Angelica Semenec

CMPE 127 Lab

Lab 1: Shared Bus Architecture

Abstract

This purpose of this lab is to implement both the hardware and the software of an address latch to control the output of the shared address/data bus. The software is written in the c++ language and flashed onto the SJTwo board, which controls the GPIO pins. The address latch is implemented via PCB using TTL gates.

Functionality

This circuit controls the flow of signals on the address/data bus. Because the bus is shared between both the address and the data, the address needs to be latched and separated from the data before the data is transmitted via the same bus.

The SJTwo board is connected to the PCB via a ribbon cable. 8 of the GPIO pins are used to transmit each bit of the 8-bit address. These are passed through a buffer (SN74LS641) and into the PCB circuit. 4 of the GPIO pins from the SJTwo board are used as control signals (write, address latch enable, memory/ input-output, and interrupt). The write signal controls the flow of the buffer and when it is set to high, the bus will transmit signals from the SJTwo board to the PCB. Write and ALE are used as the latch enable input (LE) into the latch (SN74HC373) that controls the flow of the address. The output enable (OE') is connected directly to ground for this lab because the address will need to be read from the output pins of the '373.

Design

'373 Voltage Table

D_n	LE	OE'	O_n
H	H	L	H
L	H	L	L
X	L	L	Q_0
X	X	H	Z*

H = High voltage, L = Low voltage, X = immaterial, Z = High impedence*

'641 Voltage Table

Control Inputs		Operation
G'	DIR	'641
L	L	B data to A bus
L	H	A data to B bus
H	X	Isolation

H = High voltage, L = Low voltage, X = immaterial

‘08 Voltage Table

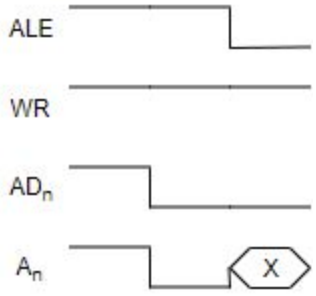
Inputs		Output
A	B	Y
L	X	L
X	L	L
H	H	H

H = High voltage, L = Low voltage

Lab1 Circuit Voltage Table

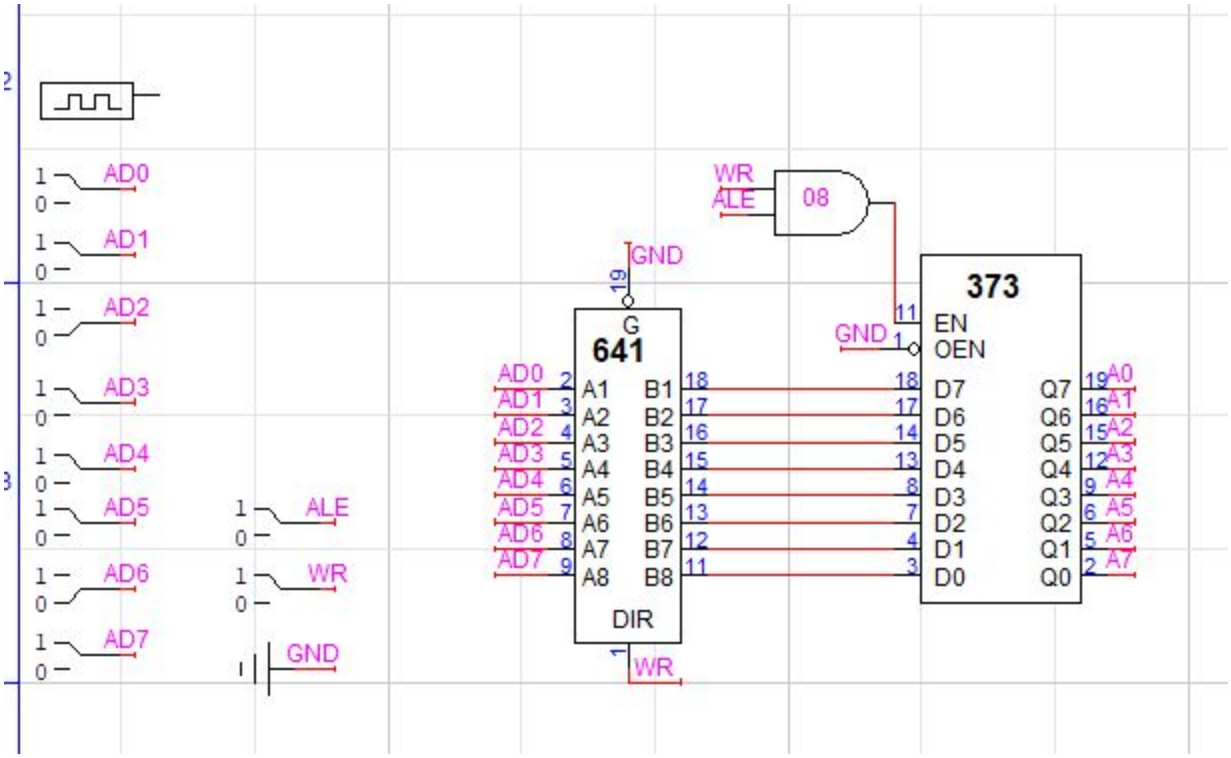
D_n	ALE	WR	O_n
L	H	H	L
H	H	H	H
X	L	X	Q_0
X	X	L	Q_0

H = High voltage, L = Low voltage, X = Immaterial



Lab 1 Timing Diagram

Schematics



Lab 1 Circuit Schematics

Parts List

1. SN74LS641: Octal Bus Transceiver
2. SN74HC373: Octal D-Type Transparent Latch
3. SN74LS08: Quadruple 2-Input Positive-AND Gates
4. 20-pin wire-wrapping headers
5. 14-pin wire-wrapping headers
6. 30 gauge wire
7. 40-pin ribbon cable
8. 1 Kohm resistors
9. SJTwo board
10. PCB

Algorithms

‘641:

A_{data} to B_{bus} = DIR * G’

‘373:

$O_n = D_n * LE * OE’$

LE input (‘373) from ‘08:

LE = WR * ALE

Demo

To demo the circuit, the program must be flashed onto the SJTwo board. To build the program, the command “make build” must be run in the WSL (Window’s Subsystem for Linux). This will create the .hex file that can be flashed onto the board. The hyperload tool is opened with the .hex file that contains the code for the project. A reset must be sent to the device so that this new program can be flashed. The hyperload tool must then be closed and the Hercules tool opened. The USB port connected to the SJTwo must be opened. The usb port on the PCB must also be connected to supply power to the devices. When the device is successfully opened in Hercules, the red button on the SJTwo board should be pressed to send a reset signal to the board so that it can run the program.

Display text will appear prompting the user to input an address as a decimal value. After the user presses “Enter”, the address will be passed to the appropriate GPIO pins as a binary value. This will be passed to the address latch and output.

```
Serial port COM4 closed
Serial port COM4 opened
{FF}Hyperload Version (1.1)
Application Reset ISR value = 000125D5
Booting Application...
Please enter the address or type '-1' to quit: 175
Please enter the address or type '-1' to quit:
```

Hercules Input Prompt

The outputs of the ‘373 should be connected to a logic analyzer where channel 0 corresponds to bit 0, channel 1 corresponds to bit 1 and so on. Using the logic analyzer software, the user can display the output waveforms in order of the binary value where a high line will correspond to a binary 1 and a low line will correspond to a binary 0. This value should match the binary value of the input address.

Logic Analyzer Output corresponding to input decimal number 175

Source Code

main()

```
#include <project_config.hpp>

#include <cstdint>
#include <iterator>

#include <math.h>

#include "L2_HAL/displays/led/onboard_led.hpp"
#include "utility/log.hpp"
#include "utility/time.hpp"
int main(void)
{
     Gpio ad[] = {
       Gpio(2, 2),
       Gpio(2, 5),
```

```
      Gpio(2, 7),
      Gpio(2, 9),
      Gpio(0, 15),
      Gpio(0, 18),
      Gpio(0, 1),
      Gpio(0, 10)
  };

  Gpio Write = Gpio(0, 17);
  Gpio ALE = Gpio(0, 22);
  Gpio M_IO = Gpio(0, 0);

  for (int i = 0; i < 8; i ++)
  {
    ad[i].GetPin().SetAsOpenDrain();
  }

  int address = 0;

  Write.SetAsOutput();
  Write.SetHigh();

  ALE.SetAsOutput();
  ALE.SetHigh();

  while(address != -1)
  {
    printf("Please enter the address or type '-1' to quit: ");
    scanf("%d", &address);
    if (address != -1)
    {
      for (int i = 0; i < 8; i++)
      {
        ad[i].SetAsOutput();
        if((address & int(pow(2,i))) > 0)
          {
            ad[i].SetHigh();
          }
        else
          {
            ad[i].SetLow();
          }
      }
    }

  }
 return 0;
}
```