CSCE 2301 - Digital Design I

Project 2 - Simple Calculator.

Dr. Mona Farouk

Ali Yassine - 900294483
Amena Hossam - 900202552

Program Design

System Block Diagram:

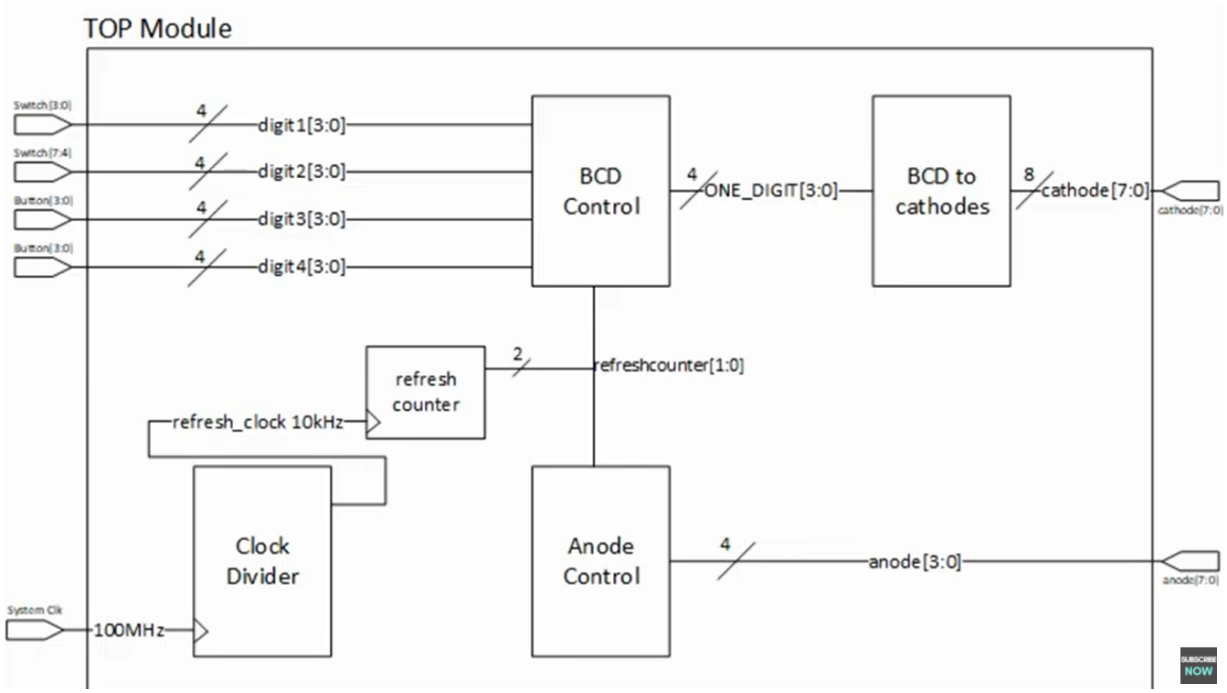


Figure 1

The used modules:

Goals:

1- Correct usage of arithmetic operations on the input.

2- Display different numbers on each digit of the seven segment display.

To achieve this, the below modules were implemented.

Calculator (To achieve arithmetic operations):

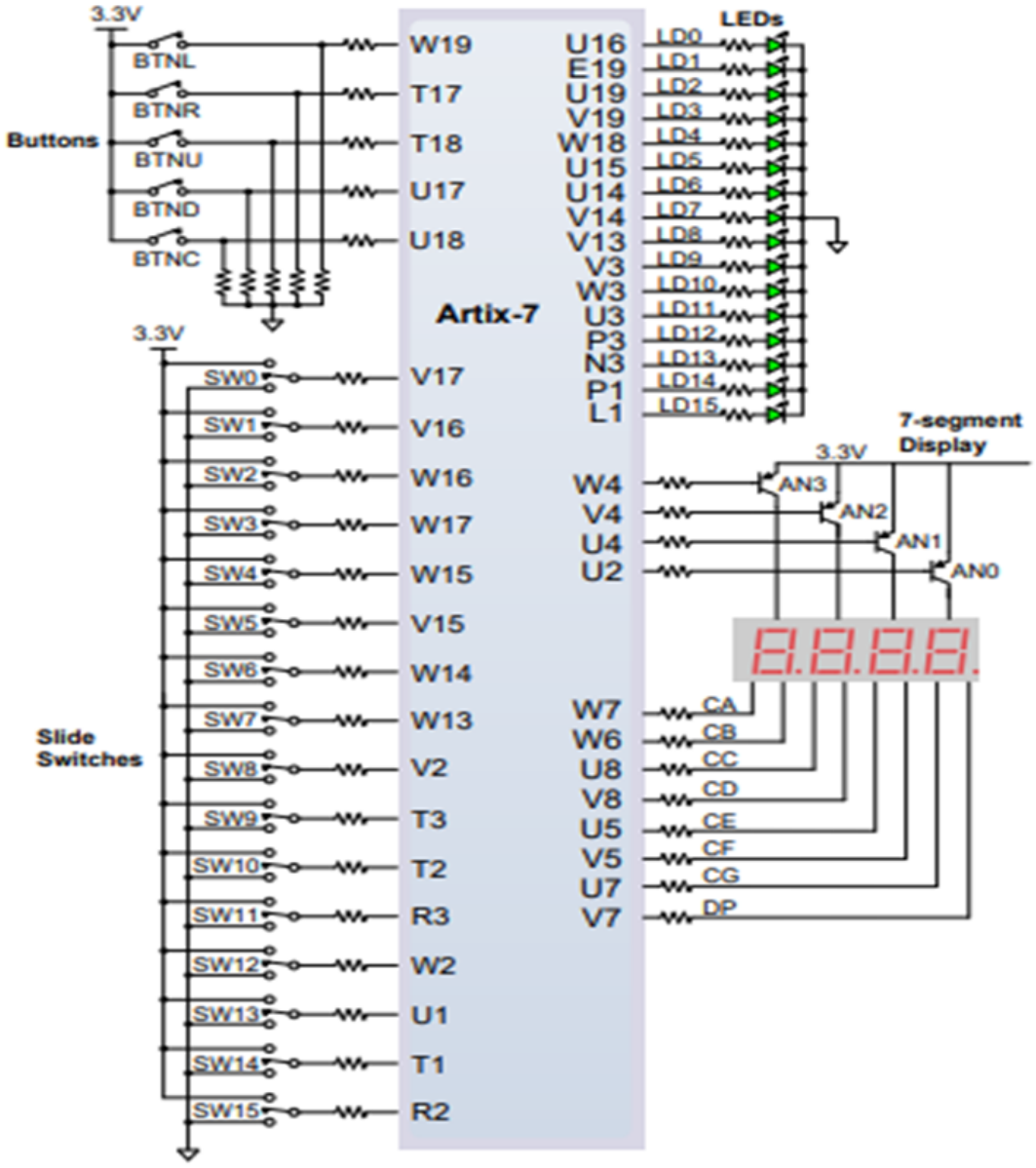


Figure 2

In this module, we defined what each switch does.

- Calculator outputs the 4 digits that will be displayed on the FPGA after applying the arithmetic operations on the input. They are out_1,out_2,out_3 and out_4, where out_1is the most left and out_4 the most right.
- If reset is 1, all outputs are set to 0, and reset is W14(refer to figure 2).

User Input (entering the two numbers):

Switch 5 to 8:

- Switch 8 is R2 (refer to figure 2). When this switch is 1 (raised) then this means that we are incrementing the tens of the first input number by 1.
- Switch 7 is T1 (refer to figure 2). When this switch is 1 (raised) then this means that we are incrementing the units of the first input number by 1.
- Switch 6 is U1 (refer to figure 2). When this switch is 1 (raised) then this means that we are incrementing the tens of the second input number by 1.
- Switch 5 is W2 (refer to figure 2). When this switch is 1 (raised) then this means that we are incrementing the units of the second input number by 1.

Applying Arithmetic operations on user input:

Switch 0to 4:

- Switch 4 is R3 (refer to figure 2). When this switch is 1 (raised) then this means that we are adding the two user inputs.
- Switch 3 is T2 (refer to figure 2). When this switch is 1 (raised) then this means that we are subtracting second user input from first user input.As stated the user inputs are 0 to 99 and 0 to 99, thus whatever the input is, the result of a subtraction regardless of the sign is a maximum of two digits. To display the negative sign in case the result is negative, we did a condition where it states that if the result is negative then the leftmost digit on the FPGA is 15 which is 1111 in binary. We then stated in the bcd_cathodes module that when the binary value is 1111 then we only put on G (refer to figure 3) and G represents the minus sign.

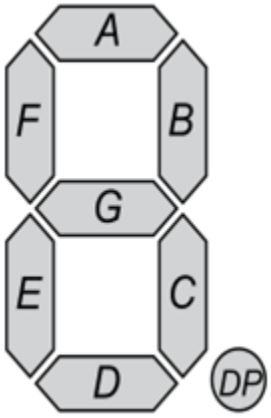


Figure 3

- Switch 2 is T3 (refer to figure 2). When this switch is 1 (raised) then this means that we are multiplying the two user inputs.
- Switch 1 is V2 (refer to figure 2). When this switch is 1 (raised) then this means that we are applying division, (first user input)/(second user input).
- Switch 0 is W13 (refer to figure 2). When this switch is 1 (raised) then this displays the result after applying the arithmetic operation.

Clock Divider:

This module is a standard clock divider. This clock divider is used to decrease the frequency clock signal from an input clock source, in our case the input clock source is the clock built in the FPGA itself which is 100Hz. Thus, this clock divider changes the value of the output after each n timeunit and n is a parameter we chose.(n=5000).

In other words, this module makes us capable of displaying different numbers on the FPGA at the same time. This is because the clock is so fast that it tricks the human eye to believe that all the numbers are displayed at the same time, although the digits are displayed one at a time based on the value of the counter from the binary counter module.

Binary Counter:

As stated before, one of our main goals is to find a way to display different numbers on the different digit areas of the FPGA(area 0, area 1 , area 2, area 3). To do this, the binary counter is a module that counts from 0 to 3 and then the output is passed to anode_controller and BCD control.

Anode_Controller:

In this module, we aim to specify which anode to put on. We used case statements for the value of the binary counter. Thus, when the output of the

binary counter is 0, anode number 0 is turned on (right most), when output of binary counter is 1, the anode number 1 is turned on and so on.

BCD Control:

This module takes the output from"calculator" as inputs and takes the binary counter value. If the binary counter value is 0, it assigns the output to the 0th digit, if the binary counter value is 1, it assigns the output to 1st digit,...

Bcd cathodes:

Used case statements to specify the binary value of each cathode according to the "digit", (output of BCD_Control).

Main:

Since each module in our projects depends on the one before, or in other words they are driven by each other, thus the output of a module is the input of another, the main is very essential to call all the modules with the appropriate parameters.

In the main, we call the clock divider, then pass its output as an input for the binary counter. After that, the output of the binary counter is an input for anode controller and BCD control. Outputs from "calculator",(answer after applying operations), are also inputs to BCD Control.Last but not least, the output of the BCD control is used as an input for BCD_to_cathodes.

Testbench & Simulation

A testbench is created for the module “calculator.” In the testbench, there are the following inputs that are passed to the instance of calculator (c1) in addition to the outputs of calculator (c1):

- **clk :** Input to the calculator that keeps changing values between 1 and 0 in the testbench. Its purpose is to simulate the clock of the FPGA.
- **reg [8:0] sw :** A 9 bit register variable that represents the switches responsible for the incrementation of the numbers, performing the operations, and displaying the original numbers.
- **digit1 :** Incremented by sw[8] and represents the tens value of the first number and should be displayed as the leftmost digit.
- **digit2:** Incremented by sw[7] and represents the unit value of the first number and should be displayed as the second digit from the left.
- **digit3:** Incremented by sw[6] and represents the tens value of the second number and should be displayed as the third digit from the left.
- **digit4:** Incremented by sw[5] and represents the tens value of the second number and should be displayed as the rightmost digit.

All the digits change after the performance of any operation to display the result.

- **operation_result:** stores the result after any operation is chosen and is shown only in the simulation.
- **dot:** dot simulates the decimal point between the two numbers to separate them. dot should only be = 1 (present) while displaying the two numbers, and it should be = 0 (off) when displaying the result of an operation to not confuse the output as containing decimal point.

All the required test cases are created. The simulation of incrementing the digit is done by setting the corresponding bit of the switch responsible for incrementing the number to 1.

For example: To increment the first digit (digit1) which is controlled by sw[9], we write the following: sw = 9’b100000000.

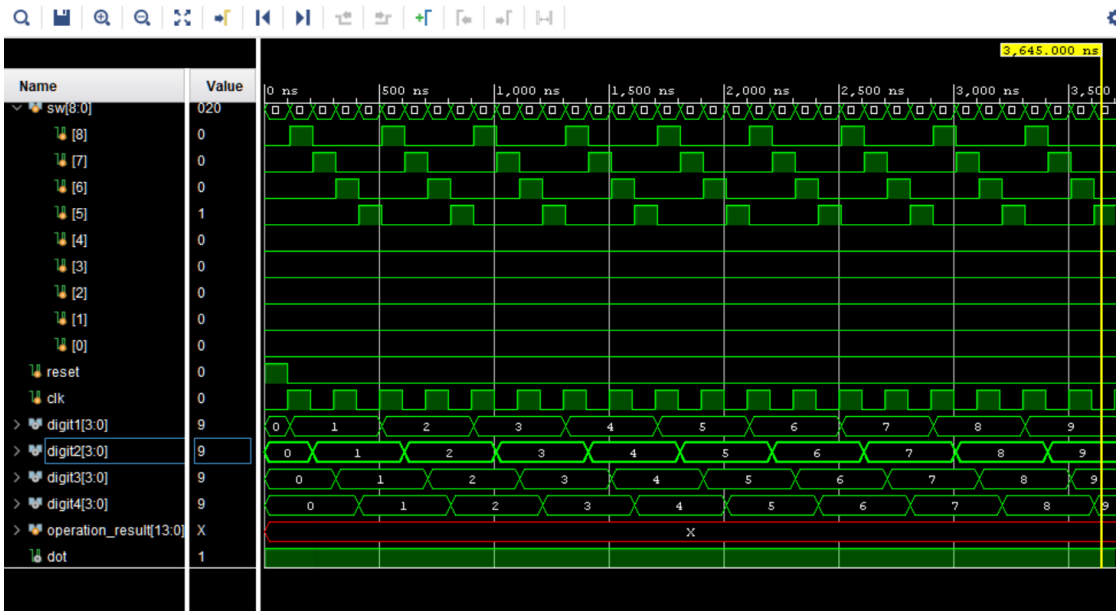
To perform an operation, we do the same. For example, multiplication is chosen by sw[2], so perform multiplication, we write the following : sw = 9’000000100.

Simulation results:

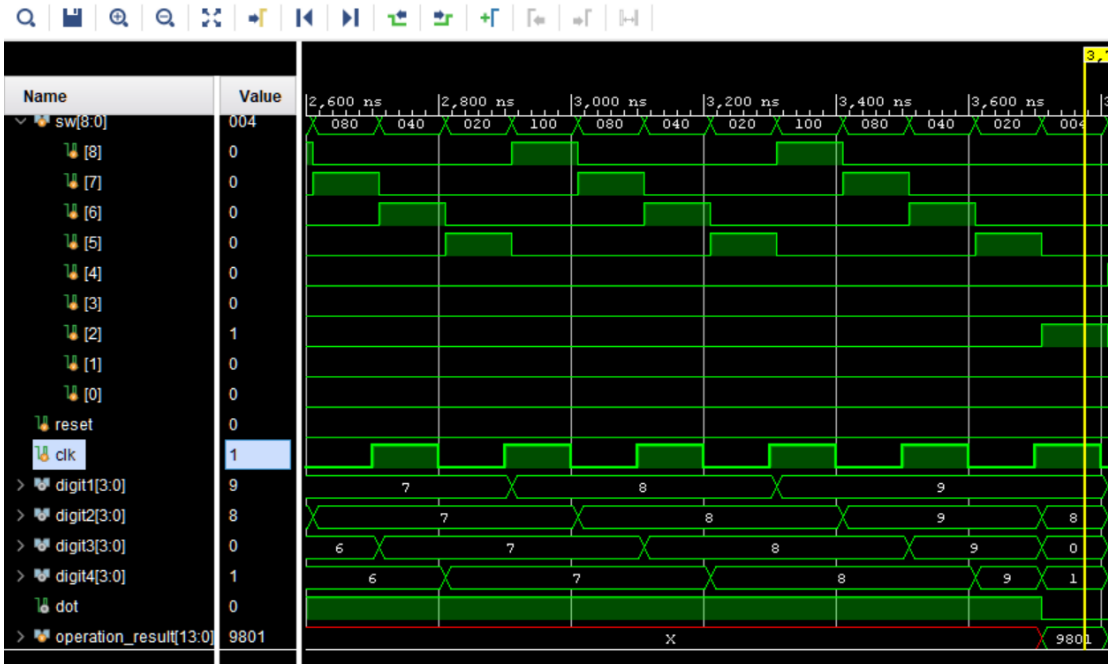
In the simulation, all the changes in the switches, digits1, digits2, digits3, digits4, dot, clk, reset, and operation_result are shown.

Note: For all the results to show, we need to set the time to 8000ns.

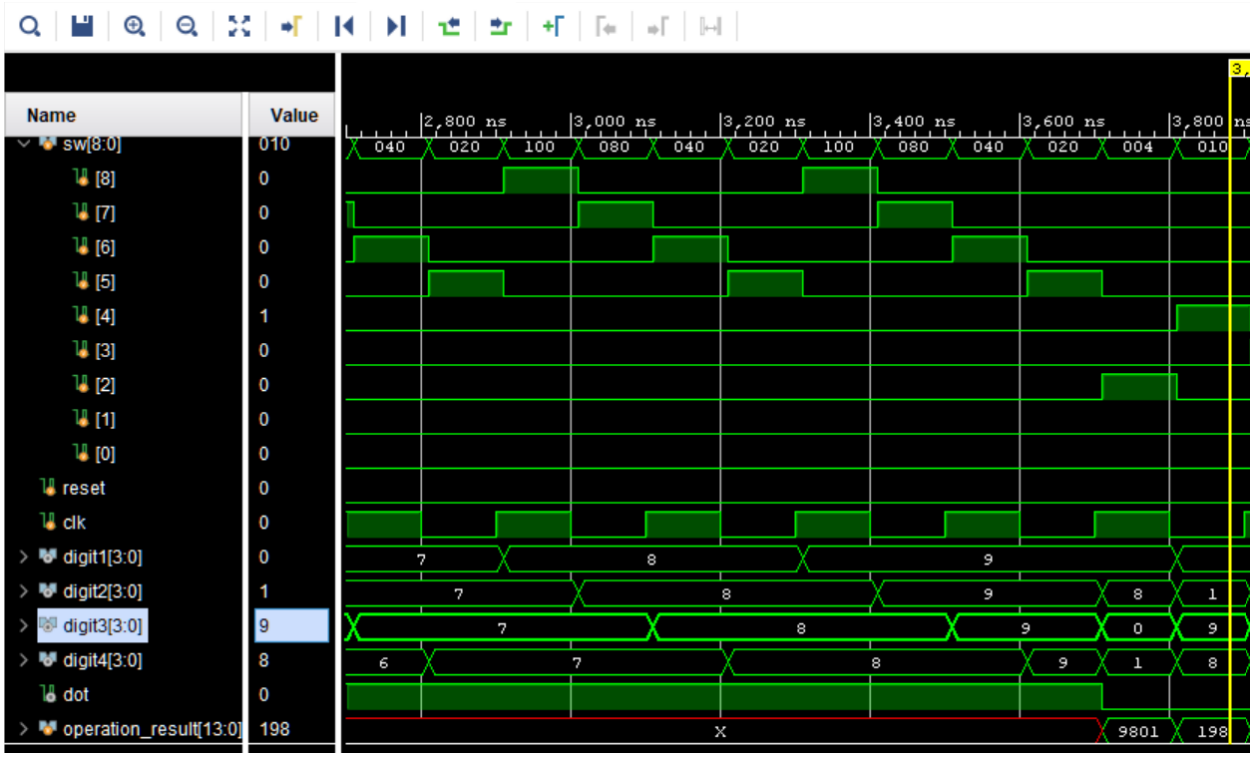
1. **Adjusting the two numbers to 99 and dot = 1 to separate them (99.99)**



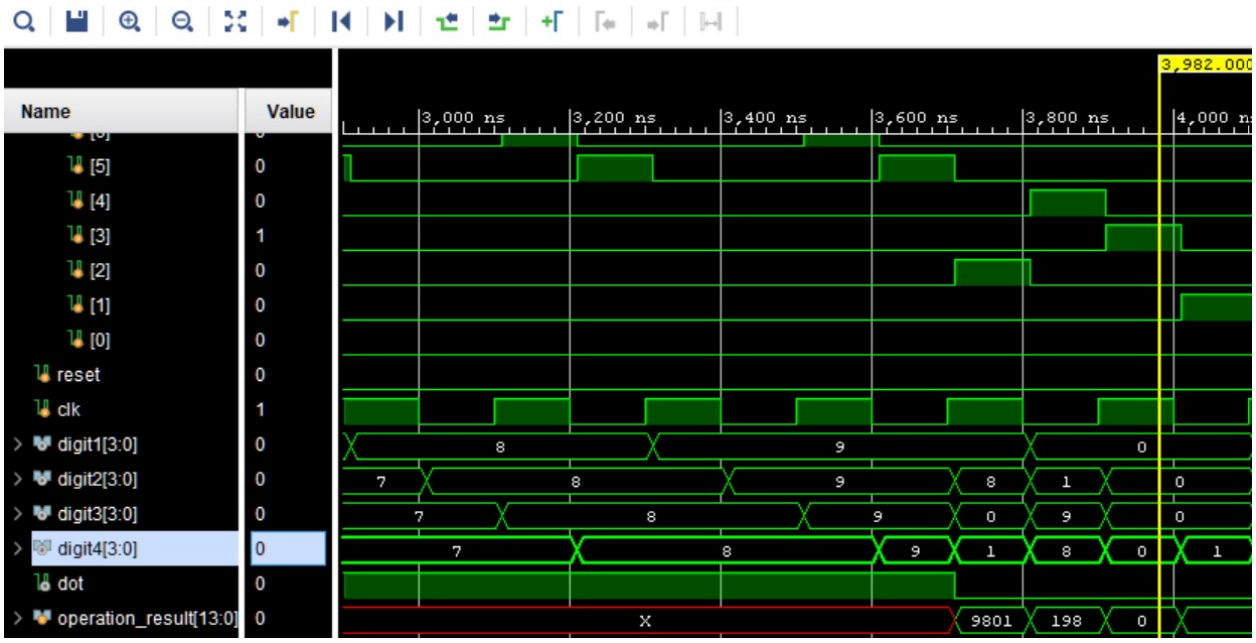
2. **Multiplying the two numbers (99*99 = 9801).**



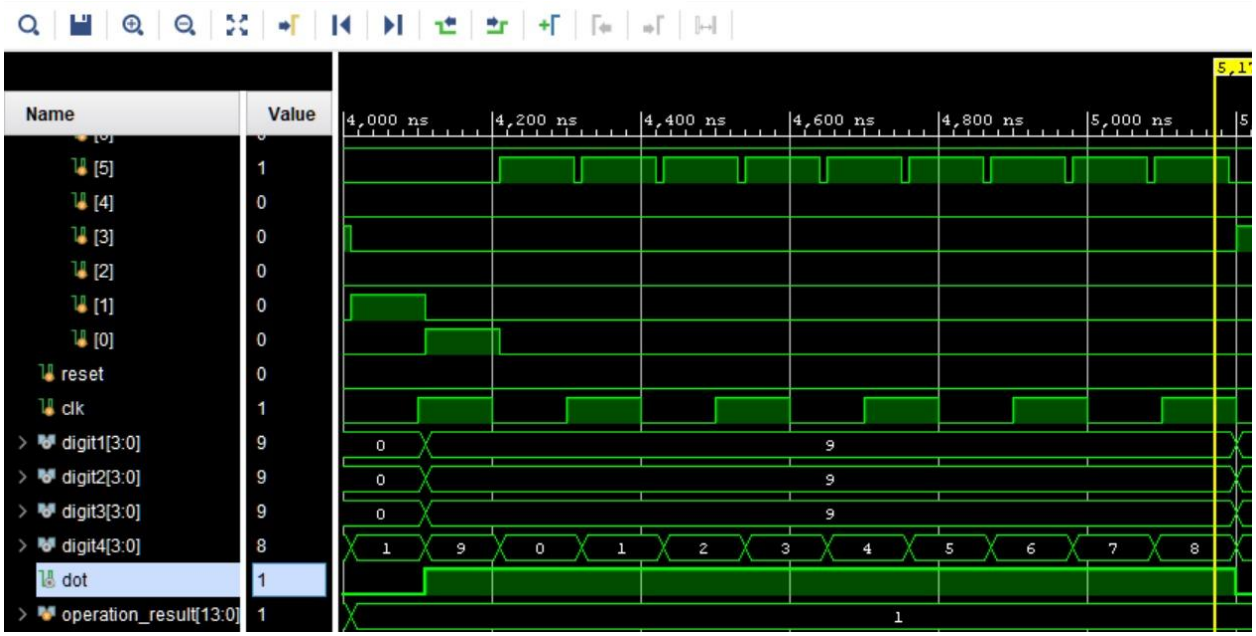
3. **Adding the two numbers (99+99 = 198).**



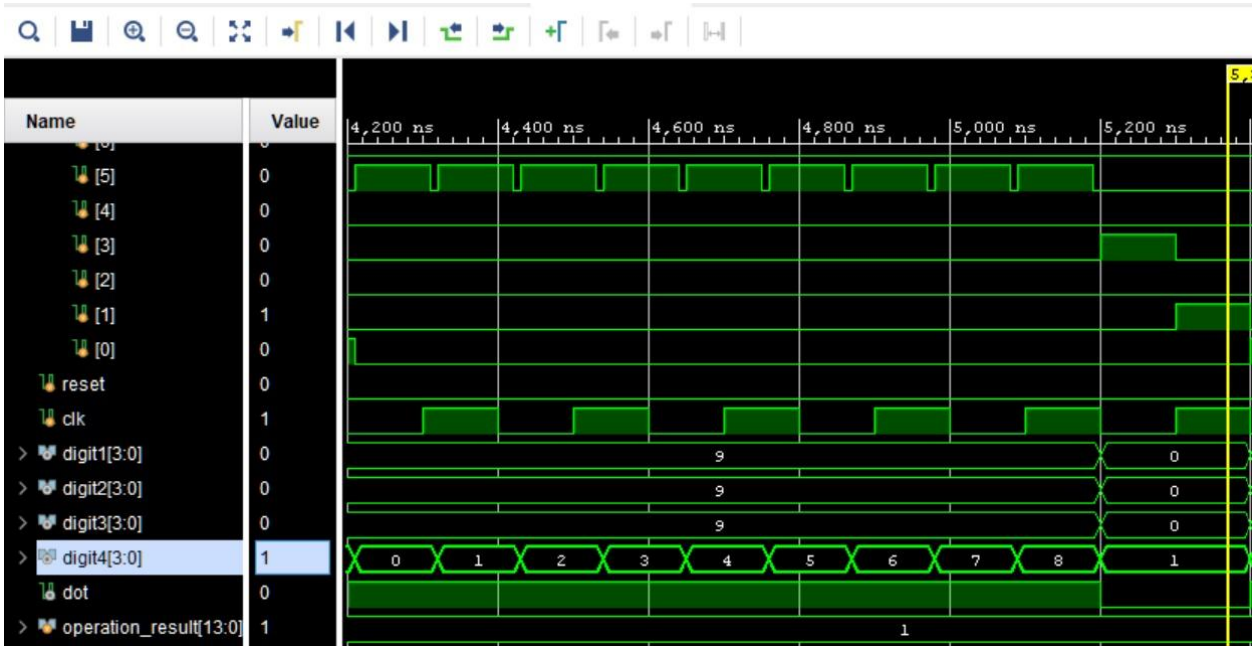
4. **Subtracting the two numbers (99-99 = 0).**



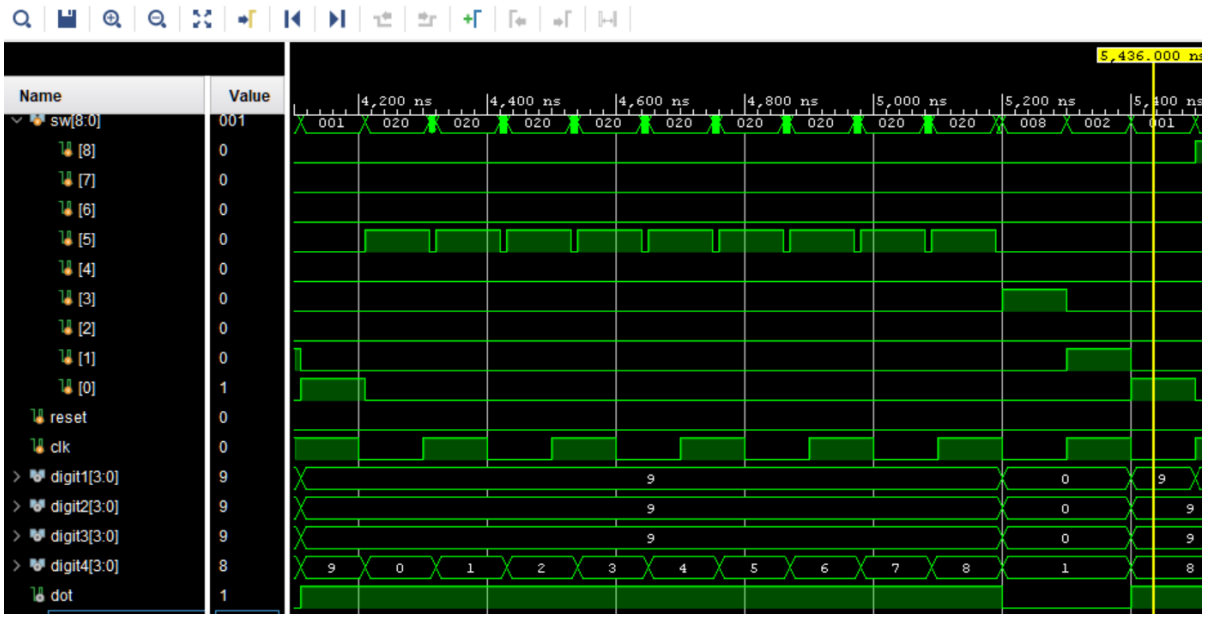
5. **Dividing the two numbers (99/99 = 1) which is shown as 0001 by digits and operation_result = 1. Then display the original numbers again (99.99) and adjust the second number to be 98 and the first number remains the same (99.98).**



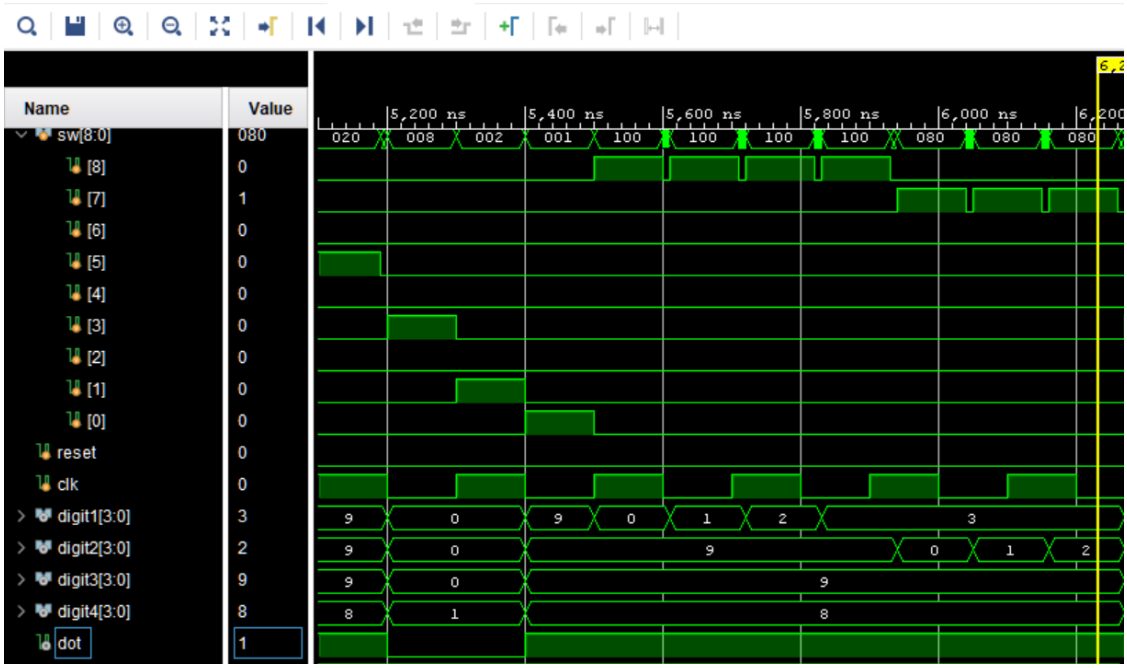
6. **Perform subtraction (99-98 = 1) and division (99/98=1). Both are equal to 1 so digits displayed and operantion_result remains 1 while both operations are performed.**



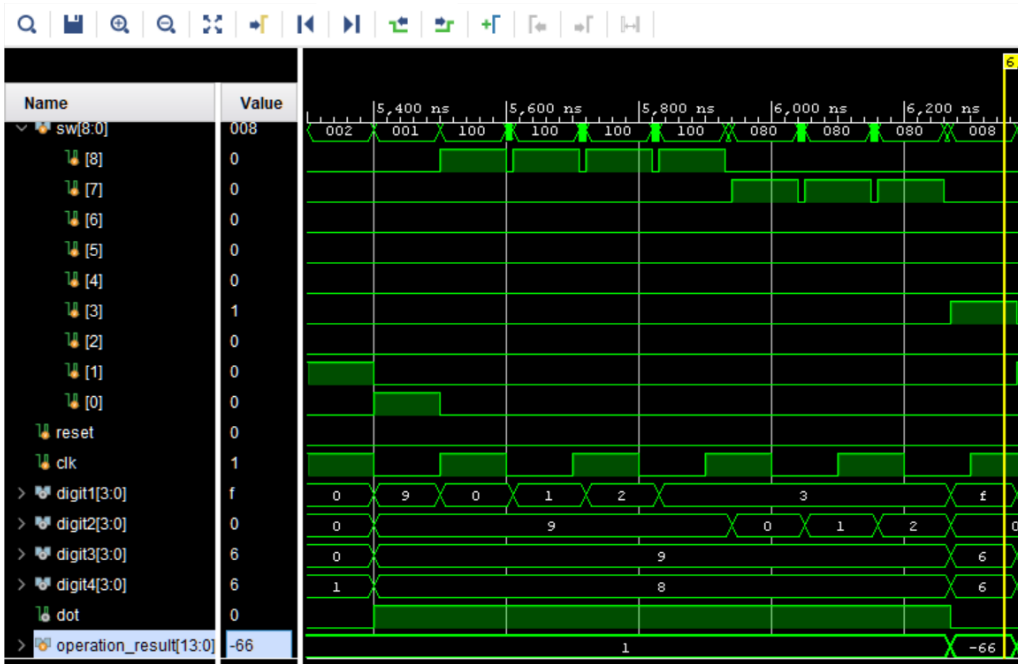
7. **Display the original numbers (99.98) again.**



8. **Adjust the first number to be 32 and the second remains the same (32.98)**



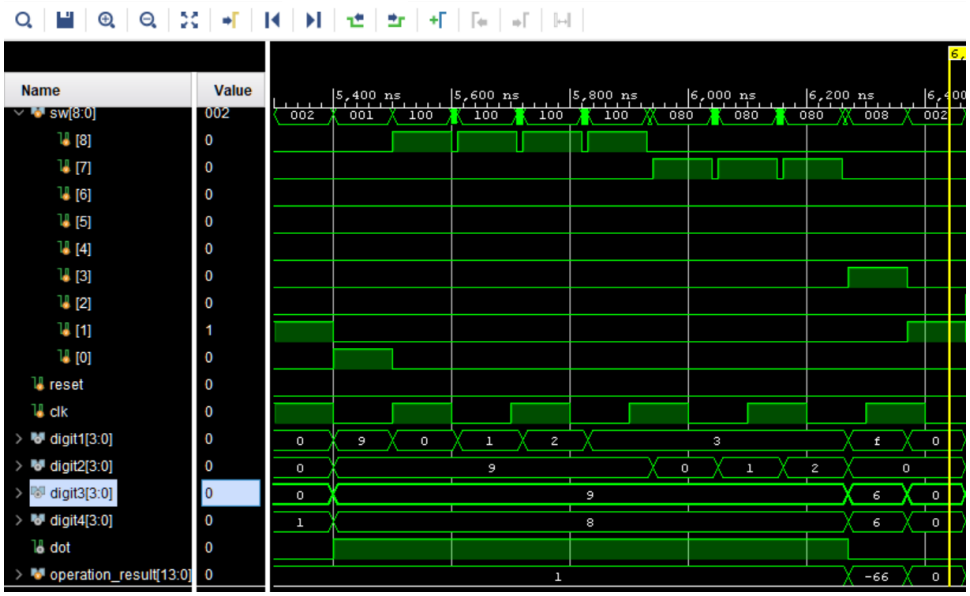
9. Perform subtraction (32-98 = -66)



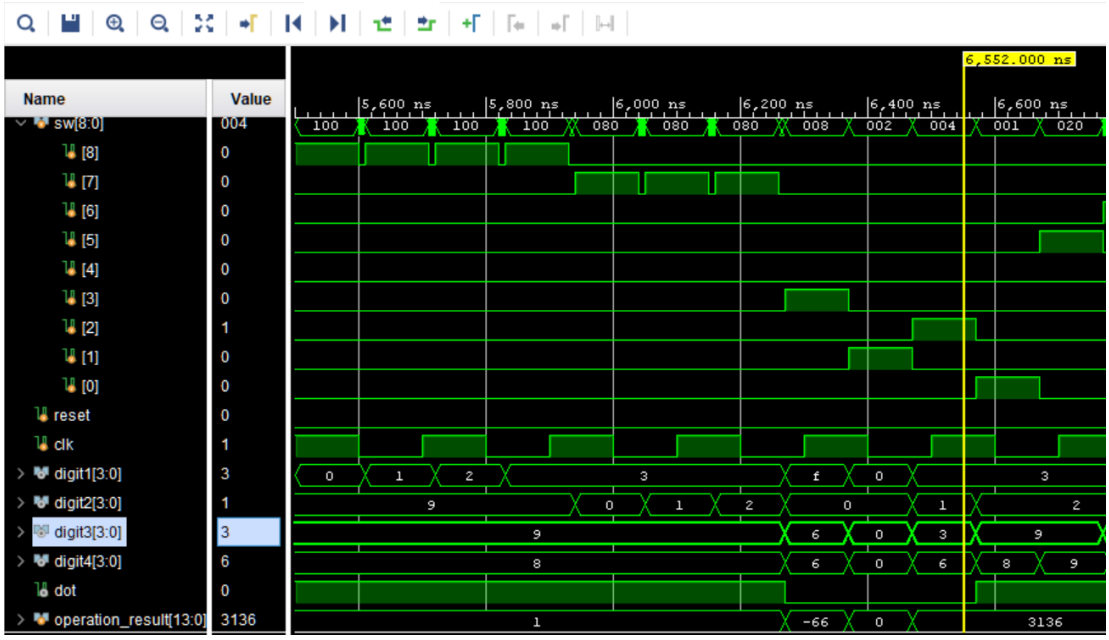
- Note: f in digit1 here represents the value 15. The bcd_to_cathodes (bcd to seven segment display) is adjusted so that if it receives the binary value of 15, then it displays the negative sign on the leftmost LED.

To show operation_result = -66 change radix from unsigned decimal (which shows all other values) to signed decimal.

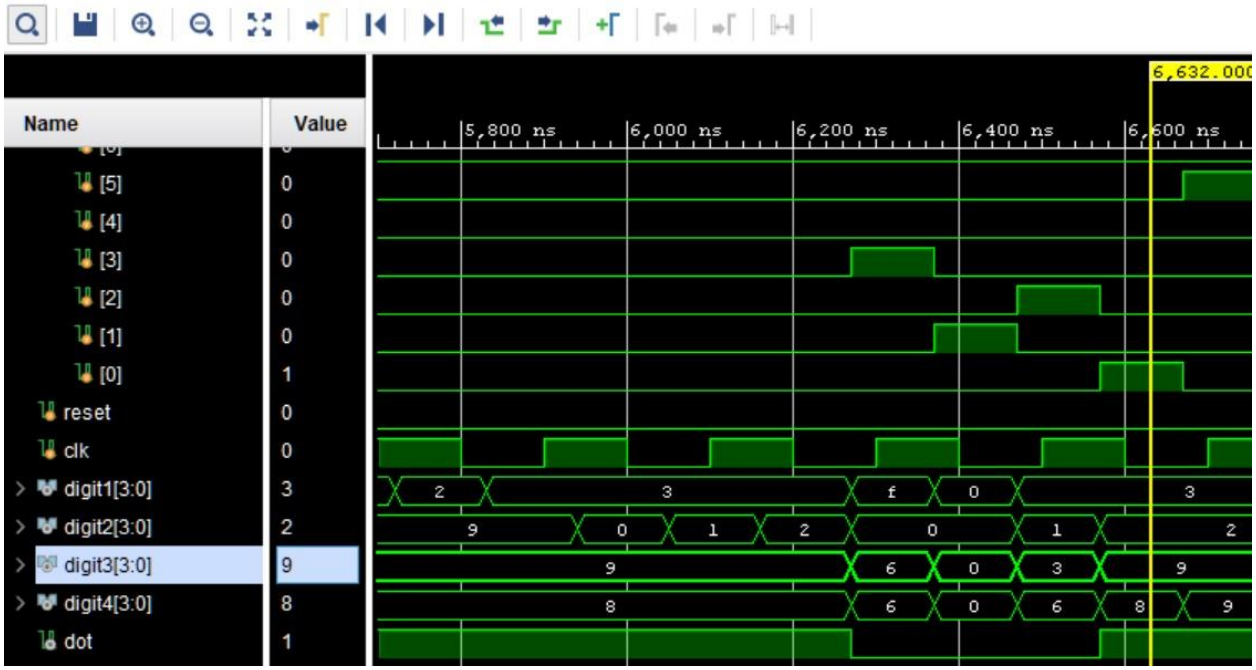
10. Perform division to the nearest integer (32/98 = 0).

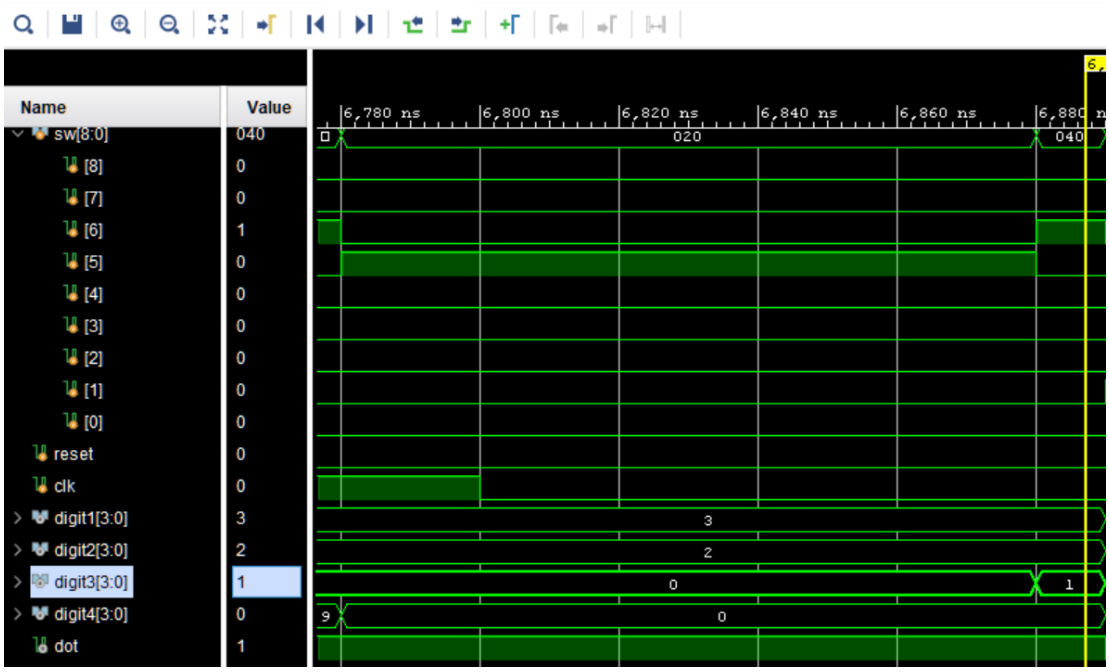


11. Perform multiplication (32 * 98 = 3136).

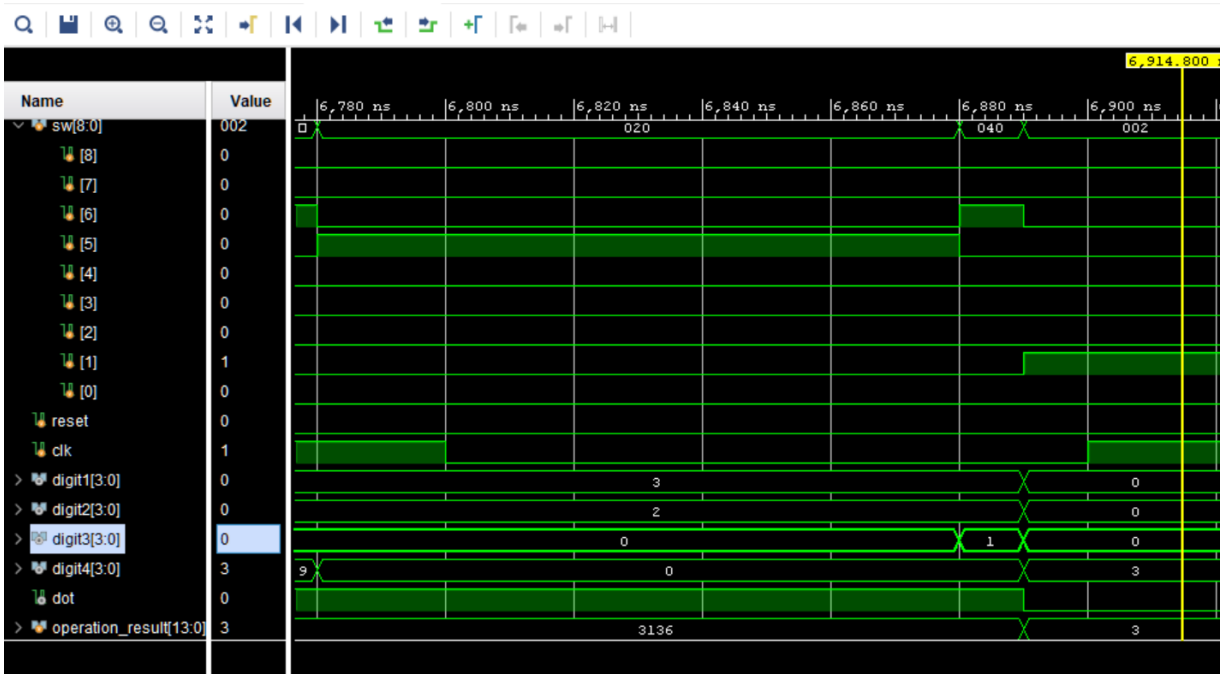


12. Display the original number (32.98) then adjust it to be (31.10)





13. Perform division (32/10 = 3).


Name
Value
sw[8:0]
002
[8]
0
[7]
0
[6]
0
[5]
0
[4]
0
[3]
0
[2]
0
[1]
1
[0]
0
reset
0
clk
1
digit1[3:0]
0
digit2[3:0]
0
digit3[3:0]
0
digit4[3:0]
3
dot
0
operation_result[13:0]
3
6,914.800
6,780 ns
6,800 ns
6,820 ns
6,840 ns
6,860 ns
6,880 ns
6,900 ns
020
040
002
3
0
2
0
0
1
0
9
0
3
3136
3