VADCOPs: Leveraging STARKs for Tailored Proof Generation

Héctor Masip & Felicià Barceló

Prover Engineers, Polygon Labs

Section 1

Introduction

Execution Trace, Transition Function and Arithmetization

Rules of the **transition function**:

- a starts at 1 and b at 7.
- The sum of a,b,c equals 2·d.
- The next value of c is d.
- Each two states, b is zero.
- …

state	a	b	c	d
0	a_0	b_0	c_0	d_0
1	a_1	b_1	c_1	d_1
2	a_2	b_2	c_2	d_2
3	a_3	b_3	c_3	d_3
4	a_4	b_4	c_4	d_4
…	…	…	…	…

Execution Trace

Execution Trace, Transition Function and Arithmetization

Rules of the **transition function**:

- a starts at 1 and b at 7.
- The sum of a,b,c equals 2·d.
- The next value of c is d.
- Each two states, b is zero.
- …

Witness Computation

Arithmetization

```
L1 * (a - 1) === 0
L1 * (b - 7) === 0

a + b + c === 2*d
c' === d
(∑L2i) * b === 0
```

state	a	b	c	d
0	1	7	0	10
1	2	0	11	13
2	1	11	0	12
3	9	0	15	24
4	7	15	1	23
…	…	…	…	…

Execution Trace

The EVM Arithmetization

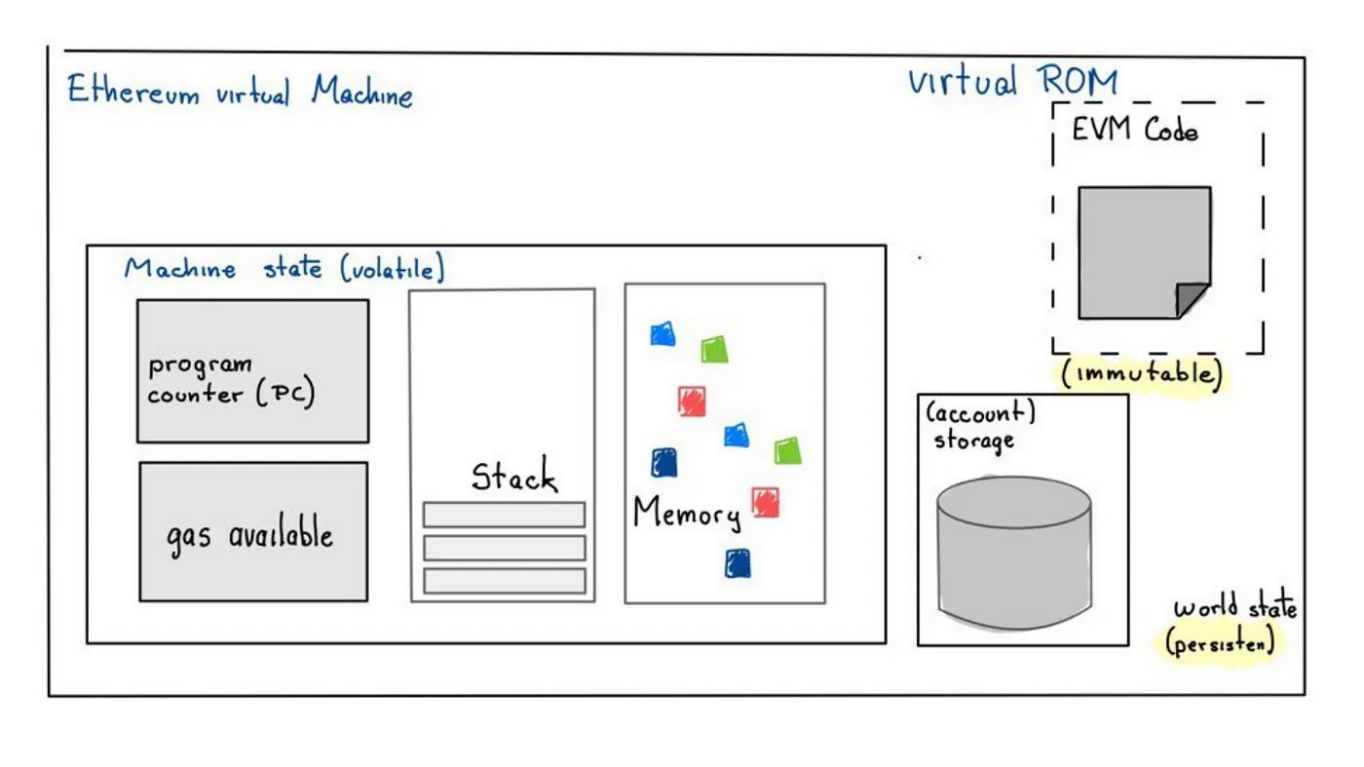
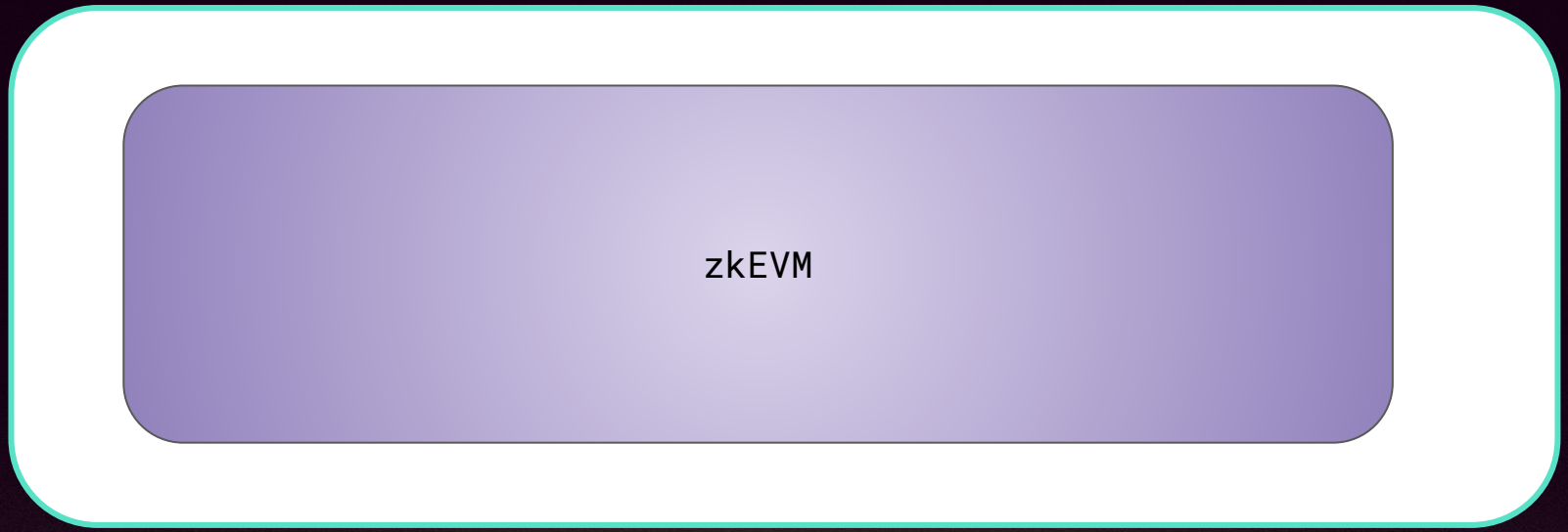
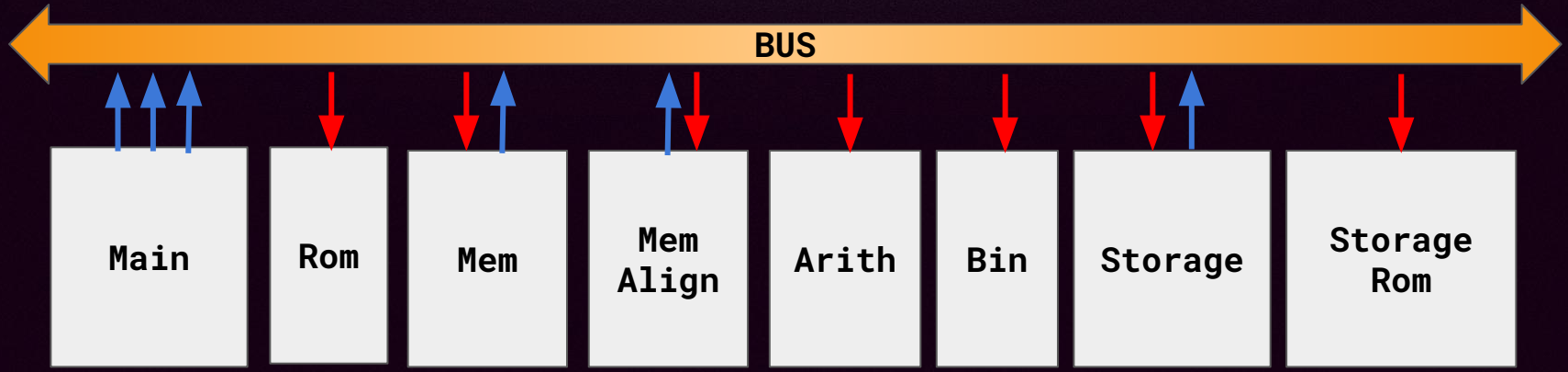


Image taken from "Understanding the EVM"

The EVM is Too Complex to Arithmetize as a Whole



Unlocking Modularity: The BUS



We didn't know at that time, but we unlocked **the BUS**:

- The main EVM component (transaction processor) could now **"assume"** operations, throwing them to the bus.
- Specialized components could now **"prove"** the operations they get, removing them from the bus.

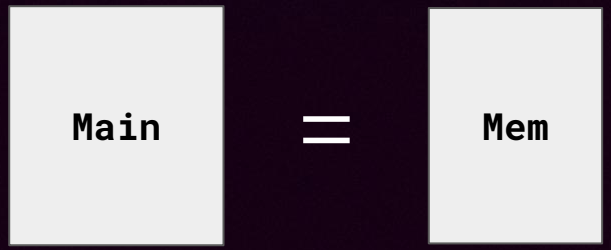
How to Throw to the BUS

There is a variety of manners that one can throw to the BUS:

How to Throw to the BUS

There is a variety of manners that one can throw to the BUS:

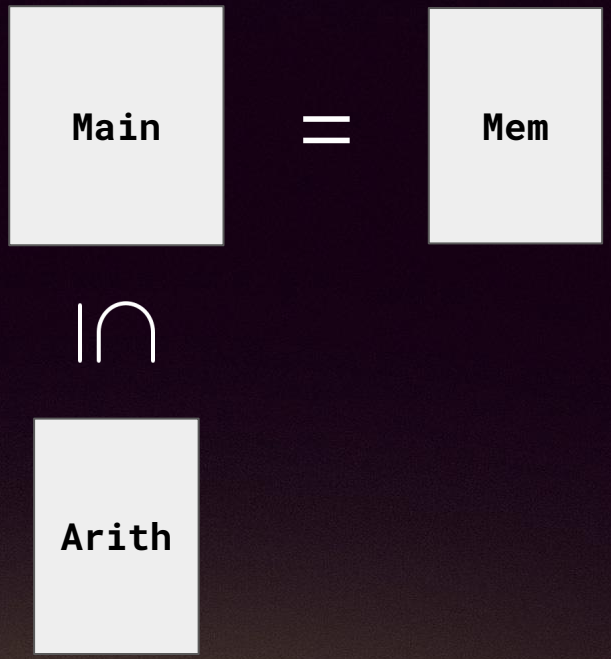
- Permutations.



How to Throw to the BUS

There is a variety of manners that one can throw to the BUS:

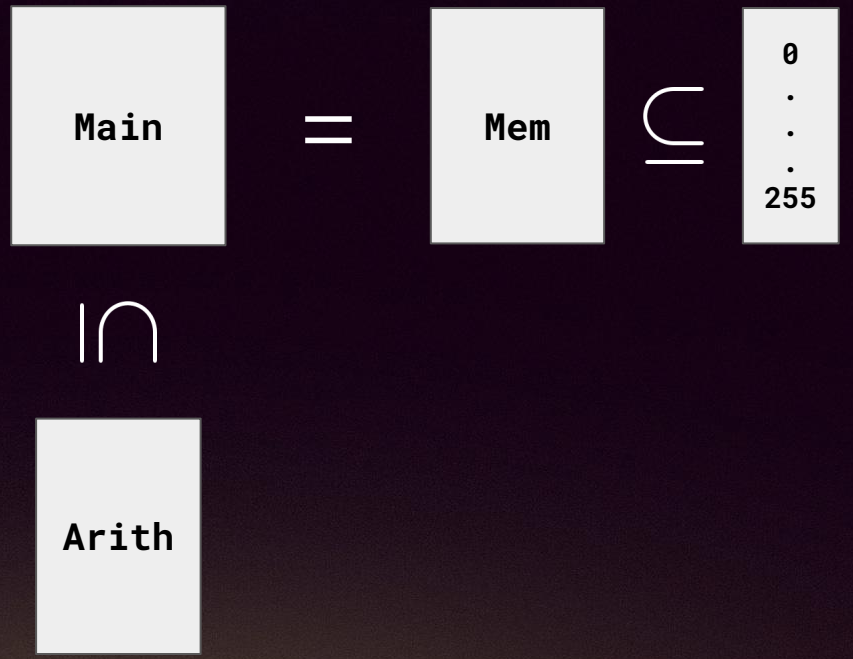
- Permutations.
- Lookups.



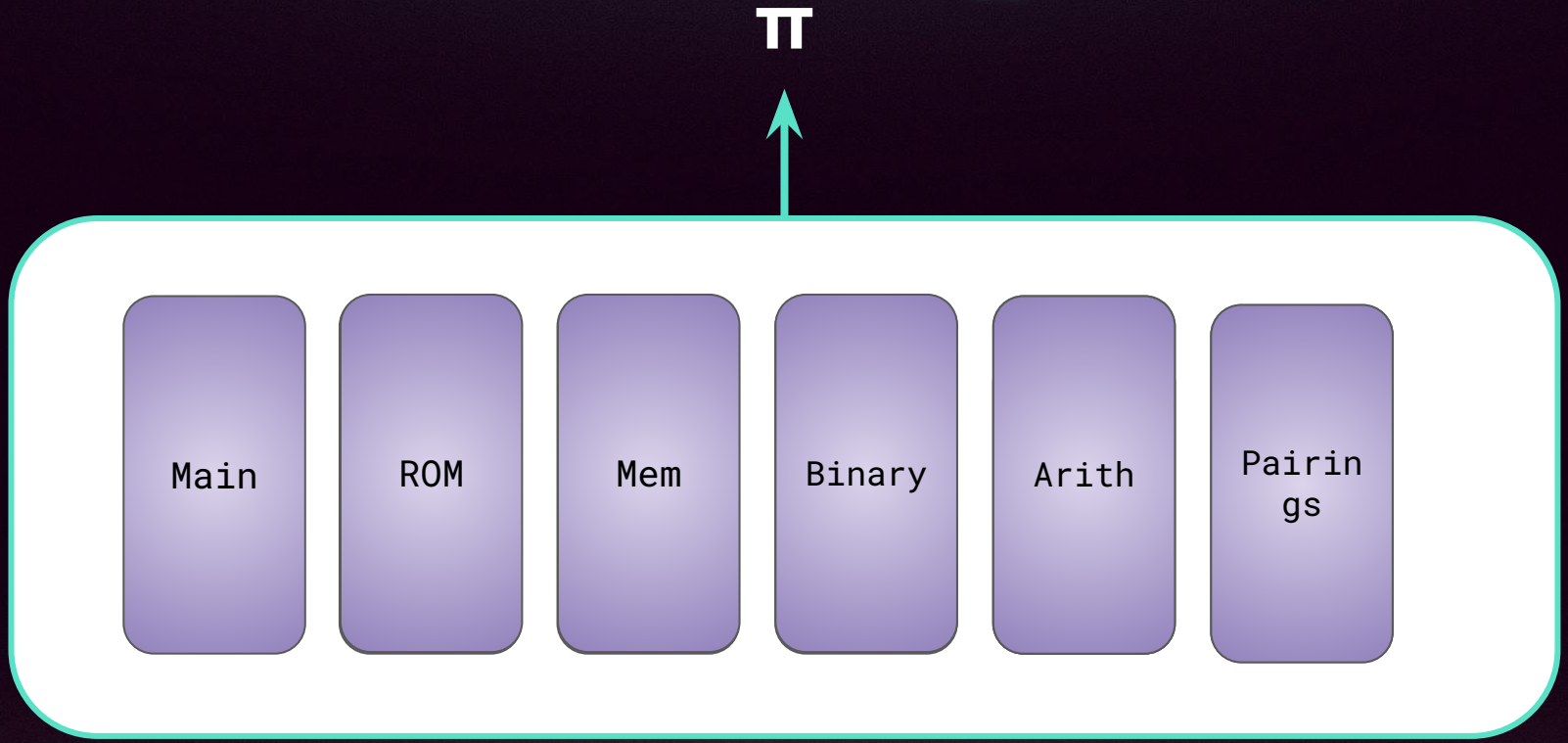
How to Throw to the BUS

There is a variety of manners that one can throw to the BUS:

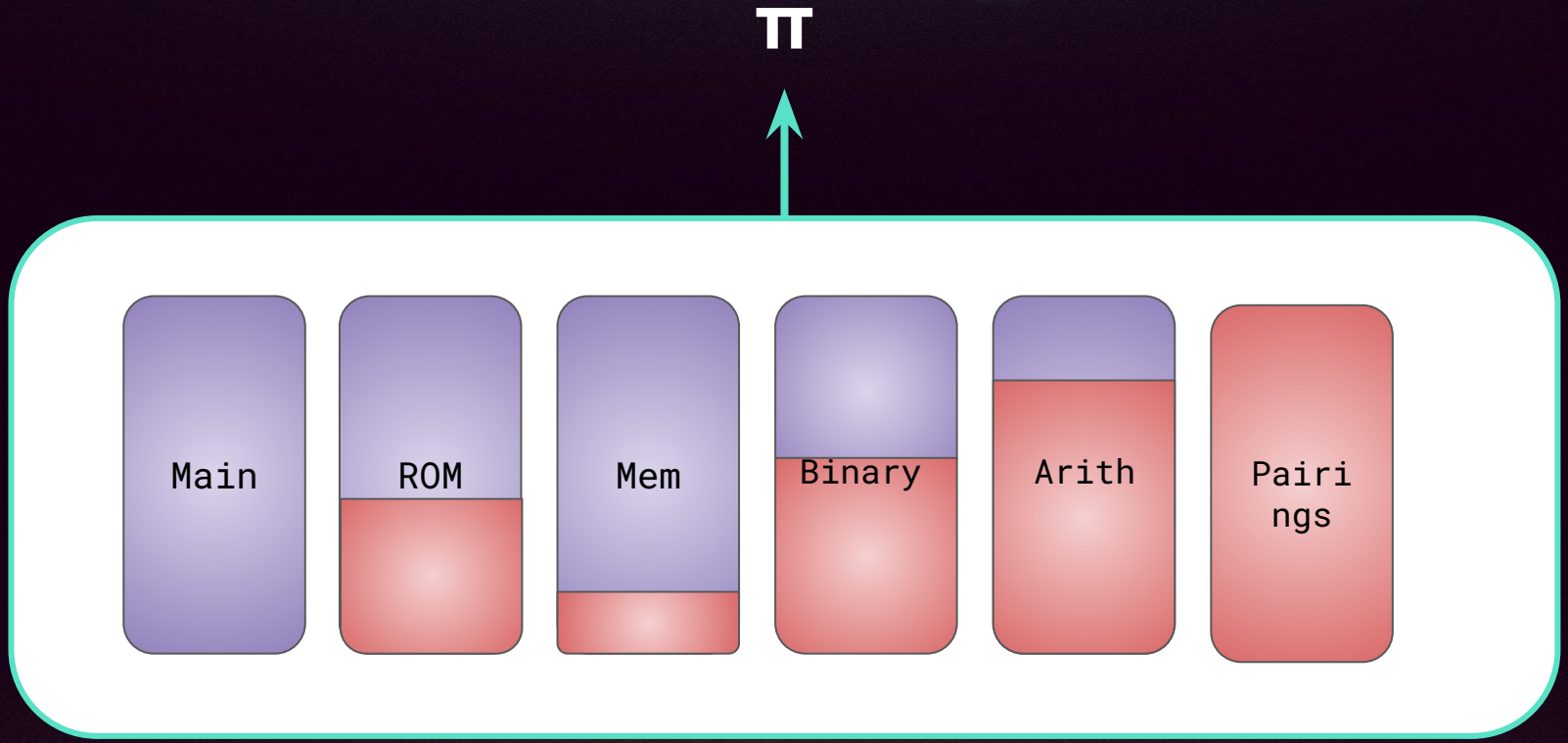
- Permutations.
- Lookups.
- Range Checks.
- Connections.
- …



Proof Generation Model: The Monolithic Approach



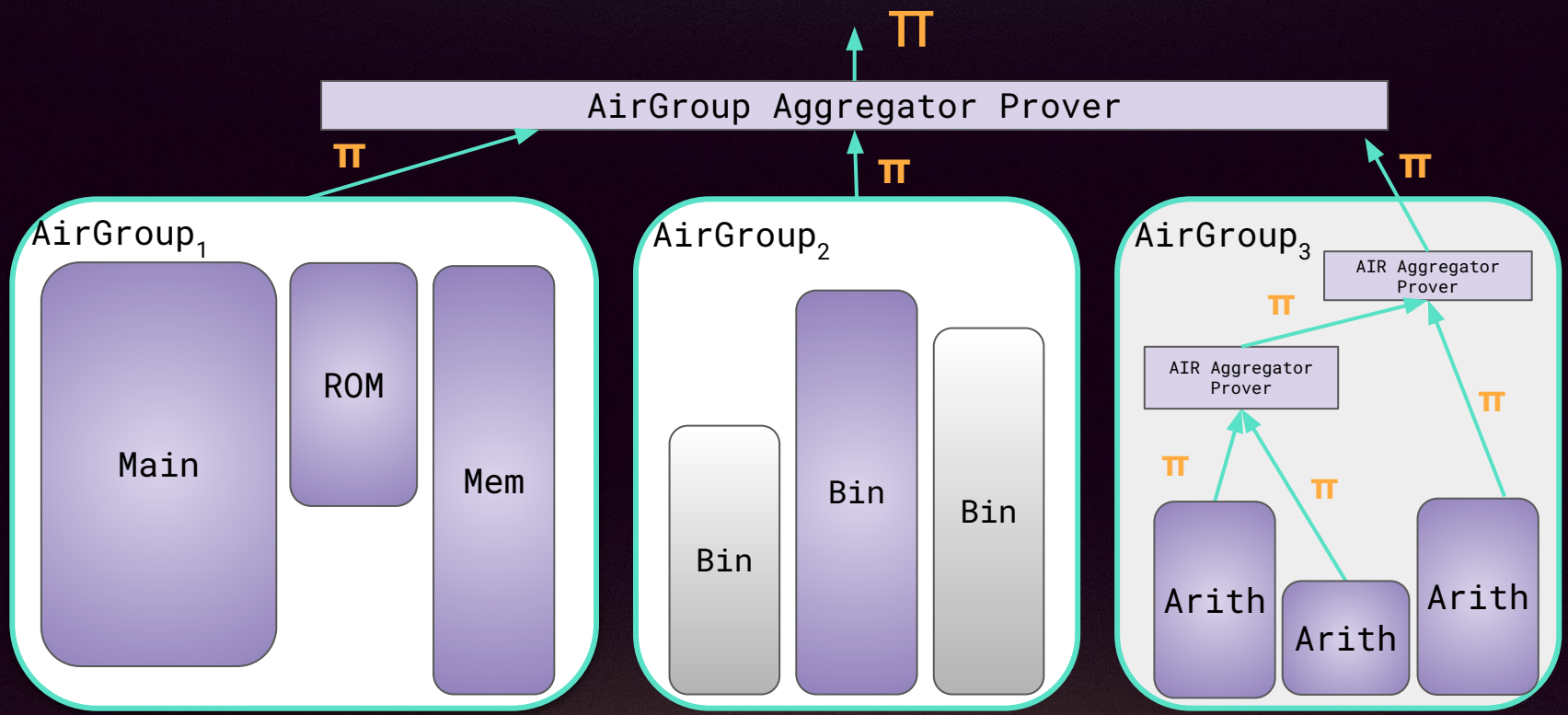
Proof Generation Model: The Monolithic Approach



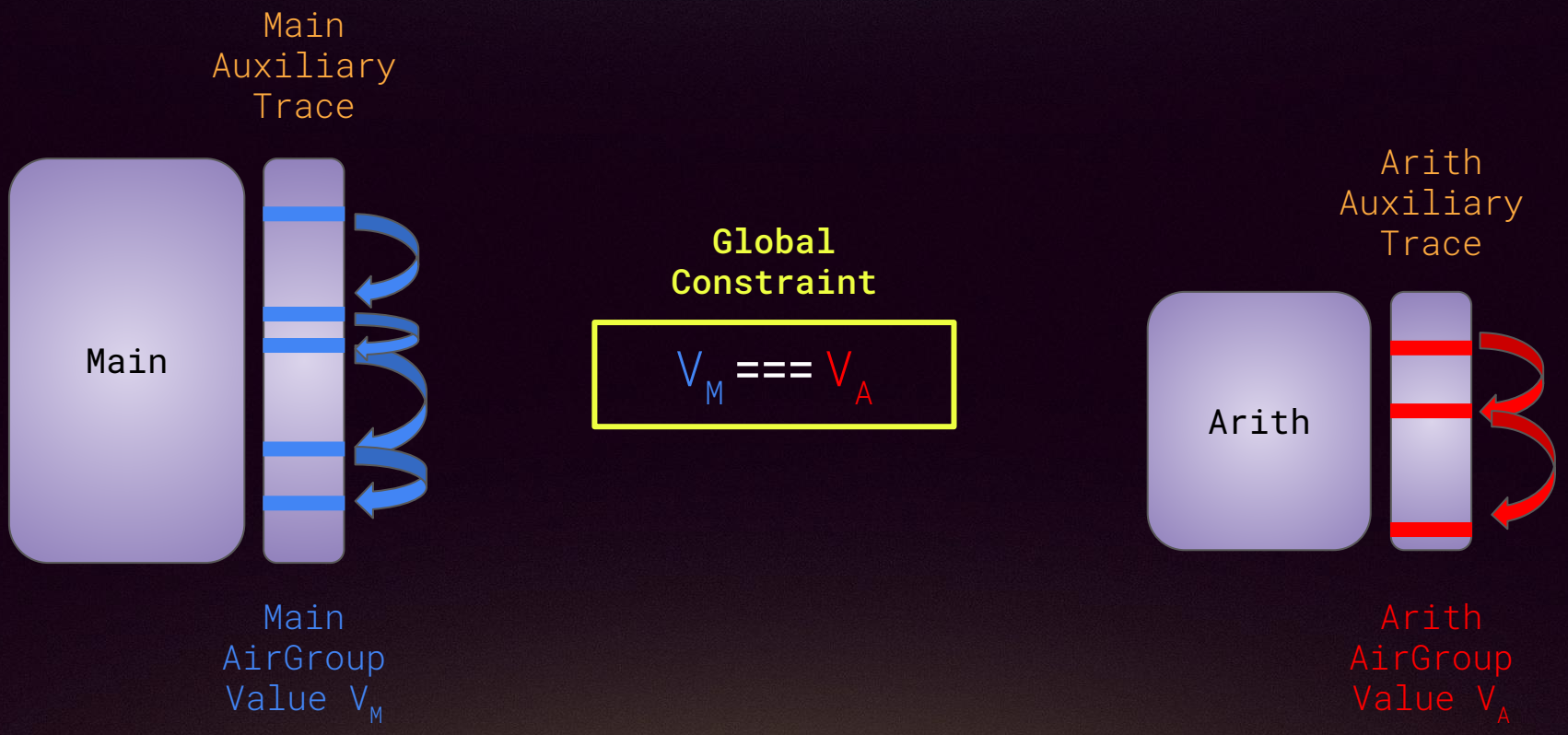
Section 2

VADCOPs

VADCOPs: Variable Degree Composite Proofs



AirGroup Values and Global Constraints



Example: Multi-Domain Lookups

f1	f2	f3	S1
8	1600	88	####
2	400	22	####
2	400	22	####
1	200	11	SUM1

$\subseteq$

mul	t1	t2	t3	S2
1	1	200	11	####
2	2	400	22	####
0	3	600	33	####
0	4	800	44	####
0	5	1000	55	####
0	6	1200	66	####
0	7	1400	77	####
1	8	1600	88	SUM2

$$S_1' = S_1 \cdot (1 - L_4) + \frac{1}{f_1' + \alpha f_2' + \alpha^2 f_3' + \beta}$$

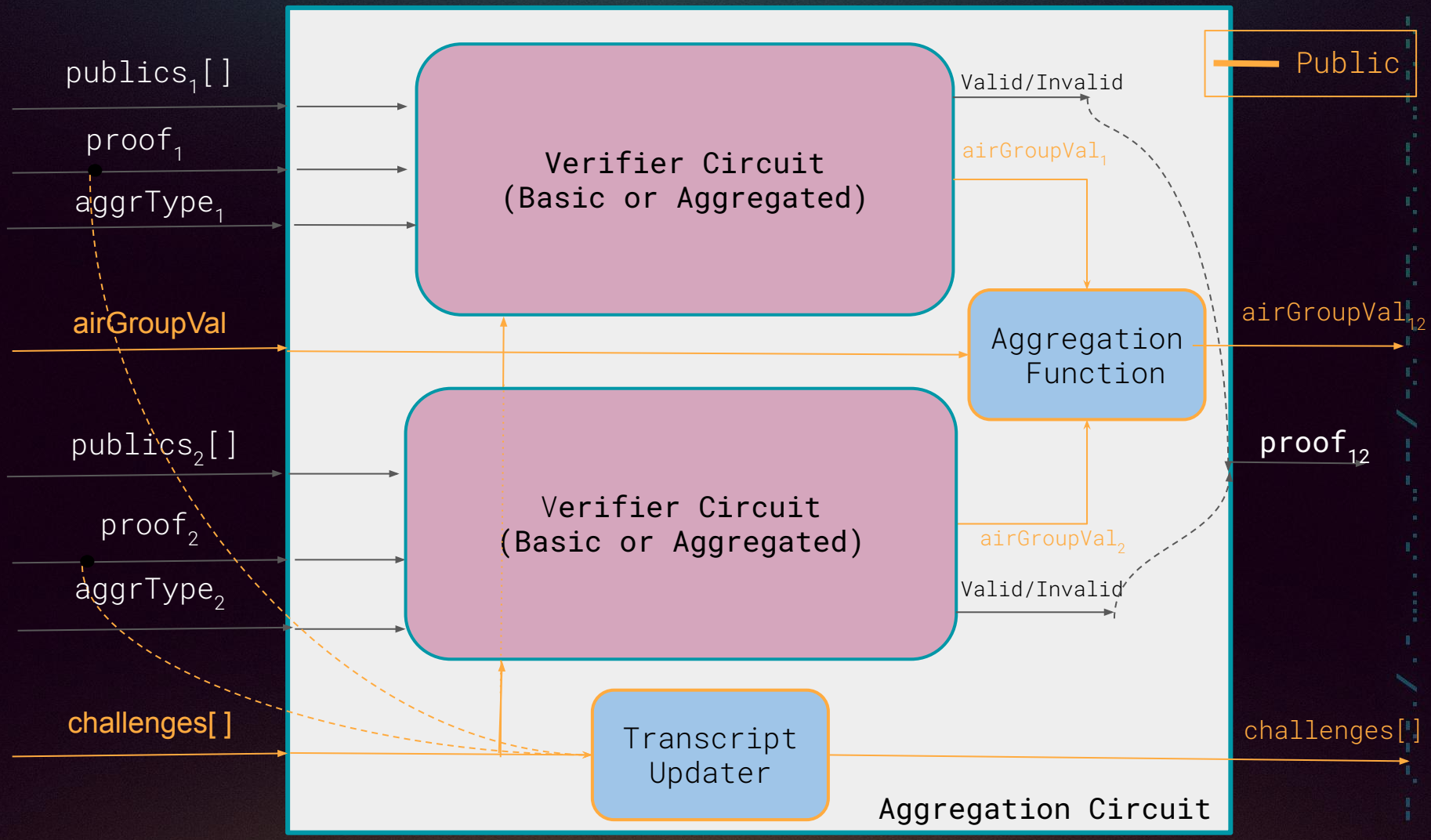
$$L_4 \cdot (S_1 - \text{SUM1}) = 0$$

$$S_2' = S_2 \cdot (1 - L_8) + \frac{\text{mul}}{t_1' + \alpha t_2' + \alpha^2 t_3' + \beta}$$

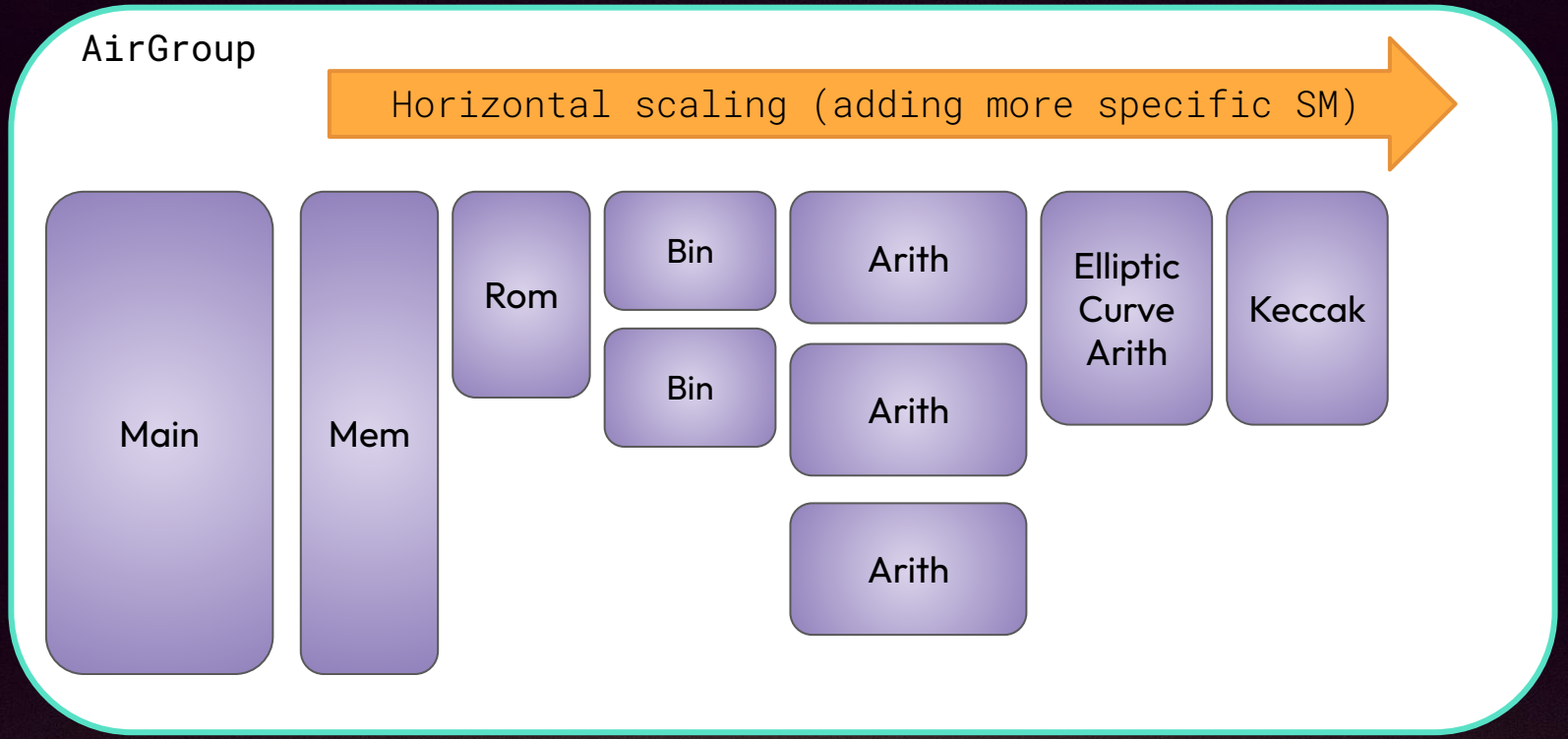
$$L_8 \cdot (S_2 - \text{SUM2}) = 0$$

Global Constraint

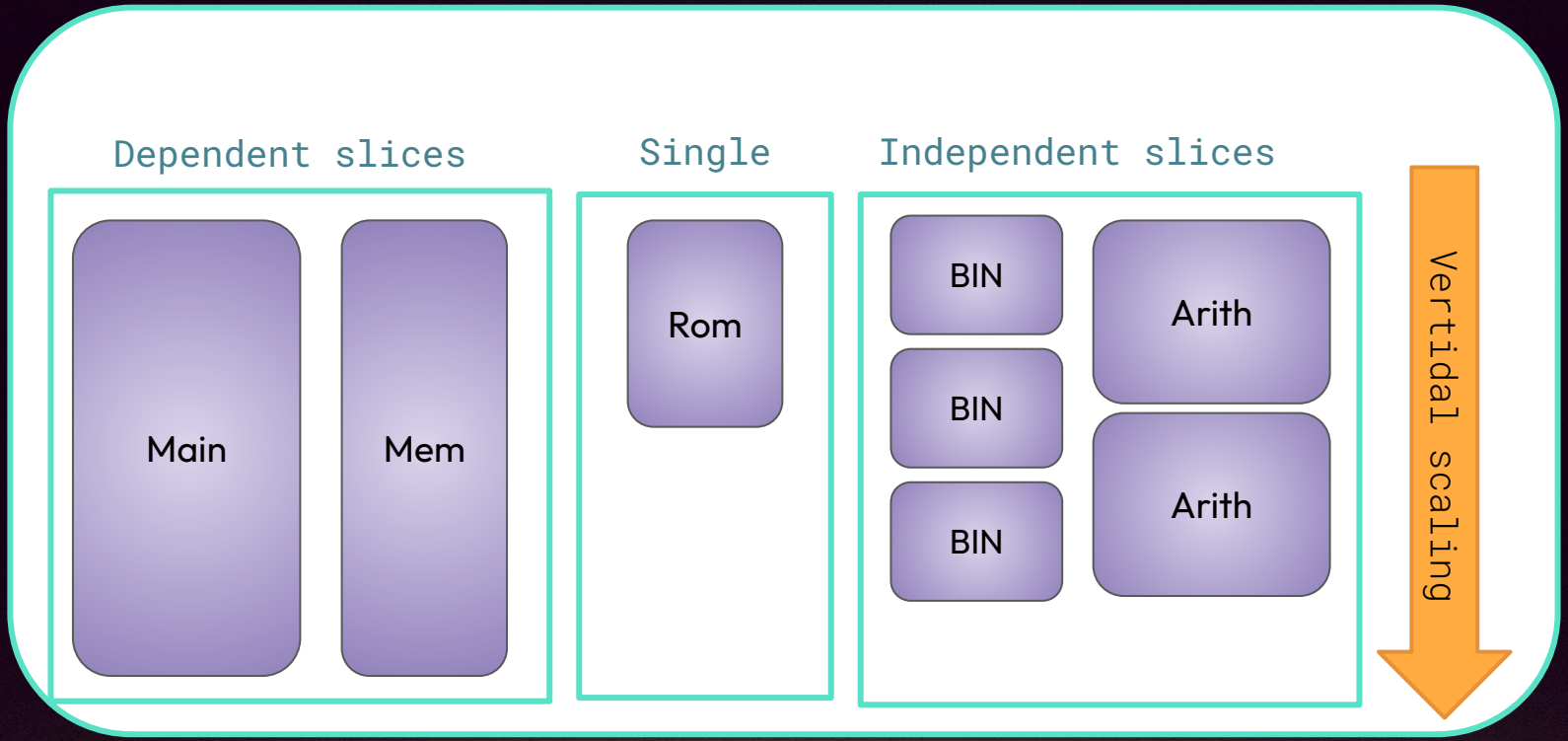
SUM1 === SUM2


Public
publics1[]
proof1
aggrType1
airGroupVal
publics2[]
proof2
aggrType2
challenges[ ]
Verifier Circuit
(Basic or Aggregated)
Verifier Circuit
(Basic or Aggregated)
Valid/Invalid
airGroupVal1
Aggregation
Function
airGroupVal12
proof12
airGroupVal2
Valid/Invalid
Transcript
Updater
challenges[]
Aggregation Circuit

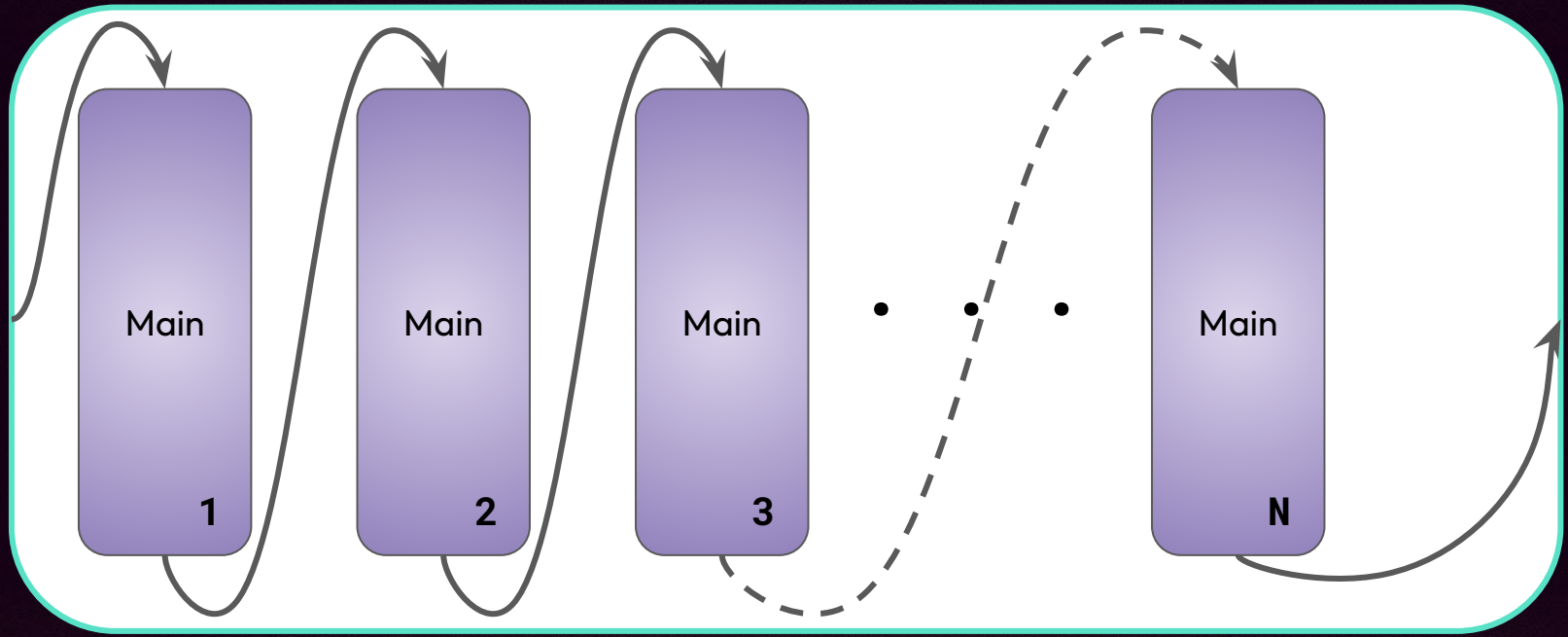
Horizontal Scaling (specific state machines)



Vertical Scaling (slice trace, types)

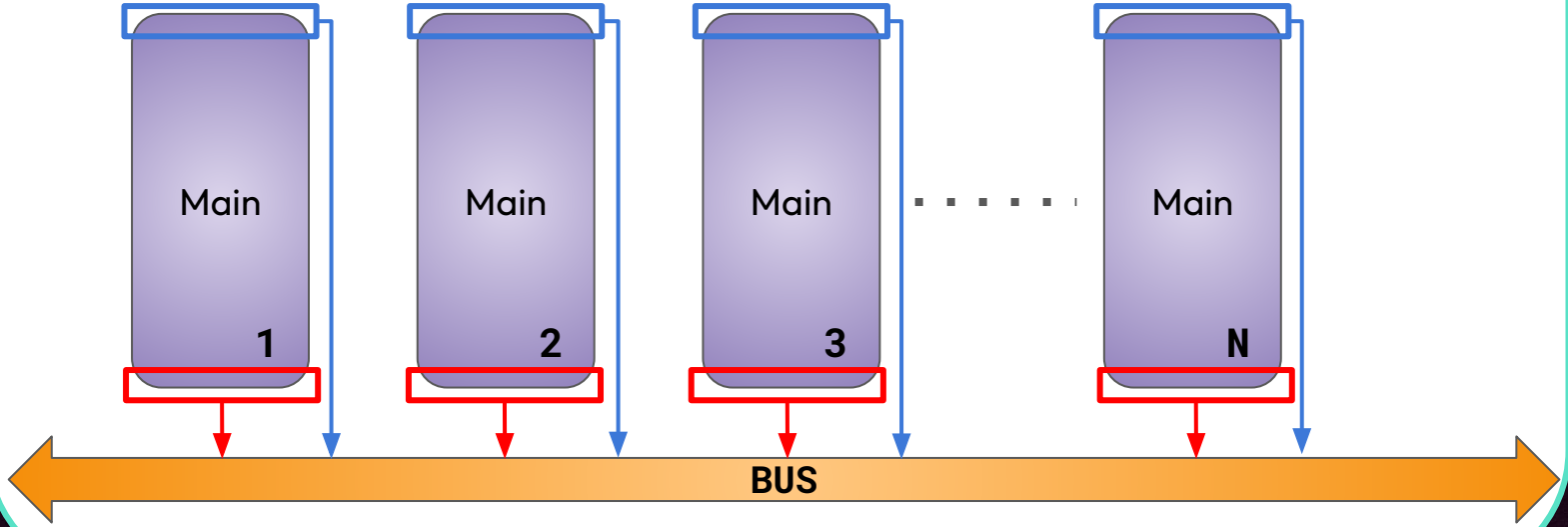


Vertical Scaling (dependent slices) ⇒ Continuations

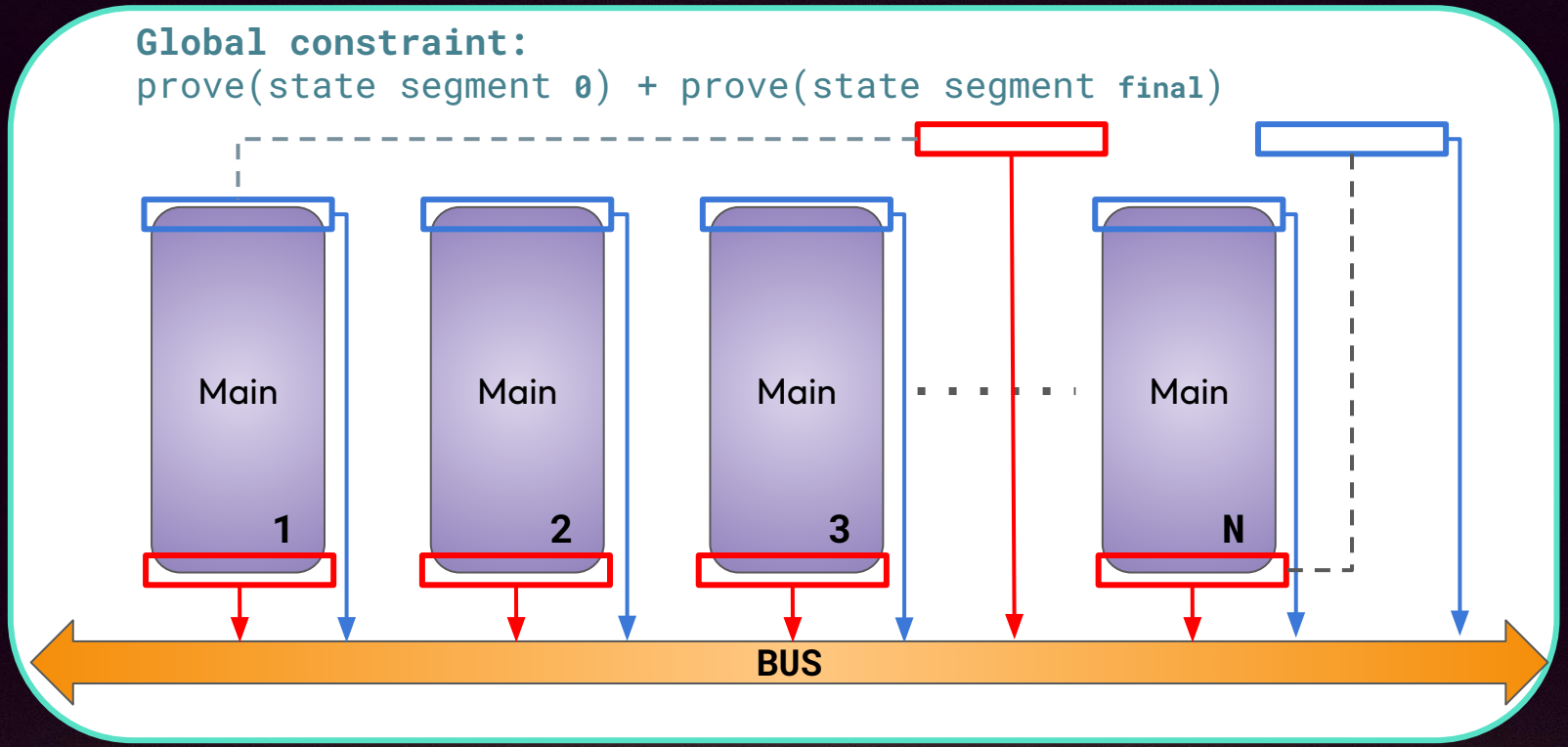


Vertical Scaling (dependent slices) ⇒ Continuations

Each segment:
`assume(state segment i) + prove(state segment i+1)`



Vertical Scaling (dependent slices) ⇒ Continuations



Continuations

- Each segment **assumes a prior state** and **proves the next**, ensuring continuity.
- **Initial and Final State Control,** with global constraints set the initial state (e.g., segment_id = 0, pc = BOOT) and assume the final state, addressing multi-cycle security concerns.
- **Segment vs. Row State.** Each segment has two distinct states, which aren't row-based but rather segment/instance states. To avoid redundant columns, we use **air values** like segment_id and pc.
- Within each segment, constraints are applied to row values and air values, allowing flexible trace continuation beyond row boundaries.
- **Unlimited Trace Continuation.** With this approach, the trace is not limited by row boundaries.

A glimpse into the future... what's next will change the game

Stay tuned for our next steps

Thank you!

Héctor Masip & Felicía Barceló

Prover Engineers, Polygon Labs

hmasip@polygon.technology

felicia@polygon.technology

@EllipticHector