Benchmarking and Analysis of Common Redstone Computation Techniques

Brandon Esebag, Aaron Hoang, Jonathan Liu, Maria N. Aponte, **Supervisor** : Mohammed El-Hadedy

***Abstract*—The abstract goes here.**

***Index Terms*—IEEE, journal, education, benchmarking, Minecraft, redstone**

I. Introduction

THERE are few video games as widely played as Minecraft, and although on the surface it presents itself as a simple children's sandbox game, Minecraft has provided countless people of all ages with the opportunity to learn the basics of computer architecture, knowingly or not. Many members of the online Minecraft community have produced computational components or whole working systems composed of in-game elements. These systems range in complexity from simple working ALUs to entire Minecraft emulations within the game itself.

A. Scope

This paper will focus on benchmarking common computational components and features of Minecraft redstone computers and describing current solutions to common computational and design problems. To benchmark redstone systems, redstone relevant benchmarking metrics will be discussed as standard benchmarking methods cannot be meaningfully applied to redstone systems.

B. An Overview of Minecraft

Minecraft is an open-world sandbox game released on November 18^{th} 2011. The game features a functionally infinite, pseudorandomly generated world composed of cubic-meter blocks. This world can be interacted with through adding or removing blocks, using resources (e.g. blocks or items obtained by breaking blocks) to craft items and tools, and by interacting with mobs (non-player entities).

The game is typically played in "survival mode" which limits player resources and requires the player to obtain food and shelter to survive, however, many players who want to explore the game without such restrictions use the game's "creative mode" which allows unlimited resources, invulnerability to damage, and player flight. This paper will look at the resource costs associated with certain computational techniques; however, it should be noted that many of the players who implement such systems are playing in creative mode, and thus have infinite resources at their disposal.

In Minecraft there are many blocks and ores that can be mined, as suggested by the name "Minecraft", although most of these ores are inspired by real life materials such as iron, copper, diamonds, gold, and lapis lazuli. One of the few ore types in Minecraft that is not modeled off of a real world material is known as redstone. As the name suggests, this ore appears in red ore veins and the material itself is red in color.

C. Minecraft Redstone

When a player mines a redstone ore block with a pickaxe (not enchanted with "silk touch") they can recover around 4-5 "redstone dust", though there are ways of obtaining more redstone dust per ore block using other enchantments. Tool enchantments such as "silk touch" are a game mechanic that falls outside the scope of this paper.

1) Redstone Dust: Redstone dust can be placed on top of blocks to form lines of dust, which can be connected at T-joints or X-joints, and these paths can traverse up the sides of blocks over a distance of 1 block. This means that for two redstone sections, or 'nets' to be isolated, they require 1 block of distance between them.

When redstone dust is placed, it has an "unpowered" state by default. Certain blocks can power redstone dust, making the dark red line of dust glow a bright red. Powered redstone can influence some items in the world, such as extending pistons, opening doors, and similar. If a redstone line terminates at the base of a block, this block will become "powered" though it will not change in appearance.

2) Redstone Torch: Using a single wooden stick, and one redstone dust, the player can craft a "redstone torch" which can be placed on the side of most blocks. This torch faintly glows red when placed, and is by default in its "powered state". If the block that anchors the redstone torch is powered, the torch will become "unpowered" and stop glowing. A redstone line that terminates facing a redstone torch can be powered by a redstone torch. This means that in a basic sense, a block with a redstone torch placed on it acts like a 1-bit logic inverter.

3) Redstone Tick: A redstone tick is a unit of time that (barring game lag) takes 0.1 seconds to complete. The power levels of each redstone net position and the application of a redstone signal to a powerable block take 1 redstone tick to complete. The game simulates 20 standard ticks per second which are used to update all other game physics. Unless mentioned, all ticks mentioned will be redstone ticks.

4) Redstone Repeater: Redstone dust can transmit power for a distance of 15 blocks, but due to signal attenuation, it cannot propagate further into the network from the nearest power source. Using three stone blocks, one redstone dust, and two redstone torches, the player can craft an item called a redstone repeater. This device has a triangle on top which

is fitting because it acts both as a signal amplifier and as a signal buffer. A redstone repeater always outputs a signal with strength 15 (maximum) so long as the input signal has a strength greater than 0 (fully attenuated).

Redstone repeaters by default impose a 1 tick delay, however this can be adjusted in increments of 1 tick up to 5 ticks, allowing a single repeater to cause 0.5 seconds of delay.

5) Redstone Comparator: Redstone comparators are a redstone component, similar in placement and appearence to a redstone repeater. Comparators have a "back input" and two "side inputs". The default mode of operation outputs the rear input's signal strength only when the rear input is stronger than both side inputs. This behavior can be modeled by Equation 1. A comparator can also be used in "subtraction mode", which will produce outputs in accordance with Equation 2.

$$\text{output} = \text{rear} \times [\text{left} \leq \text{rear} \ \wedge \text{right} \leq \text{rear}] \tag{1}$$

$$\text{output} = \max(\text{rear} - \max(\text{left}, \text{right}), 0) \tag{2}$$

6) Other Redstone Components: There are numerous ways the user can power a redstone net, such as pressure plates, buttons (placable on the side of a block), switches (top or side placement). Minecraft also has many sensors such as daylight sensors which can be used to implement logic conditional on the current world state. There are numerous other redstone items, such as redstone blocks (act as a power source), redstone lamps (redstone powered light sources), dispensers (can launch items), powered mine-cart tracks and many more.

D. A Note on Power Consumption

Although Minecraft redstone implements signal attenuation through the use of the aforementioned "redstone power" metric, redstone actuation and signal switching does not actually consume power from any source. Redstone systems are self-powering, and as such the power consumption of a Minecraft computer is not something that can be benchmarked, as power consumption is not a feature of redstone.

E. Research Questions

This research was motivated by the authors' prior experience with redstone-based digital logic, as well as the recognition that Minecraft unexpectedly encourages players to tinker with digital logic circuits without the guise of being an educational tool.

This observation is very important for modeling other forms of educational software, the fact that Minecraft unintentionally encourages its players to grapple with real engineering problems through its implementation of redstone is significant because doing so in-game feels no different than other aspects of gameplay. The game does not instruct the player to use redstone circuitry, it is simply there to be explored by players when they are ready.

Minecraft computers have become significantly more common as internet resources detailing their construction have become more commonplace. As a result of this "explosion" of player creativity, a natural question would be how these computers could be compared to each other given the significant variation and scope of these projects. The relevant research questions resulting from these lines of thinking are as such.

1) How do you benchmark a Minecraft computer?
2) What kinds of real engineering issues arise from Minecraft computing?
3) What unique educational value can Minecraft redstone provide?
4) What makes Minecraft redstone feel so approachable to such a wide audience?

II. Related Work

Ever since redstone was added in July 2010 as a feature to the game, players and the community have experimented by creating all sorts of automated tools including our project's focus. In the first few years of the addition, players first created simple gates and latches. Overtime, players then moved on to more complex digital circuits including full working ALUs, CPUs, and computers. Many websites such as Minecraft Wikipedia and Minecraft 101 and videos on Youtube have information and demonstrations on how to build and create these circuits.

A. Community Redstone Designs

The first post on the official Minecraft forum page was posted on December 20, 2010 titled "Redstone Logic Gates and FAQs Compendium". This included all logic gates, latches, flip flops, and all digital design elements and circuits. By April 2011, community member "Salaja" posted the first recorded post including a combination of community member designs for ALUs, CPUs, and Computers. To this day, community members have researched and experimented with designs creating more optimized limiting clock speed, propagation delay, and build size.

B. Minecraft Wiki Contributions

The Minecraft Wiki is a website that extensively documents Minecraft's vast features and mechanics, and in great detail discusses redstone in many of its articles. In the Minecraft Wiki article "Redstone circuits/Memory" for example, a detailed explanation of SR latches, T-Flip Flops, and comparisons of redstone implementations of them. These comparisons implement size and resource cost benchmarking which will be examined further in section III-D below.

C. Minecraft Community Computers

There have been numerous examples of fully functioning computer systems built in Minecraft, though they have only been documented informally. Below is a short list of some notable examples that can be found on Youtube, as most of the people producing these systems are Minecraft Youtubers.

- The Youtuber "mattbatwings" produced a simple computer that can perform mathematical operations and display to a "black and white" screen.

- The Youtuber "Torb" produced an emulation of Tetris on a new "RGB display" variant.
- The Youtuber "sammyuri" produced a "black and white" emulation of Minecraft.
- The Youtuber "sammyuri" produced a working pre-trained redstone LLM.
- The YouTuber "ModPunchtree" produced a computer called "Iris" which can run Doom (1993), ray-tracing, and a 16-color emulation of Minecraft.

A major source of innovation in the Minecraft redstone community has always been display "technology" as the game only provides a limited selection of output devices that these computers can display to. Though this is outside the scope of this paper, the engineering challenges of producing working displays in Minecraft are similarly complex to producing a working computer system.

III. Benchmarking Methodology

Normal benchmarking methodology is not completely applicable to redstone logic for three important reasons which are as follows:

1) Redstone does not consume power
2) Redstone circuits (at most) operate in the 1-10 Hz range
3) Redstone circuits occupy many cubic meters and require a meter of separation between nets/traces

Metrics like cycles per instruction, data transfer speeds (by bits per Tick) and similar work similarly to normal benchmarking metrics. Because of the relatively slow Tick speed limitations imposed by Minecraft, metrics like "Millions of Instructions per Second" or MIPS become inconvenient, as the fastest device conceivably possible (1 Instruction per Tick) would only achieve 10 instructions per second, or $1 \cdot 10^{-5}$ MIPS. Similarly any analysis of circuit size will need to be adjusted, as even a simple redstone torch inverter (the simplest redstone logic gate) occupies two cubic meters of space.

A. Performance Metrics

1) Ticks per Operation: For devices that perform read, write, or logic computations, performance can simply be measured in Ticks per Operation or TpO. A device having a TpO of 0 implies the device is not amplified and only contains redstone nets, thus practically the lowest TpO a device should have is 1 outside of special cases.

2) Ticks per Computation: For devices that perform more complex tasks such as instruction execution or computation, performance can be measured in Ticks per Computation, or TpC. Similarly to TpO, the lowest TpC a device should normally have is 1, except any computer that can perform parelell processing can achieve lower TpC values. This is not commonly used in Minecraft redstone design because the true bottleneck of computer speed is often the game engine and the computer running Minecraft in the real world, meaning actual Ticks often last much longer than 0.1 seconds.

B. Clocking and Synchronization Considerations

A redstone clock can be created by forming a loop of redstone with two repeaters (they must be separated). One net must be enabled for 1 tick, which will result in the system oscillating between its two stable states. This can be done with larger even numbers of repeaters to make slower clocks if needed. Since redstone suffers signal attenuation, redstone repeaters are often needed to amplify signals over long nets. This means that clock delay will be present in large designs, resulting in H trees and other clock synchronization methods being used. Similarly, worst case analysis needs to be performed on complex circuits to determine the maximum viable clock speed. These are both real life design challenges that are often faced when implementing sequential logic using redstone, and this will effect both TpO and TpC values. We can normalize TpO and TpC in relation to the clock-speed, resulting in Cycles per Operation (CpO) and Cycles per Computation (CpC).

C. Chunk Loading Considerations

In Minecraft, a "chunk" is a 16×16 block region (from the bottom to the top of the world). Only a predetermined number of chunks are loaded at any given time (based on player position), and both redstone and general ticks only trigger updates in loaded chunks. Often techniques are used to keep many chunks loaded, such as abusing in game item properties, but often server commands (in creative worlds) or mods are used to prevent chunk loading issues for large computer projects. the details of chunk loading are outside of the scope of this project, however this is an important consideration when designing a large system.

D. Cost Metrics

1) Material Costs: A computational element can be evaluated based on the recourse costs in producing it, which would simply be a list of all items needed to construct it.

2) Spatial Footprint: Since Minecraft redstone computers often occupy a 3-D volume, the footprint given for any computational element or system will be given in cubic meters instead of square meters.

3) Routing Considerations: As mentioned in Section III-B, signal attenuation often results in using redstone repeaters for spatially long nets. This will impart a delay of at least 1 tick, encouraging efficient routing when designing redstone computers. This is another example of a real engineering problem being disguised as a game-mechanic.

IV. Benchmarking Results

A. 1-Bit Memory Cell Designs

B. Register Bank Designs

C. RAM Designs

D. Arithmetic Unit Designs

E. Multiplexer and Decoder Designs

V. Conclusion

VI. References

Acknowledgment

The authors would like to thank...

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.