Digital Calculator

Anas A. Ibrahim 900204611
Shahd Khaled Elmahallawy 900194441
Abdelaziz Yehia Zakareya 900203361

November 30, 2021

The American University in Cairo

CSCE230101 - "Digital Design I"
Fall 2021
Under the Supervision of Dr. Mona Farouq

1 Program Design: Sub-Modules

1.1 Debouncer and Synchronizer

The module is named `debouncer`, which has a Synchronizer embedded in it. The Synchronizer provides enough time for the metastable condition to be resolved. The debouncing circuit generates a single pulse with a period of the slow clock without bouncing as we expected; it resolves the bouncing (rapid change back-and-forth between low and high) of the button after pressing by the user.

1.2 Rising-edge detector

The module is named `risingEdgeDet`. The rising-edge detector is a circuit that generates a short one-clock-cycle tick when the input signal changes from 0 to 1. It is used to indicate the onset of a slow time-varying input signal.

1.3 Counter

This module is named `counter`. It takes two parameters n and k. It increments register `count` of k-bits from 0 to $n-1$ and then back to 0 again (modulo n) according to each positive edge of an input `in`.

1.4 Clock-Divider

The module is named `clockDivider`. The main function of this module is to take the 100 MHz clock of the FPGA and outputs a clock with a frequency of choice according to some parameter n. This works by initializing the output clock `clk_out` to 0 with the reset being set to 1, and then changing the level of the output clock `clk_out` every n clock cycles. That way, $2n$ clock cycles is one clock cycle of the output clock. Since 100 MHz means 10 nanosecond-period, we know that a clock cycle of the output clock cycle takes $20n$ nanoseconds and so to get a frequency of x Hz, we need

$$x = \frac{1}{20n} \cdot 10^9 \text{ Hz} \iff n = \frac{x}{5 \cdot 10^7}$$

1.5 Segment Display

The module is named `SegmentDisplay`. It is used to turn on the LEDs of the seven segment display according to the number we want to display. This module takes an input of 4 bits, `digit` which is a number that decides the digit displayed. So, if `digit` is from 0 to 9, it displays a number from 0 to 9. If `digit` is 10, we display the negative sign. Otherwise we turn off all LEDs. We note that there is an 8th bit for the decimal point, and a decimal point flag which is, if high, turns on the decimal point, and if low turns it off.

1.6 Digit Switching Module

This module is named `display`. This is one of the most important modules. This module deals with making every digit of the 7-Segment Display distinct. Since the 7-Segment display shows one digit on all four digits, we take the clock of the FPGA and convert it using `clockDivider` to a clock of 1000 Hz named `clk_1000Hz`. Accordingly, every clock cycle of `clk_1000Hz` we light up one digit of the 7-Segment display and turn off the rest. This one digit will be selected from four digits (in 7-Segment Display format) that are inputs of the module. Note that the module outputs the selected digit and an enable to deal with the FPGA and turning on one digit and turning off the rest (noting that the FPGA has an active-low enable).

2 Program Design: Main Module Flow

The main module is named `calculator`. It takes those inputs

- `button`: a 4-bit input which is the pressing of the button (high if the button is pressed and low if not). The 0^{th} bit is for the right-most digit (button B4), 1^{st} bit is for the button B3 and so on.
- `operation`: a 4-bit input to represent the switches of operations. The 0^{th} bit is for B5 switch (addition), and the 1^{st} is for the B6 switch (subtraction) and so on.
- `clk`, `rst`: the clock of the FPGA and a reset switch.
- `return`: the B9 switch.

It outputs those outputs:

- `finalToDisplay`: an 8-bit output, this is the digit that is going to be displayed on the 7-Segment Display. Note that this digit changes every 1 ms (clock cycle of the 1000 Hz clock) since we need it to assume one digit of the 4 digits displayed on the output every 1 ms.
- `enable`: a 4-bit output, this is the enable that lights on one digit and turns off the rest according to the clock.

The module supports the 4 operations as required using normal operators like `+, -, *, /` that Verilog supports. There are two caviats to note here in subtraction and division. In subtraction, we take the absolute of the result on have a negative flag assigned to put the negative sign. In division, the division in Verilog only supports floor division. So, we noted that

$$\text{round}(x) = \lfloor x + 0.5 \rfloor \text{ for any real number } x$$

$$\implies \text{round}\left(\frac{a}{b}\right) = \left\lfloor \frac{a}{b} + \frac{1}{2} \right\rfloor = \left\lfloor \frac{2a+b}{2b} \right\rfloor$$

and used the floor division of Verilog for rounding.

2.1 The Program Flow

The module takes the buttons and inputs them into a debouncer, a synchronizer, and a rising Edge detector. We use the wire outputs of the rising Edge detector as a positive edge trigger for a counter that increments 4 wires called `digitsIn` to represent the digits of the operands of the calculator. Each digit is 4 bits. These digits are then passed on to the `SegmentDisplay` module to get the output `segmentsIn` which represents the digits of the operands on the display, noting that the decimal point flag will be on only on the third digit from the right.

The wire `segmentsIn` which is a 4-by-8 array, 4 digits, each digit is 8 bits for the display, will be selected from only one digit that changes according to the clock. This digit is called `selectSegmentInput`. Accordingly, there will be `enableOfInput` to turn on and off the digits. These wires will be dealt with by the `display` module.

The program performs the operation on each change of the input switches and results in 4 4-bit digits which are the output digits `digitsOut`. Note that one digit of `digitsOut` may take 10, which is a negative sign (according to the module `SegmentDisplay`) or 11, which is an off bit. This will be needed in result of subtraction. Each of the 4 `digitsOut` will be passed on the `SegmentDisplay` to get `segmentsOut`. Similar to the input state, there will be only one digit to select from the 4 `segmentsOut` which will be called `selectSegmentOutput` and will have `enableOfOutput`.

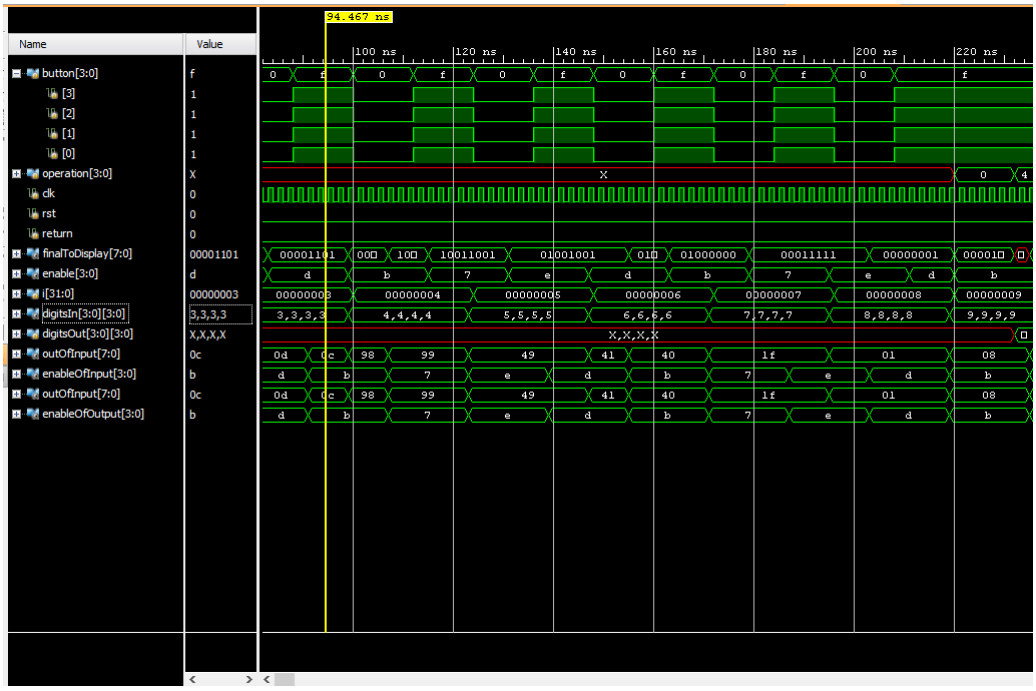
The final output `finalToDisplay` will select from `selectSegmentInput` and `selectSegmentOutput` according to some constraints. If the reset or `return` is on, we select the `selectSegmentInput` to display the operands of input, otherwise we select the output if the operation is an addition, subtraction, multiplication or division. Note that addition is equivalent to the array `operation` being 1, and subtraction 2, multiplication 4, and division 8 (according to the switches).

3 Testing

Every one of the important arrays and bits are displayed in the pictures.

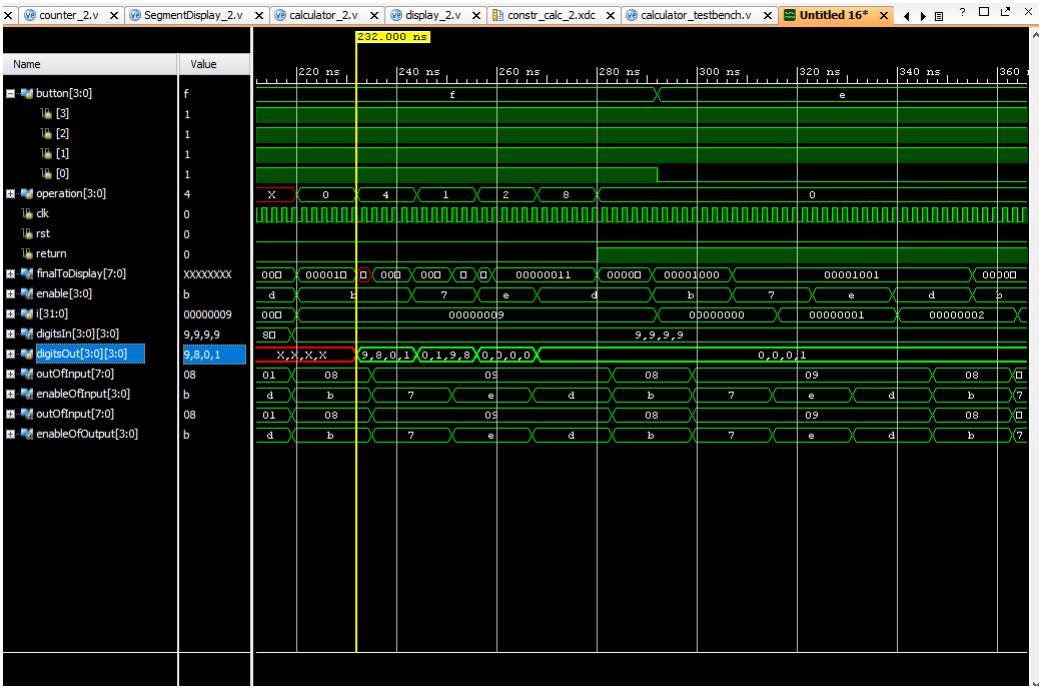
1. Adjusting Numbers to 99

We note that `digitsIn` all increment to 9s simultaneoutly according to the button presses, and `finalToDisplay` changes accordingly, assuming one digit of the 4 every clock cycle.



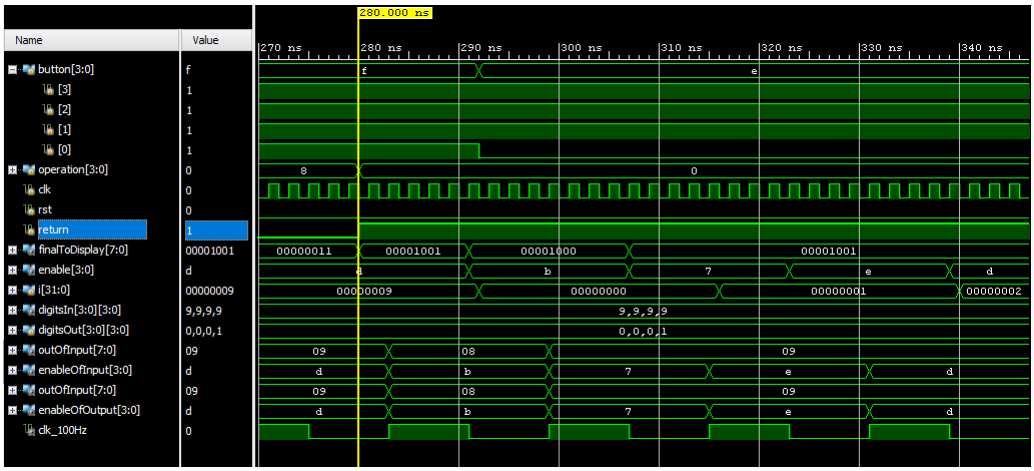
2, 3, 4, 5. Multiplication, Addition, Subtraction, and Division

We note that `digitsOut` are 9801 which is the correct result of multiplication, same for addition, subtraction and division.

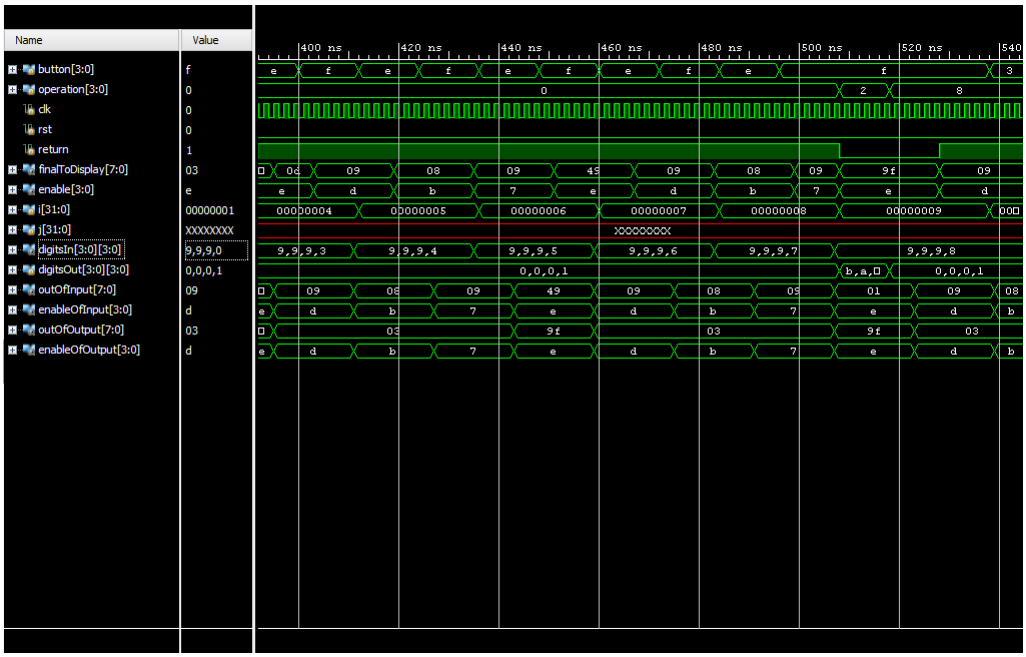


6. Display the original Numbers

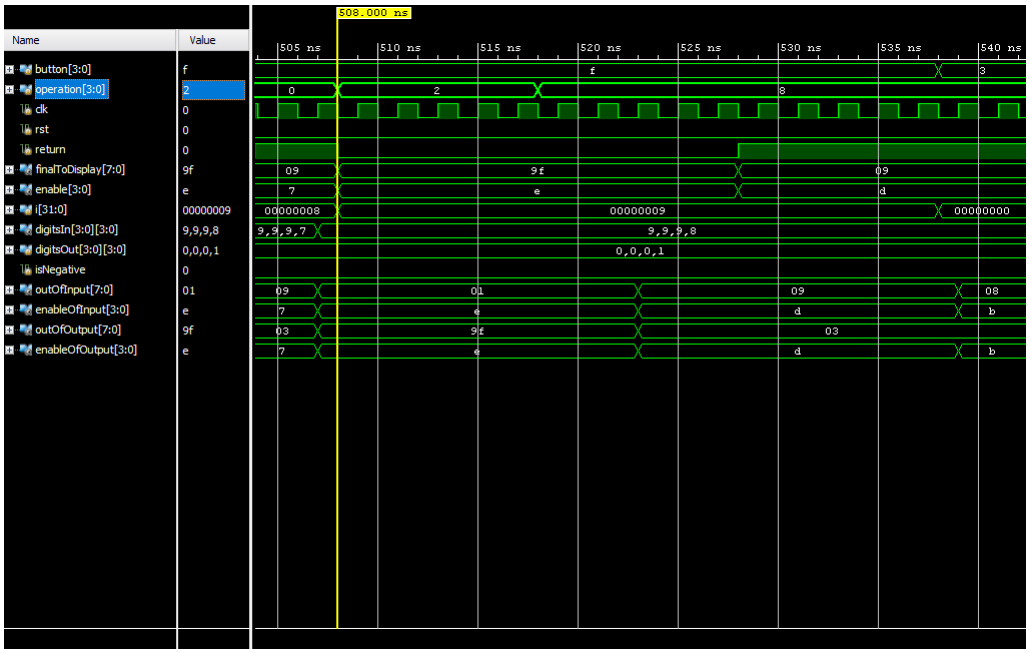
When the `return` is turned into 1, `finalToDisplay` changes immediately to assume a number of the inputs (according to the clock cycles of `clk_100Hz` (we changed its name to `clk_1000Hz` later).



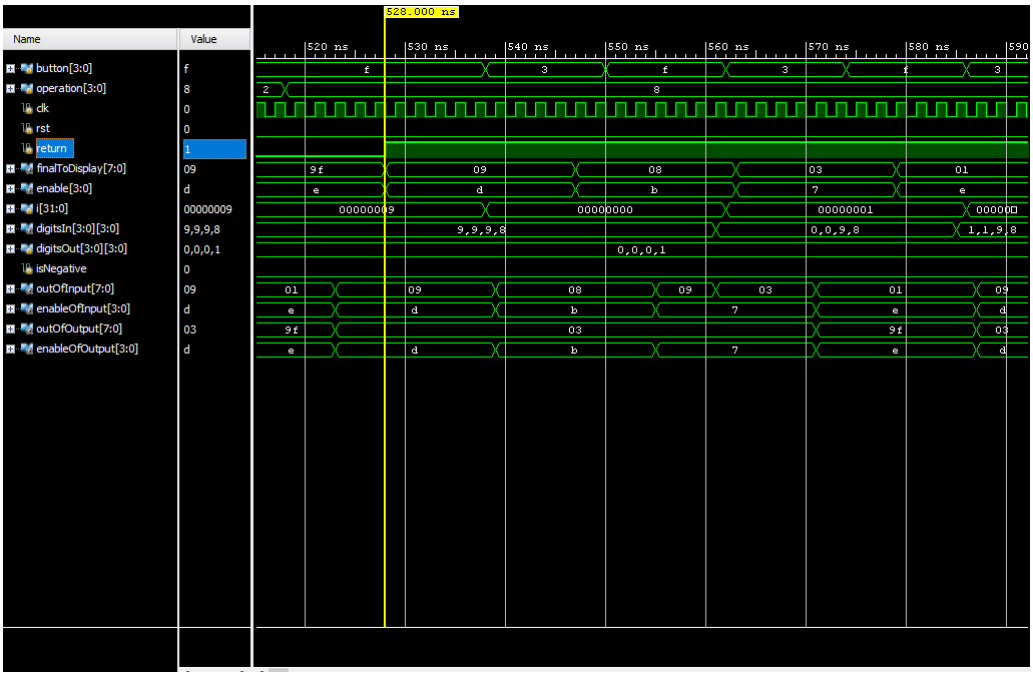
7. Adjusting the second number to 98



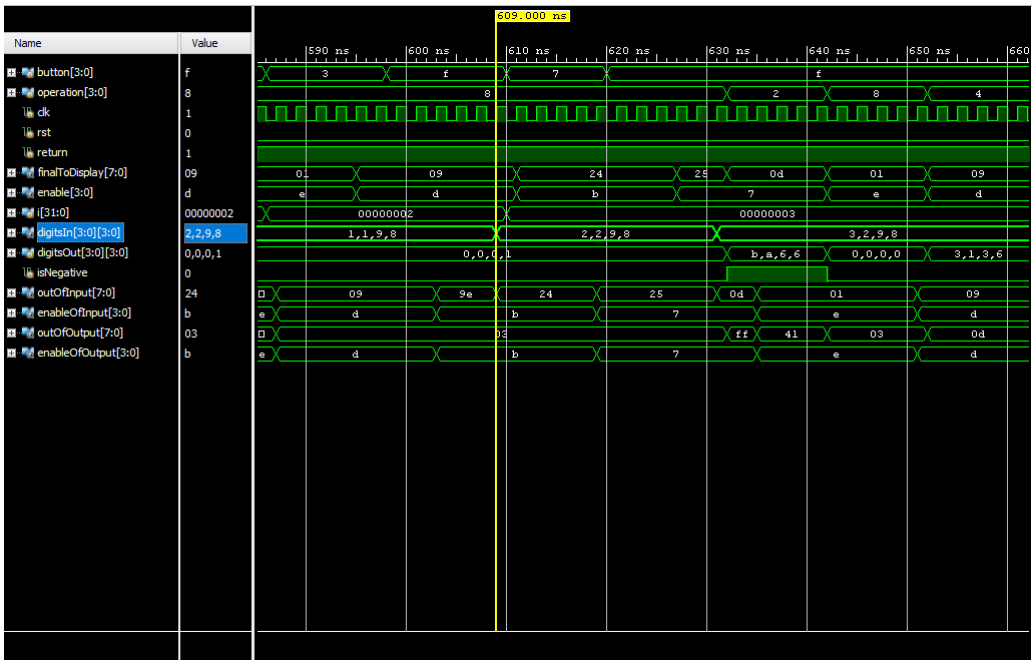
8, 9. Division and Multiplication



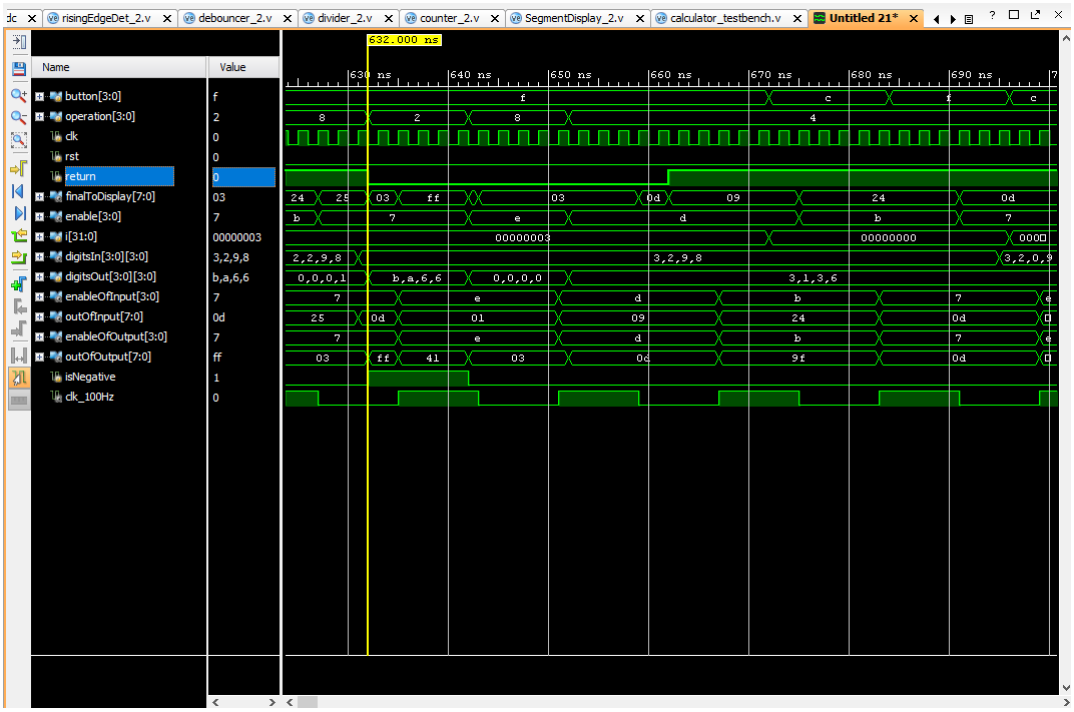
10. Displaying the original numbers



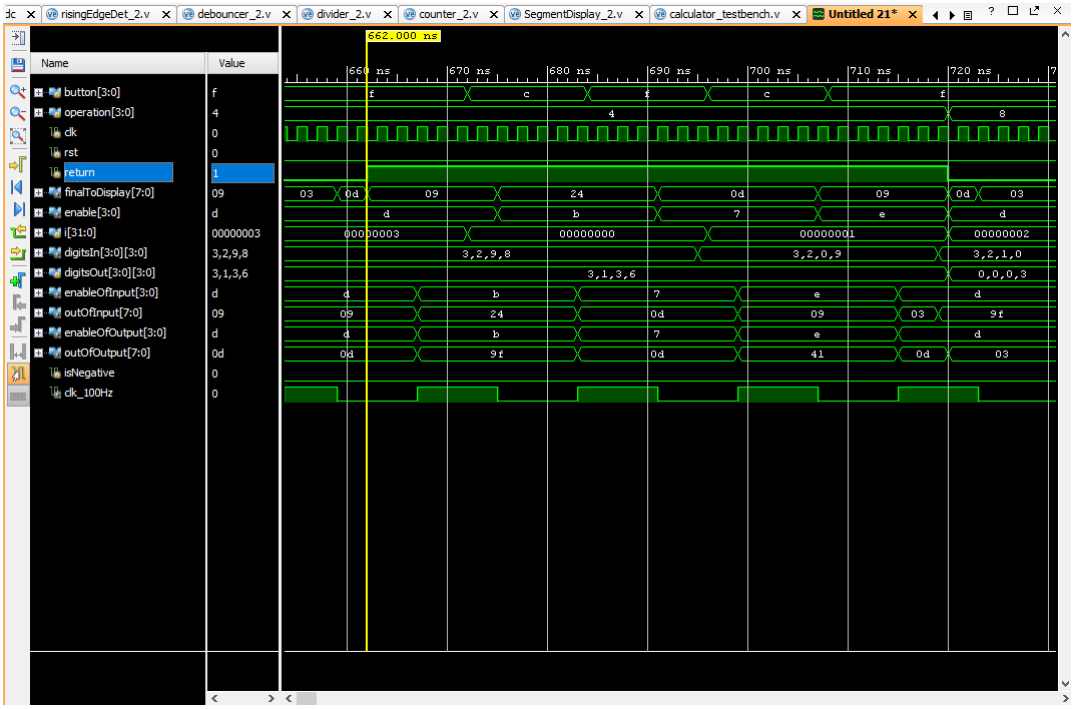
11. Adjusting the 1st number to 32



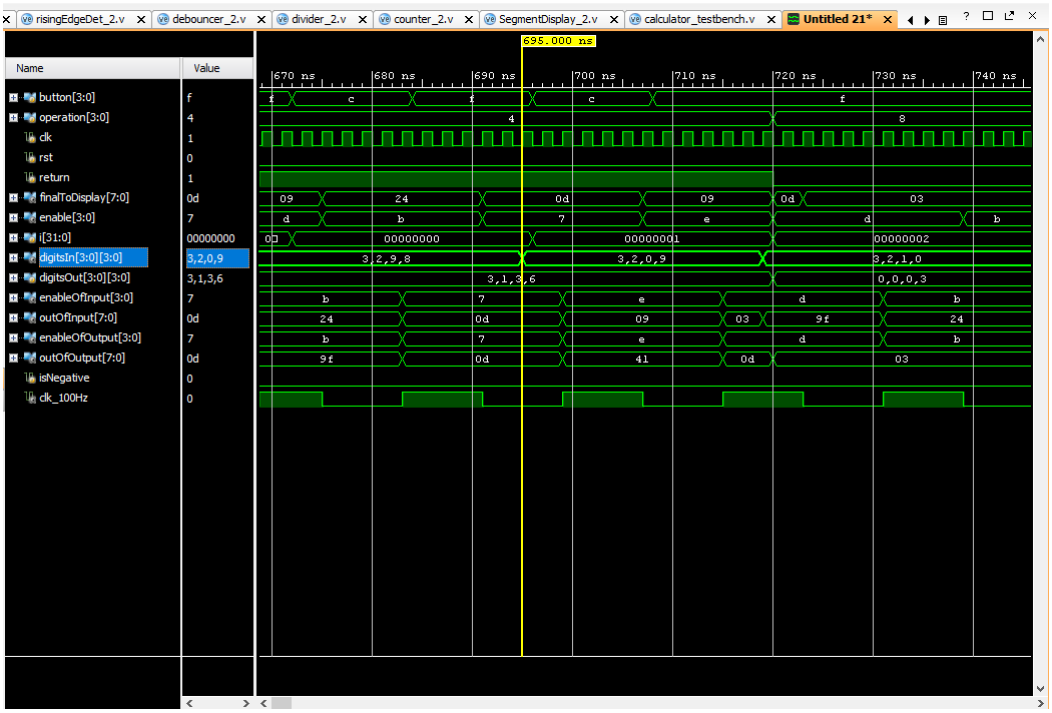
12, 13, 14. Subtraction, Division Multiplication Respectively



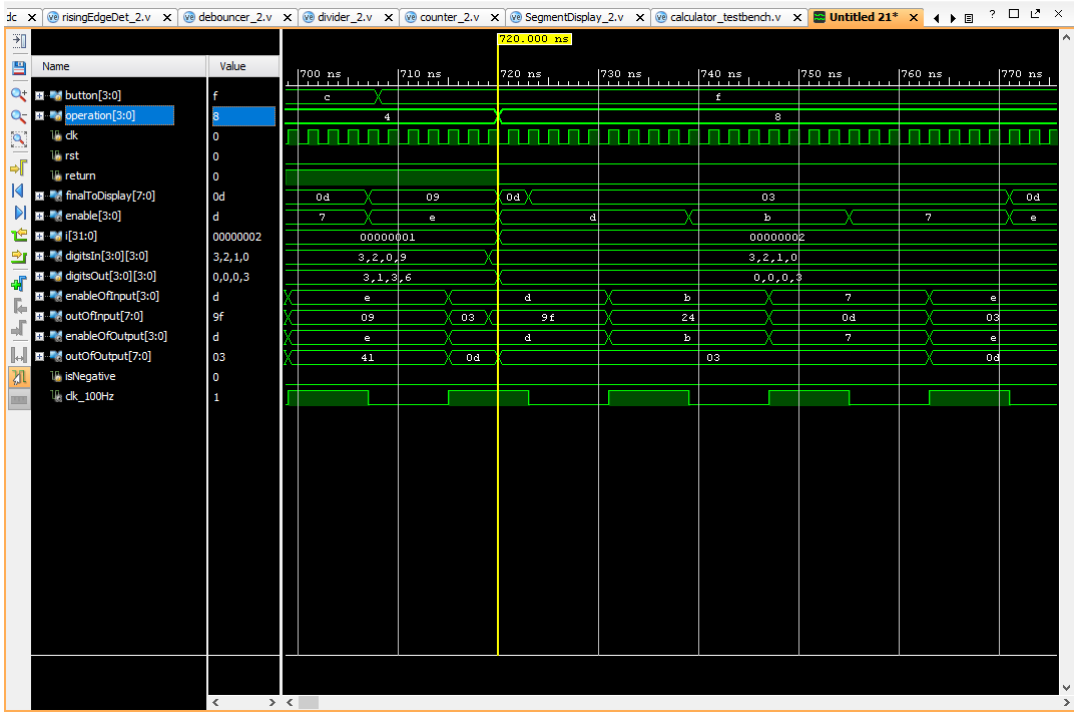
15. Displaying the original numbers



16. Adjust the second number to 10



17. Division


risingEdgeDet_2.v
debouncer_2.v
divider_2.v
counter_2.v
SegmentDisplay_2.v
calculator_testbench.v
Untitled 21*
720.000 ns
Name
Value
700 ns
710 ns
720 ns
730 ns
740 ns
750 ns
760 ns
770 ns
button[3:0]
operation[3:0]
clk
rst
return
finalToDisplay[7:0]
enable[3:0]
i[31:0]
digitsIn[3:0][3:0]
digitsOut[3:0][3:0]
enableOfInput[3:0]
outOfInput[7:0]
enableOfOutput[3:0]
outOfOutput[7:0]
isNegative
clk_100Hz