Project-1

Objective

Design the Finite state machine to simulate a basic calculator in Verilog

Description

In this project, a simple calculator can be divided into a datapath and its control; the control mechanism is usually a finite state machine (FSM). The FSM will control what is being displayed (either the result or the input) and will determine when the operations should take place. The datapath design is provided, so you need implement in Verilog. However, you will have to design the finite state machine (ASM chart) and then simulate all in Verilog.
The block diagram of the system is given in Figure 1.

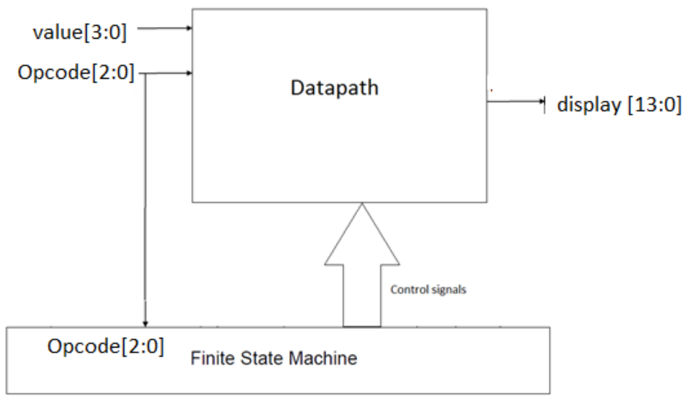


Figure 1: Block Diagram of Calculator

Design Constrains:

Functionality of the system has been sacrificed for simplicity:

- Calculator can only operate on a number from 0-9.
- The datapath can operate up to 14 bits wide (display can show numbers up to 9999)
- On some operations, such as multiply, more bits are required. In this case, we will simply use the 14 least significant bits.
- The circuit operates on positive integer numbers.

Datapath

The datapath diagram is shown in the Figure-2. The heart of the datapath is the Arithmetic Logic Unit (ALU). This performs all the calculations on the data. It stores its results in the accumulation register (AccReg). The contents of this register is used every computation cycle. The other operand comes from another register (InputReg) which stores in the input entered by the user (Value).

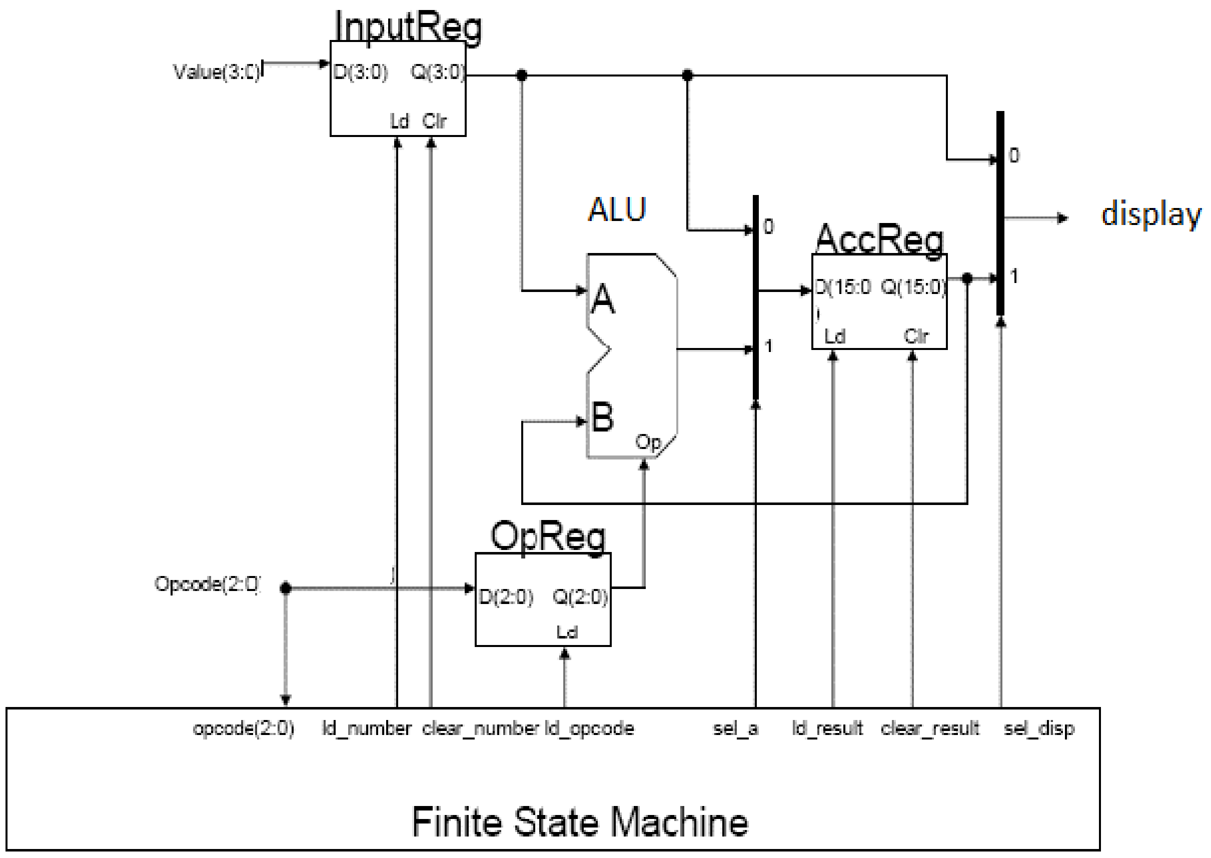


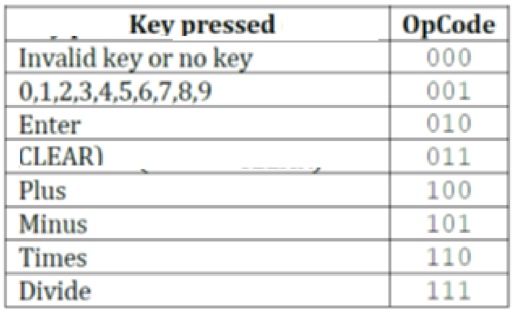
Figure 2: Datapath Diagram

The inputs and outputs of the datapath are shown in the figure-2. The ouput is the display. The internal inputs are from the FSM. The external inputs are two:

a) Value [3:0] -> number entered by the user.

b) Opcode [2:0] -> Operation code which depend of the key entered by the user. For example if the user entered a number then opcode = 001 and the value is in Value [3:0]. If key plus, minus, enter, etc., is pressed then it outputs a specific code depending on the key. The table below list all operation codes.

Table 1: Operations codes



Key pressed	OpCode
Invalid key or no key	000
0,1,2,3,4,5,6,7,8,9	001
Enter	010
CLEAR)	011
Plus	100
Minus	101
Times	110
Divide	111

Finite State Machine (FSM)

The FSM is responsible for two things: displaying the correct value at the appropriate time, and keeping track of what keys have been pressed. If you have used a simple calculator before, then the follow sequence should make sense. If you enter a number, then that number is displayed. If you push an operator key (+, - , * , /) then the accumulated total is displayed. The order of operations is as follows (these are the four states of your FSM):

Table 2: Description of FSM

State	Description
S0	On startup, there is no data in the calculator. This is also, where the calculator will be when the CLEAR button in pressed. In the S0 state, have the output displaying the Input Register (InputReg). It will enter the next state when any number is pressed. It will also load the value of that number into the InputReg
S1	In the S1 state, it should display the number key that was pressed (InputReg). If another number key is pressed, it will replace the current value and stay in the same state. If an operation key is pressed, then it will enter the next state. This is either +,-,*, or /. It will have to transfer the value of InReg into the Accumulation Register (AccReg) to get ready for the next number key pressed. It will also load the operator into the Opcode Register (OpReg) until it knows the next operand.
S2	It should display the AccReg in the S2 state. If a different operation key is pressed, then it should replace the one stored in the Opcode Register. However, if a number key is pressed, then it must load the number into InputReg and enter the last state
S3	In the final state (S3), it should display the InputReg. If a number key is pressed, it should replace the InputReg. If an operation key is pressed, it should load the operation in the OpReg, load the computed value into AccReg, and return to the third state

The inputs and outputs from the FSM are listed below.

Inputs:

a) Opcode[2:0] - see Table 1 for the definitions of each code
b) Clk - clock input

Outputs:

a) ld_opcode – loads the opcode into a register for the ALU
b) ld_number – load the value of the number key into a register
c) ld_result – loads the accumulation register
d) sel_disp – selects what to display on the LEDs
e) sel_a – selects what to store into the accumulation register
f) reset – Clears all registers
g) state[1:0] – outputs the current state

Task to do:

For this assignment, you need to create an ASM chart that performs the above operations, then implement in Verilog and simulate the system. The calculator must perform addition, subtract, multiplication and division. In other words, you need to do the following:

1. Use the operation list in Table 2 to create an ASM chart using the inputs and outputs listed above.
2. Implement the datapath in verilog using the diagram in Figure-2. Creat the modules that you need.
3. Implement a FSM using the ASM chart in Verilog. It should have a total of four states.
4. The divide function is not elementary in Verilog and cannot be synthesized by a simple statement such as *A<=B/C; therefore*, the divide function for this circuit operates as follows: When the divide key is pressed, the number in the AccReg will be divided by 2. You must still press a number key, but it will have no effect on the computation.
5. Synthesize and simulate your code on Vivado. Create a testbench that include the states in the simulation. Show this to professor.

Enter Function

A key not described in the previous section is the ENTER key. When this key is pressed, then the current string of operations is halted and a new set of operations is about to begin. It only works after you press a number key and after a completed operation. Refer to a simple calculator to understand how the key works. On many calculators, it is the equals (=) key.

Include the enter function in the ASM chart and FSM implementation

Project Parts

The project includes three parts:

Part	Description	Due	Percent of grade
1	Finite State Machine section (ASM)	November 8 (class)	10%
2	Simulate the calculator in Verilog (include enter section)	November 15 class	80%
3	Final Report (include all section) and modules in Verilog.	November 15 (midnight)	10%

NOTE: Submit all files (*.v). Include a table with the files and description for each one.

Report

1) FSM design:
    a. List of all ports. Include the port name, direction, vector width, *and* a 1-2 sentence description of each port.
    b. Include an ASM chart of your FSM.
2) Enter Function.
    a. Explain what are the modification realized to FSM
    b. Include the enter function in the ASM chart and FSM implementation
3) For all modules implemented by you, describe the following:
    a. A list of all ports. Include the port name, direction, vector width.
    b. A list of all sub-modules instantiated in the top-level. The name of each sub-module should be a clickable link to a page describing that module.
4) Testing and Troubleshooting: Describe all major problems that you found and how you resolved it.
5) Conclusion:
    a. Analyze your design in terms of how the results met your expectations.
    b. Things you tried which did not work.
    c. Recommendations for future work (What might you do differently next time?)