A Taxonomy of Circuit Languages

Alex Ozdemir, “ZK Language Event” 2021
Core Problem

Idea → Circuit Language → Circuit

Today → Proof System → Proof
A Cambrian Explosion
2013 - present
Circuit Target

R1CS
- Single circuit
- Unit-cost \times \text{`free' } +

Plonk
- Single circuit
- Unit-cost \times \text{custom gates}
- Verifier randomness

AIR
- Repeated circuit
- Unit-cost \times \text{`free' } +

\text{True "circuits"}

\text{RAM-like}
Circuit Target

R1CS
- bellman
- gadgetlib
- pequin
- Zokrates

Plonk
- snarky
- circom
- Leo
- Noir
- UPA

AIR
- Cairo
- Air Script
- Distaff
- Assembly
- Air Assembly
- Zinc
Language Type

Library

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

HDL

```
```

(VHDL)

PL

```
fn main(…)
{
...
}
(Rust)
```
Language Type

Library
- bellman
- gadgetlib
- snarky
- uPA

HDL
- circom

PL
- Zokrates
- pequin
- Zinc
- Noir
- Leo
- Cairo
Feature: Mutable Variables

```rust
fn add(x: u64, y: u64) -> u64 {
    let mut sum = x;
    sum = sum + y;
}
```
# Feature: Mutable Variables

<table>
<thead>
<tr>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>bellman, gadgetlib, snarky, Circom, UPA</td>
<td>Leo, Zinc, Cairo, Noir, pequin, Zokrates</td>
</tr>
</tbody>
</table>
Feature: Primitive Types

```rust
fn main() {
    let x: bool = true;
    let y: u32 = 5;
    let z: f64 = 5.0e-1;
}
```
Feature: Primitive Types

Field
- Circom
- Cairo

Bool
- bellman
- gadgetlib
- UPA

Machine Integers
- Snarky
- Pequin
- Leo
- Zinc
- Noir
Feature: If-Statements

```rust
fn pow(mut base: i64, exp: bool) -> i64 {
    if exp {
        base = base * base;
    }
    base
}
```

Conditional branch with side-effects
<table>
<thead>
<tr>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>bellman</td>
<td>Leo</td>
</tr>
<tr>
<td>gadgetlib</td>
<td>pequin</td>
</tr>
<tr>
<td>Zinc</td>
<td>Cairo</td>
</tr>
<tr>
<td>UPA</td>
<td>Noir</td>
</tr>
<tr>
<td>snarky</td>
<td></td>
</tr>
<tr>
<td>circom</td>
<td></td>
</tr>
<tr>
<td>Zokrates</td>
<td></td>
</tr>
</tbody>
</table>
Feature: User Structures

```rust
struct CompressedPoint<F: Field> {
    x: F,
    neg_y: bool,
}
```
# Feature: User Structures

<table>
<thead>
<tr>
<th>No</th>
<th>From Host Language</th>
<th>Tuples</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noir</td>
<td>bellman</td>
<td>Leo</td>
<td>pequin</td>
</tr>
<tr>
<td>circom</td>
<td>gadgetlib</td>
<td></td>
<td>Zokrates</td>
</tr>
<tr>
<td></td>
<td>smarky</td>
<td></td>
<td>Zinc</td>
</tr>
<tr>
<td></td>
<td>UPA</td>
<td></td>
<td>Cairo</td>
</tr>
</tbody>
</table>
Feature: Arrays

```rust
fn foo(a: [u32; 4], i: usize) -> u32 {
    let x = a[i];
    a[i] = x * x;
    x
}
```
Feature: Arrays

No Arrays
- bellman
- gadgetlib
- UPA

Constant Indices
- circom
- Zinc
- Noir
- Leo

Linear Scans
- Zokrates

Better
- pequin
- Cairo
1. We need to continue to explore
2. The compilation problem is hard
The Future of Circuit Languages

3. Perhaps we can share infrastructure?

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