DSM LAB REPORT –8

Name: M.Harish

Id no.: 2020102062

Experiment: Bus based data transfer using Tri-State Buffers

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.

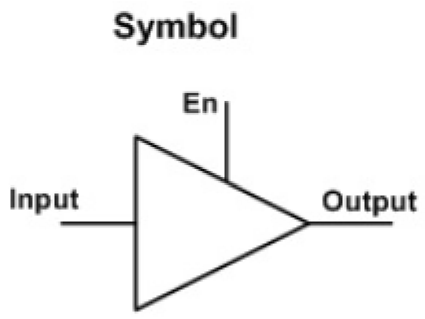


Fig. 1 Tri State Buffer

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

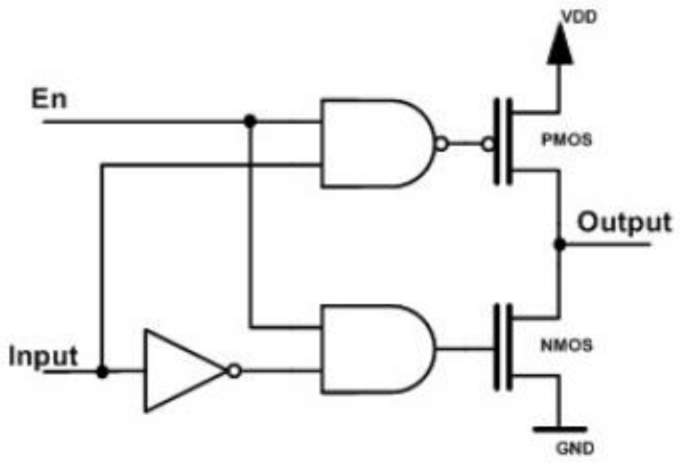


Fig. 2 Circuit of a Tri State Buffer

In Fig. 3 an example is given where five devices are connected to a bus using tri-state buffers. At any single moment, only one of them should drive the line and all others should be disabled (Hi-Z state). A device that sets its output to Hi-Z can read the value driven on the bus by other devices. In order words, only one device can write to the line, but all of the devices can read it.

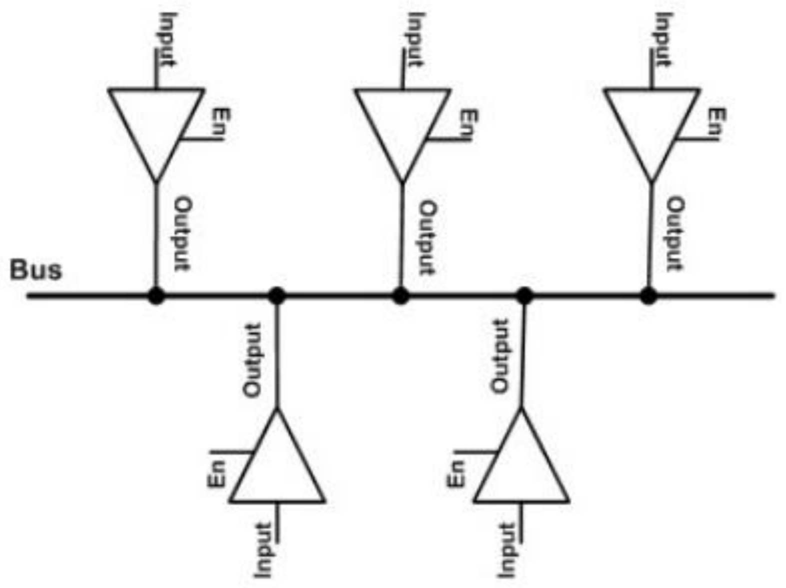


Fig. 3 Buffers connected to a data bus

In the experiment, you will verify the working of a tristate buffer and similar to the example given, 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.

Part A: ***Working of the Tri state Buffer***

Objective:

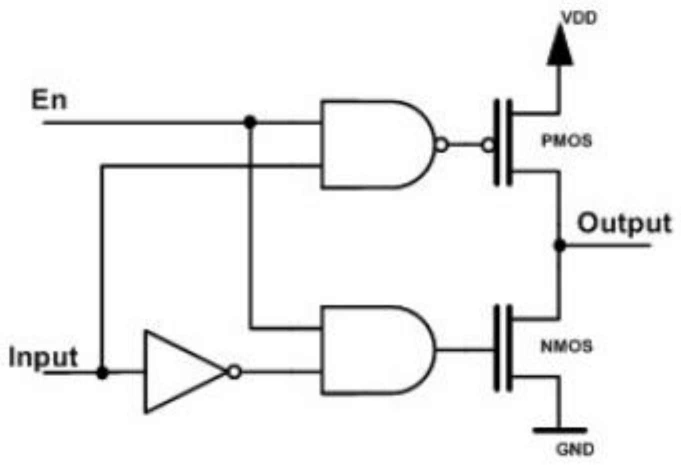
To get familiar with the working of a tri state buffer and creating the truth table.

Electronics components required:

- Arduino
- Breadboard

- Quad Nand Gate
- PMOS, NMOS
- Multimeter
- Led’s, resistors and connecting wires.

Reference circuit:



Procedure:

1).Connect the 5V pin of the Arduino to the positive strips of the breadboard and connect the GND pin of the Arduino to the negative strips of the breadboard.

2).Connect the Pin12 of the Arduino to the Input1A and Input4B of the 74HC00 – NAND gate IC.

3).This pin12 represents the Input of the buffer.

4).Connect the pin13 of the Arduino to the Input1B and Input3A of the 74HC00 – NAND gate IC.

5).This pin13 represents the enable of the buffer.

6)Drag a resistor and a LED on to the bread baord and connect the Gate pin of the Pmos to one of the terminals of the resistors and other terminal of the resistor to the anode and cathode to the ground.

Code:

```
int enable=0;

int input=0;

 void setup()

{

  Serial.begin(9600);

  pinMode(13, OUTPUT);//enable

  pinMode(12, OUTPUT);//input

}

void loop()

{

  if(Serial.available()==2)

  {

    enable=Serial.read();

    enable=enable-'0';

    Serial.print("enable:");

    Serial.println(enable);

   input=Serial.read();

    input=input-'0';

    Serial.print("INPUT:");
   Serial.println(input);
   if(enable==0)
   {
     Serial.print("high impedence state\n");
   }
digitalWrite(13,enable);
digitalWrite(12,input);
  }
}
```

Observation:

Tri-state Buffer is activated when a logic level “1” is applied to its “enable” control line and the data passes through from its input to its output. When the enable control line is at logic level “0”, the buffer output is disabled and a high impedance condition, Hi-Z is present on the output.

We can check this by connencting multimeter across the output to the terminal of the led.

Truth Table:

Enable	input	output
0	0	Hi-z
0	1	Hi-z
1	0	0
1	1	1

Link for TinkerCad Simulation:

https://www.tinkercad.com/things/eJfGDTTk2od-copy-of-tri-state-buffer/editel?sharecode=STfzl2dejRE8MQYDaIDa_hHhCaTQdtTzjjNf2_beiRM

Conclusion:

The **Tri-state Buffer is used** in many electronic and microprocessor circuits as they allow multiple logic devices to be connected to the same wire or bus without damage or loss of data.I understood the working of tri state buffers and their usage.

Part B: Data flow using Tri state Buffers

Objective:

To understand data flow control using a tristate buffer.

Electronic Components:

- Arduino
- Bread Board
- PMOS,NMOS
- Shift Registers
- Quad Nand Gate
- Led's,Resistors and connecting wires

Reference ciruit:

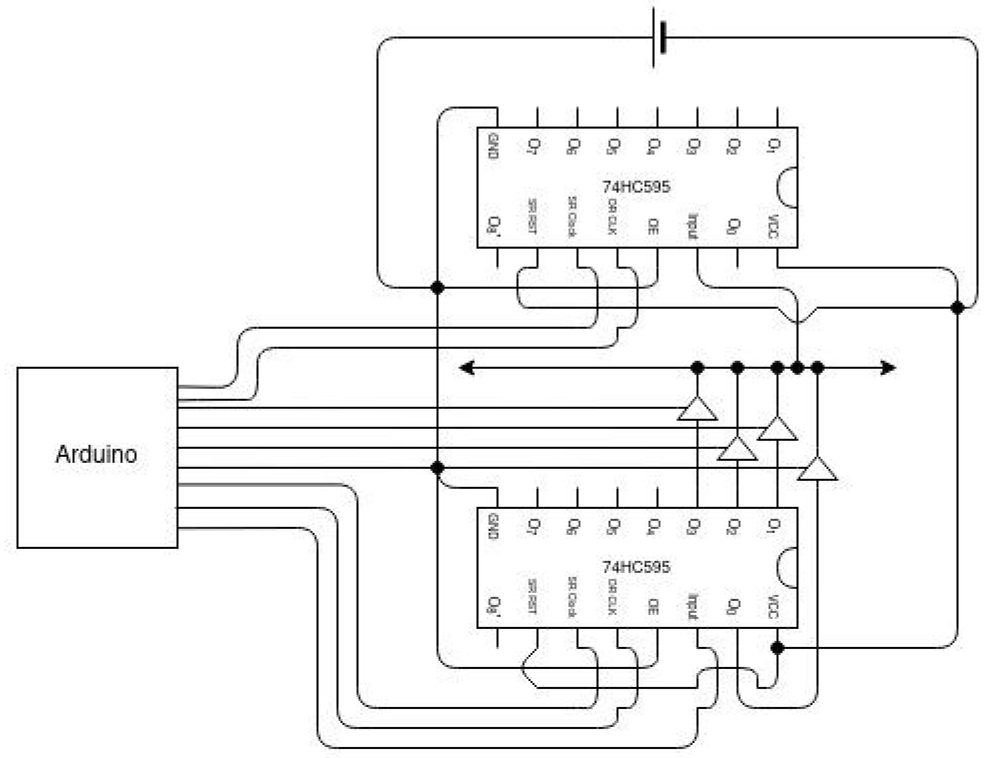


Fig. 4 Circuit Diagram

Procedure:

1).Connect the 5V pin of the Arduino to the positive strips of all the breadboards to the link given in Lab experiment.

2).Similarly, connect the GND pin of the Arduino to the negative strips of all the breadboards.

3).Now give the connections of enables and Inputs as done in the part1

->Pin 5 : Enable of Buffer 1

->Pin 6: Enable of Buffer 2

->Pin 7: Enable of Buffer 3

->Pin 8: Enable of Buffer 4

4).Drag a breadboard on to the workspace and place two 8-bit shift registers in the appropriate place and connect the 5V pin of the Arduino to the positive strips of the breadboard and GND pin of the Arduino to the negative strips of the breadboard.

5).Now make the connections of “Shift register Clock” pins, “Latch pins," and "Shift register clear” pins for both the 8-bit shift registers.

->Pin 4: SR Clk 1

->Pin 2: Input 1

->Pin 3: Latch pin 2

->Pin 10: SR Clk 2

->Pin 9: Latch pin 1

6).Now connect the Output pins 1, 2, 3, 4 of shift register 1 respectively to four LEDs. These correspond to our input.

7).Also create a data bus of output pins 1, 2, 3, 4 and send this data bus as input to the second 8-bit shift register.

8).Output enable of both the shift registers is set to GND and the shift register clear is set to 5V.

9).Connect the Output pins 1, 2, 3, 4 to four LEDs which correspond to the final output.

10).Now code the Arduino accordingly.

Code:

```
int i1;

int i0;

int input;
int inputPin = 2;

int outputCLK1 = 3;

int shiftCLK1 = 4;

int outputCLK2 = 9;

int shiftCLK2 = 10;

int En1 = 5;

int En2 = 6;

int En3 = 7;

int En4 = 8;


void setup()

{

  pinMode(2,OUTPUT);

  pinMode(3,OUTPUT);

  pinMode(4,OUTPUT);

  pinMode(5,OUTPUT);

  pinMode(6,OUTPUT);

  pinMode(7,OUTPUT);

  pinMode(8,OUTPUT);

  pinMode(9,OUTPUT);

  pinMode(10,OUTPUT);

  Serial.begin(9600);

}

void loop()
```

```
{
  if(Serial.available()==2)
  {
  i1 = Serial.read();
  i1 = i1 - '0';
  i0 = Serial.read();
  i0 = i0 - '0';
  input = 10*i1 + i0 ;
  Serial.println(input);

  digitalWrite(outputCLK1,LOW);
  shiftOut(inputPin,shiftCLK1,MSBFIRST,input);
  digitalWrite(outputCLK1,HIGH);

  digitalWrite(outputCLK2, LOW);
  digitalWrite(En1,LOW);
  digitalWrite(En2,LOW);
  digitalWrite(En3,LOW);
  digitalWrite(En4,HIGH);
  digitalWrite(shiftCLK2,HIGH);
  delay(0.5);
  digitalWrite(shiftCLK2,LOW);
  delay(0.5);
  digitalWrite(outputCLK2, HIGH);
```

```
delay(1);

digitalWrite(outputCLK2, LOW);
digitalWrite(En1,LOW);
digitalWrite(En2,LOW);
digitalWrite(En3,HIGH);
digitalWrite(En4,LOW);
digitalWrite(shiftCLK2,HIGH);
delay(0.5);
digitalWrite(shiftCLK2,LOW);
delay(0.5);
digitalWrite(outputCLK2, HIGH);
delay(1);

digitalWrite(outputCLK2, LOW);
digitalWrite(En1,LOW);
digitalWrite(En2,HIGH);
digitalWrite(En3,LOW);
digitalWrite(En4,LOW);
digitalWrite(shiftCLK2,HIGH);
delay(0.5);
digitalWrite(shiftCLK2,LOW);
delay(0.5);
digitalWrite(outputCLK2, HIGH);
```

```
  delay(1);


  digitalWrite(outputCLK2, LOW);
  digitalWrite(En1,HIGH);
  digitalWrite(En2,LOW);
  digitalWrite(En3,LOW);
  digitalWrite(En4,LOW);
  digitalWrite(shiftCLK2,HIGH);
  delay(0.5);
  digitalWrite(shiftCLK2,LOW);
  delay(0.5);
  digitalWrite(outputCLK2, HIGH);
  delay(1);
  }
}
```

Observations:

Shift register 1:

input	O3	O2	O1	O0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0

7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

These outputs are sent to the input of another register via bus using tristate buffers.

We can see the transfer of data from shift register 1 to shift register 2.

We can verify them.

Shift register 2:

Output of second shift register when the outputs of first register is passed as an input of second register.

We can see the transfer of data from register to another.

O3	O2	O1	O0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0

1	**0**	**0**	**1**
1	**0**	**1**	**0**
1	**0**	**1**	**1**
1	**1**	**0**	**0**
1	**1**	**0**	**1**
1	**1**	**1**	**0**
1	**1**	**1**	**1**

Link for TinkerCad Simulation:

https://www.tinkercad.com/things/erYFb91glgF-copy-of-data-transfer-using-tri-state-buffer/editel?sharecode=wqXOqgw2AVAS5r9fPZNAMAn2l1b08JcGsZdD3z2I5Rs

Conclusion:

From this experiment I understood the passage of data(output) of one register as a input to the another register via bus and understood the data flow control using tristate buffers.