- b) Perform synthesis of the design and evaluate LUT's, FFs, Power and Delay of the GCD.

## 3 Theory

### 3.1 Greatest Common divisor (GCD)

The GCD (Greatest Common Divisor) of two numbers is the largest positive integer that divides both numbers without leaving a remainder. There are many ways to find the GCD of two numbers, including the Subtraction Method. The Subtraction Method is a simple and iterative algorithm for finding the GCD of two numbers. The basic idea is to subtract the smaller number from the larger number until both numbers become equal. The resulting number is the GCD of the original two numbers. Here is the step-by-step process for finding the GCD of two numbers using the Subtraction Method:

Take two numbers "a" and "b", where "a" is greater than or equal to "b". Subtract "b" from "a". If the result is greater than or equal to "b", replace "a" with the result and go back to step 2. Otherwise, let "a" be the smaller number and let "b" be the difference. The resulting number is the GCD of the original two numbers.

For example, let's find the GCD of 42 and 28 using the Subtraction Method:

$a = 42$  and  $b = 28$

Subtract "b" from "a":  $42 - 28 = 14$ . Since 14 is less than 28, let "a = 28" and "b = 14".

Subtract "b" from "a":  $28 - 14 = 14$ . Since both numbers are equal, stop.

The GCD of 42 and 28 is 14.

The Subtraction Method is a simple and easy-to-understand algorithm for finding the GCD of two numbers. However, it can be inefficient for large numbers or when the numbers are relatively prime (have no common factors). Other algorithms, such as the Euclidean Algorithm, are more efficient for these cases.

## 4 Verilog Code

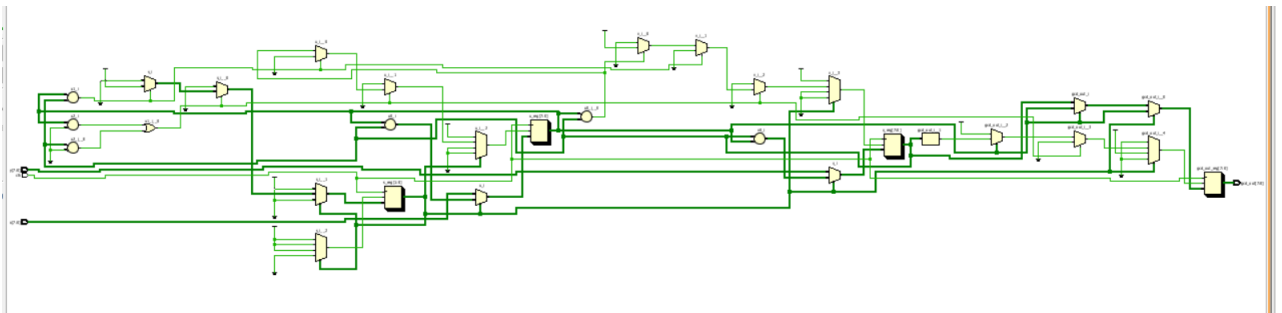
### 4.1 Code

```
1  *timescale 1ns / 1ps
2  module gcd_n(
3  input wire [7:0]x,
4  input wire [7:0]y,
5  input clk,
6  output reg [7:0]gcd_out);
7
8
9  reg [7:0]v;
10 reg [7:0]u;
11 reg [1:0]s=0;
12
13
14 always@(posedge clk)
15 begin
16     case(s)
17     2'b00: begin
18         u=x;
19         v=y;
20         s=2'b01;
21     end
22
23     2'b01:begin
24         if(u==0||v==0)
25         begin
26             if(u==0)
27                 gcd_out=v;
28             else if(v==0)
29                 gcd_out=u;
30             s=2'b00;
31         end
32     else
33     begin
34         if(u!=v)
35             if(u>v)
36                 begin
37                     u=u-v;
38                     s=2'b01;
39                 end
40             else
41                 begin
42                     v=v-u;
43                     s=2'b01;
44                 end
45             else
46                 s=2'b11;
47             end
48         end
49         2'b11:
50         begin
51             gcd_out=u;
52             s=2'b00;
53         end
54     endcase
55 end
56 endmodule
57
```

## 4.2 Test Bench

```
1 `timescale 1ns / 1ps
2 module gcd_n_tb();
3
4
5 reg [7:0]a,b;
6 reg clk;
7 wire [7:0]c;
8
9 gcd_n g(a,b,clk,c);
10
11 initial
12 begin
13   clk=1'b0;
14
15   forever #5 clk=~clk;
16 end
17 //in=1'b0;
18 initial
19 begin
20   a=8'd4;b=8'd0;
21   #100
22   a=8'd128;b=8'd64;
23   #100
24   a=8'd22;b=8'd33;
25   #100
26   a=8'd45;b=8'd81;
27   #200
28   $finish;
29
30 end
31 endmodule
```

## 4.3 RTL schemetic



4.4 Functionality

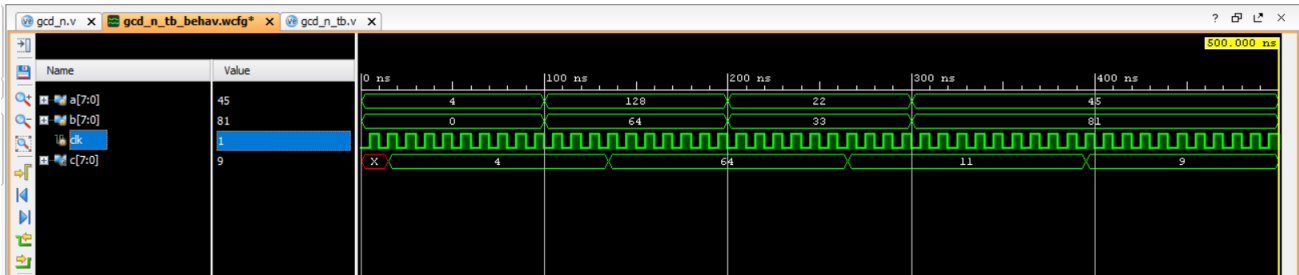


Figure 1: Output Waveform

4.5 LUTs,Flip-Flops,Power,Delays Observations

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM	URAM	PCIE %	Start	Elapsed	Strategy
synth_1	constrs_1	synth_design Complete!								49	26	0	0	0.000	9/27/23 6:02 PM	00:00:36	Vivado Synthesis Default
impl_1	constrs_1	Not started															Vivado Implementation C

Figure 2: Number of LUT's

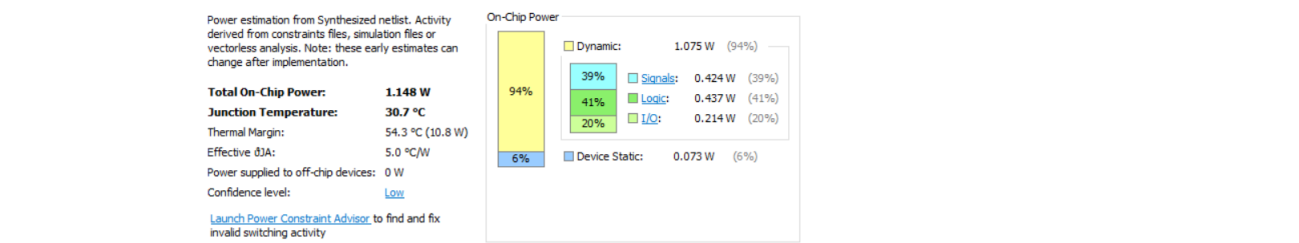


Figure 3: Power

Location	Delay type	Incr(ns)	Path(ns)	Netlist Resource(s)
(clock sys_clk_pin rise edge)				
		0.000	0.000 r	
		0.000	0.000 r	clk (IN)
net (fo=0)		0.000	0.000	clk
IBUF (Prop_ibuf_I_O)		0.152	0.152 r	clk_IBUF_inst/O
net (fo=1, unplaced)		0.337	0.489	clk_IBUF
BUFG (Prop_bufg_I_O)		0.026	0.515 r	clk_IBUF_BUFG_inst/O
net (fo=26, unplaced)		0.114	0.629	clk_IBUF_BUFG
FDRE				r s_reg[1]/C
FDRE (Prop_fdre_C_Q)		0.141	0.770 f	s_reg[1]/Q
net (fo=7, unplaced)		0.213	0.983	s[1]
LUT3 (Prop_lut3_I2_O)		0.099	1.082 r	s[0]_i_1/O
net (fo=1, unplaced)		0.000	1.082	s[0]_i_1_n_0
FDRE				r s_reg[0]/D
(clock sys_clk_pin rise edge)				
		0.000	0.000 r	
		0.000	0.000 r	clk (IN)
net (fo=0)		0.000	0.000	clk
IBUF (Prop_ibuf_I_O)		0.340	0.340 r	clk_IBUF_inst/O
net (fo=1, unplaced)		0.355	0.695	clk_IBUF
BUFG (Prop_bufg_I_O)		0.029	0.724 r	clk_IBUF_BUFG_inst/O
net (fo=26, unplaced)		0.259	0.983	clk_IBUF_BUFG
FDRE				r s_reg[0]/C
clock pessimism		-0.209	0.774	
FDRE (Hold_fdre_C_D)		0.091	0.865	s_reg[0]
required time			-0.865	
arrival time			1.082	
slack			0.217	

Figure 4: Delay

## 5 Observation

Circuit parameters of GCD

Circuit	LUT's	flip flops	Delay(ns)	power(W)
GCD	49	26	1.082	1.075

## 6 Results

I Sucesfully Calculated the no:of LUT's, flip flops , delay and power required for a GCD (Greatest Common Divisor) computation for two numbers which are noted above in table.