# Unlocking Scalable QED with Design for Verification

Clark Barrett

(Joint work with Subhasish Mitra)

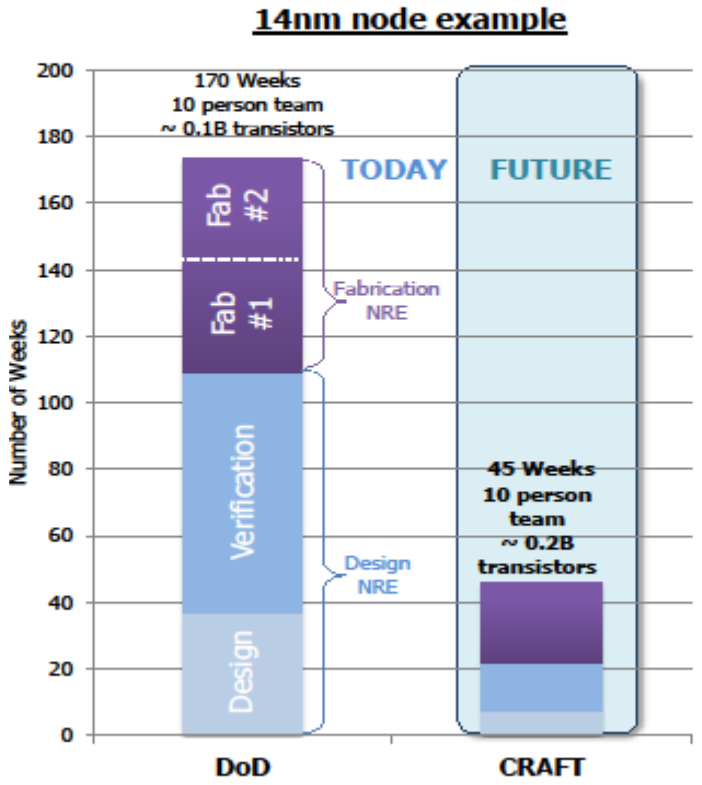
Stanford Agile Hardware Retreat

August 26, 2021

# Verification Dominates Design Time

Existing DoD custom IC product cycle time can take as long as **2.5 years**.

- 60%: Design (most of which is verification)
- 40%: Fabrication (20%/fab spin)



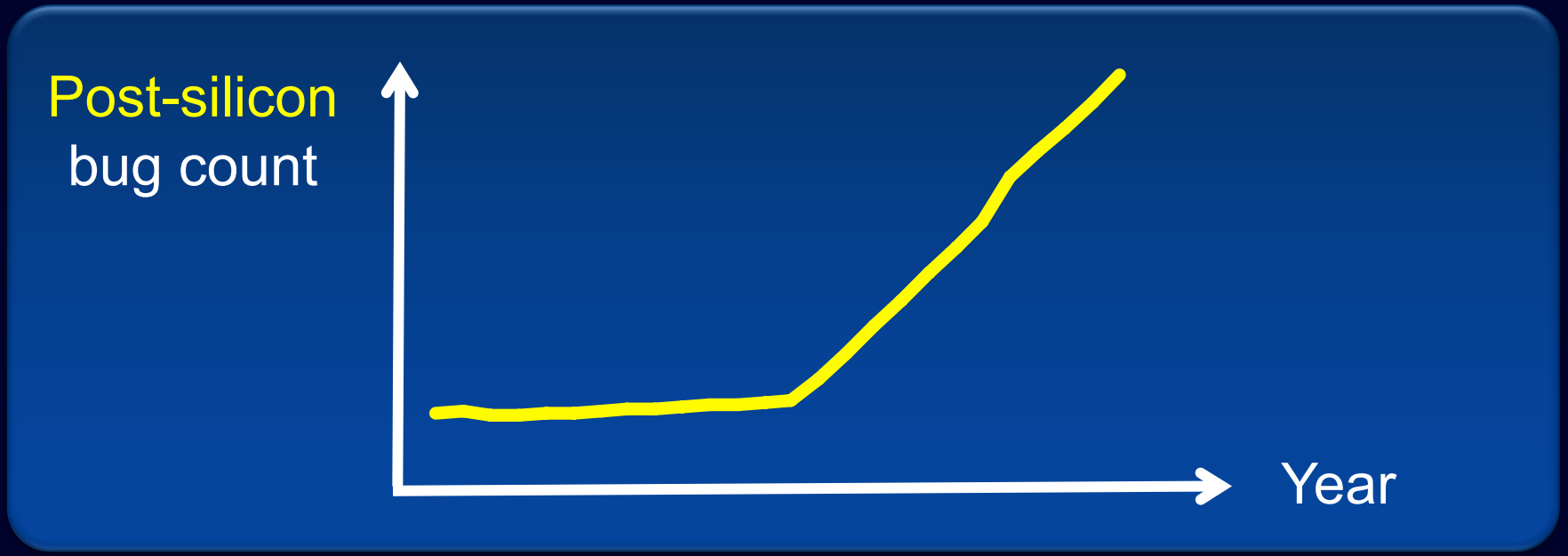

*Data from industry survey by DARPA consultants*



*Slide from DARPA CRAFT proposer's day*

# Pre-Silicon Verification Inadequate


**Getting worse: custom hardware, complexity, security**



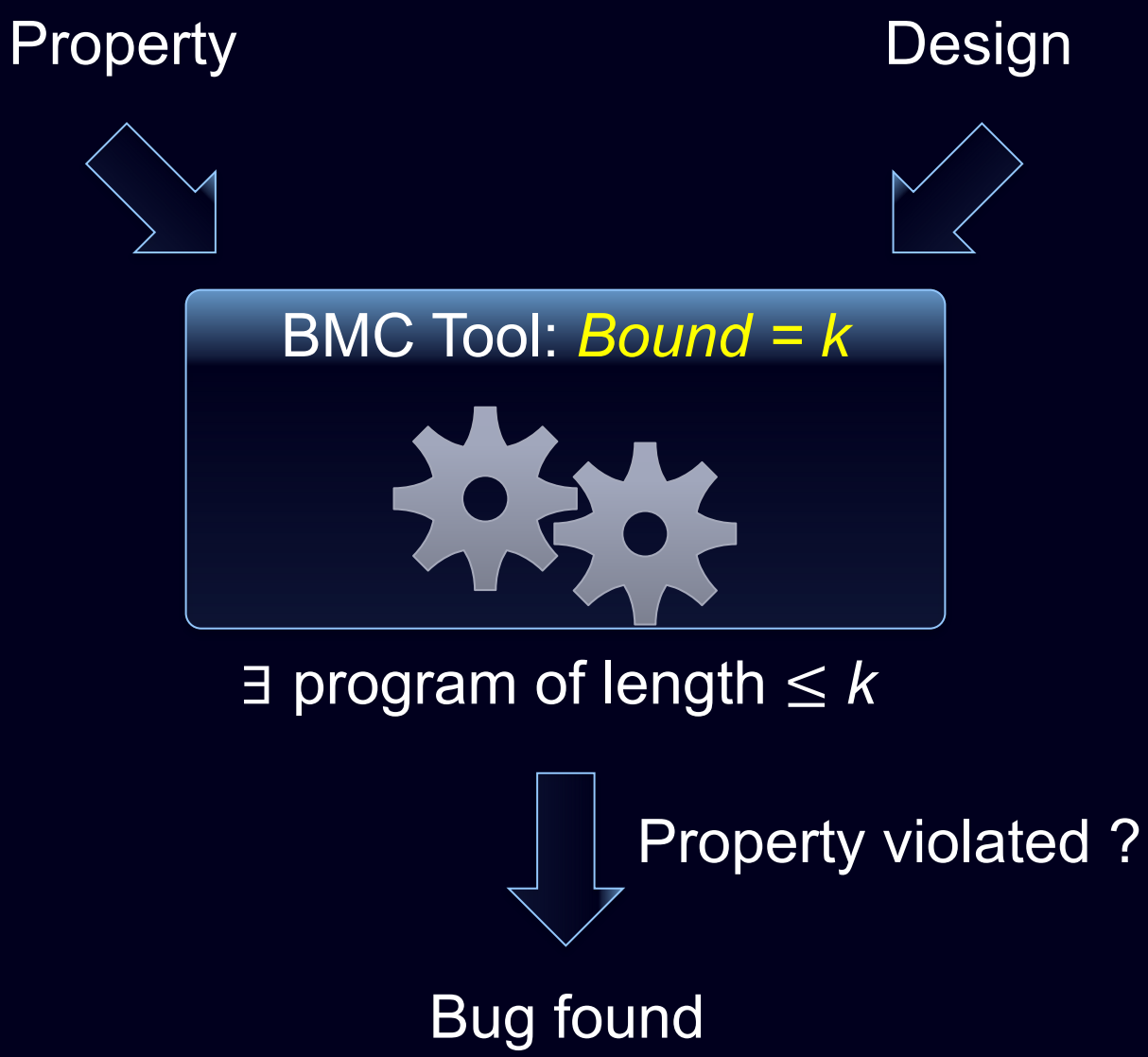

Source: Intel

# Scalability Barriers

- System-level failure reproduction
- Full system simulation

J. Stinson (ex-Intel)

# Traditional Bounded Model Checking



# “Universal” Property: QED Check

**CMP Ra == Ra’**

- Ra – original register
- Ra’ – corresponding duplicated register
- Ra ≠ Ra’ – error detected

# Symbolic QED Implementation

Chosen by Model checker

```
R1 ← R1 + 5
R2 ← R2 – R1
```

QED Module

**Interleavings covered**

```
R1 ← R1 + 5
R2 ← R2 - R1
R16 ← R16 + 5
R18 ← R18 – R16
(R1 == R16)∧
(R2 == R18)
```

Checked by Model checker

Design **+ QED Module** Instructions

Model checker: *Bound* = *k*

**QED program** length ≤ *k* **that fails ?**

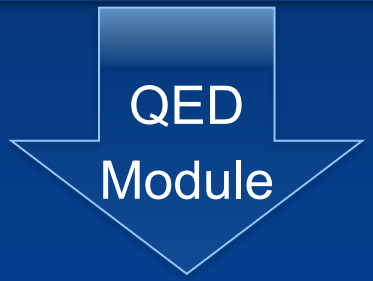
**QED Module: no hardware cost**

[Lin ITC 15, Singh IEEE TCAD 18]

# Alternative Implementation

Chosen by Model checker

```
R1 ← R1 + 5
R2 ← R2 – R1
```




**Interleavings covered**

```
R1 ← R1 + 5
R2 ← R2 - R1
R16 ← R16 + 5
R18 ← R18 – R16
(R1 == R16)∧
(R2 == R18)
```

Checked by Model checker

```
Reset
R1 ← R1 + 5
R2 ← R2 - R1
Checkpoint C
Reset
R1 ← R1 + 5
Soft Reset
R2 ← R2 – R1
(R1@C == R1)∧
(R2@C == R2)
```

[Lonsing, FMCAD ‘20]

# How To Implement?

- Checkpoint
  - Formal tool has simultaneous view of all time steps
- Soft Reset (Requires Design Support)
  - Reset microarchitectural state but leave architectural state unchanged
- Strong Correctness Guarantees
  - But Scalability still an Issue

# Symbolic Starting States

Chosen by Model checker

```
R1 ← R1 + 5
R2 ← R2 – R1
```

QED Module

**Interleavings covered**

Checked by Model checker

```
R1 ← R1 + 5
R2 ← R2 - R1
R16 ← R16 + 5
R18 ← R18 – R16
(R1 == R16)∧
(R2 == R18)
```

```
Symbolic State S1
R1 ← R1 + 5
R2 ← R2 - R1
Checkpoint C
Symbolic State S2
R1 ← R1 + 5
Soft Reset
R2 ← R2 – R1
(Arch@S1 == Arch@S2) ->
(R1@C == R1)∧
(R2@C == R2)
```

# How To Implement?

- Arch@S1 == Arch@S2
  - Formal tool has simultaneous view of all time steps
  - Designs need to have clean separation of architectural/non-architectural state
- Symbolic State
  - What about invalid/unreachable states?
  - Can we make designs that are QED-compatible, even for invalid states?

# A-QED for Hardware Accelerators

1. Loosely-coupled accelerators

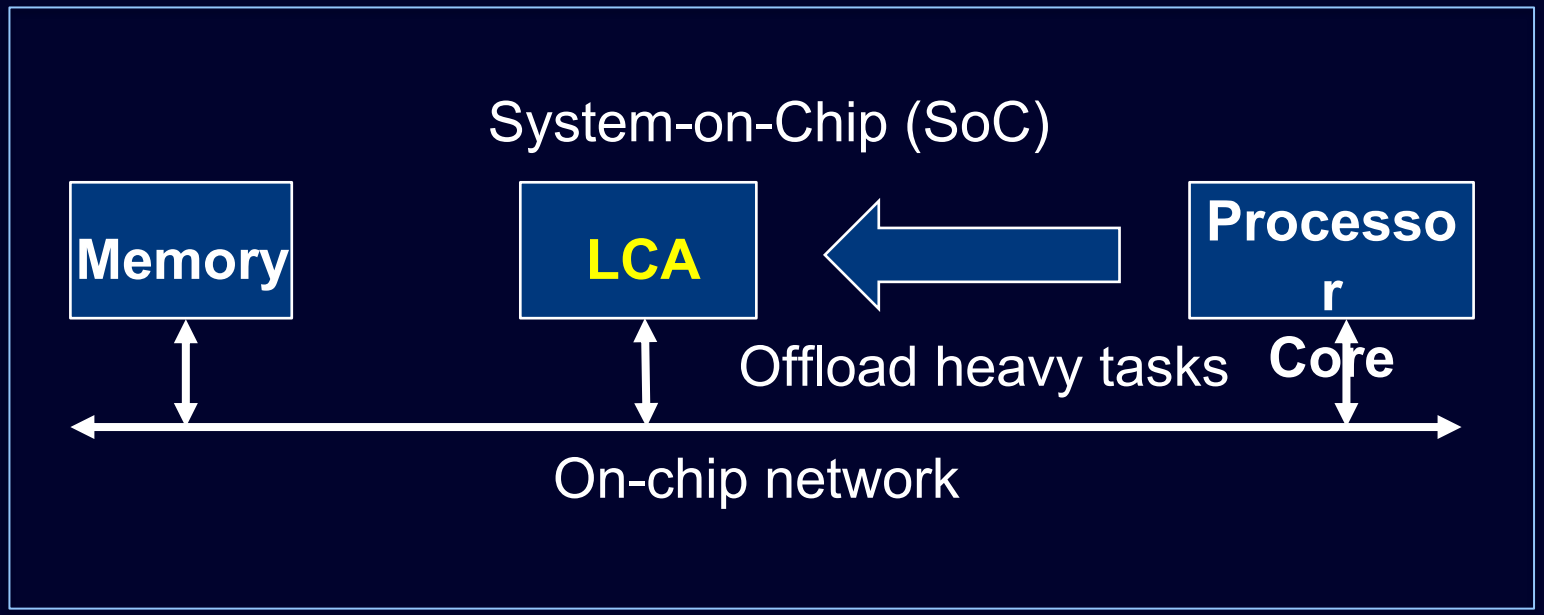
2. Non-interfering execution

Ongoing work: expand A-QED for other classes



[Singh, DAC ‘20, Chattopadhyay, FMCAD ‘21]

# Loosely Coupled Accelerators



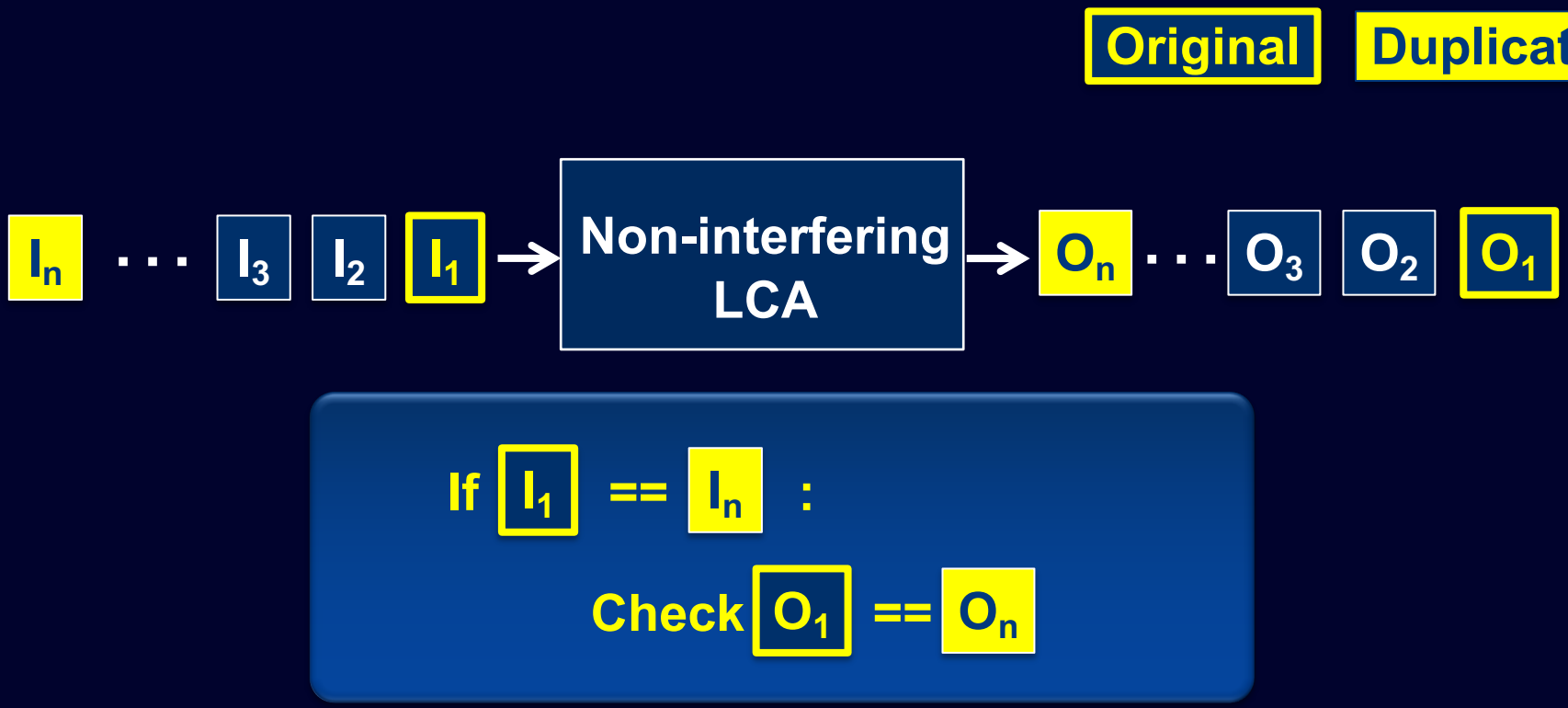
# Non-Interfering LCAs


$\forall$ j, $O_j$ = f( $I_j$ )

Value of $O_j$ independent of any other inputs

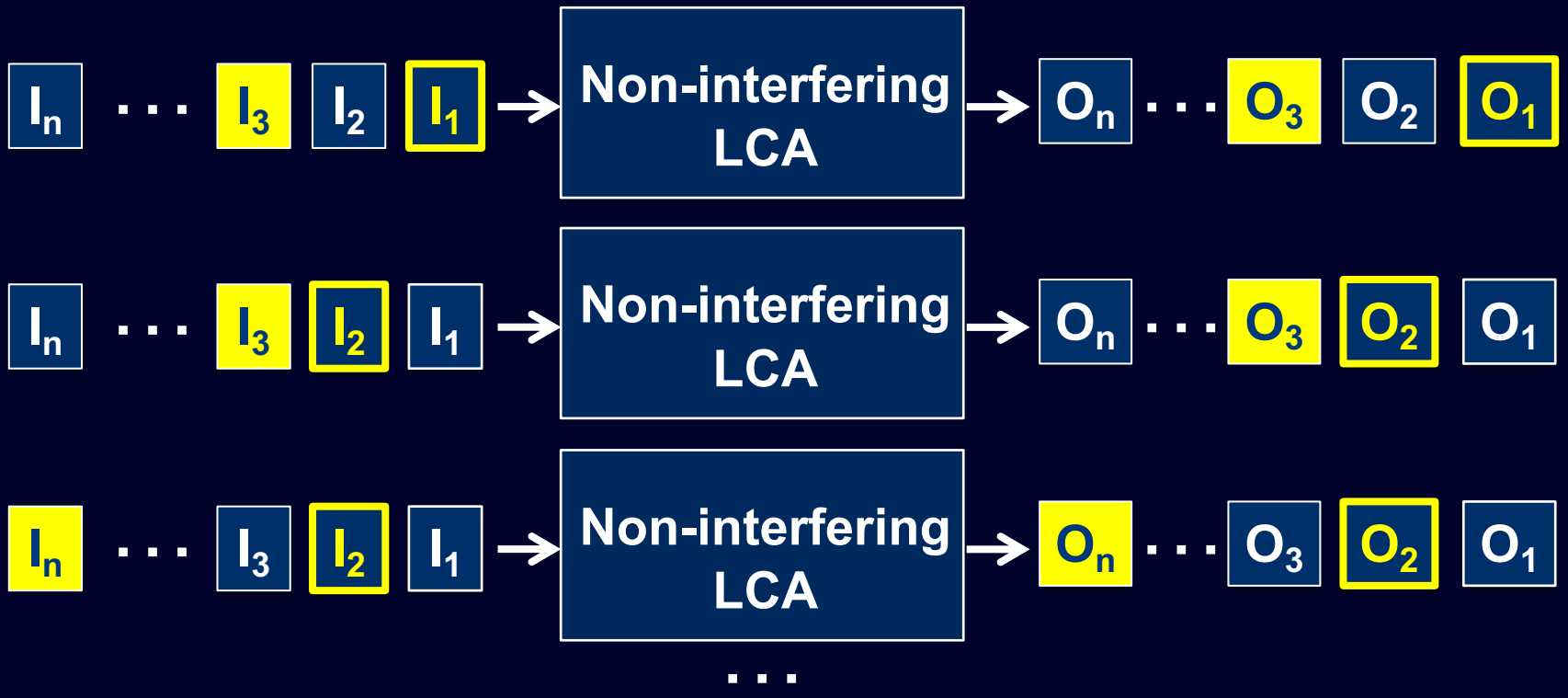
Non-interfering accelerators ≠ combinational circuits

# Functional Consistency

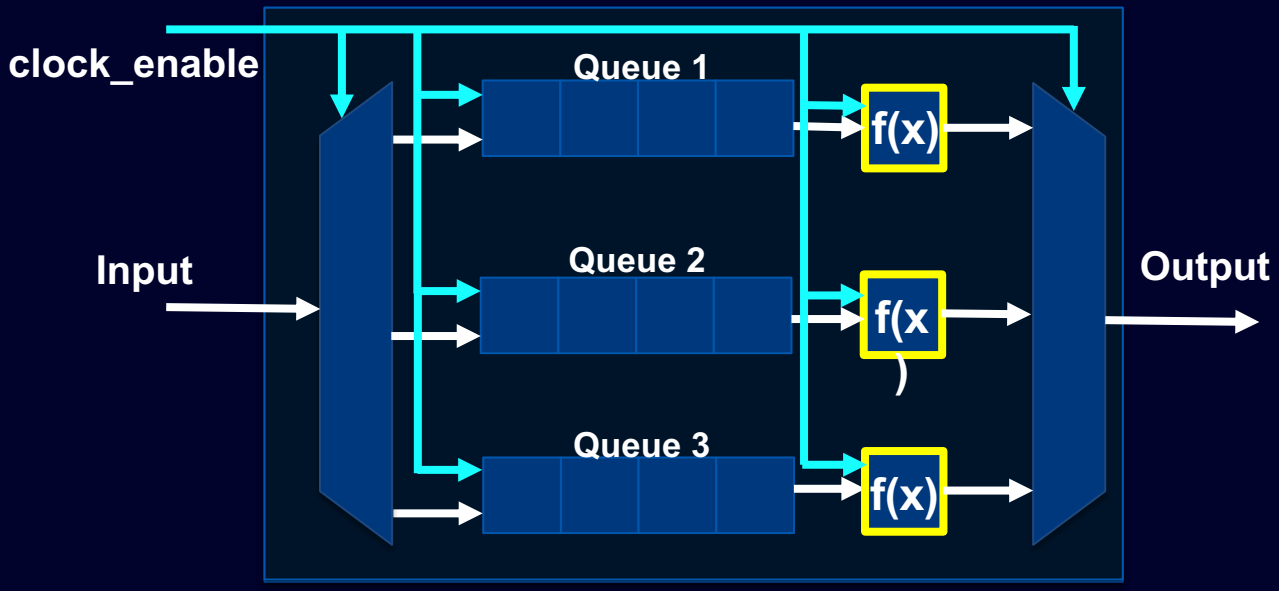


# Functional Consistency

∀ possible interleavings of **Original** **Duplicate**



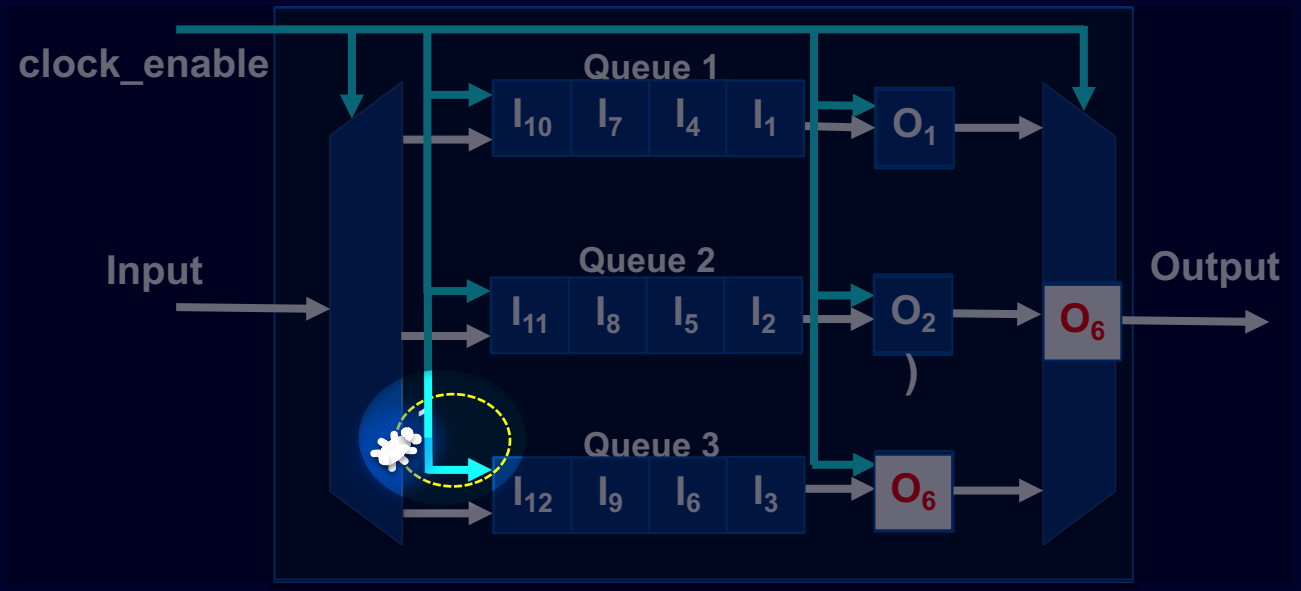
# LCA Example




- 3 internal queues, 3 execution units

# Bug Example

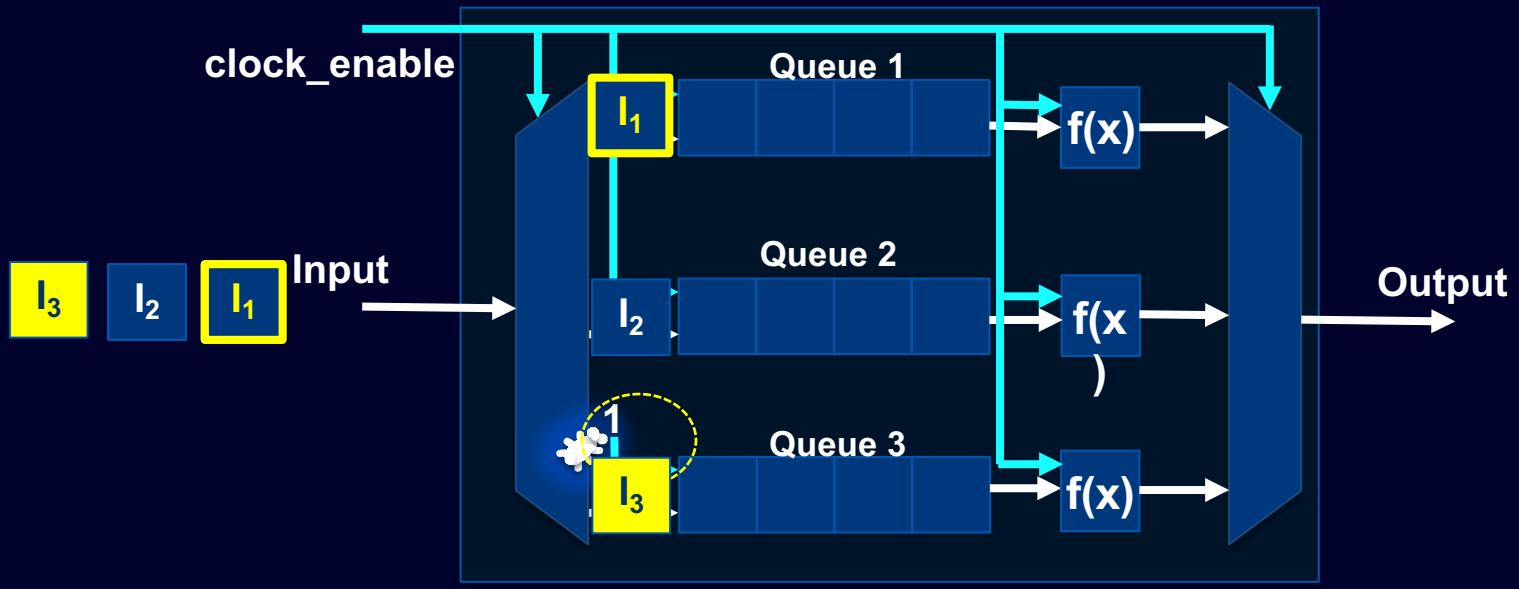
- **Bug:** Queue 3 always enabled




- $I_3$ pushed to f(x) **but not latched**
- $O_6$ **produced instead of** $O_3$

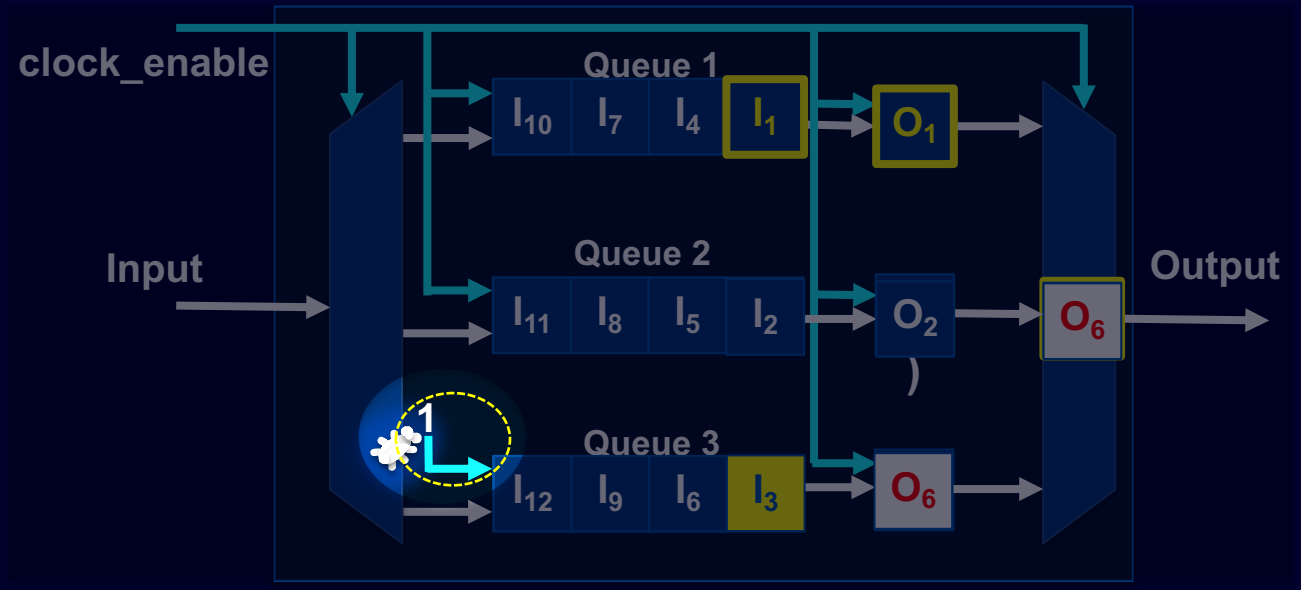
# A-QED: Functional Consistency

Functional Consistency Example **Original** **Duplicate**



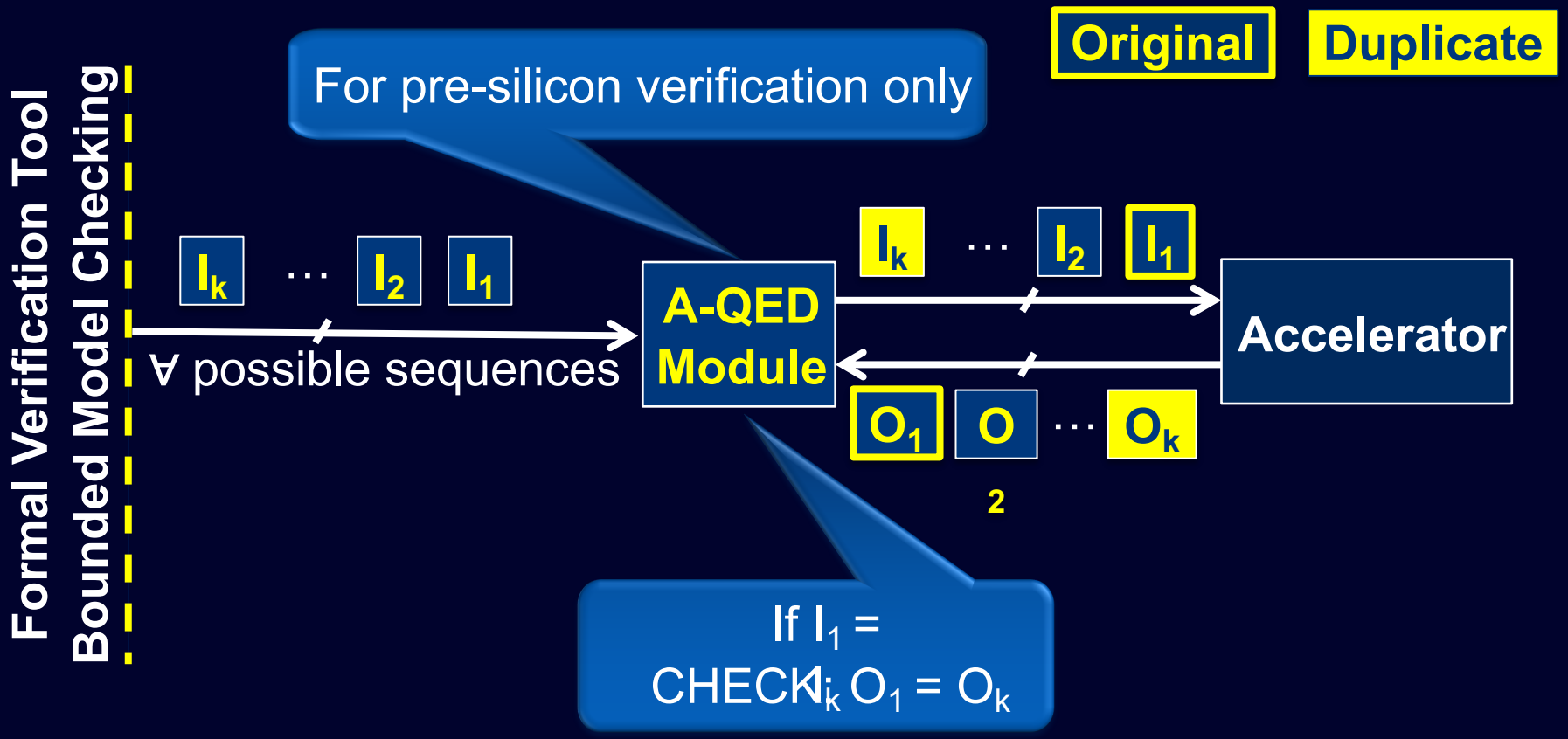

**If $I_1 = I_3$: expect $f(I_1) = f(I_3)$**

# A-QED: Functional Consistency




$I_1 \neq I_6$, therefore $O_1 \neq O_6$ BUG DETECTED

# A-QED Setup


Original
Duplicate
For pre-silicon verification only
Formal Verification Tool
Bounded Model Checking
$I_k$ … $I_2$ $I_1$
∀ possible sequences
A-QED Module
$I_k$ … $I_2$ $I_1$
Accelerator
$O_1$ $O_2$ … $O_k$
If $I_1$ = … = $I_k$ CHECK $O_1$ = $O_k$

# Need For Decomposition

- General challenge: A-QED scalability limited by large design sizes.
- Compositional verification: check correctness of sub-modules.
- Traditional techniques: complex setup, assumptions, properties.

# A-QED$^2$: A-QED with Decomposition

- Functional consistency is inherently compositional
- Designs consist of functional sub-modules: sub-accelerators
- Sub-accelerators produce partial outputs
- Functional decomposition of Acc in $Acc_1$ and $Acc_2$:
  - Input I and output O of Acc
  - $I = I_1 \rightarrow Acc_1 \rightarrow O_1 = I_2 \rightarrow Acc_2 \rightarrow O_2 = O$

[Chattopadhyay, FMCAD '21]

# A-QED$^2$: A-QED with Decomposition

- **Unique Design for Verification opportunity**
  - support A-QED$^2$ design decomposition
- Break computation into chunks, vertically or horizontally
- Potential integration in HLS workflows
- Decomposition difficult for conventional formal verification
  - Need to rethink properties (manually), false fails

# Design for Verification

- How ready are designers to prioritize verification above other goals?
  - How do we better trade-off between verification goals and other goals?
  - How important is DfV in an agile design flow?

- Where are the sweet spots?  Big ROI?
  - Making a design that decomposes easily
  - Adding soft-reset capabilities
  - Adding logic to simplify reasoning about symbolic starting states

# THANK YOU