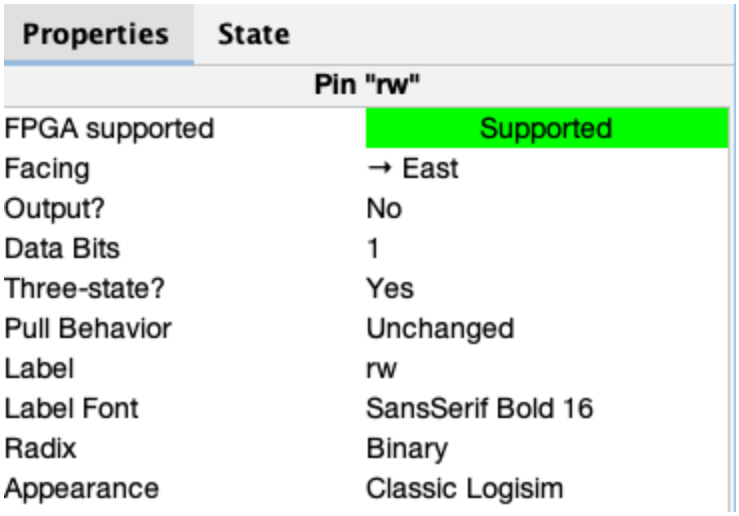
Study Guide: Floating Inputs in Digital Logic and Their Application

Introduction

This study guide explores the concept of floating inputs in digital logic and how they can be manipulated using pull-up and pull-down resistors to implement useful circuit behaviors. We'll examine how these concepts are implemented in the memory circuit (oneKram) to effectively disconnect it from read or write operations without the need for physical switches. Specifically, the oneKram “chip” uses this feature on the read/write input line. If you look at how its designed you will see that it has a state “U” - in Logisim, this is the “high impedance” or “floating” state. Logisim uses blue lines for things that are “floating” or do not have a digital value. We will learn more about how this can be used as a third state shortly, but for now lets just remember that we need to enable this feature in the properties (see below) by setting “Three State?” to “Yes”:



The approach used in the provided circuit is common in realworld circuits. We have a single bit for read/write - 0 is for read, 1 is for write, and use the floating state to effectively disconnect this circuit when integrated into another circuit (as we will see in the main circuit)

1. The Three States of Digital Logic

Traditional Binary Logic

In standard digital logic, we typically recognize two states:

- **Logic 0**: Represented by a LOW voltage (usually 0V)
- **Logic 1**: Represented by a HIGH voltage (usually VCC, like 5V or 3.3V)

All digital operations are based on manipulating these two states.

The "Third State": Floating

There exists a third state in digital systems that is often overlooked:

- **Floating state**: Neither definitively HIGH nor LOW, its essentially disconnected (in logisim, shown as blue lines).
- Not directly represented in logic values (not a "2" or a "Z" in logic terms)
- A physical state of being disconnected rather than a logical value
- Unpredictable behavior when directly used

Important Note About Floating Inputs

Floating inputs can't be used directly as a third logic value:

- Their behavior is unpredictable
- They can't be reliably detected as "floating" by other logic gates
- They require conversion to standard logic values (0 or 1) to be useful. This is super important and will be covered next. The floating state cannot be used directly, it must be detected and then converted to either a zero or a one.
- This conversion is done through pull-up or pull-down resistors.

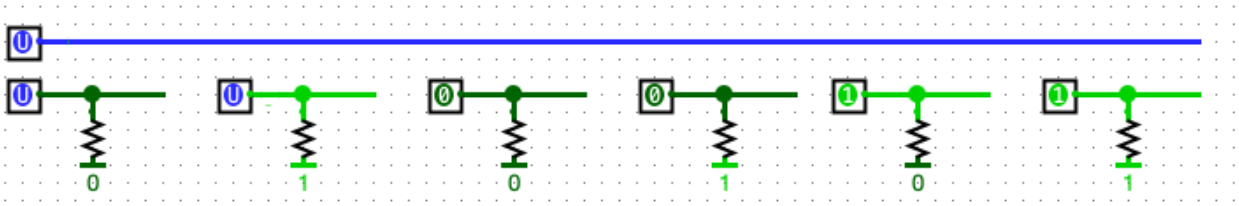
2. Pull-Up and Pull-Down Resistors: Converting Floating to Useful Logic

How Pull Resistors Work in Digital Context

Pull resistors convert the ambiguous floating state into a defined logic level:

- **Pull-up resistor**:
    - Converts floating → Logic 1 (HIGH)
    - Provides a "default HIGH" when no active signal is present
- **Pull-down resistor**:
    - Converts floating → Logic 0 (LOW)
    - Provides a "default LOW" when no active signal is present

See how the picture below shows the behavior of pullup and pulldown resistors. They are found in the wiring section of logisim. To start, look at the top line, the input is floating “U” and the line is blue indicating no connection. Below that we show how the pull resistors work on various inputs. On the left we have inputs in the floating state. Notice how with the pulldown resistor (the zero on the bottom) pulls the the value of the line to zero. With a pullup resistor (second from left) it pulls the value up to a 1. However, the remaining 4 examples show what happens when the input is a zero or a one. Notice in these cases the pulldown or pullup resistors have no effect on the line.



The Critical Role of Buffers with Pull Resistors

An important aspect of the oneKram circuit is the use of buffers with the pull resistors:

1. **What is a Buffer?**

    The buffer in logisim is found under the “Gates” tools (see below). Its basically a triangle with a single input and a single output.

- A simple logic gate that passes its input value to its output unchanged
- It provides isolation between input and output circuits
- It maintains signal integrity while preventing "backflow" of effects

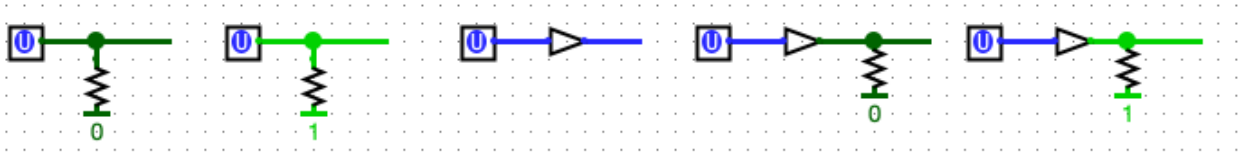
2. **Why Buffers are Necessary:**

    - If we connected pull resistors directly to floating inputs, we'd eliminate the floating state completely
    - The resistor would pull the input itself up or down
    - The original floating condition would be lost
    - We couldn't distinguish between "actively driven" and "floating but pulled"

3. **Buffer + Pull Resistor Arrangement:**

    - The buffer preserves the floating state at its input
    - The pull resistor acts only on the buffer's output
    - This maintains the floating condition at the input while providing a defined logic level at the output
    - This arrangement allows us to: a) Preserve the original floating state for other parts of the circuit b) Generate a usable logic value for the memory control logic

Notice how buffers and pull resistors work together. In the two configurations on the left we see introducing pull resistors to a floating input pulls the line to a zero or one based on the type of the pull resistor. We often do not want this, preserving the floating value to be used as a third digital value. The buffer in the center picture shows this. Its just connected inline to the floating value. Where it becomes useful is in the two figures on the right side. The output of the buffer is combined with a pull resistor. Therefore the input is preserved, but the output is driven to a zero or one, which can then be used in other logic, as I have done in the oneKram circuit.

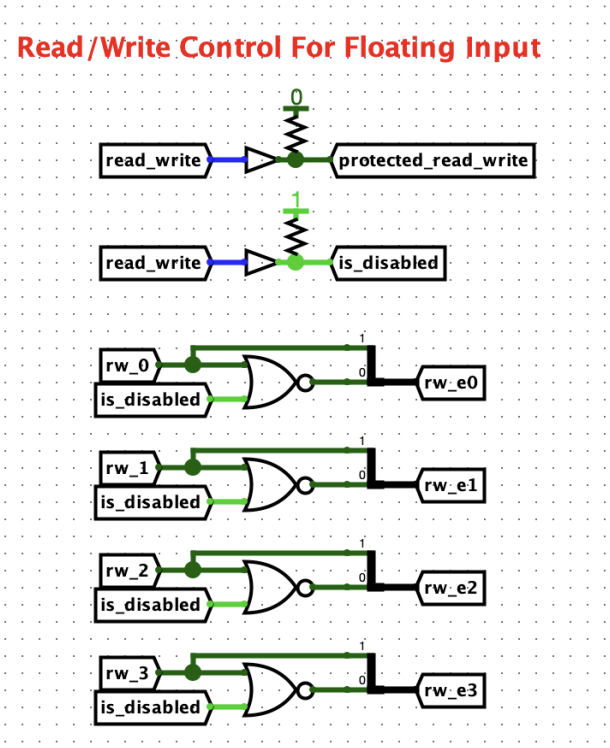


The Digital Logic Perspective

From a purely digital standpoint:

1. Pull resistors ensure every input has a defined logic state
2. They create a default state that exists when no active driving signal is present
3. They allow a stronger signal to override the default state when present
4. They enable detection of the "disconnected" condition indirectly

3. Logical Implementation in the oneKram Circuit



Key Components from a Digital Logic Perspective

Looking at your circuit file, we can analyze how floating inputs are manipulated:

1. **Buffer Components**: The first 2 items at the top of the picture above

    - These are digital buffers that pass their input to their output unchanged
    - They isolate the input from the output, preserving the floating state at the input
    - Without these buffers, connecting pull resistors would eliminate the floating condition
    - They provide a connection point for the pull resistors at their outputs

2. **Pull Resistors**:

- Pull-Up resistor: Forces a default HIGH state at the buffer output
- Pull-Down resistor: Forces a default LOW state at the buffer output
- These convert floating inputs into usable logic values without affecting the original input

3. **Logic Gates**:

    - NOR gates at the bottom of the picture above are used to setup the read/write enable pins on the memory chips.
    - These combine the pull resistor outputs with other signals

Preserving Floating Inputs

The ingenious aspect of this circuit is how it preserves the floating state while also generating usable values:

1. **Input Side (Buffer Input)**:

    - When control signals are floating, they remain floating at the buffer inputs
    - Other circuit components that need to see the true input state still see the floating condition
    - The original floating state is preserved and not altered

2. **Output Side (Buffer Output)**:

    - The buffer outputs would also be undefined/floating
    - But the pull resistors force these outputs to specific default values
    - These defined values can be used by the rest of the logic circuit

3. **Dual Nature**:

    - The circuit maintains both the original floating state and a defined default logic value
    - This allows the circuit to both preserve the input condition and generate a reliable response

How the Circuit Uses Floating Inputs

The circuit cleverly uses the concept of floating inputs to implement a memory disconnection feature:

1. **Normal Operation**:

    - When inputs are actively driven (not floating)

- The signals pass through the buffers
- The pull resistors are overridden by the stronger output signals
- Memory operates normally based on the provided signals

2. **Disconnected State (Floating Inputs)**:

   - When control signals are left floating (disconnected)
   - The buffer inputs remain floating, preserving this state
   - The buffer outputs would also be floating, but:
   - The pull-up resistor forces "protected_read_write" to HIGH
   - The pull-down resistor forces "is_disabled" to LOW
   - This specific combination disables all memory operations
   - The memory is effectively "disconnected" through logic

3. **Default State Logic**:

   - The NOR gates receive specific default values from the pull resistors
   - These values propagate through the circuit
   - They create a logical state that prevents memory access
   - No physical disconnection is needed - it's all done in logic

Digital Logic Interpretation of Floating Inputs

From a purely digital logic perspective:

- The floating state itself isn't directly used in calculations
- Instead, the circuit detects "no active signal present" indirectly
- It does this by establishing default values through buffers and pull resistors
- These default values then trigger specific circuit behaviors

4. Converting Floating to Useful Logic: Practical Applications

Using Floating Inputs in Digital Systems

While floating inputs themselves are not directly usable as logic values, their detection and conversion via pull resistors enables several useful digital patterns:

1. Default State Mechanism

- Pull resistors establish known default states
- These defaults apply whenever active signals are absent
- This creates predictable behavior without requiring explicit control signals

2. Implicit Disconnection

As seen in the oneKram circuit:

- No explicit "disconnect" signal is needed
- The mere absence of control signals (floating state) triggers disconnection
- This simplifies control logic significantly

3. "Don't Care" State Handling

- In truth tables, we often use "X" for "don't care" conditions
- Floating inputs with pull resistors implement this concept in hardware
- They provide a default value for otherwise unspecified conditions

4. Mode Selection

- Multiple operating modes based on whether pins are:
    - Actively driven HIGH
    - Actively driven LOW
    - Left floating (pulls to default)
- This allows 3 distinct behaviors from a single physical pin

How the oneKram Circuit Uses This Principle

The oneKram circuit uses floating inputs as a control mechanism:

1. **Logical Disconnection**:
    - When read/write control signals are floating
    - Pull resistors establish specific default values
    - These defaults propagate through NOR gates
    - The resulting output prevents memory access
    - Memory is effectively disconnected without physical switches
2. **Fail-Safe Default**:
    - Memory protection is the default state
    - Active signals are required to enable access
    - If control is lost (signals become floating), protection automatically engages
3. **Silent Operation**:
    - The circuit doesn't require explicit "disconnect" signals
    - It recognizes the absence of control signals

- This simplifies the control interface

5. Hands-On Experiments with Floating Inputs

Experiment 1: Observing the "Third State" and Its Conversion

Setup:

1. Open the provided circuit in Logisim
2. Navigate to the oneKram subcircuit
3. Locate the buffer components (at coordinates 1100,460 and 1100,520)
4. Add probes to the buffer inputs and outputs

Procedure:

1. First, apply a logic 1 to the buffer inputs and observe the outputs
2. Next, apply a logic 0 and observe the outputs
3. Now, disconnect the inputs (make them float)
4. Observe how:
    - The output with the pull-up resistor goes to logic 1
    - The output with the pull-down resistor goes to logic 0
5. This demonstrates how the "third state" (floating) is converted to usable logic values

Experiment 2: Memory Disconnection via Floating Inputs

Setup:

1. In the main circuit, locate the read_mode and write_mode inputs
2. Add probes to:
    - The read_mode and write_mode inputs
    - The protected_read_write and is_disabled signals
    - The memory data lines

Procedure:

1. Write a value to a memory address with write_mode enabled
2. Read the value back with read_mode enabled
3. Now disconnect both read_mode and write_mode (make them float)
4. Try to write a different value to the same address
5. Try to read from the address
6. Observe that neither operation succeeds - the memory is logically disconnected
7. This shows how floating inputs can implement a disconnection feature

Experiment 3: Modifying the Default Behavior

Setup:

1. Make a copy of the oneKram circuit
2. Locate the pull-up and pull-down resistors

Procedure:

1. Swap the positions of the pull-up and pull-down resistors
2. Repeat Experiment 2 with this modified circuit
3. Observe how the default behavior changes:
    - With the original configuration, floating inputs disabled memory access
    - With the modified configuration, the behavior will be different
4. This demonstrates how the specific arrangement of pull resistors defines how floating inputs are interpreted

6. Advanced Concepts

Tri-State Logic

The concept of floating signals extends to tri-state logic:

1. **Three states**: HIGH, LOW, and high-impedance (floating)
2. **Applications**: Shared buses, memory interfaces
3. **Implementation**: Uses enable signals to activate/deactivate drivers

Open-Collector and Open-Drain Outputs

Related concepts include:

1. **Open-collector/drain**: Outputs that can pull LOW but not HIGH
2. **Wired-AND/OR**: Multiple open-collector outputs connected to a single pull-up
3. **Applications**: I2C bus, certain interrupt lines

7. Study Questions (you don't need to answer these questions but you should understand these concepts).

If you are still confused about floating states and how they are managed do a little online research, watch youtube videos, find additional information. These are the things you should understand with respect to this important topic.

1. What happens to an unconnected (floating) input in a digital circuit?
2. Why are floating inputs problematic in digital systems?
3. What is the purpose of a pull-up resistor? A pull-down resistor?
4. How do pull resistors interact with actively driven signals?
5. In the oneKram circuit, what is the purpose of the buffer gates connected to pull resistors?
6. How does the circuit use pull resistors to implement memory protection?
7. What would happen if the pull resistors were removed from the circuit?
8. Why are specific resistance values chosen for pull-up and pull-down resistors?
9. Design a simple circuit that uses a pull-up resistor to create a default state for a button input.
10. How could you modify the oneKram circuit to change its default protection behavior?

8. Key Terms and Definitions

- **Floating input**: An input pin not connected to any defined logic level
- **Pull-up resistor**: A resistor connected between an input and the positive supply voltage
- **Pull-down resistor**: A resistor connected between an input and ground
- **Buffer gate**: A digital gate that provides isolation and signal restoration
- **High impedance**: An electrical state with very high resistance to current flow
- **Tri-state logic**: Digital logic with three possible states (HIGH, LOW, high-impedance)
- **Noise margin**: The amount of noise a digital circuit can tolerate while maintaining correct operation

6. Case Study: Floating Inputs for Memory Disconnection

Let's analyze how the oneKram circuit uses floating inputs and pull resistors to implement logical memory disconnection:

The "Third State" Implementation

The circuit demonstrates that while we can't directly use floating as a logic value, we can:

1. Detect the absence of a driving signal

2. Convert this "third state" into usable logic values
3. Use those converted values to trigger specific behaviors

Component Analysis

1. **Buffers at (1100,460) and (1100,520)**:

    - These simple logic gates pass their input values to their outputs unchanged
    - They preserve the floating state at their inputs while allowing their outputs to be pulled to defined levels
    - Without buffers, connecting resistors directly would eliminate the floating condition
    - This isolation is crucial for the circuit's operation

2. **Pull-Up and Pull-Down Resistors**:

    - Pull-Up at (1110,460): Converts floating → HIGH at buffer output
    - Pull-Down at (1110,520): Converts floating → LOW at buffer output
    - Together they establish a specific pattern of defaults without affecting the inputs

3. **NOR Gates** at (1120,590), (1120,650), (1120,710), and (1120,770):

    - These combine the default values with other signals
    - When is_disabled is LOW (default from pull-down), they force outputs LOW
    - This disables memory operations regardless of other inputs

The Dual Nature of the Circuit

What makes this design particularly elegant is its dual handling of the floating state:

1. **Preservation**:

    - The buffer inputs see and preserve the true floating state
    - This floating condition remains available to other circuit components
    - The original state is not altered or affected

2. **Conversion**:

    - The buffer outputs would also be undefined (floating)
    - But the pull resistors convert this to useful logic values
    - These defined values drive the rest of the circuit

This dual nature allows the circuit to both maintain the integrity of the floating inputs and generate predictable behavior from them.

Signal Flow Analysis

1. **When Control Signals are Present**:

    - Strong active signals pass through the buffers
    - These strong signals at the buffer outputs override the pull resistors
    - Memory operations proceed normally based on these signals

2. **When Control Signals are Floating**:

    - Buffer inputs remain in their true floating state
    - Buffer outputs would also be floating, but are pulled to defined levels:
        - Pull-Up forces "protected_read_write" → HIGH
        - Pull-Down forces "is_disabled" → LOW
    - NOR gates receive is_disabled = LOW, forcing outputs LOW
    - This specific combination disables all memory operations
    - Memory is effectively "disconnected" through pure logic

Why This Design is Clever

1. **No Explicit Disconnect Signal Needed**:

    - The mere absence of control signals triggers protection
    - This simplifies the control interface

2. **Fail-Safe by Default**:

    - If control is lost for any reason, protection engages automatically
    - This prevents accidental memory corruption

3. **Pure Logic Solution**:

    - No physical switches or additional control signals required
    - The floating state is leveraged as an implicit control mechanism

4. **Preserves Original Input Condition**:

    - The buffer maintains the floating state at its input
    - This allows other parts of the circuit to still see the true input condition
    - The floating state is converted to usable logic only where needed

10. Further Reading

1. Horowitz, P., & Hill, W. (2015). *The Art of Electronics*. Cambridge University Press. (Chapter 3: "Field-Effect Transistors" covers pull-up/down applications)
2. Wakerly, J. F. (2017). *Digital Design: Principles and Practices*. Pearson. (Section 5.6 on "Input Interfaces")
3. Tocci, R. J., Widmer, N. S., & Moss, G. L. (2017). *Digital Systems: Principles and Applications*. Pearson.
4. Application Note: "Pull-up Resistors in Digital Systems," Texas Instruments.
5. Floyd, T. L. (2019). *Digital Fundamentals*. Pearson.
6. Application Note: "Design Guidelines for Pull-up and Pull-down Resistors," Microchip Technology.
7. Harris, D. M., & Harris, S. L. (2015). *Digital Design and Computer Architecture*. Morgan Kaufmann. (Section on "Special Inputs and Outputs")
8. Kainka, B. (2012). *Electronic Circuits for the Evil Genius*. McGraw-Hill. (Good practical examples of pull resistors)

Conclusion

The oneKram circuit demonstrates an elegant application of floating inputs and pull resistors to achieve logical disconnection of memory. This technique is both simple and powerful, creating a fail-safe system where disconnection is the default state when signals are floating. By using buffers with pull-up and pull-down resistors, the circuit ensures predictable behavior even when control signals are not actively driven.

Understanding these principles is crucial for designing reliable digital systems, especially in applications where component protection, fail-safe operation, or power management are concerns. The professor's circuit provides an excellent case study in how seemingly simple components like pull resistors can be used to implement sophisticated control mechanisms.

By mastering these concepts, students will gain valuable insights into digital design techniques that extend far beyond this specific application, preparing them for challenges in real-world circuit design and troubleshooting.