Department of Computer Engineering
Faculty of Engineering, University of Peradeniya

CO221 : Digital Design
Project Specification for Phase 1

Deadline for Phase 1: 17th March

In this project your task is to build a simple computer system. Computer systems you see today are too complicated and hence this is an extremely simple version which probably existed at the beginning of the computer era. But yet the core concepts still have similarities.

Your main task will be to build the Arithmetic and Logic Unit (ALU) of the system. Each group's implementation will consist of 2 operations. Allocation of operations are given in Table 1 at the end of this document. As the inputs you will use switches. Memory would be just few 3 bit registers. Output would be displayed in binary via LEDs.

You will work in groups of 3 which are the same as your laboratory groups.

The project consists of 3 phases. Outcomes of the 3 phases are given below.

Phase 1
A computer simulation of the system using Proteus or similar software.

Phase 2
Physical implementation of the system on a Printed Circuit Board (PCB).

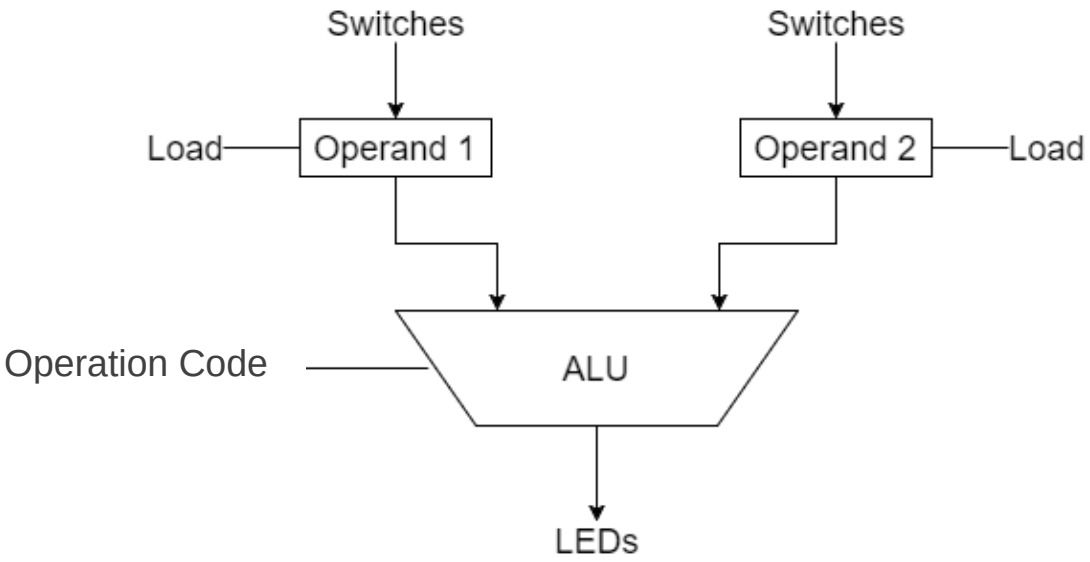
Phase 3
Verilog implementation of the system.

Description of Phase 1 is given below. Details of next two phases will be released later.

Phase 1:

As the first phase, you are required to develop a computer simulation of the ALU (Arithmetic and Logical Unit). It is a core part of the CPU that is responsible for carrying out arithmetic and logical operations. The ALU in your processor is a simple combinational circuit that supports 2 operations.

The block diagram of what you have to implement in Phase 1 is given in the figure below. *Operand1* and *Operand2* are 3-bit registers.



First the user sets the value for the operand on the switches and gives the *Load* signal to load it to the corresponding register. Then the user selects the operation to be performed (out of 2 available operations) through the *Operation Code* or commonly known as *Opcode*.

Some extra knowledge:

The instructions that we give to a computer are converted by the compiler to low-level commands called assembly language commands that a computer can understand. Each command has Opcode and operands, in the format ***Opcode, Operand1, Operand2****. Based on the operation, Operand2 could be optional.*

In **our** computer*, Opcode* is a 4 bit value. Table 2 given at the end of the document gives the values you must use for the Opcode. Select the 2 *Opcodes* which are corresponding to your 2 operations. It also gives assembly language representations of the operations. Note that the operation names in these representations may not be standard assembly language names.

Once the ALU calculates the result, it is displayed using LEDs. Here, *Load* and *Opcode* are called control lines which can be implemented through switches.

Here, note that this is a block diagram and hence a single line or an arrow can represent multiple wires as well.

You are expected to implement these operations using combinatorial logic circuits.

You are allowed to use any 7400 series IC if you can find them but the ICs that we can provide from the lab are given at the end of the document.

The outcome of Phase 1 should be a computer simulation via Proteus or similar software.

Also, you are required to submit the computer simulation (a single zip file) before the deadline to the link in FEeLs.

Notes:

- On anything which is not explicitly stated on the description you can do reasonable assumptions. But make sure you are able to explain those when required. Also if there are any ambiguities you are welcome to use the discussion forum in FEeLS to clarify those.
- So far you haven't covered theory on registers but it is not necessary for you to know the theory to start the project. Registers are memory elements that remember binary values. When a clock pulse (square wave) is given on the clock input of a register,

values on the input pins are stored. Until we give a clock pulse again, this saved data does not change even when the inputs change. You may use 74273 IC which is an 8-bit register. Refer the datasheet on how to use it.

- If you run into any problem somewhere, you can always use the forum or meet the instructors and get the problems solved rather than just giving up.

Table 1: Allocation of operations to groups

Group	Operation 1	Operation 2
A1	Addition	Bitwise AND
A2	Bitwise XOR	Bitwise OR
A3	Multiplication	Bitwise NAND
A4	Shift Left	Bitwise NOR
A5	Shift Right	Bitwise AND
A6	Addition	Bitwise OR
A7	Bitwise XOR	Bitwise NAND
A8	Multiplication	Bitwise NOR
A9	Shift Left	Bitwise AND
A10	Shift Right	Bitwise OR
B1	Addition	Bitwise NAND
B2	Bitwise XOR	Bitwise NOR
B3	Multiplication	Bitwise AND
B4	Shift Left	Bitwise OR
B5	Shift Right	Bitwise NAND
B6	Addition	Bitwise NOR
B7	Bitwise XOR	Bitwise AND
B8	Multiplication	Bitwise OR
B9	Shift Left	Bitwise NAND
B10	Shift Right	Bitwise NOR

Note

- Bitwise XOR operation must be implemented using basic gates (AND, OR and NOT).

- For Multiplication, it is sufficient to display only the 4 least significant bits of the output.
- Shift Left and Shift Right operations of our computer shifts the input binary value to the left or to the right by **two** bits. For example, 001 shifted left by two bits is 100.
  Hint: You can implement a one bit shifter and extend it to a two bit shifter by performing another shift to the first result.

Table 2: Operation codes

Operation	Operation code
Addition: add, operand1, operand2	0000
Bitwise XOR: xor, operand1, operand2	0001
Multiplication: mult, operand1, operand2	0010
Shift Left: shiftl, operand1	0011
Shift Right: shiftr, operand1	0100
Bitwise AND: and, operand1, operand2	0101
Bitwise OR: or, operand1, operand2	0110
Bitwise NAND: nand, operand1, operand2	0111
Bitwise NOR: nor, operand1, operand2	1000

Note

- Based on Table 2, you must implement some combinational logic to select the operation to be implemented using *operation code* input. Please note that this logic should only correspond to the operation selector codes corresponding to your group.

Components that we can provide:

- IC : 7400, 7402, 7404, 7408, 7410, 7411, 7427, 7432, 7474, 7486, 7483, 74273
- DIP switches
- IC bases
- LEDs