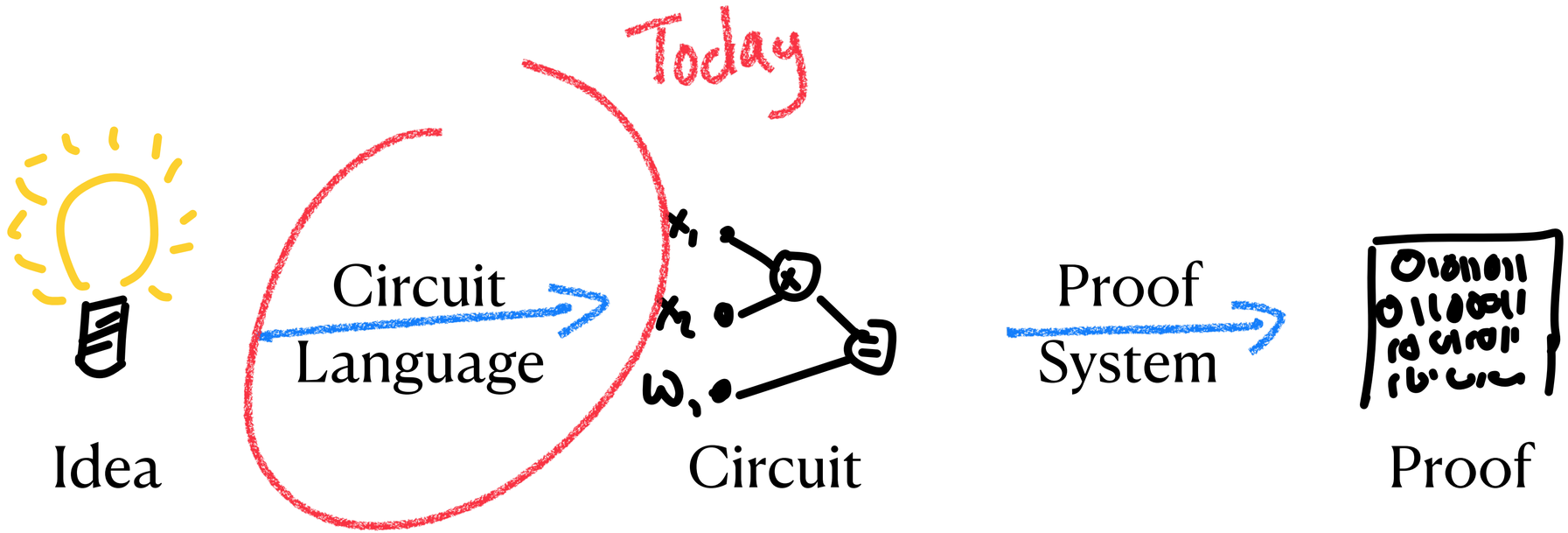
A Taxonomy of Circuit Languages

Note:
If you notice an error in this talk's material, please let Alex know.

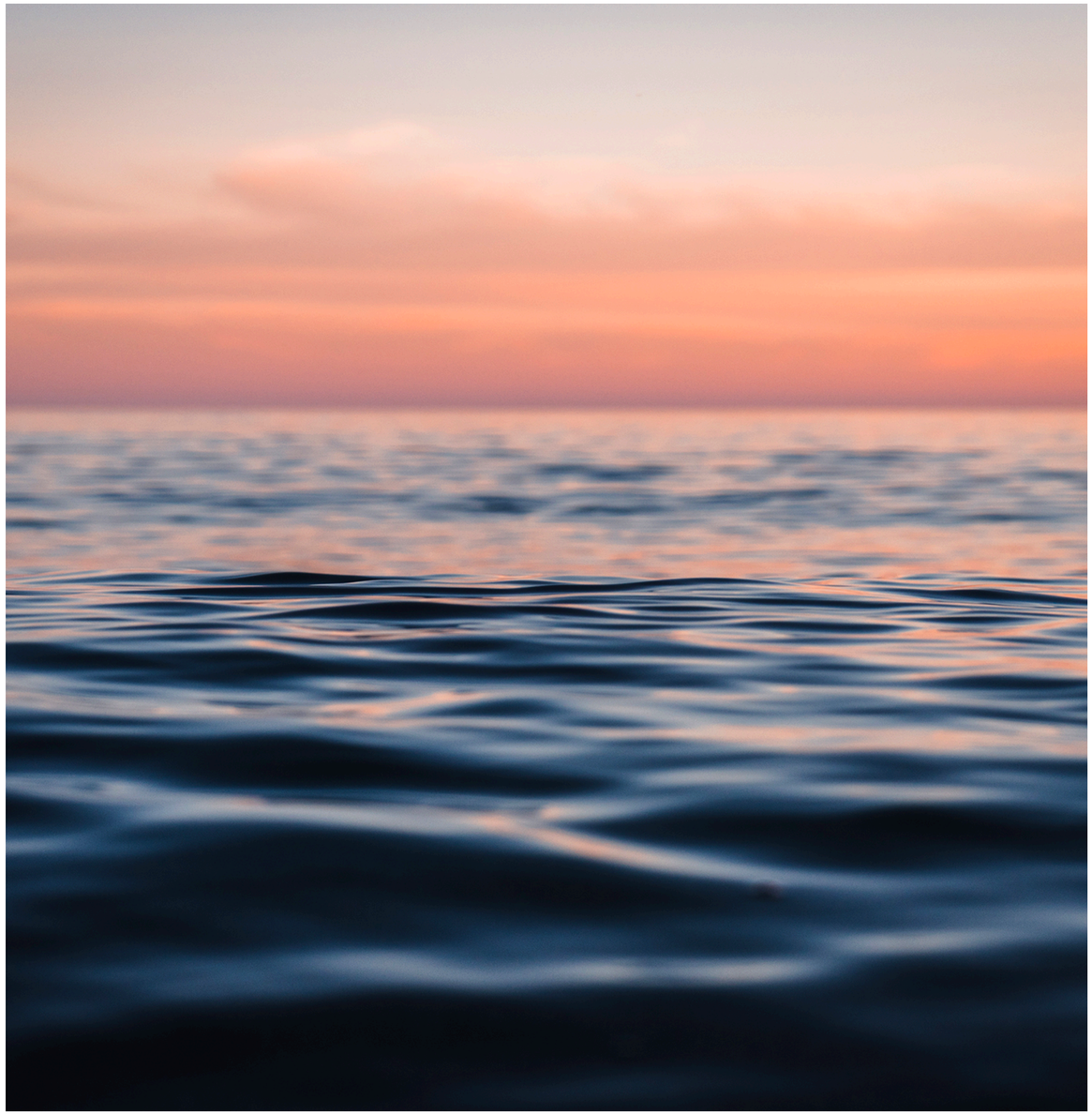
Alex Ozdemir, “ZK Language Event” 2021

Core Problem

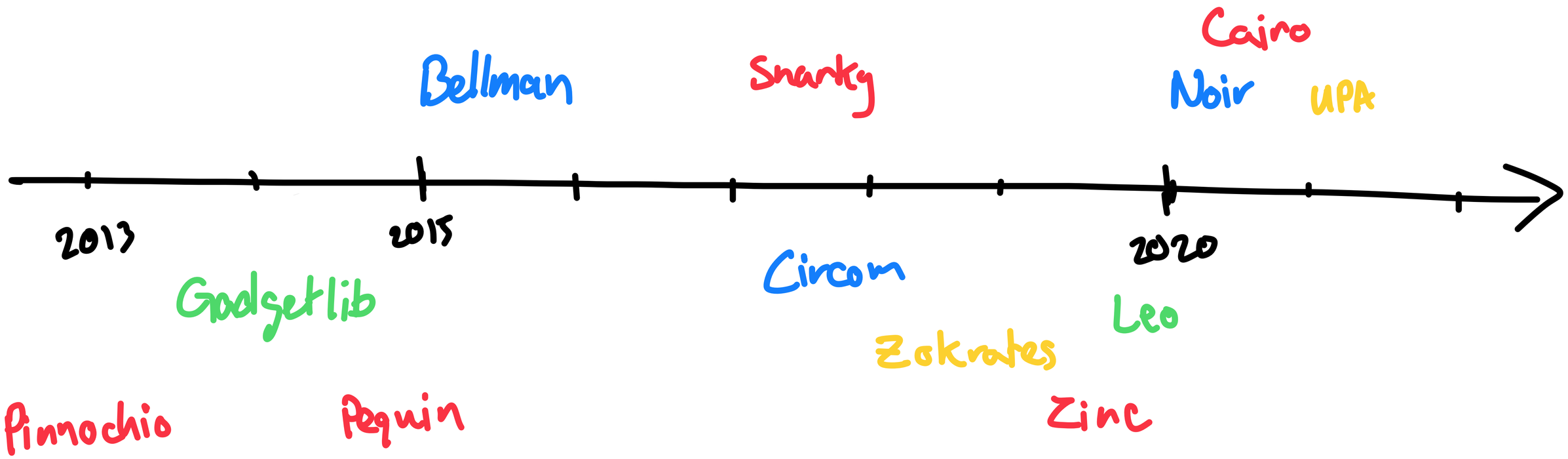


A Cambrian Explosion

2013 - present



Timeline


Cairo
Bellman
Snarky
Noir
UPA
2013
2015
2020
Circom
Goodgetlib
Leo
Zokrates
Pinnochio
Pequin
Zinc

Circuit Target

R1CS	Plonk	AIR
single circuit	single circuit	repeated circuit
unit-cost x	unit-cost x	unit cost x
'free' +	unit-cost +	'free' +
	custom gates	
	verifier randomness	

True "circuits"

RAM-like

Circuit Target

R1CS

bellman

gadgetlib

pequin

Zokrates

snarky

circom

Leo

Zinc

Plonk

Noir

UPA

AIR

Cairo

Air Script

Distaff Assembly

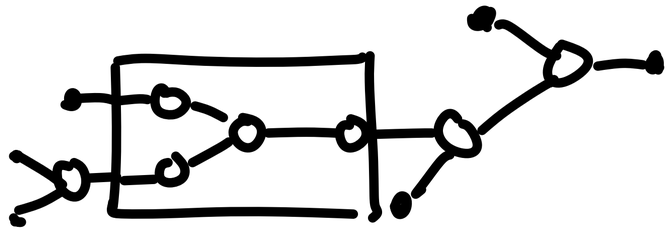
Air Assembly

Language Type

Library

```
circ.add_wire(...)
circ.add_gate(...)
```

HDL



(Verilog)

PL

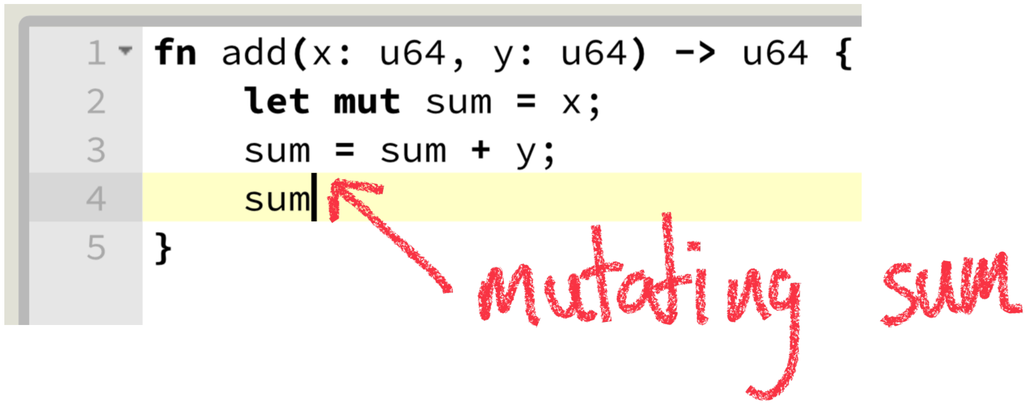
```
fn main(...) {
    ...
}
```

(Rust)

Language Type

Library	HDL	PL
bellman	circom	Zokrates
gadgetlib		Zinc
snarky		Leo
UPA		pequin
		Noir
		Cairo

Feature: Mutable Variables



Feature: Mutable Variables

No	Yes
bellman	Leo
gadgetlib	Zinc
snarky	Cairo
circom	pequin
UPA	Zokrates
	Noir

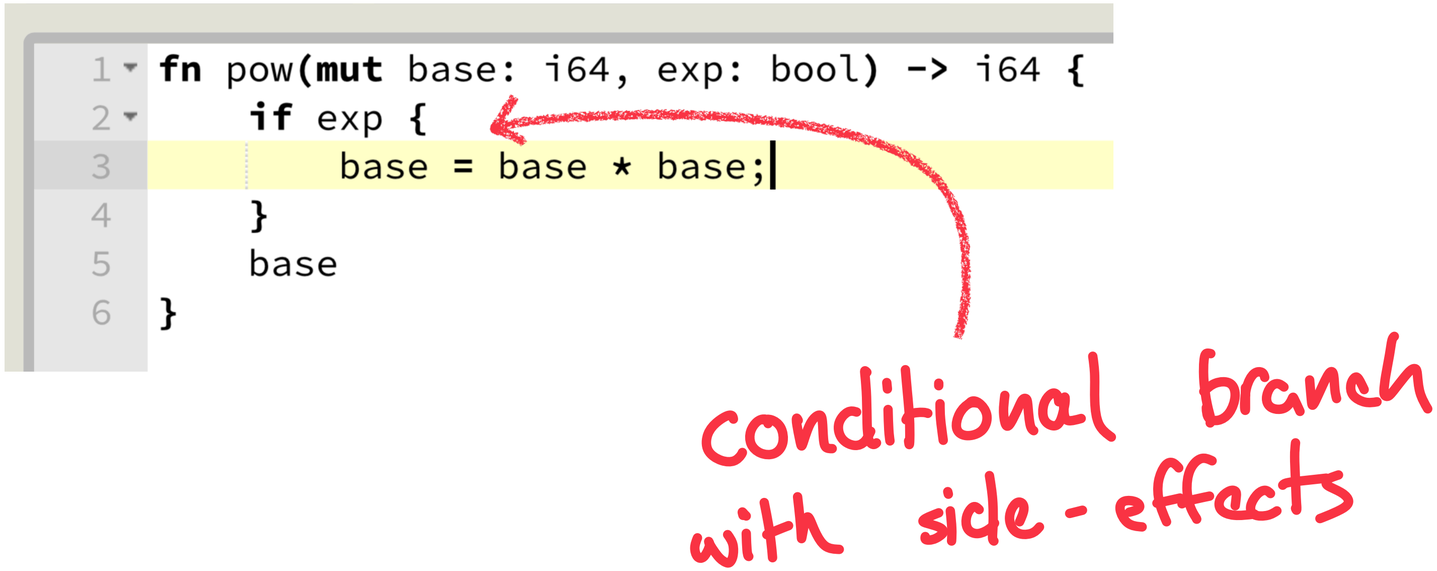
Feature: Primitive Types

```
1  fn main() {
2      let x: bool = true;
3      let y: u32 = 5;
4      let z: f64 = 5.0e-1;
5  }
```

Feature: Primitive Types

Field	Bool	Machine Integers
circom	bellman	snarky
Cairo	gadgetlib	Zokrates
	UPA	pequin
		Zinc
		Leo
		Noir

Feature: If-Statements



Feature: If-Statements

No	Yes
bellman	snarky
gadgetlib	circom
Zinc	Zokrates
UPA	Noir
Leo	pequin
Cairo	

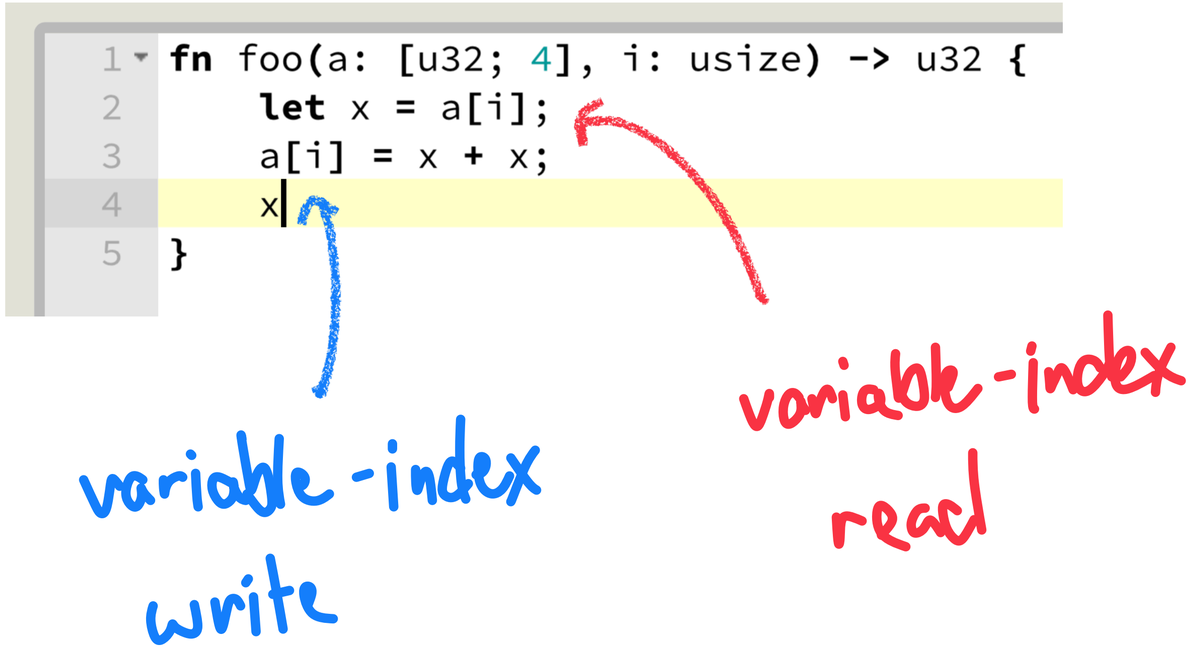
Feature: User Structures

```
1  struct CompressedPoint<F: Field> {
2      x: F,
3      neg_y: bool,
4  }
```

Feature: User Structures

No	From Host Language	Tuples	Yes
Noir	bellman	Leo	pequin
circom	gadgetlib		Zokrates
	snarky		Zinc
	UPA		Cairo

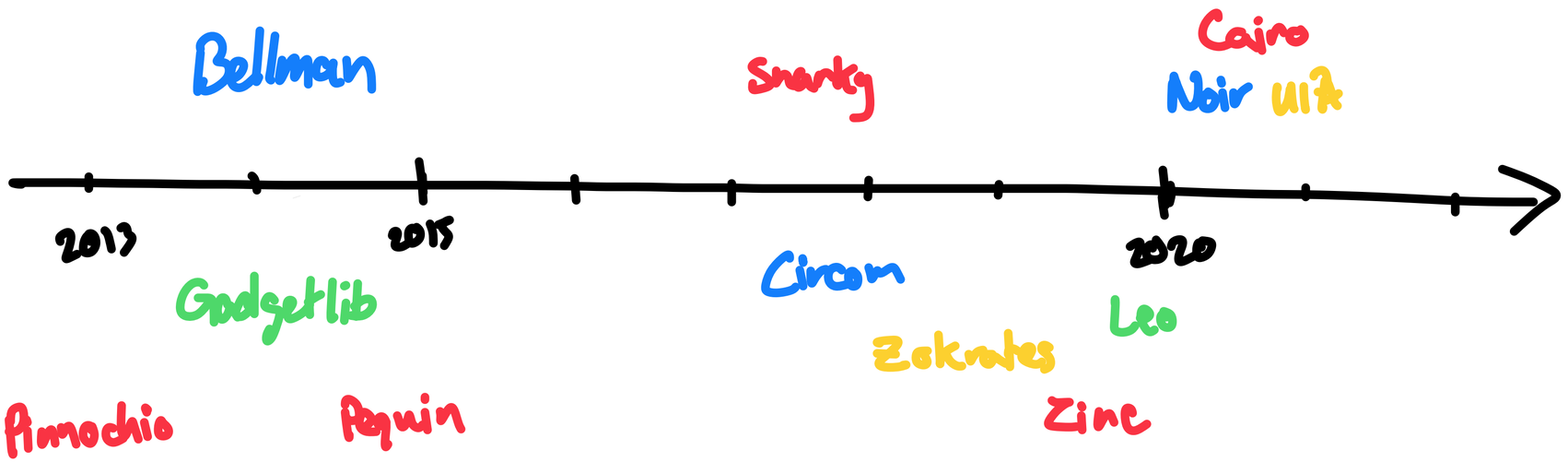
Feature: Arrays



Feature: Arrays

No Arrays	Constant Indices	Linear Scans	Better
bellman	circom	Zokrates	pequin
gadgetlib	Zinc		Cairo
UPA	Noir		
	Leo		

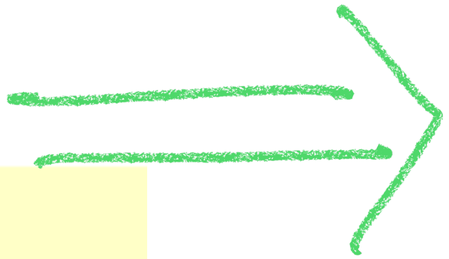
The Future of Circuit Languages



1. We need to continue to explore

The Future of Circuit Languages

```
let x = a[i];
a[i] = x + x;
x
```



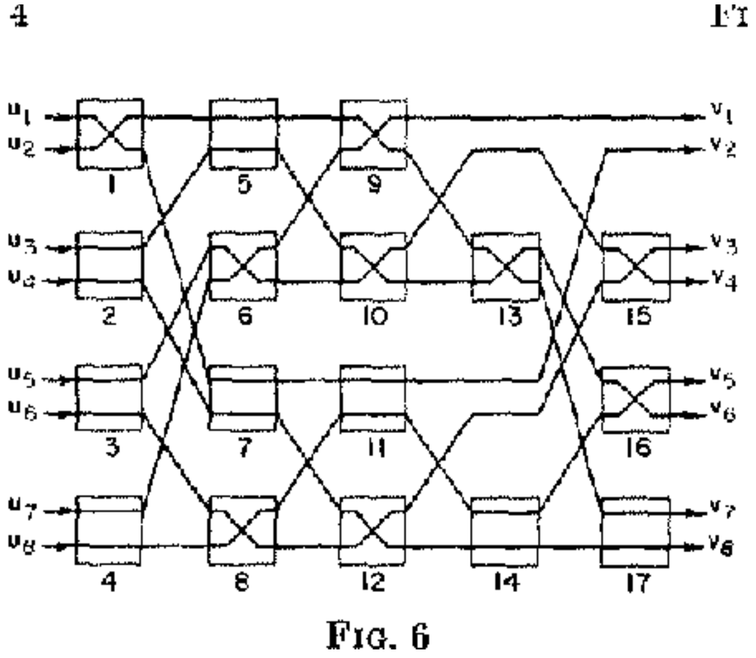
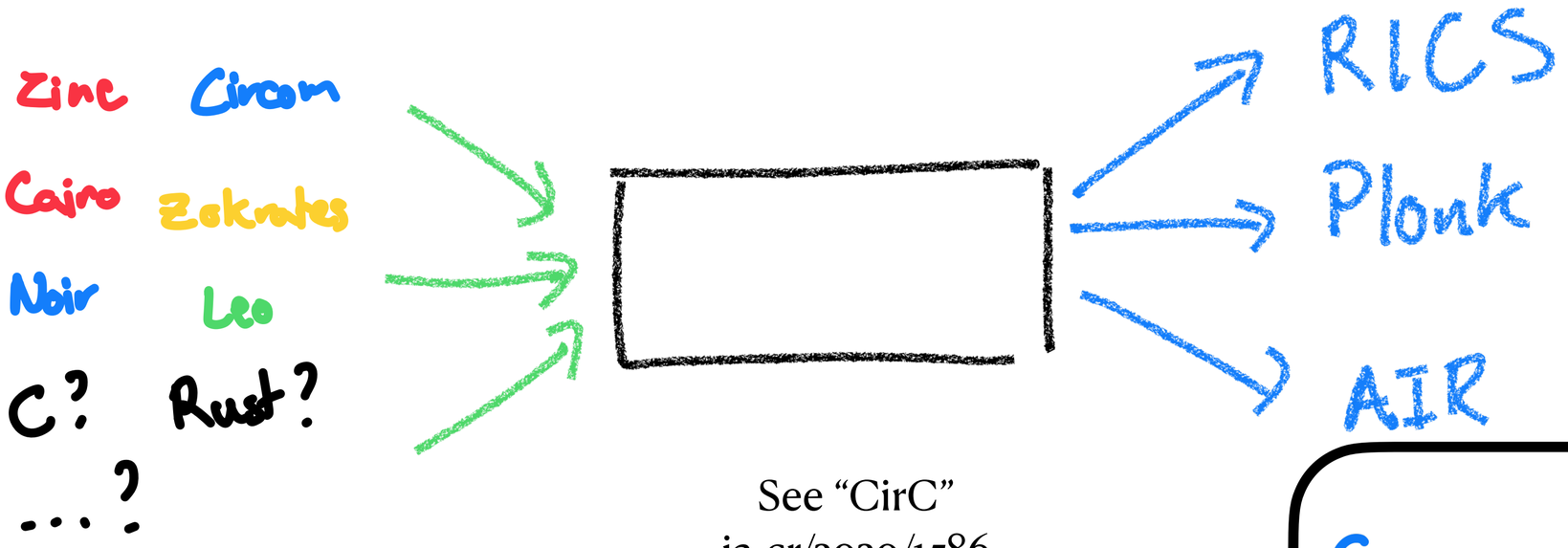


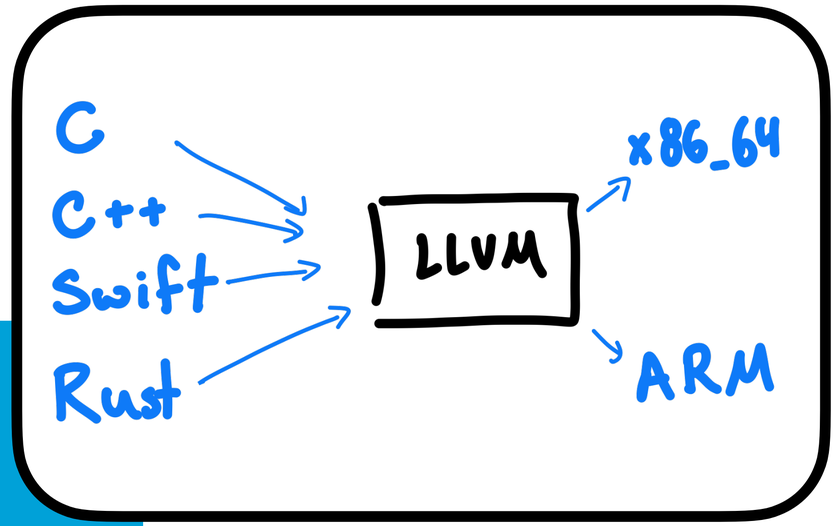
FIG. 6

2. The compilation problem is hard

The Future of Circuit Languages



See “CirC”
ia.cr/2020/1586
github.com/circify



3. Perhaps we can share infrastructure?