8051 Calculator

Designed by

Anthony Dotterer

And

Derwyn Hollobaugh

Fall 2001

Chapter 1

Objective:

A simple calculator takes input from a keypad and displays the answer on a seven-segment LED. To build a simple calculator that adds and subtracts, four stages are needed. The first stage takes the serial input of a number from a keyboard, converts the number into hex, and stores it into memory. The second stage also takes serial input from the keypad only this time it is an operation key. The operation data informs the calculator what operation to be performed on the next number after the equal sign key is pressed. The third stage repeats the first stage by storing a second hex number to memory. Finally, the fourth stage is performed when the equal sign is pressed. This stage performs the operation on the two numbers and outputs the answer to the LED.

Realized:

The 8051 calculator takes a parallel input of a number or operation from the FPGA on the Xilinx board, performs the operation on two numbers and outputs the answer back to the Xilinx board to display the answer on the LED. The Xilinx board obtains numbers or operations from a keyboard. The 8051 calculator uses five stages to complete the process. The first stage makes the 8051 wait till a valid data has been received by the FGPA. Once a valid number is typed, the second stage occurs, which involves the processor saving the number in hex to RAM. After the second stage completes, the processor returns to the first stage and waits for and operation key to be pressed. As soon as an operation key is pressed except for the equal sign key, the third stage is invoked. The processor takes in the hex value from the FGPA and compares it with its instructions to determine the operation code to perform on the next number entered. To make a note, the processor does not store the hex value of the operation. The hex value is only used to make comparisons inside its instruction code. When it matches the hex value, the 8051 will follow the instruction code for that operation. Once again, the processor returns to the first stage. The fourth stage occurs when the next number is pressed. The same procedure for the second stage is repeated again for this stage. The fifth stage occurs once an equal sign has been typed on the keyboard. The answer generated by the processor is displayed on the LED and the processor returns to the beginning of its instruction and waits for stages one through five to be repeated again. The 8051 calculator is able to do addition, subtraction, and multiplication between two numbers.

Chapter 2

Xilinx and 8051 I/O:

CLK: The clock provided by the FPGA, which is set to 10 Mhz.
Kbclk: The clock from the keyboard.
Kbdata: The data line from the keyboard.
nPSEN: The 8051's program-store enable.
PC_D: The data from the parallel cable. PC_D(0) is used to reset the 8051 and Xilinx.
nWR: The 8051's active-low write line that also controls the RAM.
ALE: The 8051's active-high address latch enable.
AH: The high address line used by Xilinx and 8051.
AD: The address data line used to pass data to either 8051, Xilinx, or RAM.
AL: The low address line used by Xilinx and the 8051 to get data off the RAM using the address data line.
nCE: Active-low chip enable is used to enable the RAM.
nRD: The 8051's active-low read.
nOE: The RAM's active-low output enable.
AH_16: The RAM's upper 64 k enable.
RST: The 8051 and Xilinx active-high reset
XTAL1: The 8051 clock that comes from the FPGA.
LED: The seven-segment active-high output to the LED on the XS-40 board.

Transformation of input to output:

The 8051-calculator circuit reads data from the Xilinx board in form of a number, operation, and number in this order and outputs the answer back to the Xilinx board to be displayed on a seven-segment display. Unfortunately, it does not exactly work this way. Almost every clock cycle, the processor reads the data/address line of the FPGA and compares it to see if the data is a valid or not. The FPGA gets its data from an external keyboard that is plugged into the Xilinx board. Once the user has pressed a valid key, the FPGA converts the data into hex. Once in hex, the data is combined with two zeroes in the most two significant bits followed by done signal and kind signal. The hex data consists of the last four least significant bits. After all the bits have combined into one byte, it sends the data to the parallel data line for the processor to read the data and detect it is valid. When the data becomes valid, the microprocessor starts executing its instructions to store only the hex value of the key in one of its registers on the RAM. Then, the microprocessor sends the data back to the FPGA so it can save the number in its register and display the number on its seven-segment display. Once this has completed, the microprocessor jumps to its instruction that waits for a valid operation key from the FPGA. The FPGA waits for the user to press a valid operation key. When the valid operation key occurs, the FPGA stores the data into a byte the same way as the first number and sends the data to the data lines. The microprocessor detects the data and

compares it to find which operation done. Take note, the processor does not store the value of the operation. It is only used to determine which set of instruction codes to follow next. After determining its next course of action, the 8051 sends a zero hex value to the FPGA to be displayed on the seven-segment display to verify the processor is ready for the next number. Once again, the 8051 waits for another valid key to be entered. The same process for the FPGA happens with the second number, but not for the microprocessor. After the microprocessor detects the second number on the data lines, it stores the value of the number into memory, sends the number back to FPGA to be displayed, and then, performs the operation on the two numbers and stores the answer to memory. The 8051 will not send the answer to the FPGA until the user types the key for equal sign. When the processor detects the equal sign key, it sends the answer to the FPGA to be displayed on the seven-segment LED.

Control over data path

Reading Data:

In order to read the data line, the microprocessor sends an active low read signal to the FPGA. The FPGA converts the active low read to active high read inside of it. The active high read triggers a tri-state buffer in the FPGA to send the data out the data lines to the 8051.

Writing Data:

In order to write on data line, the microprocessor sends an active low write signal. The active low signal triggers the RAM to allow the 8051 to write to it while the FPGA converts the active low to an active high internally so it can store the data to its register that outputs to the seven-segment screen.

Latch Enable:

The latch enable signal tells the FPGA what address the 8051 is using on the RAM. The address line is updated every time the microprocessor moves to a new address.

Chapter 3

Decode:

The Decode process is made to look at the device address and tell the device targeted by the micro-controller to send or load. This process in the FPGA will use the READ and WRITE lines as well as the higher bits of the address from the micro-controller. When the micro-controller wants to read from the scan-code register the decode process will look at the device address. Once the device has been determined to be the scan-code register, the process ensures that the micro-controller intends to read. The process then gives the tri-state buffer the command to stop cutting off the output from the scan-code register. Then as soon as the next clock cycle hits, the micro-controller changes its address and the decode process enables the tri-state buffer to allow the micro-controller to use the RAM. The process is similar for writing to the LED register, but there are some differences. The first that instead of reading the decode process ensures that the micro-controller intends to write. The second difference is that the process keeps the tri-state in cut-off. The third difference is that the decode process enables the register to load, so that it can gather the information from the micro-controller.

Micro-Controller Programming:

The micro-controller in our calculator both controls our calculator and the processing of data. The micro-controller is unlike the FPGA; it needs to be program using the hex file from an 8051 complier. The micro-controller on the Xilinx board is essentially an 8031, which is a version of the 8051 only without a ROM. The asm51 complier is located in the BIN folder of XSTOOLS. The assembly code that we made for the micro-controller has four sections to it. The first section is the process of waiting for the first valid input from the keyboard and reading in numbers as well as updating the LED until an operation is encountered. The second section is the process of getting an operation, storing it for the evaluation, and updating the LED to zero. The third section is the same as the first section only that it defines the second operand instead of the first. The finial section of code will calculate the operands according to the operation defined.

Chapter 4

Our Calculator's design has a minor bug and a couple of disadvantages. Our design bug is that at random times the micro-controller locks. This may be linked to some of our designs lengthy timing delays. There is no apparent solution to this problem, only a way to decrease the chance of it happening. The process of decreasing the likely hood of lock ups is done be decreasing the speed of the clock.

Some of our disadvantages are only one number per operand, no stacking of numbers and operations, and no ability to represent numbers over 15. The first and last disadvantages are linked to the fact that the Xess board has only one LED for us to access. The Xtend board's LED's are inaccessible to us because the micro-controller's sharing of the wires to the Xtend LED's. The second disadvantage is that our design does not allow stacking numbers and operations (calculating new operations from old operations).

Even with the disadvantages of the micro-controller, there are more advantages to using this valuable tool. If used properly, the micro-controller can accomplish tasks that would originally be to complex to make in the FPGA. Some designs that can be done with the micro-controller are as follows:

- Music Generator
- Counter
- Run a Stepper Motor
- Virtual Clock
- Reflex Timer
- Serial Communication with PC
- Temperature Sensor

Appendix A

To set the clock frequency

Follow the following instructions:

1. Open GXSSETCLK in windows
2. Provide the necessary information about the XESS board
3. Input the clock divisor that you wish
4. Click SET
5. Follow the instructions from GSSETCLK

To compile your 8051 assembly code:

1. Copy the ASM file from 8051 Assembly Code to a known folder
2. Find the BIN folder in XSTOOLS
3. Copy mod51 into the folder that contains the assembly file
4. At the DOS prompt type “path C:\XSTOOLS\BIN” (change to correct path if necessary)
5. Open the folder containing the assembly file
6. At the DOS prompt type “asm51 WALK.ASM”

After following these steps and getting a successful compile, asm51 will have created your code HEX file.

To create the uCcalculator bit file:

1. Load all of the VHD files in 8051 VHDL code into a folder of your choice
2. Create a new project in Xilinx called CAL
3. Add all of the VHD files to the project
4. Click on the Synthesis button and select uCcalculator at the top-level
5. Once done click on the implementation button
6. Select a Custom ucf file
7. Select the ucf file from 8051 VHDL
8. Click RUN

This will create the BIT file used to create the calculator FPGA

To download the uCcalculator:

1. Copy the BIT file and HEX file to the folder of your choice
2. At the DOS prompt type “path C:\XSTOOLS\BIN” (change to correct path if necessary)
3. Double check that the XESS board has been properly connected and has a keyboard attached to the PS/2 port

4. Once ready type at DOS prompt “XSLOAD WALK.HEX CAL.BIT”
5. Open GXPORT
6. Toggle the last bit to ‘1’
7. Click Strobe
8. Toggle the last bit to ‘0’
9. Click Strobe

These steps will load the uCcalculator into the XESS board.