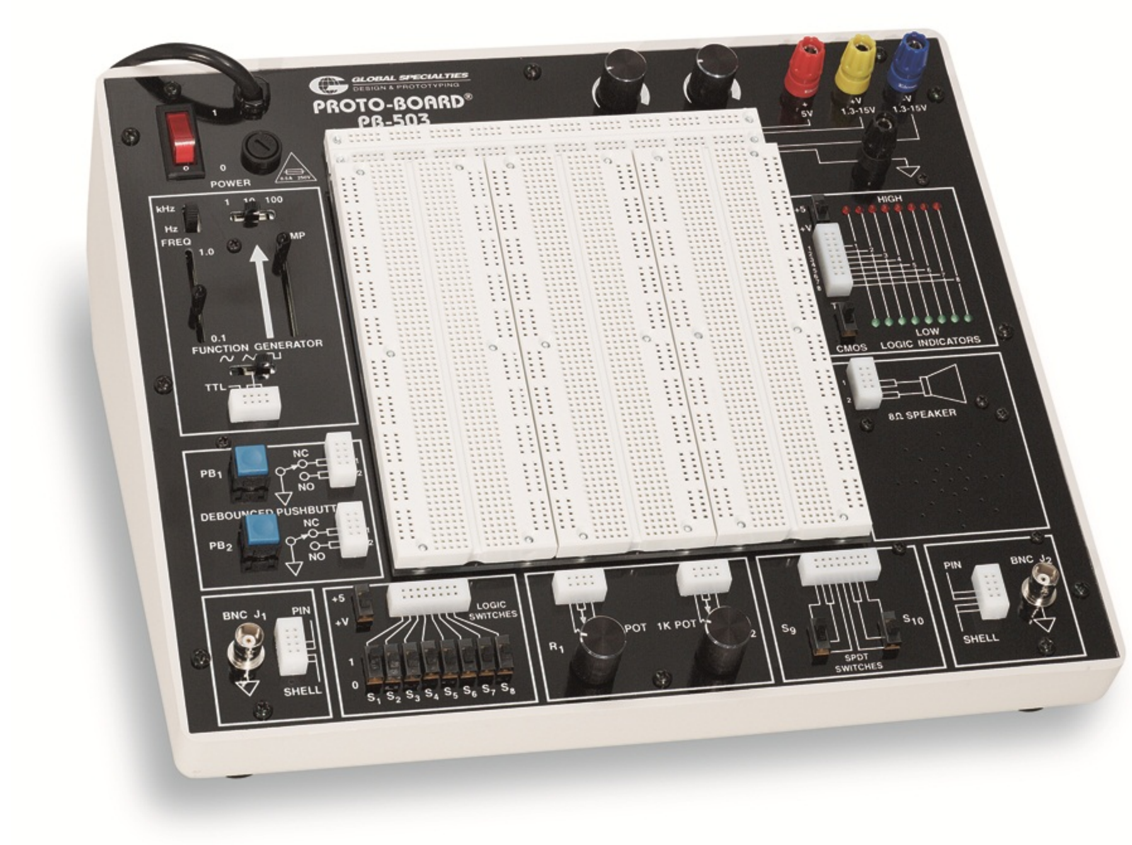
CS M152A DIGITAL DESIGN LAB



Lab 2

4-Bit Serial Adder Implementation

In this lab, you will learn how to utilize more complicated integrated circuits by using them to build a 4-Bit Serial Adder.

CS M152A DIGITAL DESIGN LAB

4-BIT SERIAL ADDER IMPLEMENTATION

Prelab

- Compare a half adder and a full adder in terms of their functionality and architecture.
- Briefly compare a 4-bit serial adder to a 4-bit parallel adder.
- Read through this lab and design your system. Sketch the design with the components you will use. You need not have all the pin numberings or wiring at this point but you should be able to explain what each component is used for.

Introduction

You will build your circuit onto a breadboard that you can continue working on later. Make sure you write your team members' names on a piece of paper that you keep with your breadboard in the lockers.

You are going to design a system to carry out serial addition. Specifically, the system will store two 4-bit binary numbers in two registers (A and B) and perform serial addition on them. Serial arithmetic implies that operations are performed on the operands one bit at a time. Since the input operands are represented using 4 bits, it will take 4 clock cycles to complete each operation.

Operation	Description
LOAD	Load Operands: $A \leftarrow A_{in}$, $B \leftarrow B_{in}$
ADD	Add: $B \leftarrow A + B$

The first step will be to serially LOAD (shift) your operands into the serial-in registers. Since the input is serial (meaning you will need to load one bit at a time), it will take 4 clock cycles to load all 4 bits. Another 4 clock cycles are needed to ADD the operands after they are loaded into the registers. If you need to keep track of the number of cycles that have passed, you can use the lower 3 bits of a 4-bit counter.

After the arithmetic operation, the result should appear in the B register. The values of the registers can be read at any time so you'll need to use 5 LEDs to display the output (4 for the result and 1 for the overflow).

You ***must*** use a one-bit full adder, and not a four-bit parallel adder. You should use the first bit of the 4-bit adder (IC chip: 74LS83). Since it is possible for the operations to produce a result that is larger than 4 bits, you should also output an overflow bit.

Inputs:

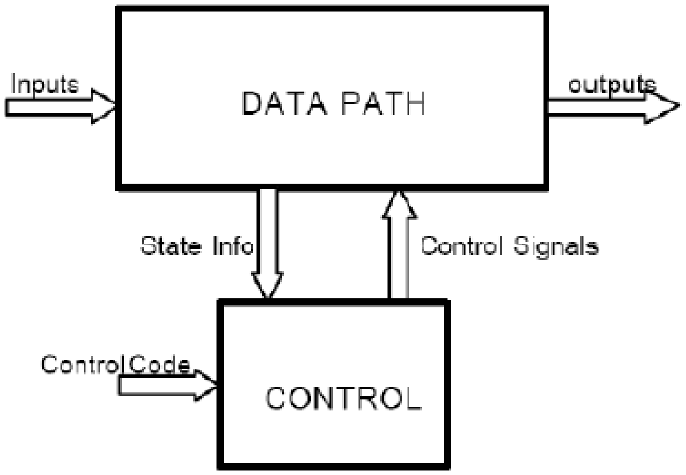
A_{in}, B_{in} : 4-bit Operands. Use 8 switches on PB-503 protoboard.
CLOCK: Clean Clock Signal. Use a push button on PB-503 protoboard.
RESET: Reset button. Use a push button on PB-503 protoboard.

Outputs:

B_{out} : 4-bit Result on 4 LEDs.
OVERFLOW: Bit indicating overflow/carry-out on a LED.

Data Path & Control Design

A good design strategy for many systems similar to this one is to have a data path and a control unit. The data path will perform all of the necessary arithmetic operations while the control unit will tell the data path what to do. The control unit does this based on the current state of the data path in addition to an external control input.

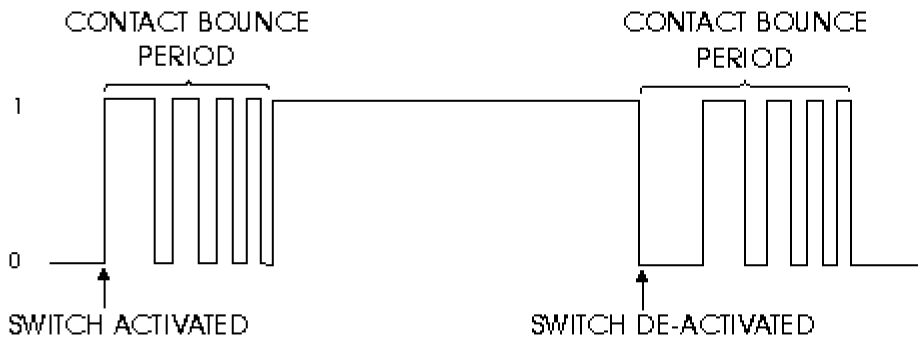


First, design the data path to perform all the required operations and define all the necessary control signals to route the bits appropriately and sequence the operations. Then, design the control unit to decode the control code and generate proper control signals.

The Clean Clock

In this lab, you will test and demonstrate your system with a manually operated clock, you will **not** use the 555 timer. Mechanical push buttons, such as the ones on the PB-503 proto-boards, do not provide a clean switching action. Instead of making or breaking a clean connection, they bounce. Hence, the contact is not a clean ON/OFF connection, but rather a series of brief contact connections that may last several milliseconds. Note that the circuitry in the FPGA or most TTL ICs operate in nanoseconds, hence great attention must be paid to provide switching signals that have clean ON/OFF characteristics. Such a signal is called a "clean clock".

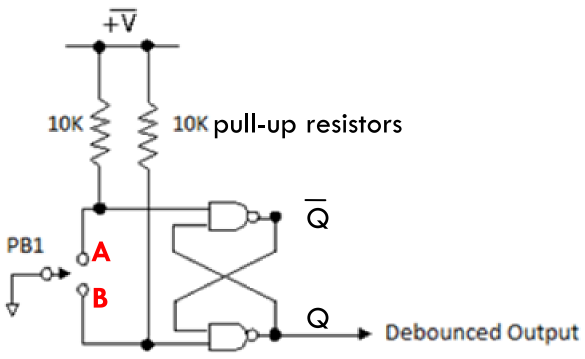
In push button switches, it's the metal contacts that make and break the circuit and carry the current in switches and relays. Because they are metal, contacts have mass. And since at least one of the contacts is on a movable strip of metal, it has springiness. Since contacts are designed to open and close quickly, there is little resistance to their movement. As a result, these contacts will be "bouncy" as they make and break. The image below represents a "bouncy" circuit.



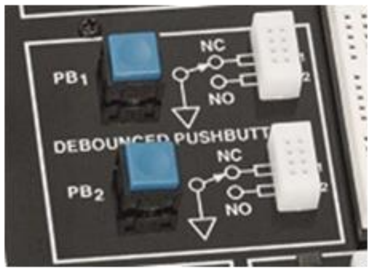
Setting Up the Push Buttons

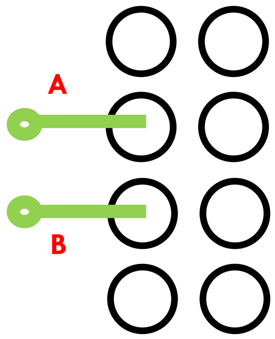
You may have noticed that the pushbutton available in the proto-board is already debounced, which implies that you may not need a latch in order to create a debouncer circuit. However, we will be double debouncing these push buttons so that you can understand why debouncing is necessary. Be sure to implement this feature because your TA will be looking for it.

The diagram below shows how to debounce the push buttons output by using two 10K resistors and a Set-Reset latch. In this diagram, two NAND gates will be used to replicate the Set-Reset latch. You will be able to test the push button using the LEDs on the proto-board. Plus the output of the doubly debounced signal into any LED slot and you should see the "LOW" indicator for that LED light up. Pressing the push button will get the "HIGH" indicator to light up. Use this to debug your circuit until you have correctly double debounced your push button input.



In order to build the debouncer circuit above, you may use the pushbutton on the Protoboard as follows. The two push buttons on the proto-board each have 8 holes in their respective breadboards to make connections. For this lab, we will be working with PB1 though you may use PB2 if you feel inclined to do so. The push buttons use the 8 holes as 4 rows of 2. Each set of 2 holes corresponds to a connection. You may use any one of the lower 4 holes (the NO "normally open" connection).

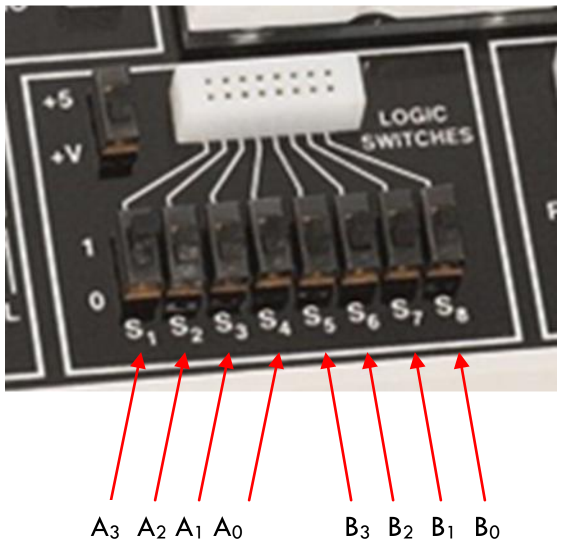




Push Button

Input/Output Setting

You **must** use the 8 switches available on the proto-board in order to provide the 4-bit inputs, A_{in} and B_{in}, for the serial adder. Please refer to the following diagram for detail order of the inputs.



Note that A_0 and B_0 are the least significant bits of A_{in} and B_{in}, respectively. This setting requires you to implement an additional module that feeds an appropriate input bit at different clock cycle (e.g., A_0 and B_0 must be loaded to the shift register at the first clock cycle, then A_1 and B_1 at the next clock cycle and so on.)

For the output, you can use 4 LEDs that are available in the upper right corner of the proto-board. Please use the 4 right-most LEDs to represent the 4-bits of the output. You may use the remaining LEDs for your debugging purposes. To represent the carry-out bit, you can use an extra LED from the component drawers in the lab. Note that component LED's are diodes and have polarity, i.e. they must be wired in the correct direction to function.

Lab Work

Implement your design using the chips on the breadboard. Wiring is an accurate task, so make sure that you plan where you are going to put the ICs. **Check your circuit design as you go. DO NOT wait until you have wired all the components to start debugging.** It is almost impossible at that stage. You should test the functionality of each part individually before connecting it into your main design: shift register, adder, etc.

Notes:

1. Implementation of the automatic control path is mandatory in this lab. This means that you must build a control module which changes the control automatically from LOAD mode to ADD mode.
2. You can only use 1 bit of a single 74LS83 Adder.
3. The maximum number of shift register you can use is limited to 2. This implies that you must recycle one of the two shift registers to store the output value.

Part Numbers (Datasheets Available On Courseweb):

- 74LS83 – 4-Bit Full Adder (You are only allowed to use 1 bit)
- 74LS393 – Dual 4-Bit Binary Counter
- 74LS157 – 4-Bit 2-to-1 Multiplexor (Note that there is only one control for all 4 bits)
- Various Gates (NOT, AND, OR, NAND, NOR, XOR, etc…)
- 74LS194 – Universal Shirt Register
- 74LS74 – D Flip-flop

Demonstration

1. Reset your circuit by pressing the RESET button. This button need not be debounced (but you may need a 10k pull-up resistor connected to the Normally Open pin).
2. Load two 4-bit inputs using switches. The test inputs will be given by the TA's.
3. Clock the circuit 8 times (4 times to load, 4 times to add).
4. Monitor the result of the addition in B_{out} by looking at the LEDs.
5. After you are done, make sure you take your circuit apart and return everything to its original place. Do not leave anything in the lockers. Failure to do so will result in a 10 point penalty.

Report

Please write a report describing your design of 4-bit serial adder. You must have an overall description of your design as well as descriptions for each sub-component of your design. Please include diagrams to support your explanation. Be sure to describe any issues and how you solved them. The maximum length of the report is 2 pages, not including diagrams.