"Craft your IC"
Learning hardware design in Minecraft[1]

École Normale Supérieure, Paris (ENS Ulm)

Gabriel Doriath Döhler, Constantin Gierczak–Galle

FSiC 2023

July 12, 2023

[1] Reproduce using: nix build github:gabriel-doriath-dohler/Craft-your-IC-FSiC-2023/9a0c01e1148f9503c68a324526491221c34f204e#slides

Story time

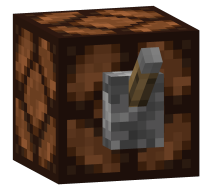
Conway's game of life, by Lucas Vieira - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=101736

Table of Contents

Minecraft Redstone

Redstone = equivalent of electronics

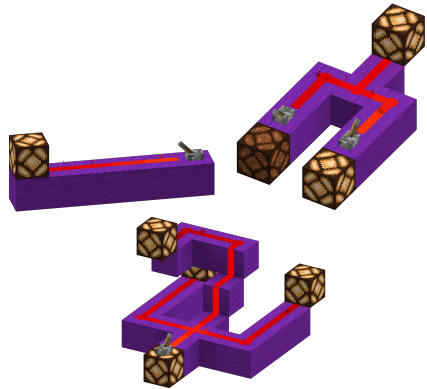
- Lever = player input
- Lamp = output



Minecraft Redstone

Redstone = equivalent of electronics

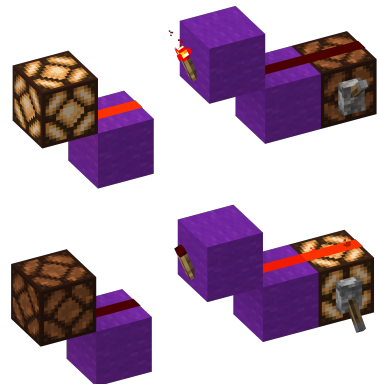
- Lever = player input
- Lamp = output
- Redstone = wires



Minecraft Redstone

Redstone = equivalent of electronics

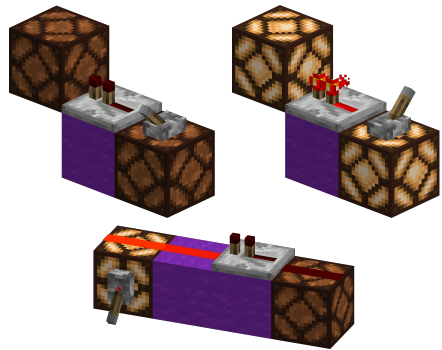
- Lever = player input
- Lamp = output
- Redstone = wires
- Torch = NOT gate
- 1 tick = 0.1 sec



Minecraft Redstone

Redstone = equivalent of electronics

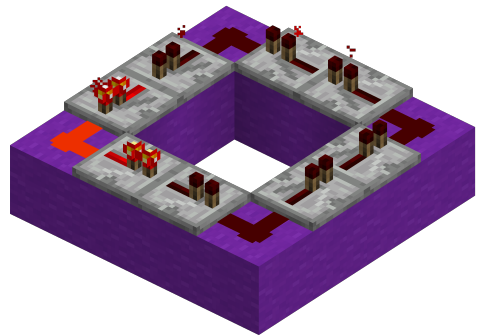
- Lever = player input
- Lamp = output
- Redstone = wires
- Torch = NOT gate
- 1 tick = 0.1 sec
- Repeater
  - delay element
  - diode
  - amplifier
  - memory cell
- Many more redstone components...



Minecraft Redstone

Redstone = equivalent of electronics

- Lever = player input
- Lamp = output
- Redstone = wires
- Torch = NOT gate
- 1 tick = 0.1 sec
- Repeater
  - delay element
  - diode
  - amplifier
  - memory cell
- Many more redstone components...



Minecraft Redstone

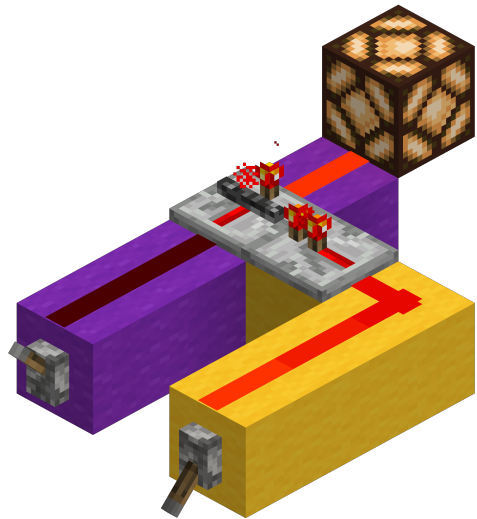
Redstone = equivalent of electronics

- Lever = player input
- Lamp = output
- Redstone = wires
- Torch = NOT gate
- 1 tick = 0.1 sec
- Repeater
  - delay element
  - diode
  - amplifier
  - memory cell
- Many more redstone components...

Minecraft Redstone

Redstone = equivalent of electronics

- Lever = player input
- Lamp = output
- Redstone = wires
- Torch = NOT gate
- 1 tick = 0.1 sec
- Repeater
  - delay element
  - diode
  - amplifier
  - memory cell
- Many more redstone components...



Minecraft Redstone

Redstone = equivalent of electronics

- Lever = player input
- Lamp = output
- Redstone = wires
- Torch = NOT gate
- 1 tick = 0.1 sec
- Repeater
  - delay element
  - diode
  - amplifier
  - memory cell
- Many more redstone components...
- MC = netlist simulator for 4-bit valued signals

Minecraft vs "Real" Hardware: what's different?

A few oddities

- Power strength logic
- Instant diodes

Fewer tooling

- HDL
- Version control
- GTKWave
- Test framework

Constraining physical layer

- Performance => short wires => compact
- Large wires => compact is hard

Roadmap

- Design an ISA
- Implement it in Minecraft
- Build a toolchain around it
- Do some formal verification
- Discover the existing tools and community along the way

Instruction Set Architecture

32-bit instruction words

8-bit registers:

- `%0`: 00000000 – like `RISC-V`
- `%1`: 11111111 – for 1–op `inc dec`, `not`
- `%2-%14`: GP registers
- `%15`: Random register

Instruction Set Architecture

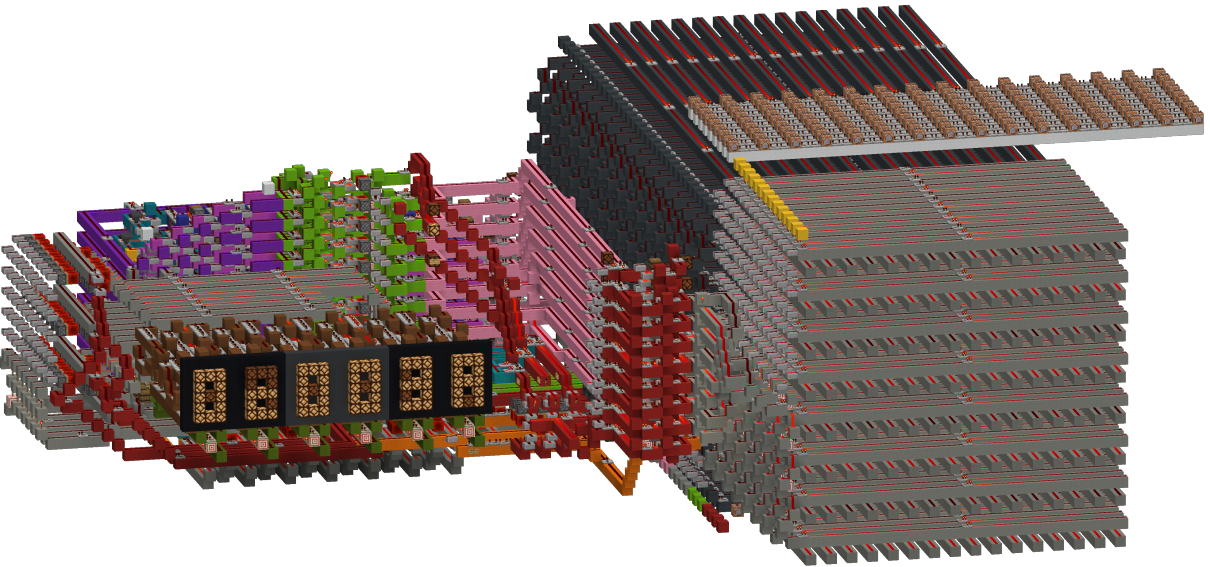
32-bit instruction words

8-bit registers:

- `%0`: 00000000 – like `RISC-V`
- `%1`: 11111111 – for 1–op `inc dec, not`
- `%2-%14`: GP registers
- `%15`: Random register

Instructions:

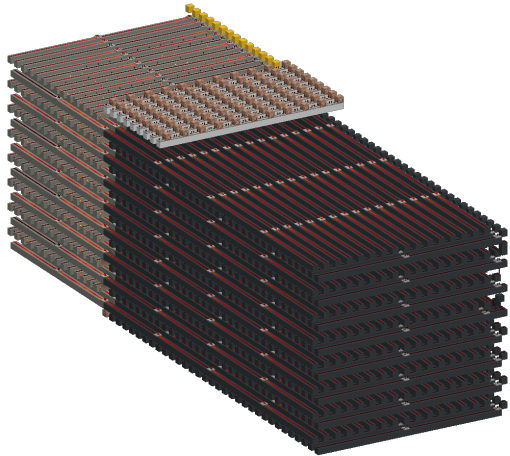
- **Binops**: `add`, `sub`, `or`, `xor`: 3 register operands
- **Jumps**: unconditional or `oveflow/negative/zero`
- `loadi`
- **RAM**: load & store
- `print`: print byte to 7-segment display



ROM

ROM bank

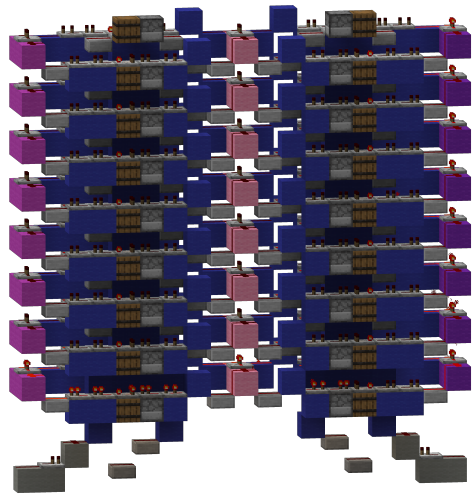
- 128 instructions ROM
- Bits encoded as presence/absence of power blocks in a grid



Registers

Register

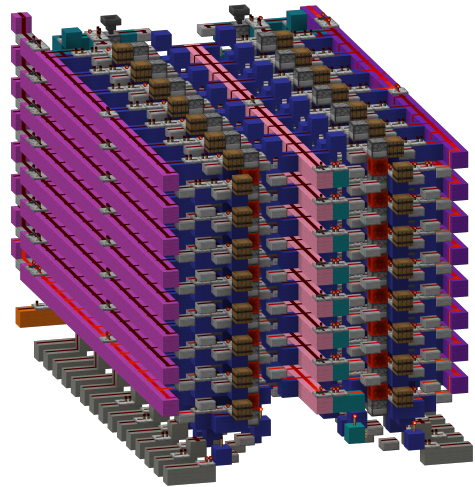
- Tileable design for easier development
- All registers doubled for dual–read
- `%0` and `%1` hard-wired
- `%15` random register uses Minecraft randomizers



Registers

Register file

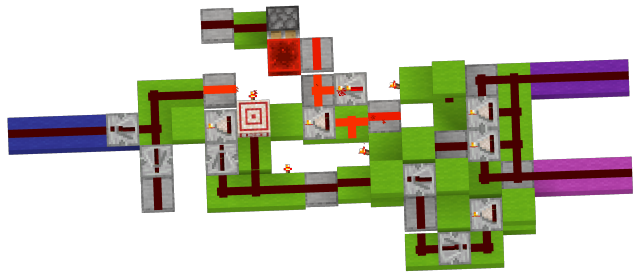
- Tileable design for easier development
- All registers doubled for dual–read
- `%0` and `%1` hard-wired
- `%15` random register uses Minecraft randomizers



ALU

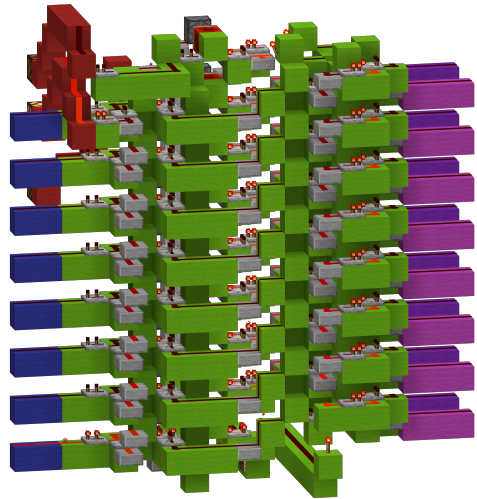
ALU slice

- Tileable design
- Compact for speed.
- Supports all common simple arithmetic and logic operations



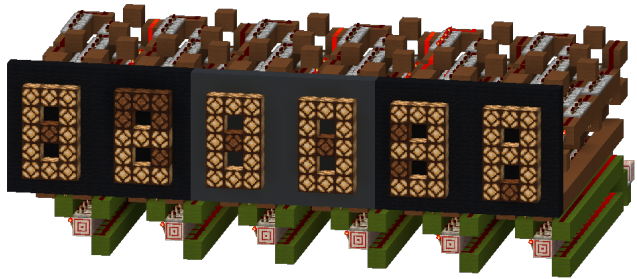
ALU

- Tileable design
- Compact for speed.
- Supports all common simple arithmetic and logic operations



I/O

- 3 individually-addressable displays



7-segment hex display

Some stats

Performance:

- In-order, single core
- 0.1Hz clock (on normal game speed)
- 1 IpC

The actual circuit

- 70k blocks
- 3.6k registers/repeaters
- 30k unit lengths of wiring

Toolchain

- Assembler

```
.sec
        add %5 %2 %2
        xor %2 %1 %9
        add %5 %9 %9
        add %6 %9 %0
        jneg min
        jmp print_sec
.min
        loadi $0 %2
        add %5 %3 %3
        xor %3 %1 %9
    ...
```

Toolchain

- Assembler

```
.sec
        add %5 %2 %2
        xor %2 %1 %9
        add %5 %9 %9
        add %6 %9 %0
        jneg min
        jmp print_sec
.min
        loadi $0 %2
        add %5 %3 %3
        xor %3 %1 %9
    ...
```

- Linker

Resolves label references and computes jumps

```
add %5 %2 %2
xor %2 %1 %9
add %5 %9 %9
add %6 %9 %0
jneg 0x10
jmp 0x03
...
```

Toolchain

- Assembler

```
.sec
        add %5 %2 %2
        xor %2 %1 %9
        add %5 %9 %9
        add %6 %9 %0
        jneg min
        jmp print_sec
.min
        loadi $0 %2
        add %5 %3 %3
        xor %3 %1 %9
    ...
```

- Linker

Resolves label references and computes jumps

```
add %5 %2 %2
xor %2 %1 %9
add %5 %9 %9
add %6 %9 %0
jneg 0x10
jmp 0x03
...
```

- ROM generator

```
setblock -272 197 -1402 torch
setblock -270 197 -1402 torch
setblock -268 197 -1402 air
...
```

The Demo

Link on `https://wiki.f-si.org/index.php?title=Learning_hardware_design_in_the_video_game_Minecraft`

The community

Other CPU designs

Open **R**edstone **E**ngineers `https://openredstone.org/`

- Shared circuit designs
- Collaborative building
- “school” system
- Guides and resources
- Guest lecture at Harvard
  `https://www.youtube.com/watch?v=EcGExRlHVzg`

The community

Open-source tools

In addition to shared circuit designs and techniques:

- Alternative high-performance Minecraft servers (`https://github.com/MCHPR/MCHPRS`)
- Game mods for debugging
- Community-organization tools

All in all, very similar to traditional FOSS comunities.

Community showcase

`https://github.com/sammyuri/chungus-2-assembler`

1Hz, 4-stage pipeline, HW `mult` & `div`, instruction and data cache, HW callstack

Toolchain: MIT license

Formal verification

- Existing tools do not know Minecraft
- Define Minecraft's semantics

First attempt: k-induction with Haskell

- Export the matrix of blocks
- Graph representation
- Define Minecraft's semantics
- DSL to express the goal
- Z3 SMT solving

Subset of the ALU verified!
CPU parts list:

- 7k redstone elements
- 30k redstone wires

so, probably too slow anyways...

On going second attempt: model checking

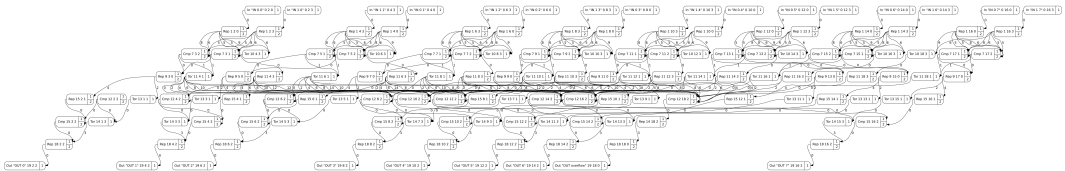


Figure: CCA graph

Each node is a state machine.

- Model checking
- Lean 4 DSL to write semantics and goals

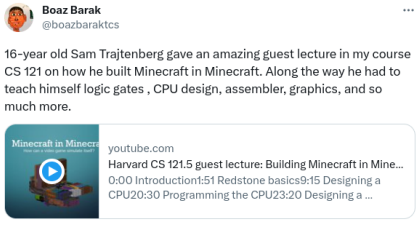
Learning Outcomes

How would you introduce hardware design to a high-schooler?

- FPGA + yosys + nexpnr + git + verilog => blinky
- Minecraft with interactive instant feedback

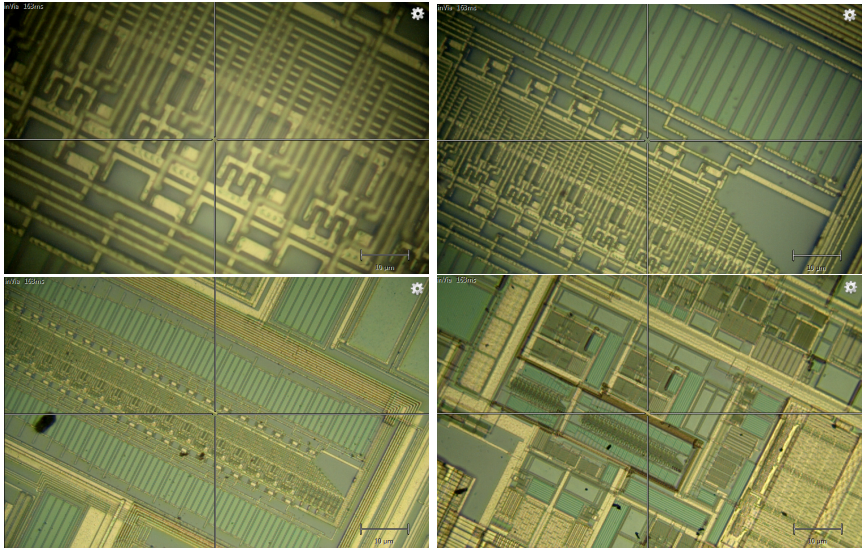
Just a silly video game? Think again:

- Open Redstone Engineers testimonials



- Learned from **first principles**: CPU design, ISA choice, debugging, assembly, linking, and formal verification

From Minecraft to "real" hardware



Acknowledgement

Special thanks to friends and collaborators who helped us on this project, including:

- Hadrien Barral
- Rémy Citerin

A big thank you to the FSiC committee for hosting our talk!

FSiC2023
Free Silicon Conference

Q/A

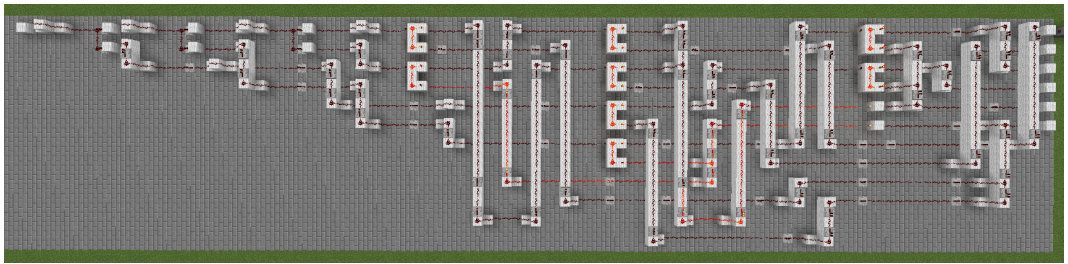
- HW design in Minecraft
- ISA and implementation
- ASM-to-ROM toolchain
- Formal verification efforts
- FOSS Minecraft engineers community

Craft your IC project wiki page

`https://wiki.f-si.org/index.php?title=Learning_hardware_design_in_the_video_game_Minecraft`

Gabriel Doriath Döhler, Constantin Gierczak–Galle

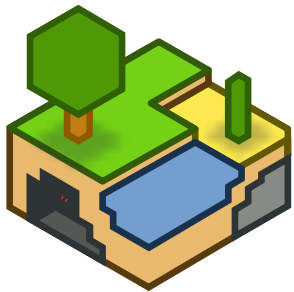
Minecraft HDL efforts



https://github.com/itsFrank/MinecraftHDL

Minetest

A FOSS Minecraft clone, with strong education goals. Licensed under LGPL2.1



https://www.minetest.net