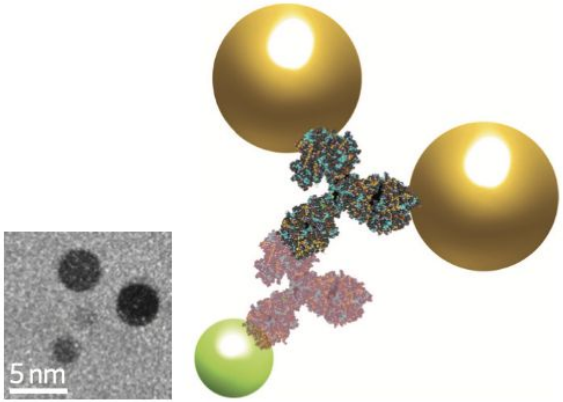
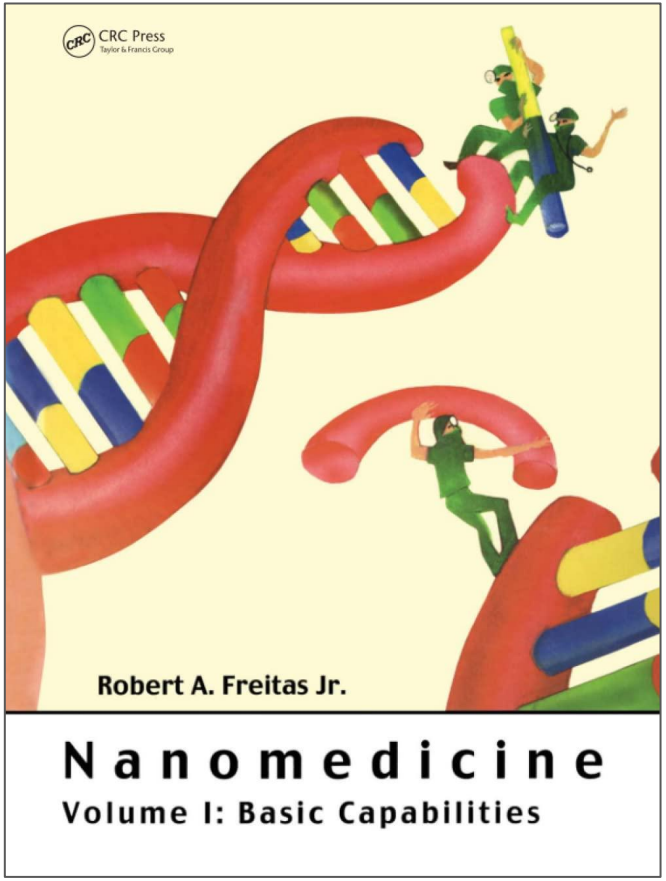
Nanomechanical Computing

Philip Turner

What is the smallest computer that existing technology can build?



Protein Transistor



Nanomedicine (1999)

Nanomedicine, Volume I: Basic Capabilities

TABLE OF CONTENTS

Dedication ..... Local Copy (HTML)

Table of Contents ..... Local Copy (HTML)

List of Figures ..... Local Copy (HTML)

List of Tables ..... Local Copy (HTML)

Foreword by K. Eric Drexler ..... Local Copy (HTML)

Preface and Acknowledgements ..... Local Copy (HTML)

Opening Quotations ..... Local Copy (HTML)

Chapter 1. The Prospect of Nanomedicine

1.1 A Noble Enterprise ..... Local Copy (HTML)

1.2 Current Medical Practice ..... Local Copy (HTML)

1.2.1 The Evolution of Scientific Medicine ..... Local Copy (HTML)

1.2.1.1 Prehistoric Medicine ..... Local Copy (HTML)

1.2.1.2 Ancient Mesopotamian Medicine ..... Local Copy (HTML)

1.2.1.3 Ancient Egyptian Medicine ..... Local Copy (HTML)

1.2.1.4 Ancient Greek Medicine ..... Local Copy (HTML)

1.2.1.5 Ancient Alexandrian Medicine ..... Local Copy (HTML)

1.2.1.6 Ancient Roman Medicine ..... Local Copy (HTML)

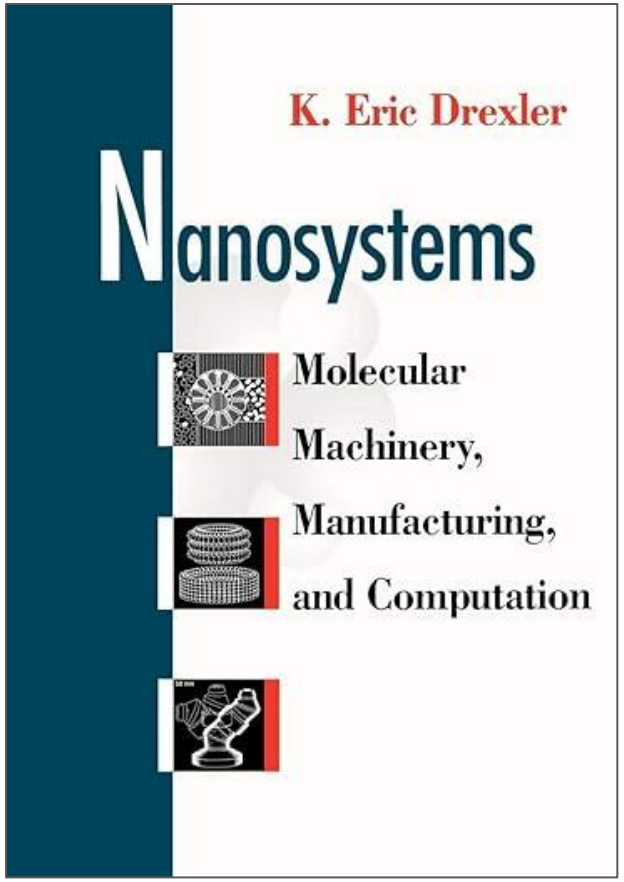
1.2.1.7 Medicine in the Middle Ages ..... Local Copy (HTML)

1.2.1.8 Renaissance and Pre-Modern Medicine ..... Local Copy (HTML)

1.2.1.9 Fully Invasive Surgery ..... Local Copy (HTML)

1.2.1.9.1 Anatomy ..... Local Copy (HTML)

Open Access Website



Nanosystems (1992)

Molecular Machinery and Manufacturing with Applications to Computation

by
K. Eric Drexler

Submitted to the Media Arts and Sciences Section,
School of Architecture and Planning, August 9, 1991,
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in an Interdepartmental Program
in the field of Molecular Nanotechnology

Abstract

Studies were conducted to assemble the analytical tools necessary for the design and modeling of mechanical systems with molecular-precision moving parts of nanometer scale. These analytical tools were then applied to the design of systems capable of computation and of molecular-precision manufacturing.

Part I draws on classical, statistical, and quantum mechanics, together with empirical force-field models developed in chemistry, to select and develop a set of practical models describing the key engineering properties of nanometer-scale mechanical systems. These properties include potential energy functions; positional uncertainties resulting from the combined effects of quantum mechanics and thermal excitation; transition rates and damage rates resulting from these same combined effects and from ultraviolet and ionizing radiation; and energy dissipation resulting from acoustic radiation, phonon scattering, thermoelastic effects, phonon viscosity, and transitions occurring between potential wells in disequilibrium states. Part I concludes with an analysis of the capabilities of mechanosynthesis, that is, chemical synthesis directed by devices capable of moving and positioning reactive moieties with atomic-scale precision.

Part II draws on the tools developed in Part I, analyzing the mechanical properties of representative structural components and of mobile nanomechanical components such as gears and bearings. Using components and devices with these properties, nanomechanical computational systems (comprising logic gates, signal transmission elements, registers, I/O mechanisms, power supply, power distribution, and clocking) are described and analyzed, yielding estimated component densities $> 10^{19}/cm^3$ and power dissipation levels $< 10^{-9}$ those of current transistor-logic devices. Part II concludes with an analysis of the capabilities of molecular manufacturing systems based on mechanosynthesis performed by nanomechanical systems, concluding that assembly cycle times of $\sim 10^{-6}$ s will commonly be compatible with error rates of $< 10^{-12}$.

Part III summarizes a study of the feasibility of performing positionally-controlled chemical synthesis using a modified atomic force microscope, concluding that assembly cycle times of ~ 1 s and error rates of $\sim 10^{-5}$ can be anticipated from devices based on combinations of existing molecules and mechanisms. Molecular assembly mechanisms in this class can provide one of several paths forward from our present capabilities toward more advanced molecular technologies of the sort assumed in the body of the present work.

Thesis supervisor: Marvin L. Minsky
Professor of Computer Science and Engineering
Toshiba Professor of Media Arts and Sciences

Open Access Thesis

Molecular Link Logic

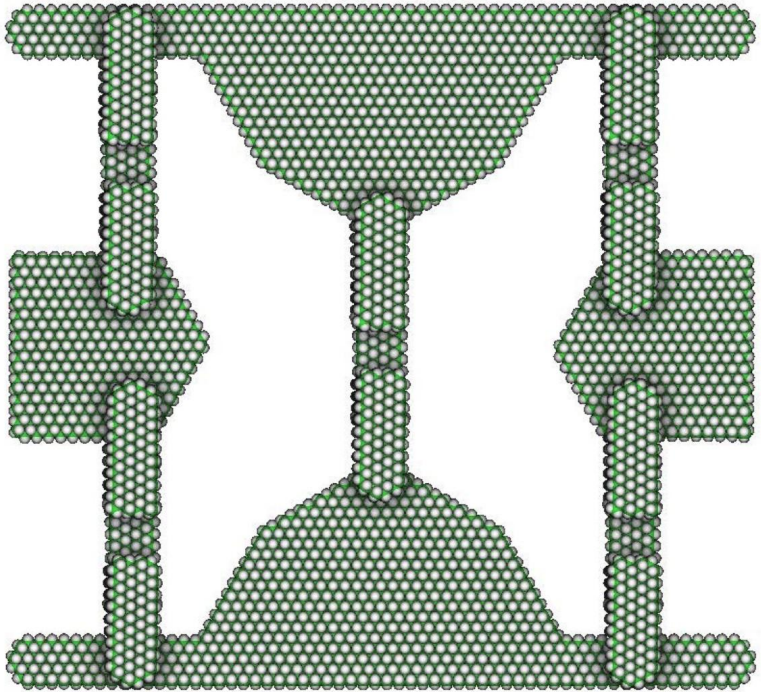


Figure 15: A molecular model of a diamond-based lock, top view.
Hydrogens are white, Carbons are green.

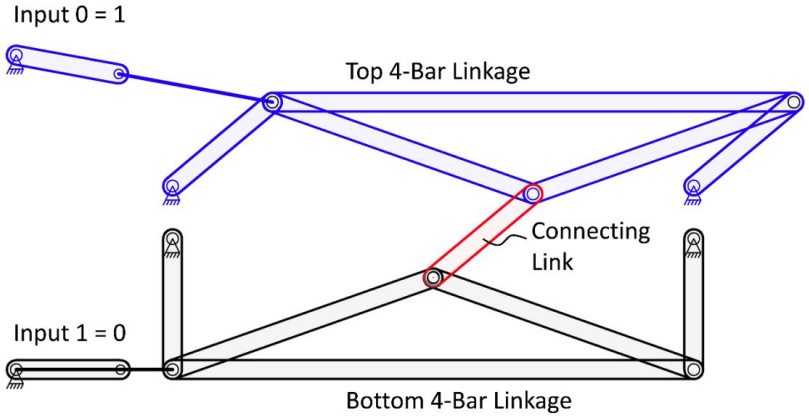


Figure 6: A lock in the (1,0) position

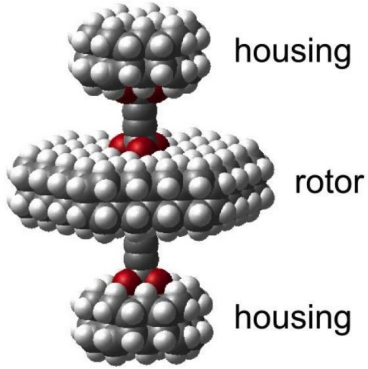


Figure 14: A rotary joint using two atomic bonds

Molecular Link Logic: Private Intellectual Property

Mechanical computing systems
Patent number: 10481866
Abstract: Systems and methods for creating mechanical computing mechanisms and Turing-complete systems which include combinatorial logic and sequential logic, and are energy-efficient.
Type: Grant
Filed: December 31, 2015
Date of Patent: November 19, 2019
Assignee: CBN Nano Technologies inc.
Inventors: Ralph C. Merkle, Robert A. Freitas, James Ryley, Matthew Moses, Tad Hogg

Rod Logic: Open Alternative

Schematic from Drexler's thesis

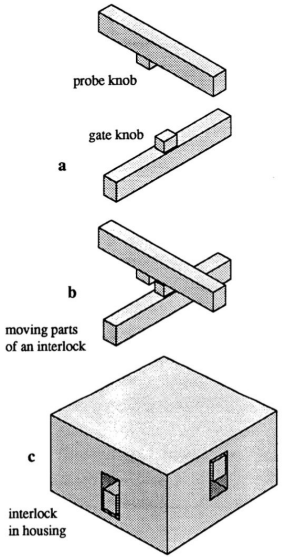
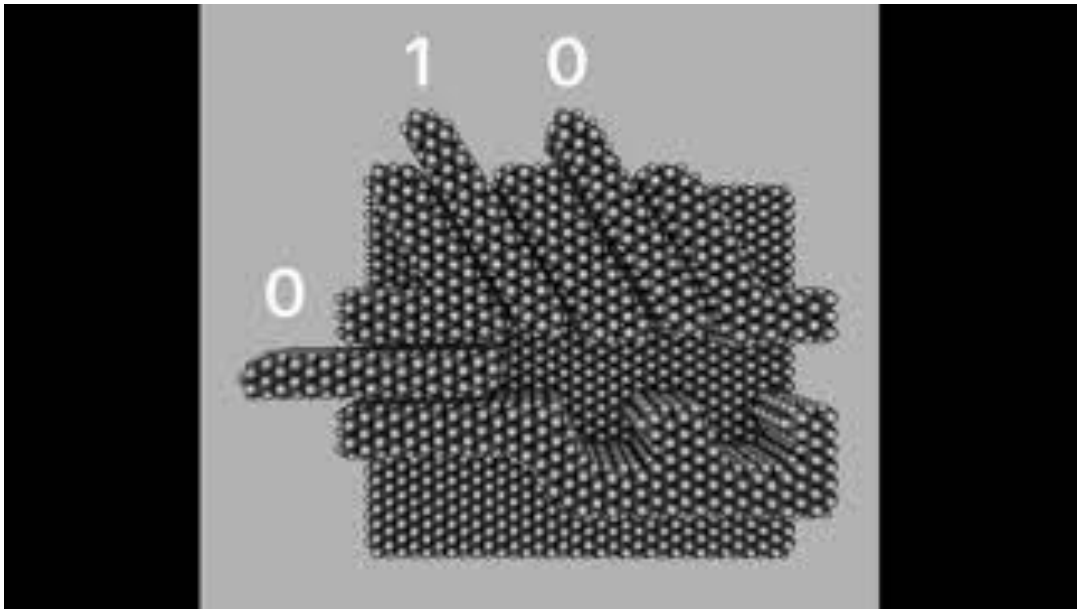


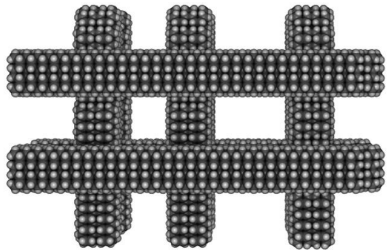
Figure 11.2. Components of an interlock: input rod with gate knob and output rod with probe knob, separated (a), in their working positions (b), and constrained by a housing structure (c).

200,000 atoms/gate → 2,000 atoms/gate

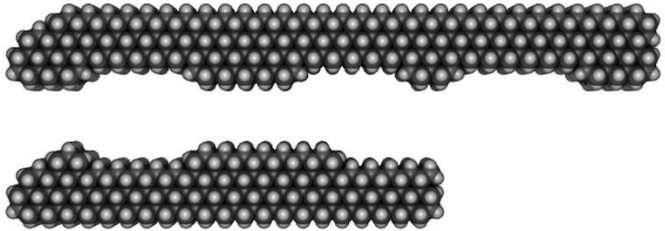


Rod Logic: Open Alternative

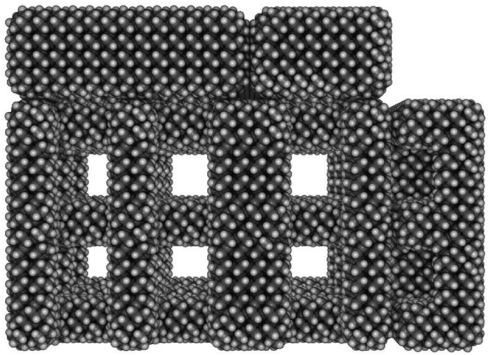
rods



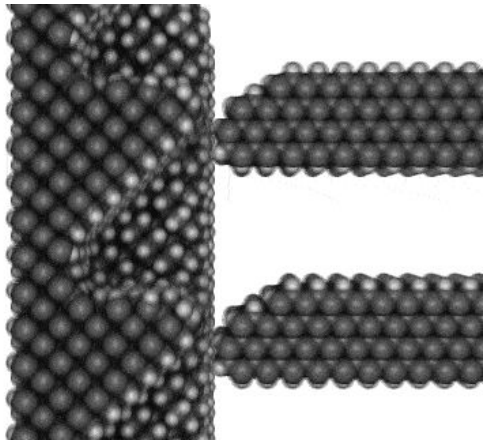
patterns



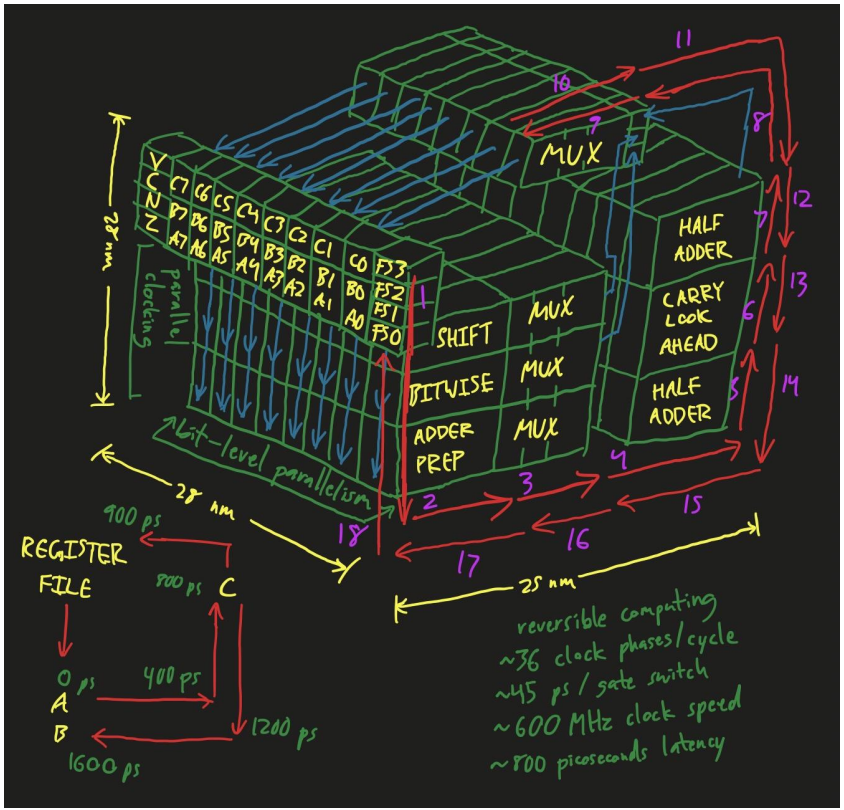
housing



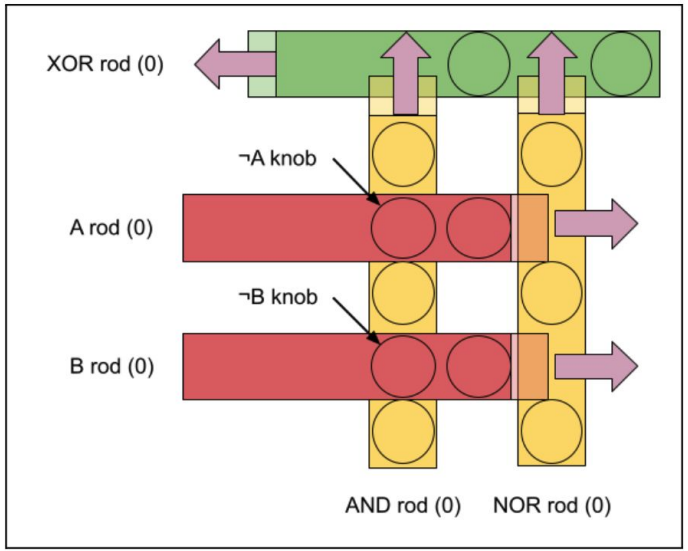
drive wall



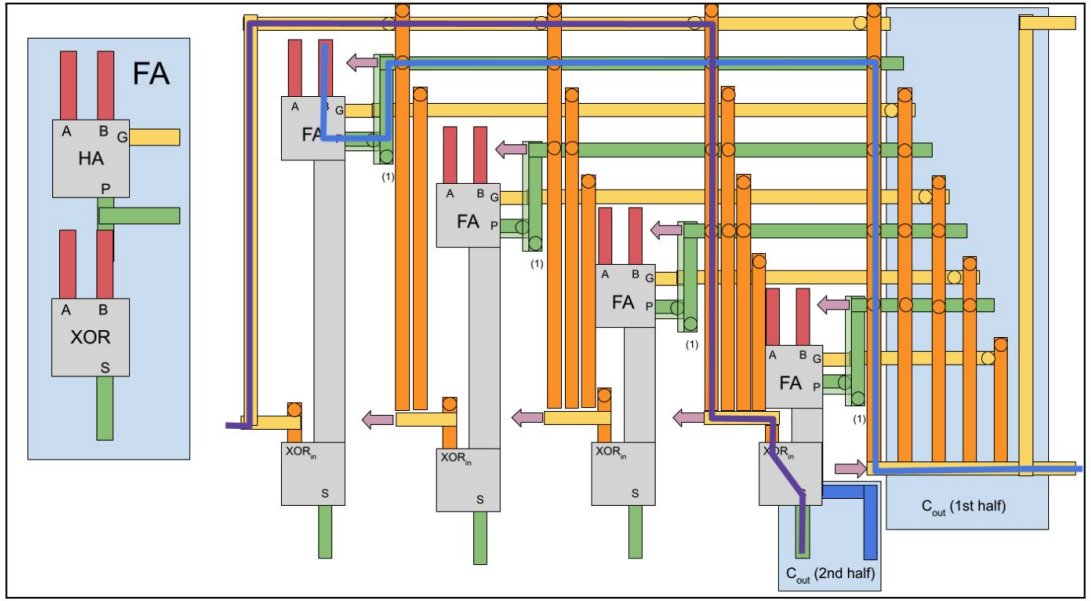
What would a rod logic computer look like?



Critical element of a computer: 4-bit CLA



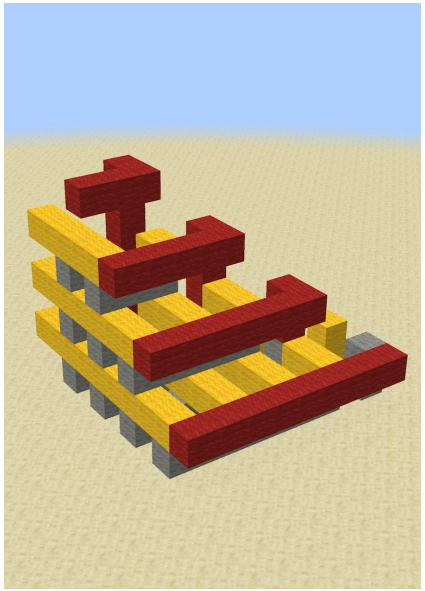
Half Adder



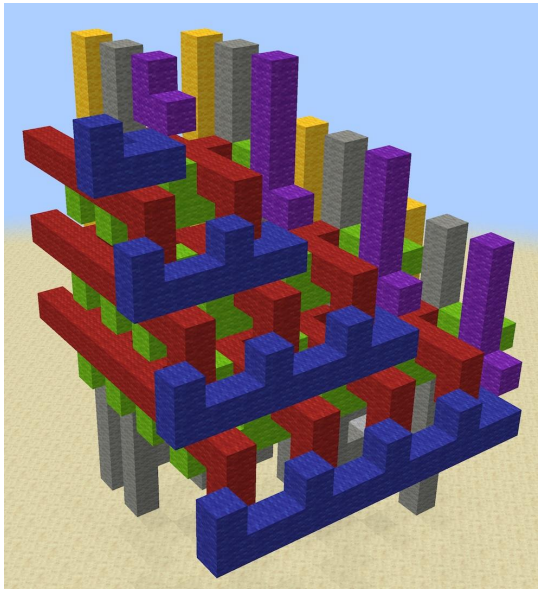
Half of an 8-bit nanomechanical carry lookahead adder.

Attempts to draft in Minecraft

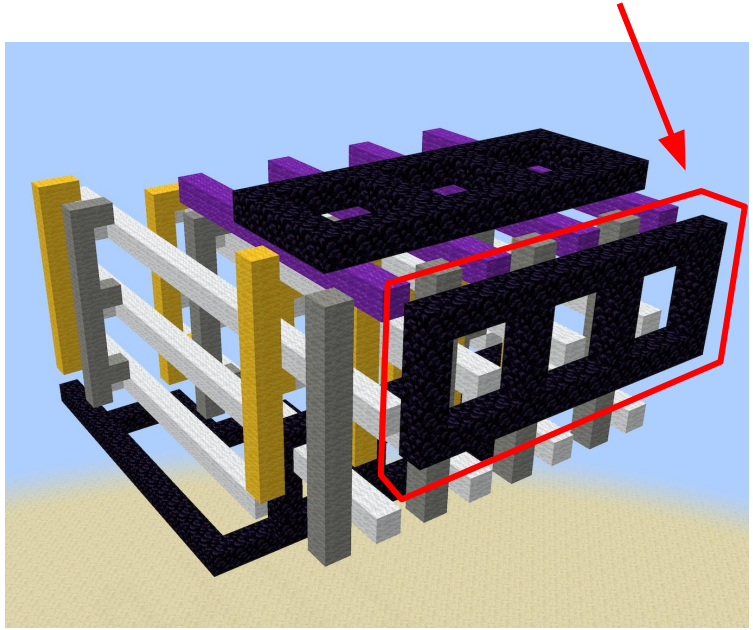
drive “wall” actuating 8 rods at once



Draft 1



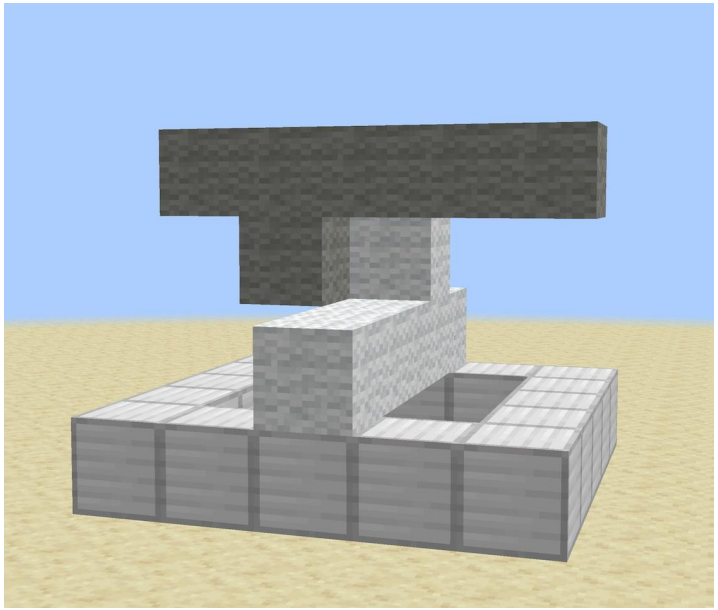
Draft 2



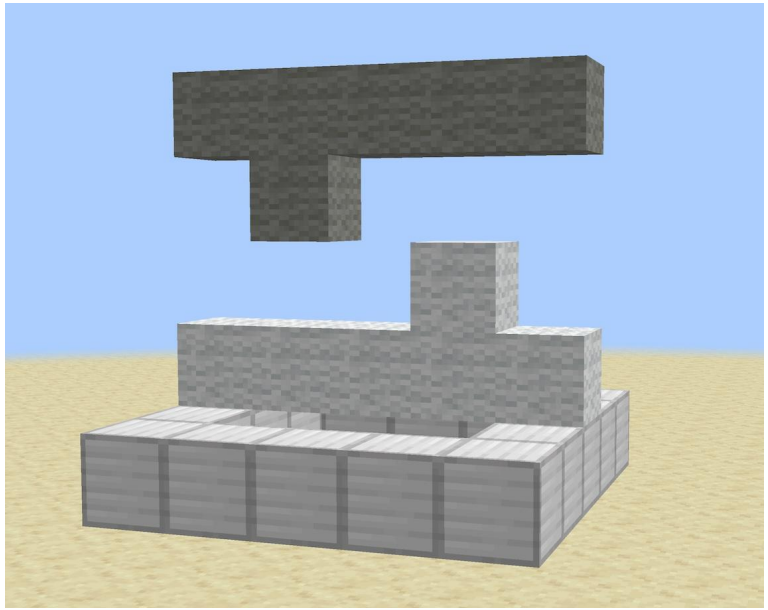
Draft 3

Formalizing Rod Logic

switch – fundamental building block



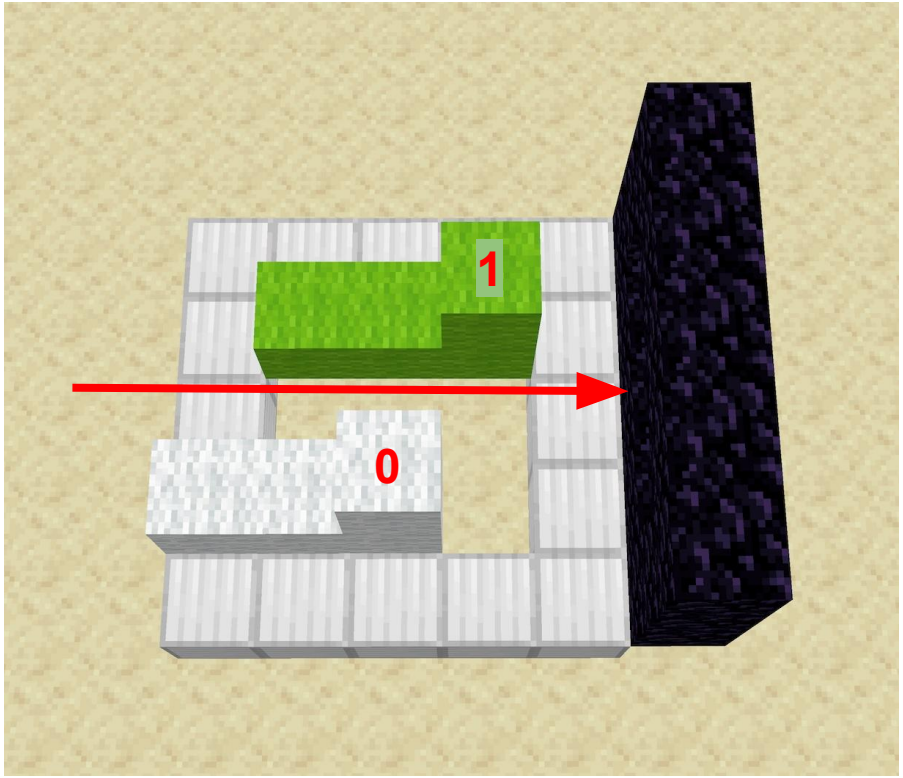
pattern – knobs, inversion of signals



Formalizing Rod Logic

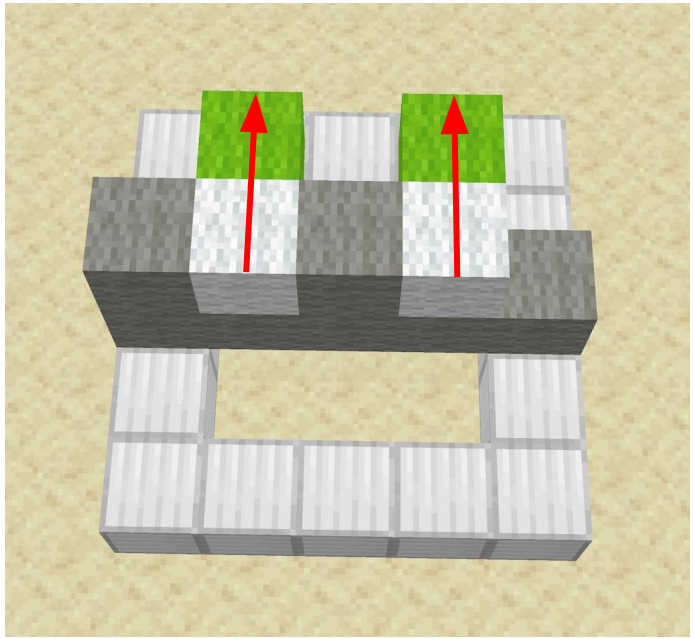
Rules:

- 0 = blocked
- 1 = mobile
- Can only invert inputs
- Cannot invert outputs

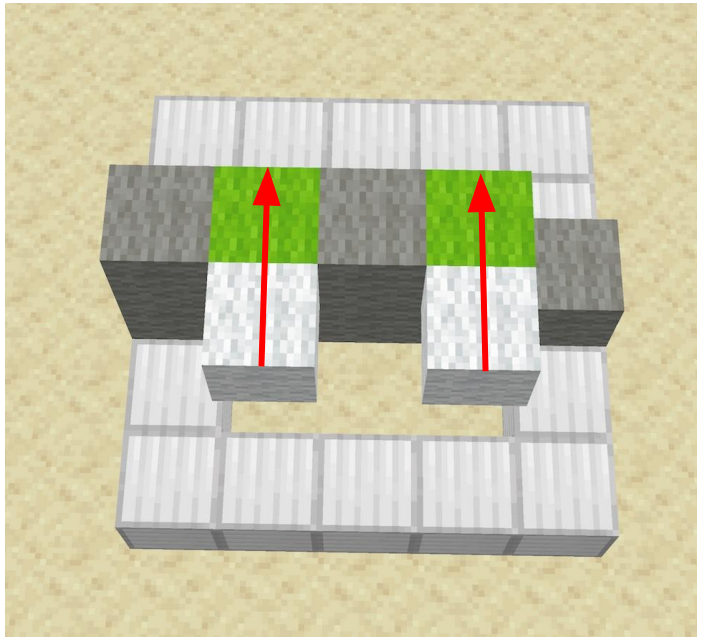


Formalizing Rod Logic

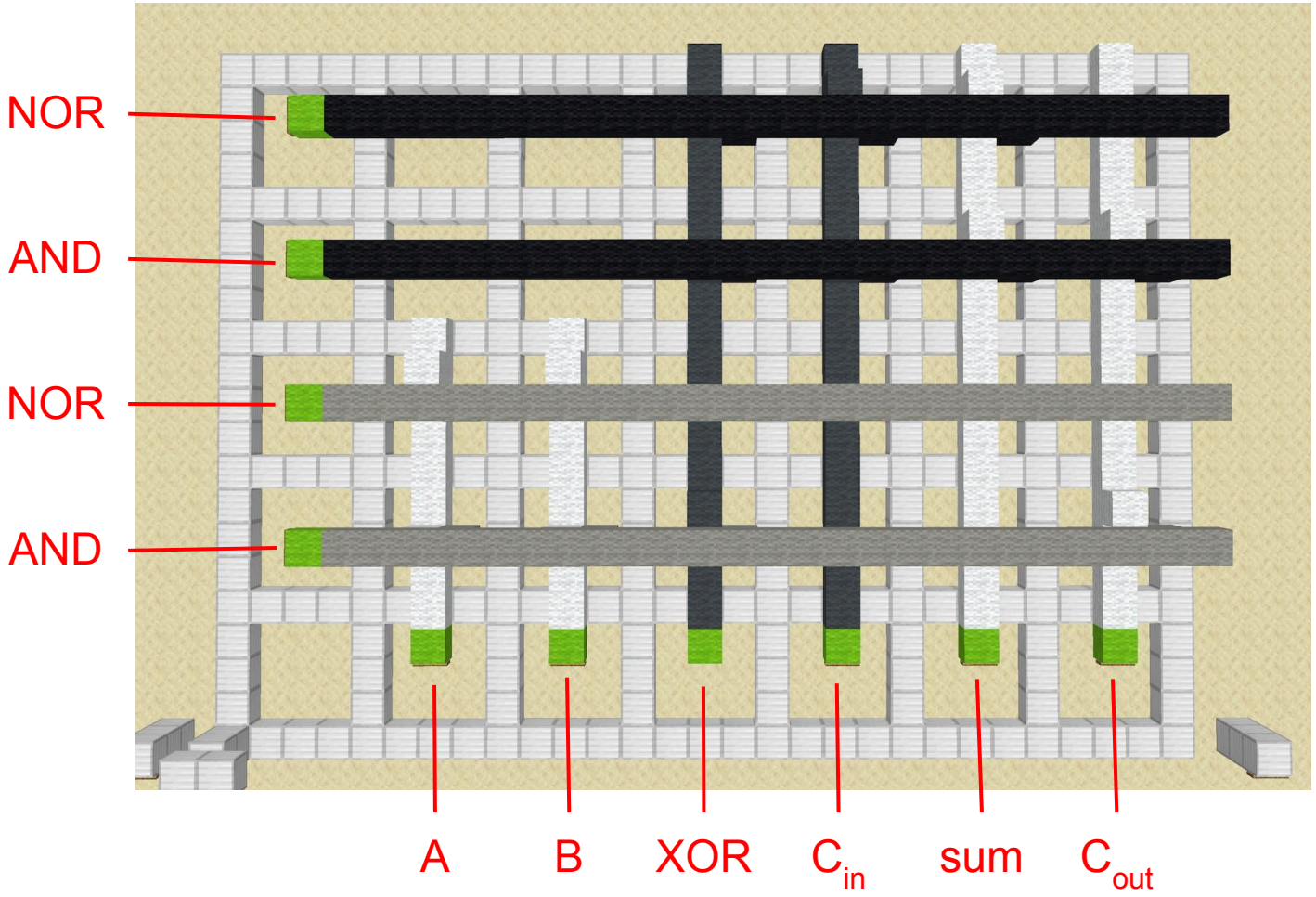
AND gate



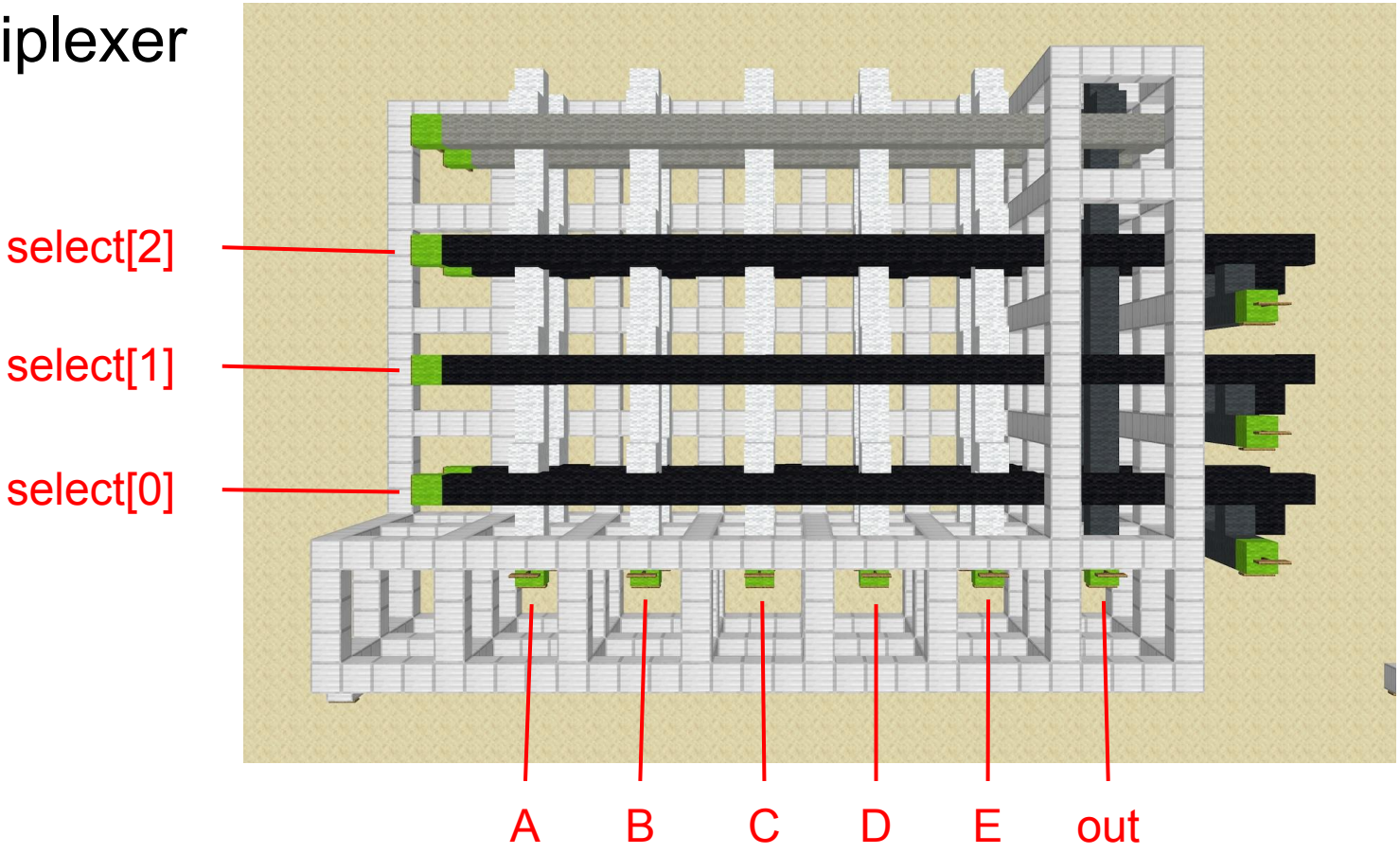
NOR gate



Full Adder



5-to-1 Multiplexer



2-Bit Logic Unit

operation select

in[1]

in[0]

out[1]

out[0]

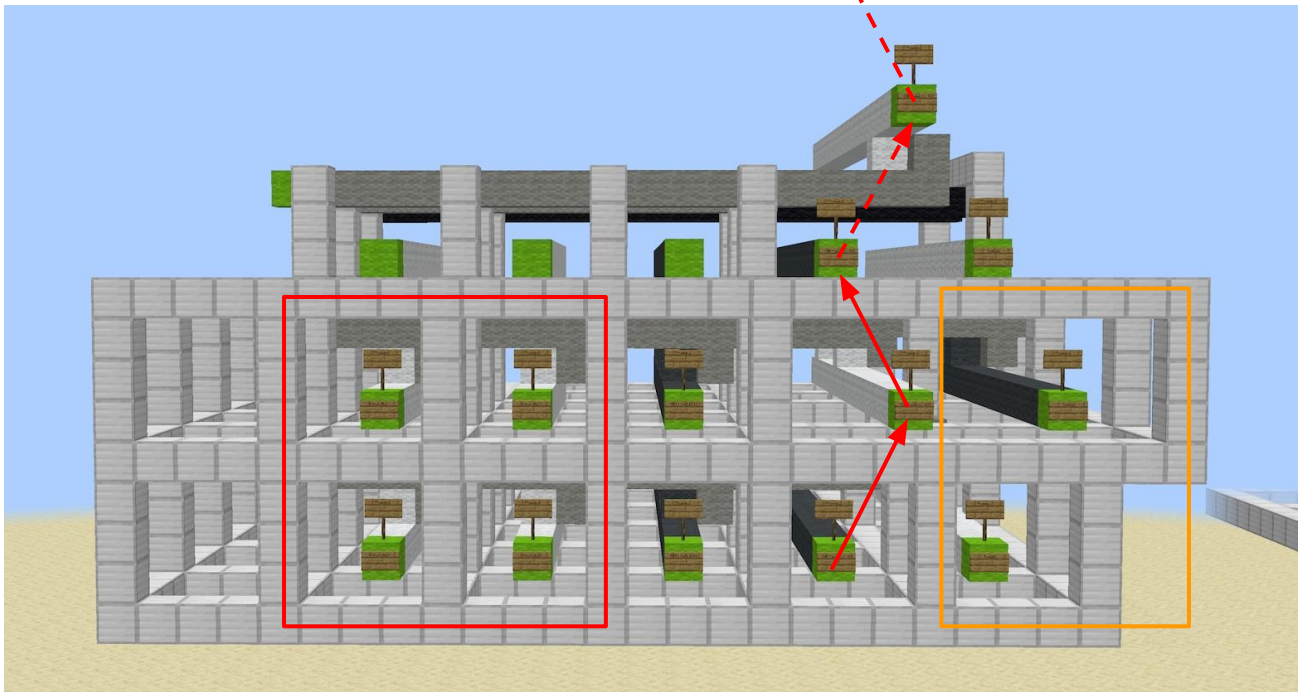
2-Bit Ripple Carry Adder

bit 3

bit 2

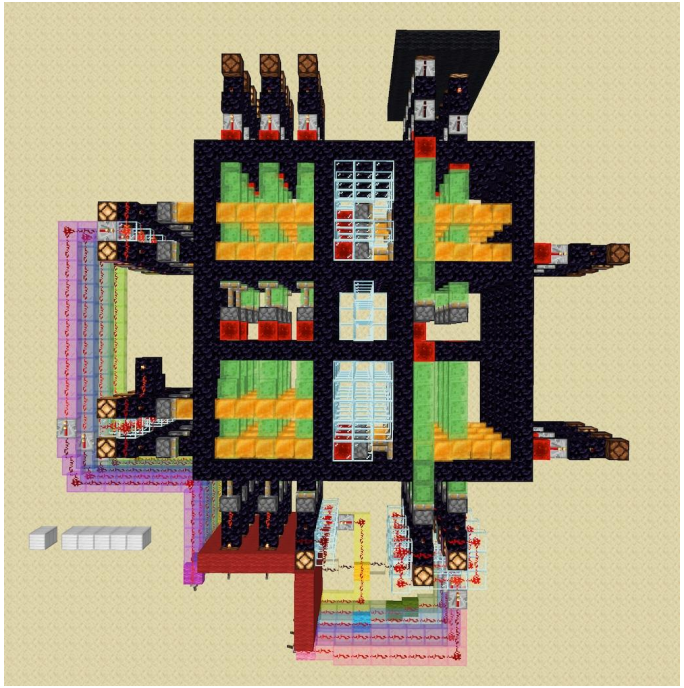
bit 1

bit 0

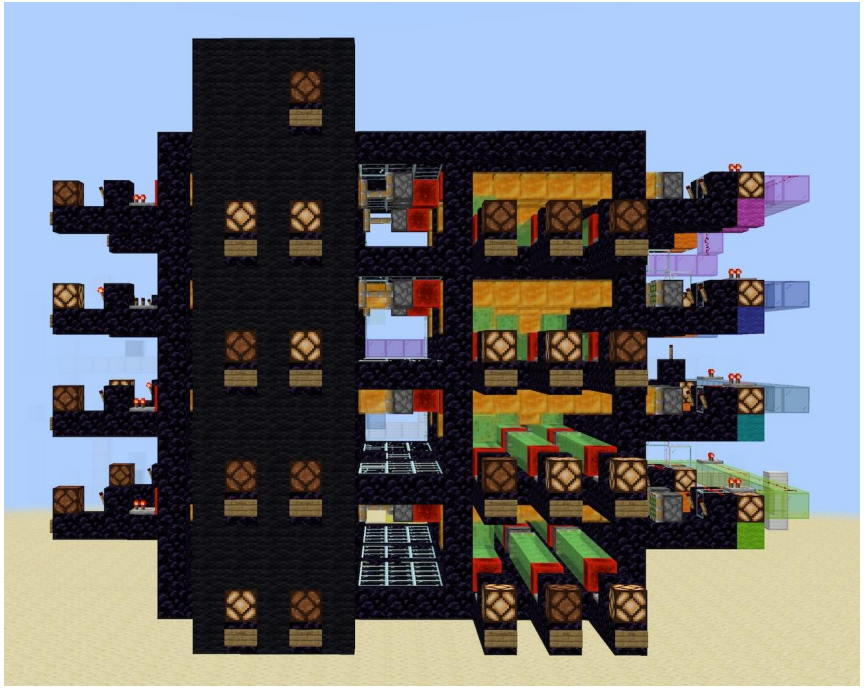


4-Bit Ripple Carry Adder

Aerial view



Output interface



4-Bit Ripple Carry Adder

Achievements:

- First reversible 4-bit adder
- Forward and reverse clock cycle
- Easy to debug

Issues:

- 10 distinct clock phases

Minecraft Carry Lookahead Adder

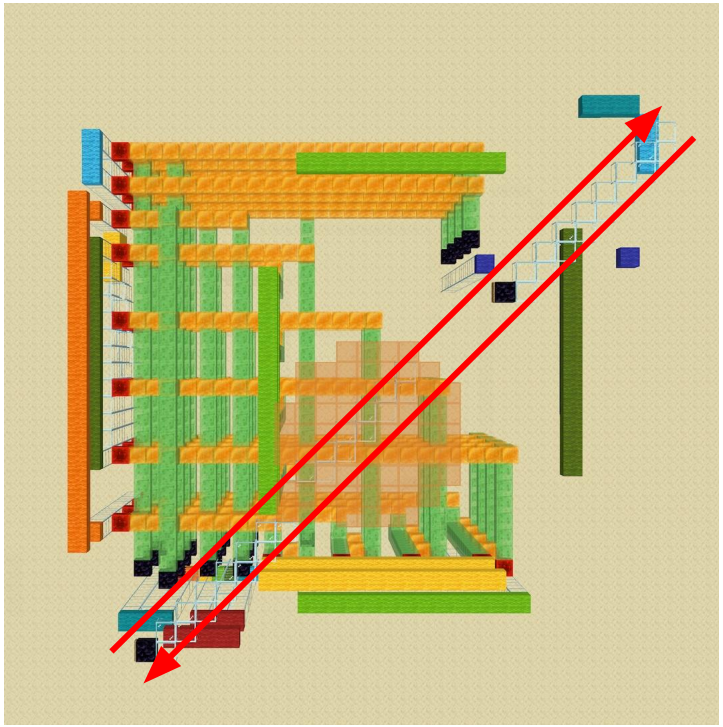
6 more design iterations

generate unit

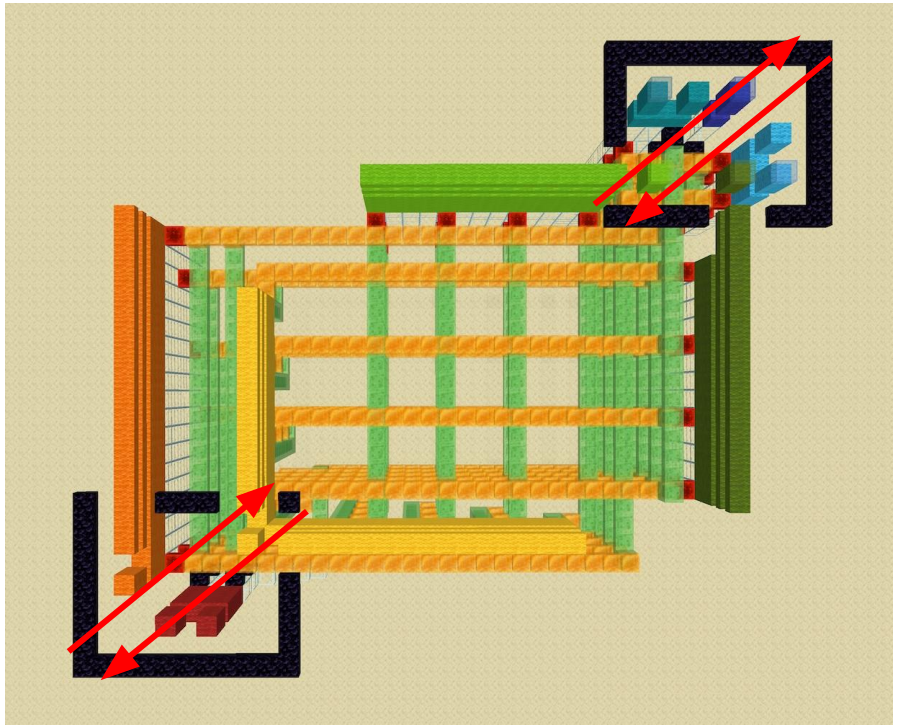
propagate unit

Minecraft Carry Lookahead Adder

clocking sequence



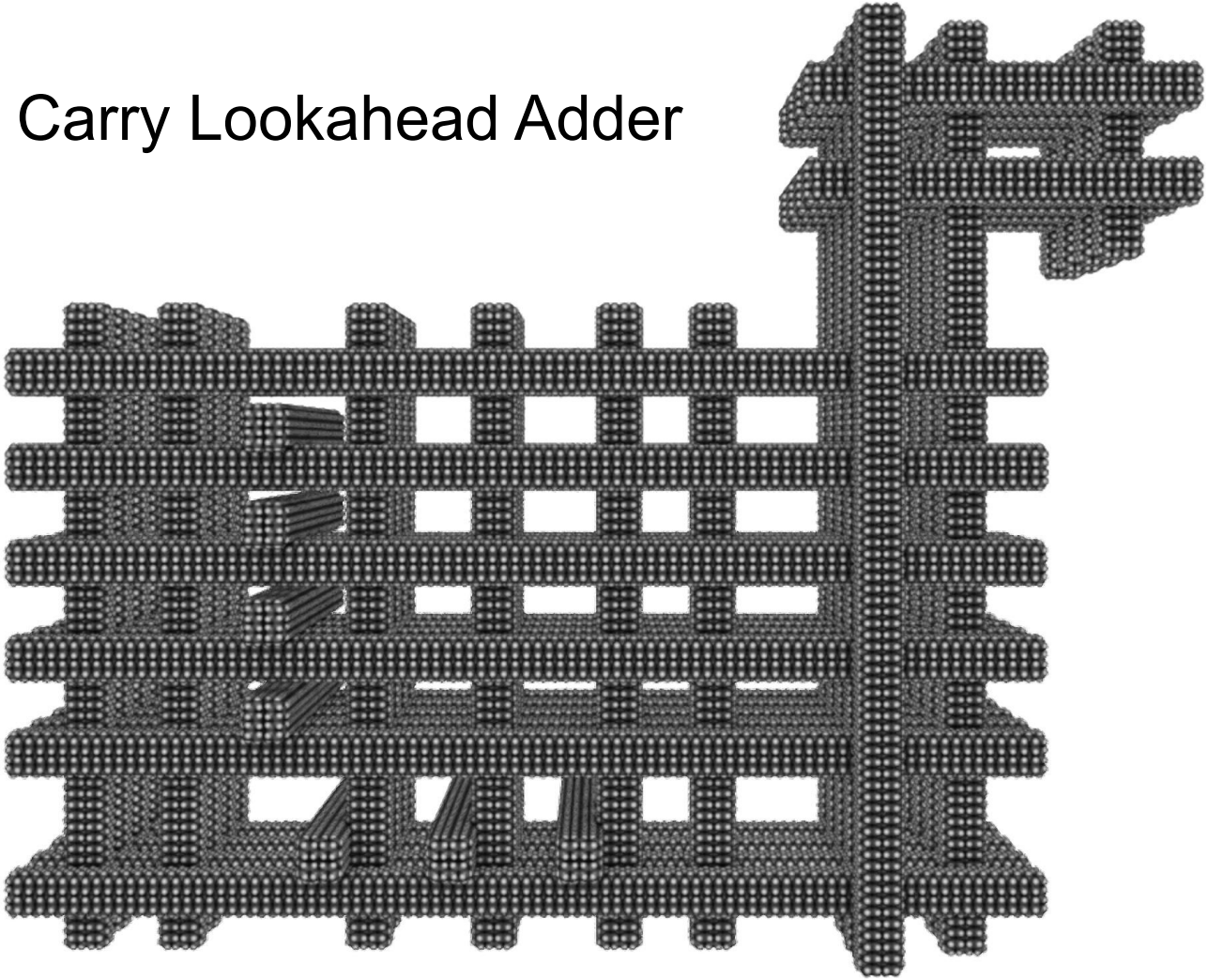
Draft 8

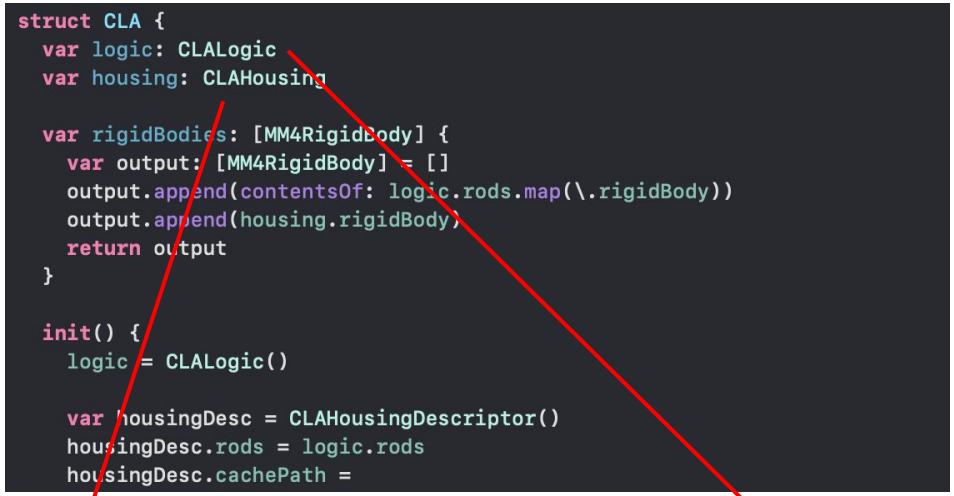


Draft 9

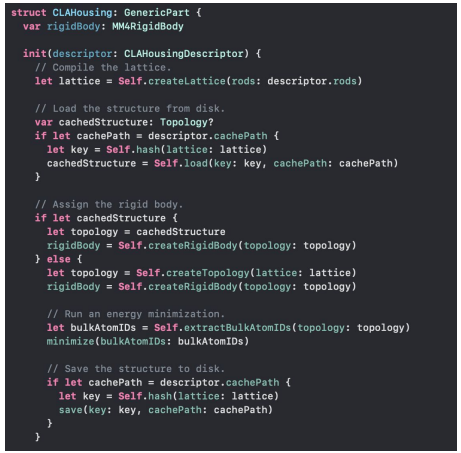
Nanomechanical Carry Lookahead Adder

Draft 13

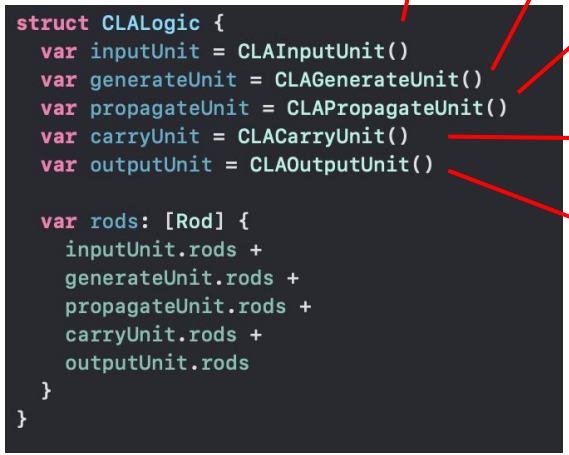




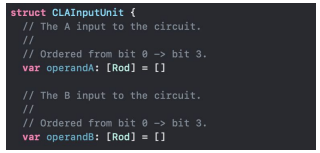
```
struct CLA {
  var logic: CLALogic
  var housing: CLAHousing

  var rigidBodies: [MM4RigidBody] {
    var output: [MM4RigidBody] = []
    output.append(contentsOf: logic.rods.map(\.rigidBody))
    output.append(housing.rigidBody)
    return output
  }

  init() {
    logic = CLALogic()

    var housingDesc = CLAHousingDescriptor()
    housingDesc.rods = logic.rods
    housingDesc.cachePath =
```



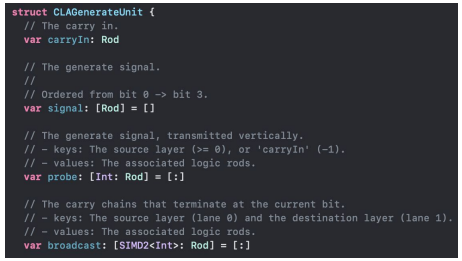
```
struct CLAHousing: GenericPart {
  var rigidBody: MM4RigidBody

  init(descriptor: CLAHousingDescriptor) {
    // Compile the lattice.
    let lattice = Self.createLattice(rods: descriptor.rods)

    // Load the structure from disk.
    var cachedStructure: Topology?
    if let cachePath = descriptor.cachePath {
      let key = Self.hash(lattice: lattice)
      cachedStructure = Self.load(key: key, cachePath: cachePath)
    }

    // Assign the rigid body.
    if let cachedStructure {
      let topology = cachedStructure
      rigidBody = Self.createRigidBody(topology: topology)
    } else {
      let topology = Self.createTopology(lattice: lattice)
      rigidBody = Self.createRigidBody(topology: topology)

      // Run an energy minimization.
      let bulkAtomIDs = Self.extractBulkAtomIDs(topology: topology)
      minimize(bulkAtomIDs: bulkAtomIDs)

      // Save the structure to disk.
      if let cachePath = descriptor.cachePath {
        let key = Self.hash(lattice: lattice)
        save(key: key, cachePath: cachePath)
      }
    }
```



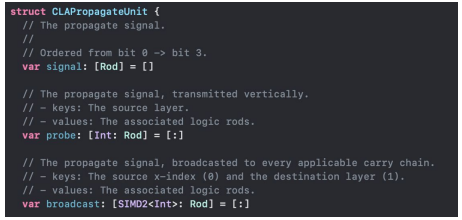
```
struct CLALogic {
  var inputUnit = CLAInputUnit()
  var generateUnit = CLAGenerateUnit()
  var propagateUnit = CLAPropagateUnit()
  var carryUnit = CLACarryUnit()
  var outputUnit = CLAOutputUnit()

  var rods: [Rod] {
    inputUnit.rods +
    generateUnit.rods +
    propagateUnit.rods +
    carryUnit.rods +
    outputUnit.rods
  }
}
```



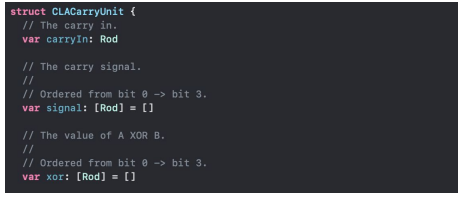
```
struct CLAInputUnit {
  // The A input to the circuit.
  //
  // Ordered from bit 0 -> bit 3.
  var operandA: [Rod] = []

  // The B input to the circuit.
  //
  // Ordered from bit 0 -> bit 3.
  var operandB: [Rod] = []
```



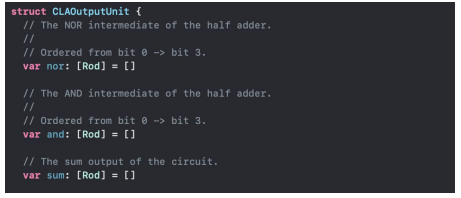
```
struct CLAGenerateUnit {
  // The carry in.
  var carryIn: Rod

  // The generate signal.
  //
  // Ordered from bit 0 -> bit 3.
  var signal: [Rod] = []

  // The generate signal, transmitted vertically.
  // - keys: The source layer (>= 0), or 'carryIn' (-1).
  // - values: The associated logic rods.
  var probe: [Int: Rod] = [:]

  // The carry chains that terminate at the current bit.
  // - keys: The source layer (lane 0) and the destination layer (lane 1).
  // - values: The associated logic rods.
  var broadcast: [SIMD2<Int>: Rod] = [:]
```



```
struct CLAPropagateUnit {
  // The propagate signal.
  //
  // Ordered from bit 0 -> bit 3.
  var signal: [Rod] = []

  // The propagate signal, transmitted vertically.
  // - keys: The source layer.
  // - values: The associated logic rods.
  var probe: [Int: Rod] = [:]

  // The propagate signal, broadcasted to every applicable carry chain.
  // - keys: The source x-index (0) and the destination layer (1).
  // - values: The associated logic rods.
  var broadcast: [SIMD2<Int>: Rod] = [:]
```

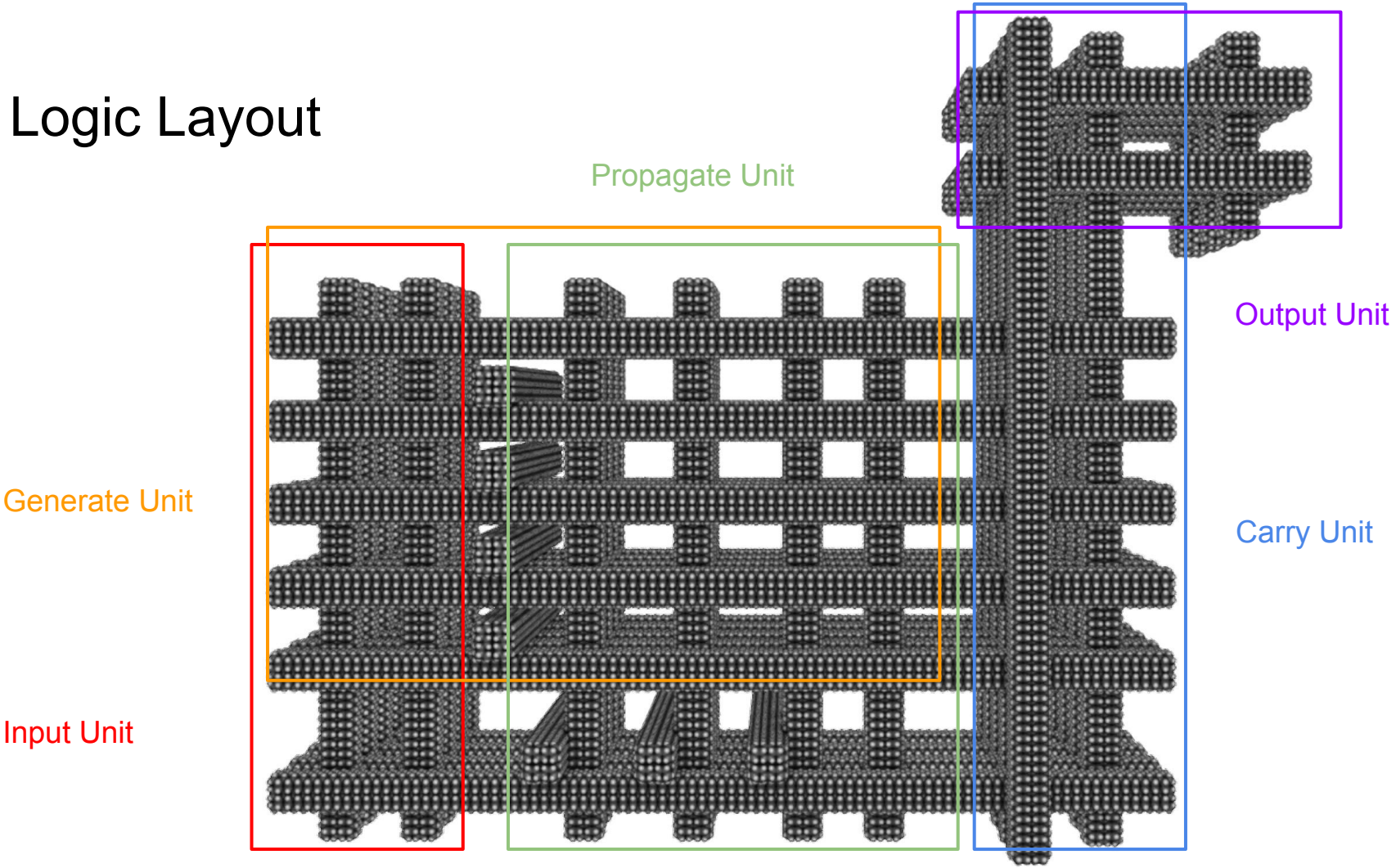


```
struct CLACarryUnit {
  // The carry in.
  var carryIn: Rod

  // The carry signal.
  //
  // Ordered from bit 0 -> bit 3.
  var signal: [Rod] = []

  // The value of A XOR B.
  //
  // Ordered from bit 0 -> bit 3.
  var xor: [Rod] = []
```

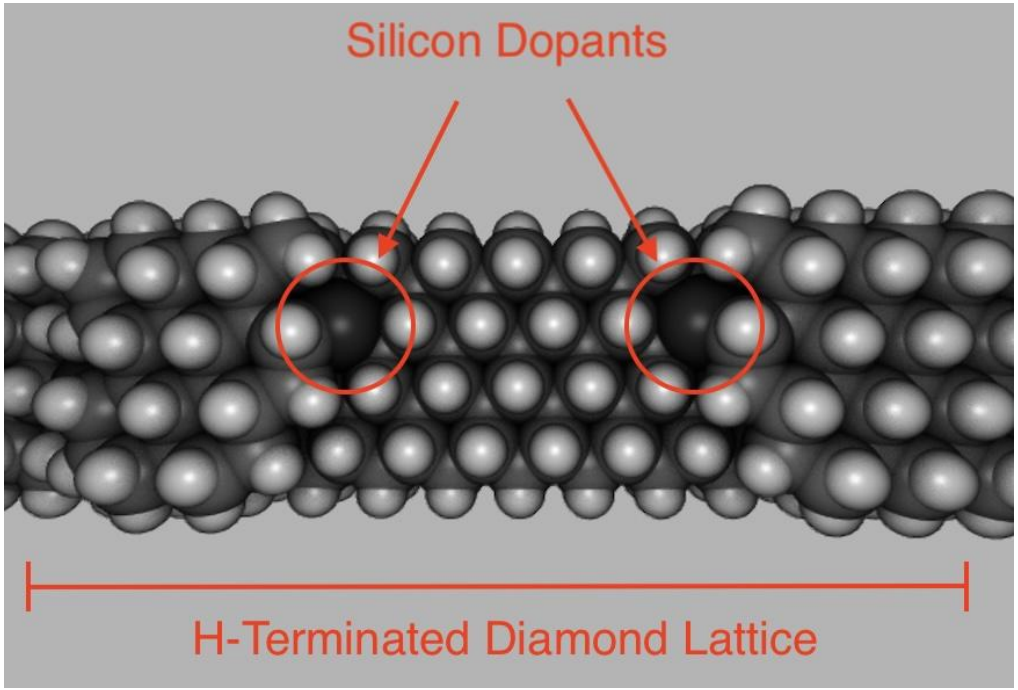


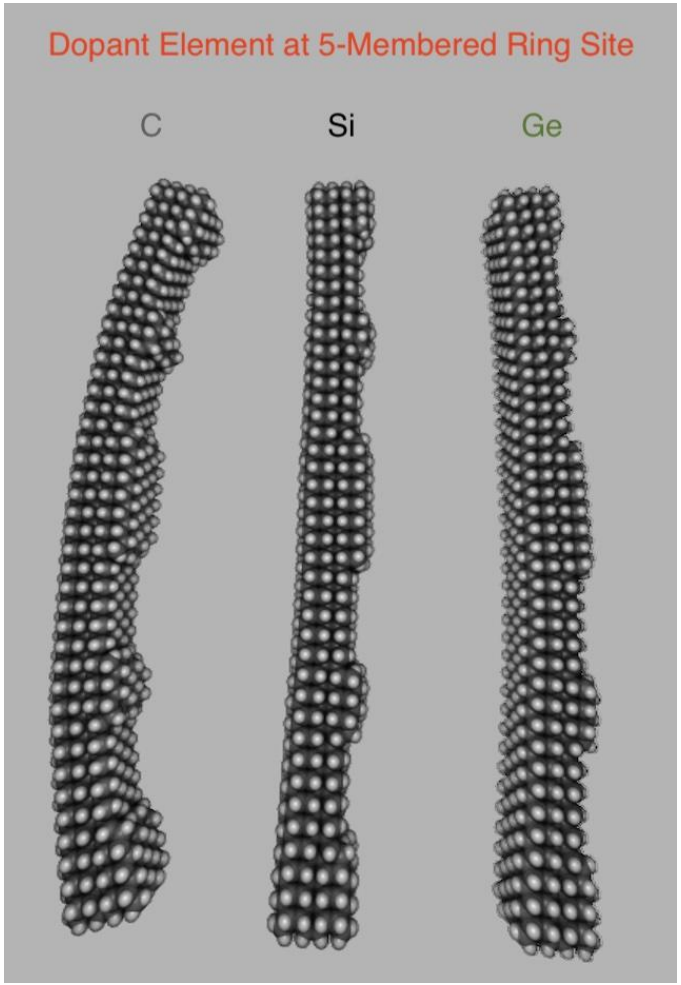
```
struct CLAOutputUnit {
  // The NOR intermediate of the half adder.
  //
  // Ordered from bit 0 -> bit 3.
  var nor: [Rod] = []

  // The AND intermediate of the half adder.
  //
  // Ordered from bit 0 -> bit 3.
  var and: [Rod] = []

  // The sum output of the circuit.
  var sum: [Rod] = []
```

Logic Layout

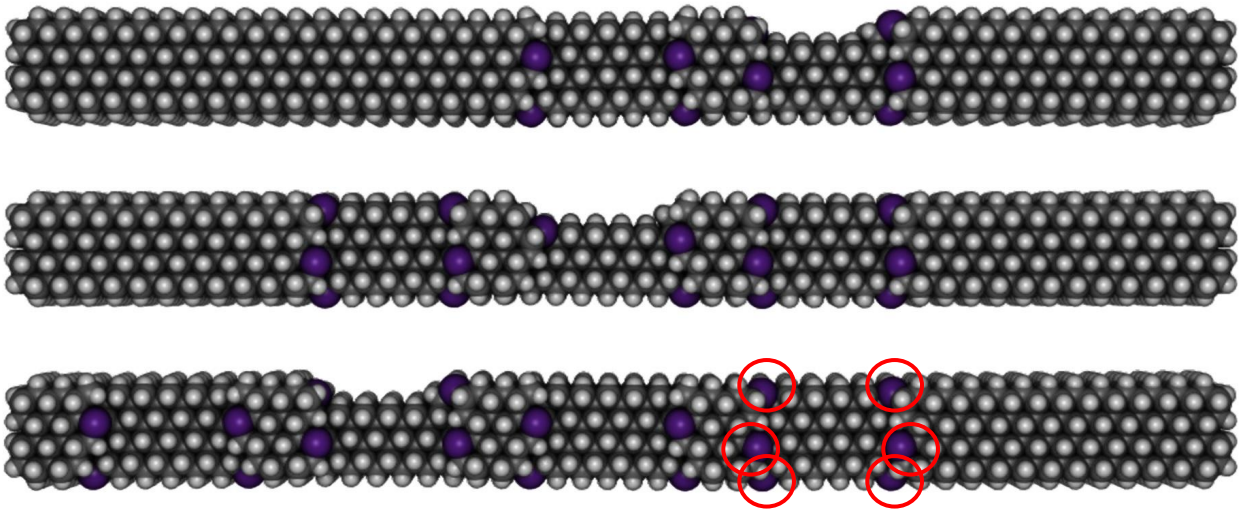


Vertical Logic Rods





Phosphorus



P dopants

MM4

Molecular Mechanics force field, version 4. The simulator used to create *Nanosystems (1992)*, but updated with modern ab initio parameters.

Documentation: philipturner.github.io/MM4

Atoms

Officially supported:

Element	Ring Types
H	n/a
C	5, 6
Si	5, 6
P (trivalent)	5, 6
S	5, 6
Ge	5, 6

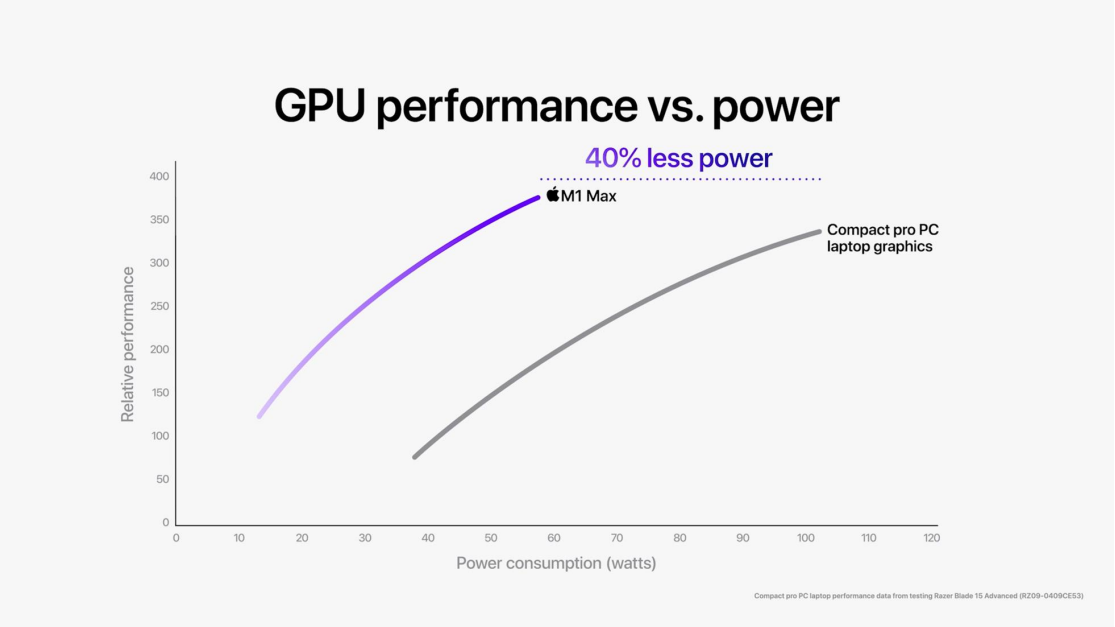
Experimental:

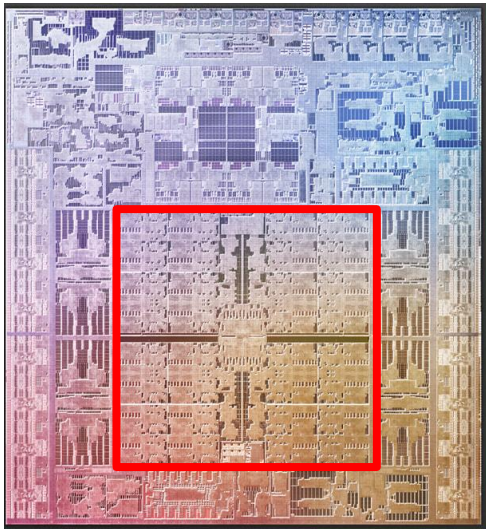
Element	Ring Types
N (trivalent)	5, 6
O	5, 6
F	n/a

Energy Efficiency

Highly Efficient Electronic Computer

Apple GPU architecture: low clock speeds, small caches





25% of die area

Energy per Switching Operation

$\geq 2707 \times 10^{-21}$ J

(50 W) / (1.296 GHz)(57 billion)(≤25%)

$\geq 2164 \times 10^{-21}$ J

(12 W) / (389 MHz)(57 billion)(≤25%)

3×10^{-21} J

Landauer Limit (300 K)

How efficient can consumer electronics get?

TSMC Node	1 MHz	389 MHz	1.296 GHz	6 GHz
N5	2,164 zJ	2,164 zJ	2,707 zJ	>10,000 zJ
N3 (35% better)	1,406 zJ	1,406 zJ	1,759 zJ	>10,000 zJ
N2 (35% more)	914 zJ	914 zJ	1,143 zJ	>10,000 zJ
Landauer Limit	3 zJ	3 zJ	3 zJ	3 zJ

Efficiency of Rod Logic

Dissipation Mode	Scaling	1 MHz	1 GHz
Vibrational Excitation	$O(v^4)$	0 zJ	0.56 zJ
Sliding-Interface Drag	O(v)	0 zJ	0.05 zJ
Cam Surface	O(v)	0 zJ	0.06 zJ
Thermoelastic Losses	O(1)	0.41 zJ	0.41 zJ

Energy loss, per rod, per cycle (*Nanosystems* PLA)

	Energy / Rod	Energy / Switching Event
Forward Cycle	1.08 zJ	0.068 zJ
Reverse Cycle	1.08 zJ	0.068 zJ
Overall	2.16 zJ	0.135 zJ
Landauer Limit	44.8 zJ	2.8 zJ

Efficiency of Rod Logic

	1 MHz	1 GHz	6 GHz
TSMC N5	2,164 zJ	2,707 zJ	>10,000 zJ
TSMC N2 (optimistic)	914 zJ	1,143 zJ	>10,000 zJ
Landauer Limit	2.8 zJ	2.8 zJ	2.8 zJ
Rod Logic	0.051 zJ	0.135 zJ	cannot operate
Molecular Link Logic	TBD	TBD	cannot operate

Rod logic is theoretically ~15,000x more efficient than CMOS.
Rod logic is theoretically ~40x more efficient than Landauer limit.

Proposed Operating Speeds

1 MHz

- Factories and robots
- Massively parallel AI training
- Energy dominated by O(1)

1 GHz

- Competitive with electronics
- Personal computers (CPUs, GPUs)
- Energy dominated by O(v)
- $K = \frac{1}{2} m v^2$
- How is kinetic energy recovered?

Nanomechanical Carry Lookahead Adder

685,431 atoms

26 nm x 13 nm x 21 nm