Pocket Calculator

DSD First Year Project

Student : Pop Joshua Ruben

Project Supervisor:

Mihai Timar

Table of Contents

[1. Specifications 3](#_Toc168518561)

[2.Design 4](#_Toc168518562)

[2.1 Black Box 4](#_Toc168518563)

[2.2 State Machine 4](#_Toc168518564)

[3.Control and Execution units 5](#_Toc168518565)

[4.Detailed Design 8](#_Toc168518566)

[4.1 Simulation sources 9](#_Toc168518567)

[5.Technical Justification for the Design 9](#_Toc168518568)

[6.User manual 9](#_Toc168518569)

[7.References 10](#_Toc168518570)

# Specifications

Create an 8-bit pocket calculator which includes all fundamental mathematical operations, such as Addition, Subtraction, Multiplication and Division.

The input and output must be identical to a normal calculator, meaning, decimal input, simple operation select, decimal output. The result has to be stored and available to use as input for the next operation chosen (ie. 1 + 1 = 2 => 2 + 1 = 3 => 3 + 1 = 4 => ....).

* The initial state is STATE1, first operand input. Decimal input is obtained by using ten switches, each switch position representing it’s value (switch 0 is value 0). After confirming the hundreds, tens and ones, CONFIRM transitions to next state.
* The second state is STATE2, operator input. First switch is addition, second subtraction, third multiplication and fourth is division. CONFIRM transitions to next state.
* The third state is STATE3, second operand input, identical to STATE1, but updates the second operand. CONFIRM transitions to next state.
* The fourth state is STATE4, based on operator input, selects the correct output. No input required. The correct result will be displayed as DECIMAL on the seven segment displays of the FPGA. CONFIRM transitions to next state.
* The fifth state is STATE5, is a signal that operation is done and LOADS first operand with the result. CONFIRM transitions to next state.
* The next state after STATE5 is state 2, and the loop continues.
* Reset reverts the state back to STATE1 after resetting each input and output to 0.

# 2.Design

## 2.1 Black Box A graph with arrows pointing to a rectangle Description automatically generated

Fig 1. Black Box of the system

This first step can be a complicated one, the machine may initially be in a unknown state, with undefined I/O. An Initial Reset is necessary to bring the machine to a known state, state 1.

At this point, the designer must identify the *use cases of the project* (i.e. what actions can be taken), take each case separately and go through it *in the imagination step by step*, preferably with the FPGA board in front of you, to take into account the restrictions of the board (number of inputs and outputs). Thus, if there are hidden, implicit actions, the designer must discover them and describe them as inputs or outputs

## 2.2 State Machine

A diagram of a flowchart

Description automatically generated Fig 2. State Machine of the system

The state machine of this system has 5 major states. State 1,2,3,4 and 5 respectively.

The Reset button, when activated resets the state machine to known state 1, asynchronously, however, for every state, a confirm is needed to transition to next state.

In some major states, such as 1 and 3, there are smaller state machines known as CurrentInputStateA and CurrentInputStateB.

A diagram of a diagram

Description automatically generatedFig 3. Inner State Machine of the system

This state Machine, known as the Inner State machine or InputStateA and InputStateB, is in control of the input of the operands.

Having only ten switches, a state machine is needed to differentiate between the hundreds, tens and ones digit, so as the input for the number “111” is possible (127 is the max), in the hundreds state we select ‘1’, in the tens ‘1’ and in the ones ‘1’. In that sense, this state machine is necessary, it transition of the activation of ConfirmInput

## 2.3 Block Design

The basis for any Digital Design of a machine the BLOCK DIAGRAM, which represents every block and the logic between them necessary to carry out the machine’s tasks.

The Pocket Calculator’s block diagram is shown below:

A diagram of a computer hardware

Description automatically generated

# 3.Control and Execution units

The Control unit in this machine is represent by the Top level STATE MACHINE which is activated by CONFIRM and RESET and the process called „InputPROCESS”, which :

 Captures input values for input1 and input2 in state1 and state3 respectively.

 Captures the operation to be performed in state2.

 Resets inputs and states on Reset signal.

This unit’s role is only to decide how the execution unit can and will be used. A suitable nickname for it would be the „brains” of the machine, while the Execution unit is the workhorse or the „body/muscles”.

The control unit gives comand and the Execution unit’s role is ONLY to execute the commands received. The comands sent are based on user input, mainly CONFIRM, RESET and SWITCHES.



The Execution unit consists of multiple blocks of code, implemented using structural VHDL programing. They are:

1.Divider

This is a binary divider which takes as input 2 8-bit numbers and through a complex algorithm returns as output an 8-bit result and an 8-bit rest

A white rectangle with black lines

Description automatically generated

2. Full Adder / Subtractor

On the left is the blackbox of the binary adder/subtractor. Two 8-bit inputs, one 8-bit sum and a Carry out which is activated when the sum does not fit in 8 bits.

A rectangular black and white rectangle

Description automatically generated

3.NegativeDetector

A diagram of a rectangular object

Description automatically generated

The negativeDetector circuit is what allows this machine to use the adder as a subtractor, when it receives a binary input, it will check whether the MSB is 1 (meaning a negative number) and if it is, it will return the Two’s complement representation of that number.

Two’s complement is a better way to represent negative binary numbers as it is easier to perform operations on it.

4.Multiplier

A black rectangle with black lines

Description automatically generated

This circuit takes two 8-bit binary numbers as input and returns a 14-bit product as output.

The reasoning behind the 14 bit size of the output is that the largest number resulted from the multiplication of largest inputs (01111111(127) x 01111111 = 11111100000001 (16129) ).

5.BinaryToBCD

A diagram of a block diagram

Description automatically generated

This circuit takes as input a 16 bit binary input. The 16 bit input comes from the largest possible output from the operations (14 bits from multiplication) and we pad 2 zeros at the front of the logic vector, for easier conversion.

The circuit has 5 outputs, representing the BCD representation of each decimal number of the final result (ie. 16129 = 0001 0110 0001 0010 1001).

6.ToSSD

A white rectangular object with black lines

Description automatically generated

This Circuit takes as input a 4-bit BCD coded number and outputs the equation for representing that number on a SSD.

The SSD is a series of leds arranged in 8 shape, through strategic activating and deactivating certain segments, we can form a known number (1,2,3,4,5,6,7,8,9,0), these strategic activations and deactivations are given by the equation outputed from this circuit.

7.Debouncer

A black rectangle with black lines

Description automatically generated

This circuit is one of the most crucial circuits for any machines operated on a FPGA board. When a certain button is pressed, due to the FPGA’s hypersensitivity, it will register not one press, but one large press and multiple smaller presses called REBOUNDS.

A screen shot of a graph

Description automatically generated

This image shows how in a 100 us time frame, one button activation is registered as multiple inputs. A non-stable input is not safe for implementation on a FPGA board, it carries the risk of jumping over certain states. A debouncer’s role is to detect Rebounds and not count them as input signals.

8.Clock Divider

A black rectangle with black lines

Description automatically generated

For the AnodeLoops, through testing, it has been proven that the native clock speed of the nexys a7 FPGA is too fast for proper display on the Seven Segment Displays. This circuit’s role is to divide the Clock’s frequency by 1024. )

9. AnodeLoops

A black and white rectangle with a flag

Description automatically generated

This circuit solves a problem that the SSD’s on a FPGA present. The problem is that one can not represent a multiple digit number on a SSD at one time, the FPGA is not built as such.

One number may be represented at ONE TIME, the work around this problem is quickly multiplexing which anode to DEACTIVATE (negative logic) and representing the digit we want on that SSD. All of this goes through each ssd and represents the respective number very fast, when going to the next ssd, it shuts the last one off, but doing so very quickly (clock speed), we have the illusion of a continuous activation of the SSDs.

EXAMPLE

Anodes 11111110 – printing only on first display, chose BCD1 to print

Anodes 01111111 – printing on the last display, chose BCD8 to print

# 4.Detailed Design

This machine has been developed in VHDL, a hardware description language, at the end of development, VHDL will generate the hardware that the user coded and will allow the user to see a detailed design of their machine in the SCHEMATIC WINDOW. At first appearance, what may seem a mix of wires, registers, inputs and outputs, on closer inspection, with a zoom on hand, it is an accurate description of the program

A green and black lines

Description automatically generated

The picture above is the hardware generated by VHDL for 8-bit POCKET CALCULATOR.

## 4.1 Simulation sources

A screenshot of a video game

Description automatically generated

The simulation above shows how the INPUTPROCESS part of the program is simulated with certain stimuli to show the transition in between each state and the correct loading of the signals based on the input switches. In this exact example, we loaded first and second operand with „11” or 0b in hexadecimal, and we chose the operation to be 3, which after decoding is multiplication.

# 5.Technical Justification for the Design

A problem that arrose from the beggining of the design faze of this machine was the INPUT. Having only 16 switches and 4 buttons to work with, the logic behind efficiently loading the operand with numbers, in decimal, was not clear.

The solution, however, was designing the interior state machine used only by inputprocess. By choosing each digit separately, the process became simpler for the user, if input „112” was wanted, the user would have to press 1, confirminput, 1 confirminput again and finally 2, confirminput, thus simplyfing the input process for the user, whom is assumed not to know binary code.

Another issue that an 8-bit pocket calculator might have is when loading the first operand with the result of the last two operands. For example, 127 x 127 (maximum possible value) is 16129, which can not fit in an 8-bit operand, so the choice has been made to discard the most significant bits until the result is 8 bits, and then loading is possible. Result displayed on SSD will still be 14 bits!

# 6.User manual

 Step1 : First, user must chose the first operand, this will work like a regular calculator on a cellphone, first chose the HUNDREDS digit ( every number is represented by a switch located at the position of the number ), confirminput, chose the TENS digit, confirminput and lastly chose the ONES digit and confirminput, when that is done, user must press CONFIRM to go to next state of program

 Step2: Second, the operations encoding is as following, switch 0 for addition, switch 1 for subtraction, switch 2 for multiplication, switch 3 for division, once desired switch is activated, ConfirmInput and then Confirm to go to next state. (0001 Addition, 0010 Subtraction, 0100 Multiplication, 1000 Division).

 Step3 : Identical to Step 1, but this will load the second operand

 Step4 : This step is synonymous to pressing „=”(The confirm after Step3)

Step5: Pressing confirm once again will load the first operand with the result, reset the second operand and user will return back to step2.

!! Pressing RESET at ANY given time will reset each operand and the states back to their initial state!!

.

A close-up of a circuit board

Description automatically generated

# 7.References

 Binary Addition : Conversion, Definitions, Examples

<https://www.splashlearn.com/math-vocabulary/binary-addition>

Binary Multiplication – Rules, Types, Steps, Examples

<https://www.splashlearn.com/math-vocabulary/binary-multiplication#:~:text=Binary%20multiplication%20is%20like%20traditional,products%20to%20get%20the%20result>.

# Binary Division

<https://www.geeksforgeeks.org/binary-division>

# **Nexys A7 Reference Manual**

<https://digilent.com/reference/programmable-logic/nexys-a7/reference-manual>

# **BCD to 7 Segment Decoder VHDL Code**

<https://allaboutfpga.com/bcd-to-7-segment-decoder-vhdl-code/>