Digital Logic Laboratory

Report On Lab #3 – GCD Algorithm

Ana Maghradze

7th November 2021

Contents

1 Description of The Assignment

For this assignment, we have to implement GCD algorithm. For 2 inputs A and B, we should enter 8-bit numbers from 4-bit switches. The numbers may be positive or negative, they are entered in 2's complement form and we should convert them to find GCD. We need such an implementation that can be clocked with more than 60 Mhz.

2 Description of The Solution

In my implementation, to receive 8-bit inputs from 4-bit switches, I have state register to count how many times the `KEY[0]` is pressed. On first press we go from 0th to first state. On the second press, 4 most significant bits of A are entered from the switch, on the third press, 4 least significant bits of A are entered. The same happens for B on next two key presses. On the 6th press I set `start = 1` that is GCD calculation trigger. On the key[1] press, state becomes 0 and all `LED`s are turned off.

I have separate module for GCD and instantiate it under main module. When both inputs are provided, I notify GCD module by `start` signal to start calculation. In GCD, on initial state I take A and B inputs and as they are in 2's complement form, firstly subtract 1 from them and then invert. When `start = 1`, calculation is on the next state follows the algorithm: if `a = 0` gcd result becomes b, if `b = 0`, gcd result becomes a, otherwise I check which number is greater and find remainder for the greater number divided by another number. When division is finished, I send ready signal and result to the main module, then return to the initial state.

3 ModelSim Simulation

3.1 GCD of -15 and -20

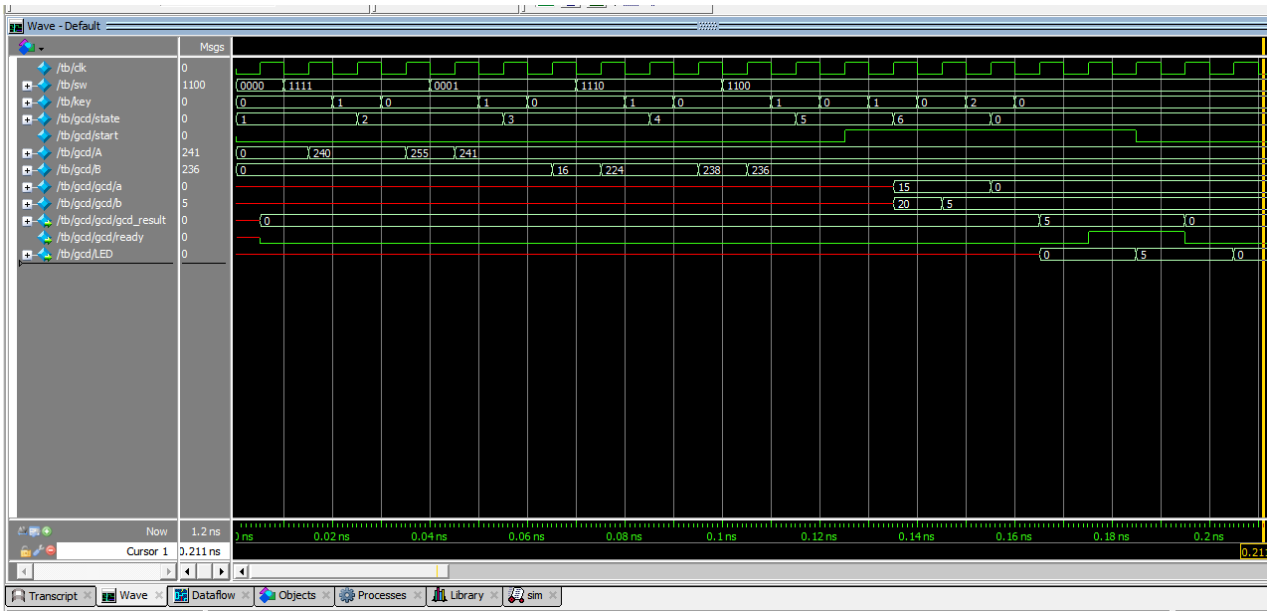

Figure 1: GCD(-15, -20) = 5

3.2 GCD of 126 and 88

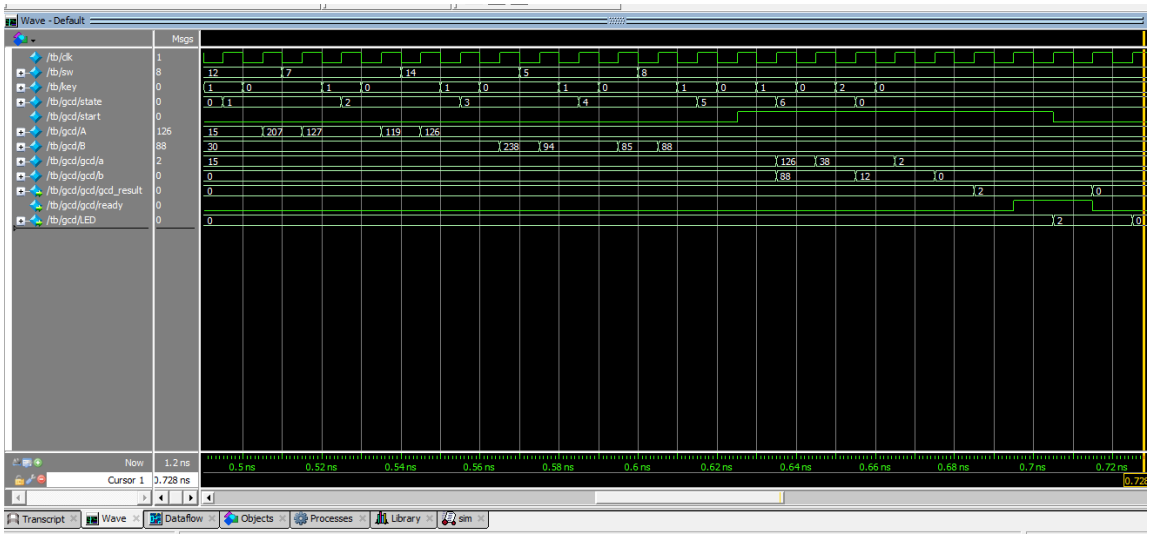

Figure 2: GCD(126, 88) = 2

3.3 GCD of -17 and 19

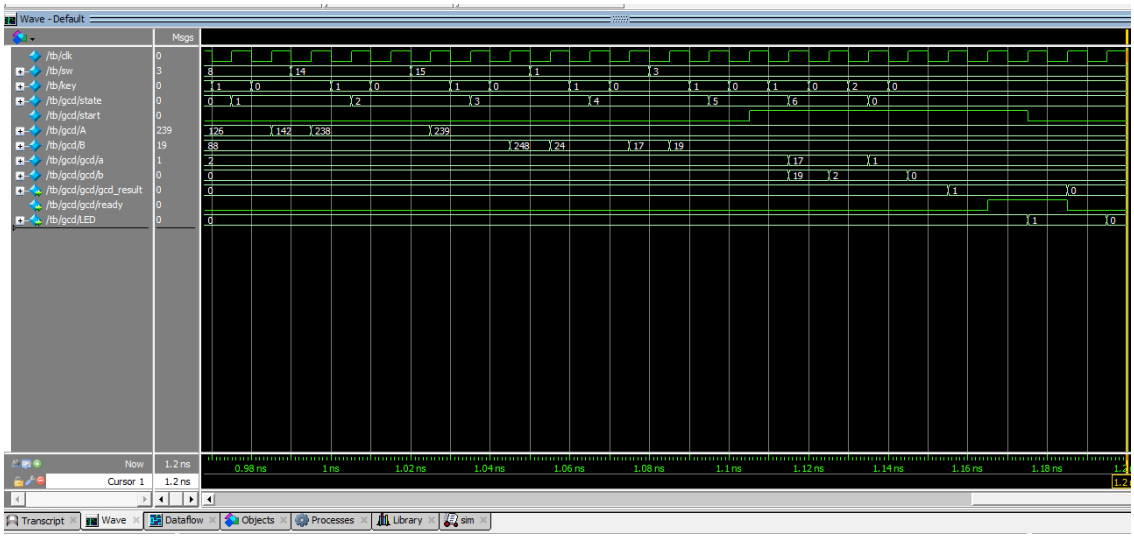

Figure 3: GCD(-17, 19) = 1

4 Appendix

```
module de0_nano_soc_baseline(
  input CLOCK_50,
  input [3:0] SW,
  input [1:0] KEY,
  output reg [7:0] LED
);

reg [1:0] prevKey = 1;
reg [7:0] A = 0;
reg [7:0] B = 0;

reg [2:0] state = 1;
reg start = 0;

wire [7:0] result;
wire result_ready;

GCD gcd(CLOCK_50, start, A, B, result, result_ready);

always @(posedge CLOCK_50)
begin

  if(KEY[0] == 1 && prevKey[0] == 0) state <= state + 1;
  if(KEY[1] == 1 && prevKey[1] == 0) state <= 0;

  case(state)
    0: LED <= 0;
    1: A[7:4] <= SW;
    2: A[3:0] <= SW;
    3: B[7:4] <= SW;
    4: B[3:0] <= SW;
    5: start <= 1;  // start gcd calculation
  endcase

  if(result_ready)
  begin
    LED <= result;
    start <= 0;
  end

  prevKey[1:0] <= KEY[1:0];
end

endmodule
```

```
 module GCD(
  input clk, start,
  input signed [7:0] A,
  input signed [7:0] B,
  output reg [7:0] gcd_result,
  output reg ready
);

reg [7:0] a;
reg [7:0] b;
reg [1:0] state = 0;

always @(posedge clk)
begin
  if(state == 0)
  begin
    gcd_result <= 0;
    ready <= 0;
    if(start)
    begin
      a <= A[7] == 1 ? ~(A - 8'b00000001) : A;
      b <= B[7] == 1 ? ~(B - 8'b00000001) : B;
      state <= 1;
    end
```

```
  end
  else if(state == 1)
  begin
    if(a == 0 || b == 0)
    begin
      gcd_result[7:0] <= a == 0 ? b : a;
      state <= 2;
    end
    else if(a >= b) a <= a % b;
    else if(b > a) b <= b % a;
  end
  if(state == 2)
  begin
    ready <= 1;
    if(ready) state <= 0;
  end
end
endmodule
```

```
module tb;
reg clk = 0;
reg [3:0] sw = 0;
reg [1:0] key = 0;
wire [7:0] led;

de0_nano_soc_baseline gcd(clk, sw, key, led);

always #5 clk <= ~clk;

initial
begin
  // gcd(-15, -20) = 5;
  #10 sw = 4'b1111;
  #10 key = 1;
  #10 key = 0;

  #10 sw = 4'b0001;
  #10 key = 1;
  #10 key = 0;

  #10 sw = 4'b1110;
  #10 key = 1;
  #10 key = 0;

  #10 sw = 4'b1100;
  #10 key = 1;
  #10 key = 0;

  #10 key = 1;
  #10 key = 0;

  #10 key = 2;
  #10 key = 0;

  #300
  #10 key = 1;
  #10 key = 0;

  // gcd(126, 88) = 2
  #10 sw = 4'b0111;
  #10 key = 1;
  #10 key = 0;

  #10 sw = 4'b1110;
  #10 key = 1;
  #10 key = 0;

  #10 sw = 4'b0101;
  #10 key = 1;
  #10 key = 0;

  #10 sw = 4'b1000;
```

```
    #10 key = 1;
    #10 key = 0;

    #10 key = 1;
    #10 key = 0;

    #10 key = 2;
    #10 key = 0;

    #300
    #10 key = 1;
    #10 key = 0;

    // gcd(-17, 19) = 1
    #10 sw = 4'b1110;
    #10 key = 1;
    #10 key = 0;

    #10 sw = 4'b1111;
    #10 key = 1;
    #10 key = 0;

    #10 sw = 4'b0001;
    #10 key = 1;
    #10 key = 0;

    #10 sw = 4'b0011;
    #10 key = 1;
    #10 key = 0;

    #10 key = 1;
    #10 key = 0;

    #10 key = 2;
    #10 key = 0;
  end

  endmodule
```