**DSM Lab Report**

**Name:** Anirudh Kaushik

**Roll No:** 2020111015

**Group Number:** 5

**Experiment Part A**

**Objective**

To get familiar with the working of a tri state buffer and understand data flow control using a tristate buffer.

**Theory**

The tri-state buffer (Fig 1) functions just as a regular digital buffer where the value at its input is propagated to its output. But it has an additional capability that allows us to configure its output to a Hi-Z (high impedance) state.

When the output of the buffer is in Hi-Z state it is basically disconnected (isolated) from the rest of the electric circuit. This makes it very useful when connecting multiple devices on a single bus, as its isolation prevents the occurrence of a short circuit event.

**Experiment setup/ procedure**

**Materials required:**

|  |  |
| --- | --- |
| 1 | Quad NAND gate |
| 1 | pMOS Transistor (MOSFET) |
| 1 | nMOS Transistor (MOSFET) |
| 1 | Arduino Uno R3 |
| 1 | 1 kΩ Resistor |
| 1 | Red LED |
| 1 | Resistance Multimeter |

**Procedure:**

1. In the experiment, you will verify the working of a tristate as shown in the circuit diagram use the tri state buffers to transfer contents of one shift register (74HC595 IC) to another shift register. Note that the output of the shift register is parallel (you will be using only the first 4 bits) while the input is serial. Therefore, you would be connecting the outputs to a single data bus using tristate buffers and reading the input to the second register from that bus.
2. Verify the working of the tri state buffer and create the truth table.

**Code:**

int latchPin = 13;

//Pin connected to SH\_CP of 74HC595

int clockPin = 12;

////Pin connected to DS of 74HC595

int dataPin = 11;

void setup() {

//set pins to output because they are addressed in the main loop

pinMode(latchPin, OUTPUT);

pinMode(clockPin, OUTPUT);

pinMode(dataPin, OUTPUT);

Serial.begin(9600);

}

int val;

int A;

int B,C;

void loop()

{

if(Serial.available() > 0)

{

while(!Serial.available())

{}

B = Serial.read(); // x would be an integer between 0 and 255

// depending on the ascii value of the character read

B = B - '0'; // Subtracting ascii value of 0 from x.

if(B == 0)

{

digitalWrite(12,LOW);

Serial.print("Enable = ");

Serial.println(B);

}

if(B == 1)

{

digitalWrite(12,HIGH);

Serial.print("Enable = ");

Serial.println(B);

}

while(!Serial.available())

{}

A = Serial.read(); // x would be an integer between 0 and 255

// depending on the ascii value of the character read

A = A - '0'; // Subtracting ascii value of 0 from x.

if(A == 0)

{

digitalWrite(13,LOW);

Serial.print("Input = ");

Serial.println(A);

}

if(A == 1)

{

digitalWrite(13,HIGH);

Serial.print("Input = ");

Serial.println(A);

}

}

}

**Observations**

The truth table for the tri-state buffer is as follows:

|  |  |  |
| --- | --- | --- |
| Input | Enable | Output |
| 0 | 0 | MΩ (High Impedance) |
| 1 | 0 | MΩ (High Impedance) |
| 1 | 1 | 5V |
| 0 | 1 | 0V |

**Conclusion**

The tri-state buffer works as expected. It gives an output corresponding to the Input value when Enable is set to 1. When Enable is set to 0 it enters high impedance state where the input doesn’t affect the output, the LED doesn’t glow and the multimeter shows MΩ (High impedance) and the voltage output is 48.6mV.

**Tinkercad link with circuit**

<https://www.tinkercad.com/things/lS4U33kh1ux-lab-8-part-a/editel?sharecode=9m__jErLqsAv0mXYa_1jd4TXE-idIIYt_a2VjBQHTFg>

**Circuit Diagram**

|  |
| --- |
|  |

**Experiment Part B**

**Objective**

To learn about Data flow using Tri state Buffers

**Experiment setup/ procedure**

**Materials required:**

|  |  |
| --- | --- |
| 4 | Quad NAND gate |
| 4 | pMOS Transistor (MOSFET) |
| 4 | nMOS Transistor (MOSFET) |
| 2 | 8-Bit Shift Register |
| 1 | Arduino Uno R3 |
| 8 | 1 kΩ Resistor |
| 8 | Red LED |
| 2 | Voltage Multimeter |
| 1 | Quad OR gate |

**Procedure:**

● Take an input number (0-15) and send it to the first register (Recall Lab 6).

● Write code to enable the tristate buffers in order and correspondingly apply clock pulses to the second shift register such that the content from the first register is transferred.

**Code:**

int latchPin = 13;

//Pin connected to SH\_CP of 74HC595

int clockPin = 12;

////Pin connected to DS of 74HC595

int dataPin = 11;

int latchPin2 = 9;

int clockPin2 = 10;

int a[4] = {0};

void setup() {

//set pins to output because they are addressed in the main loop

pinMode(latchPin, OUTPUT);

pinMode(clockPin, OUTPUT);

pinMode(dataPin, OUTPUT);

pinMode(latchPin2, OUTPUT);

pinMode(clockPin2, OUTPUT);

pinMode(7, OUTPUT);

pinMode(6, OUTPUT);

pinMode(5, OUTPUT);

pinMode(4, OUTPUT);

digitalWrite(7,LOW);

digitalWrite(6,LOW);

digitalWrite(5,LOW);

digitalWrite(4,LOW);

Serial.begin(9600);

}

int i;

int A;

int j =0;

void loop()

{

if(Serial.available() > 0)

{

A = Serial.parseInt();

digitalWrite(latchPin, LOW);

shiftOut(dataPin, clockPin, MSBFIRST ,A);

//return the latch pin high to signal chip that it

//no longer needs to listen for information

digitalWrite(latchPin, HIGH);

Serial.print("A = ");

Serial.println(A);

digitalWrite(latchPin2, LOW);

for(i=0;i<4;i++)

{

a[i]=1;

if(i>0)

{

a[i-1] = 0;

}

Copy();

digitalWrite(clockPin2, LOW);

digitalWrite(clockPin2, HIGH);

}

digitalWrite(latchPin2, HIGH);

Erase();

}

}

void Copy()

{

digitalWrite(7,a[0]);

digitalWrite(6,a[1]);

digitalWrite(5,a[2]);

digitalWrite(4,a[3]);

}

void Erase()

{

for(i=0;i<4;i++)

{

a[i] = 0;

}

}

**Observations**

Upon Entering an integer between 0 and 15 the corresponding integer is obtained in binary for as the Final Output from the Second Shift register.

**Conclusion**

Successfully verified the function of the Tri state buffer and verified the flow of data flow using tristate buffer. The integer inputted to the first state buffer was obtained as the corresponding output of the second shift register. Hence the circuit is working as expected.

**Tinkercad Link with Circuit**

<https://www.tinkercad.com/things/39EhLRM1y1a-lab-8-pat-b/editel?sharecode=BhTIh9-9cAIWVXKbSuxhsi7JpZSLn3btt-DZEHj7i5g>

**Please scroll down:**

**Circuit Diagram**

|  |
| --- |
|  |