Towards Simulator-like Observability for FPGAs:

A Virtual Overlay Network for Trace-Buffers

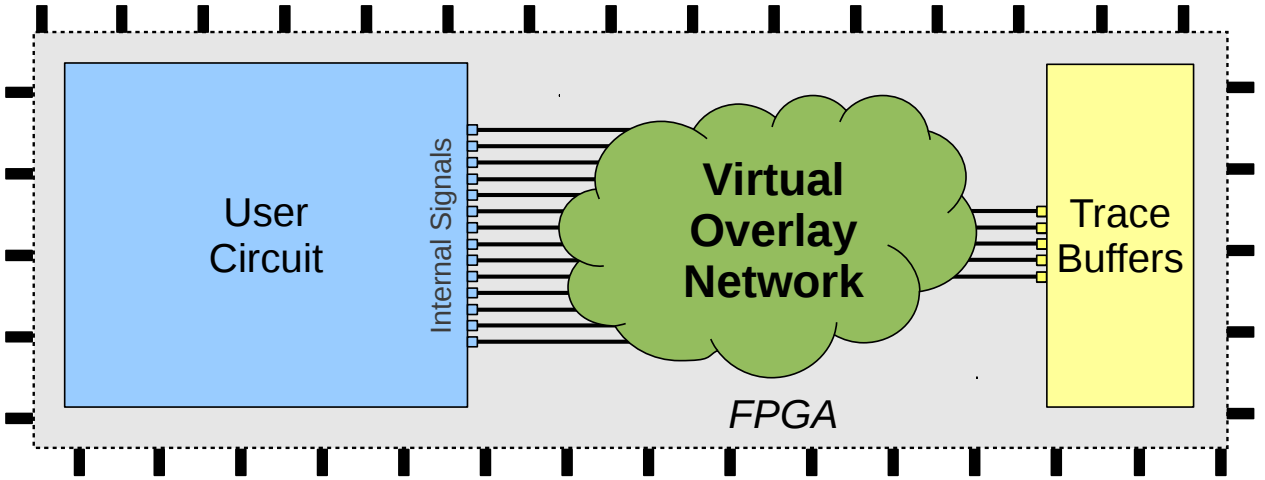
Eddie Hung, Steven J. E. Wilton
{eddieh,stevew}@ece.ubc.ca

University of British Columbia
Vancouver, Canada

FPGA :: Feb 12 2013

What this talk is about

- A network that multiplexes *all* on-chip circuit signals to a limited number of trace-buffers for rapid debug



- Key Novelties:
  1) Realized using an efficient network architecture
  2) Built using spare routing multiplexers in Switch-Blocks
  3) CAD Algorithm that can rapidly configure this
- Outcome: 10,000s observed signals for zero overhead

Introduction

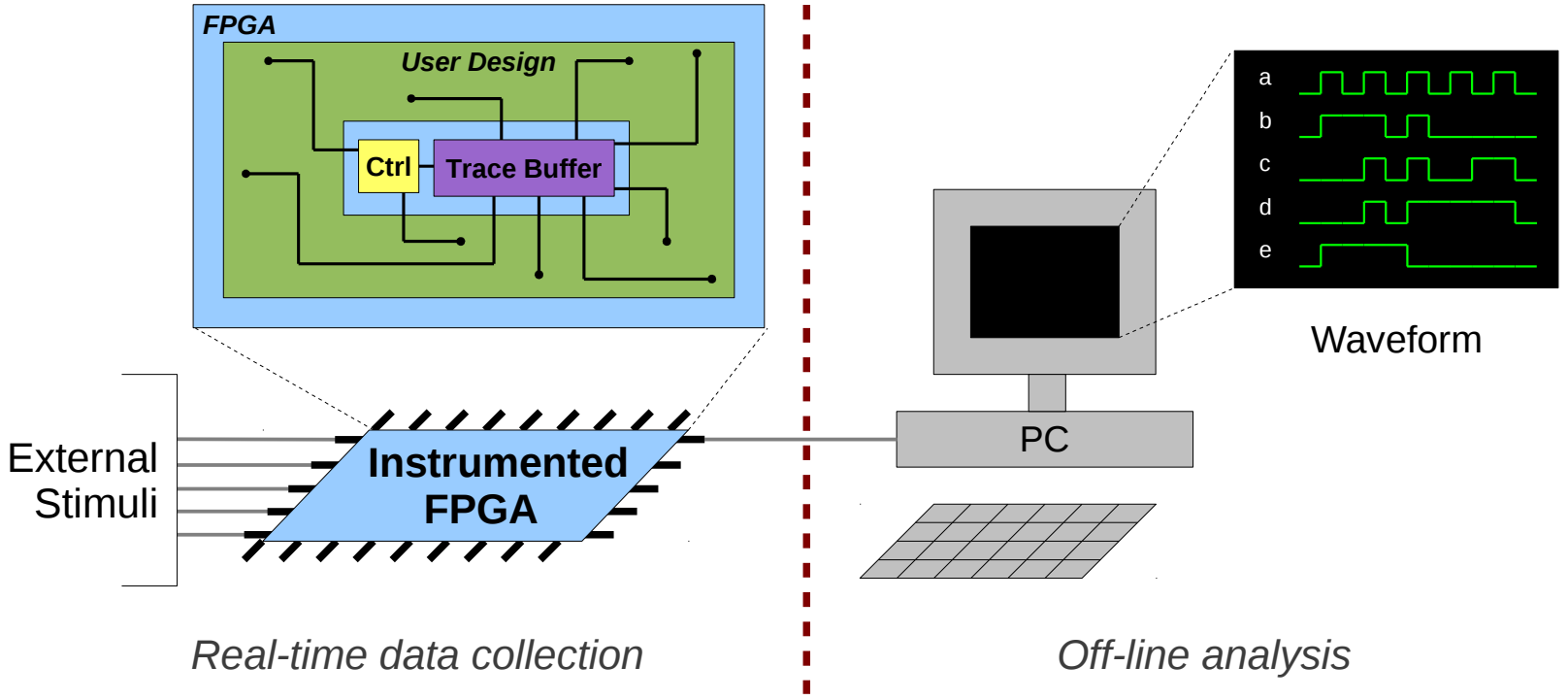
- Debug is the process of locating and eliminating design errors – 'bugs' – in ICs
- Important as mistakes in silicon cost **big** money
  - 2007: AMD K10 TLB bug – 4 months for re-spin
  - 2011: Intel 'Sandy Bridge' chipset – $700 mil. recall

Introduction

- Pre-Silicon techniques alone are insufficient
  - Software simulation effective, but slow
    - Intel: Core i7 (2.6 GHz) simulates at 2-3Hz
  - Formal verification limited to small components
  - Unable to interact with real-world stimulus
- FPGA prototypes -- fast and physical
  - Instant circuit fabrication: quick turnaround
  - Runs at near-speed: increased coverage

Primary Challenge: Observability

- Insert small amount of debug circuitry in design
  - Sample subset of signals into on-chip memory
  - Capture a sequence of states, at full speed



Introduction: Trace-Buffers

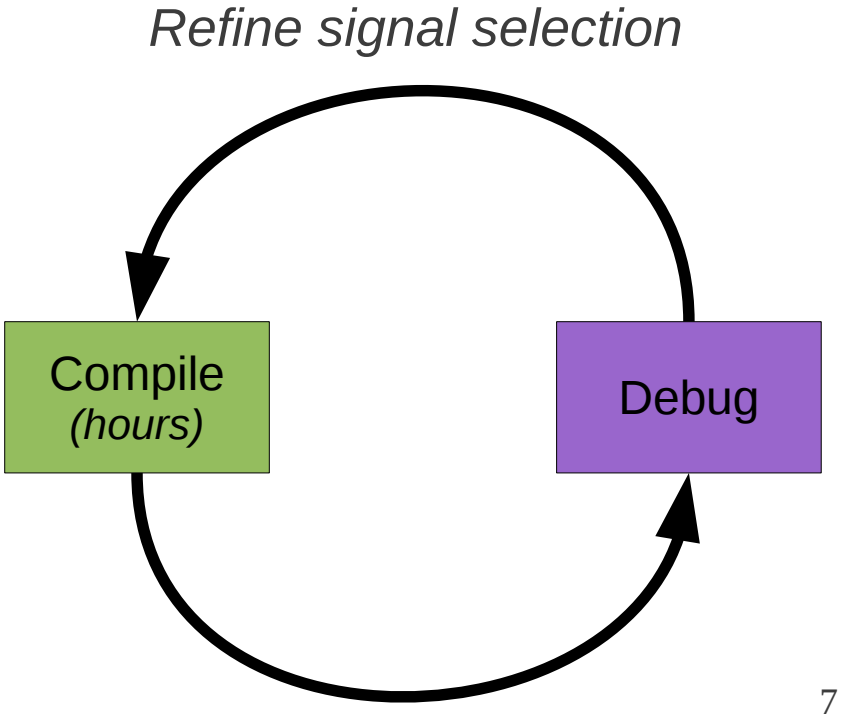
- Insert small amount of debug circuitry in design
  - Sample subset of signals into on-chip memory
  - Capture a sequence of states, at full speed
- Trace IP:
  - Xilinx ChipScope Pro, Altera SignalTap II, Tektronix Certus

Introduction: Trace-Buffers

- Insert small amount of debug circuitry in design
  - Sample **subset** of signals into on-chip memory
  - Capture a sequence of states, at full speed

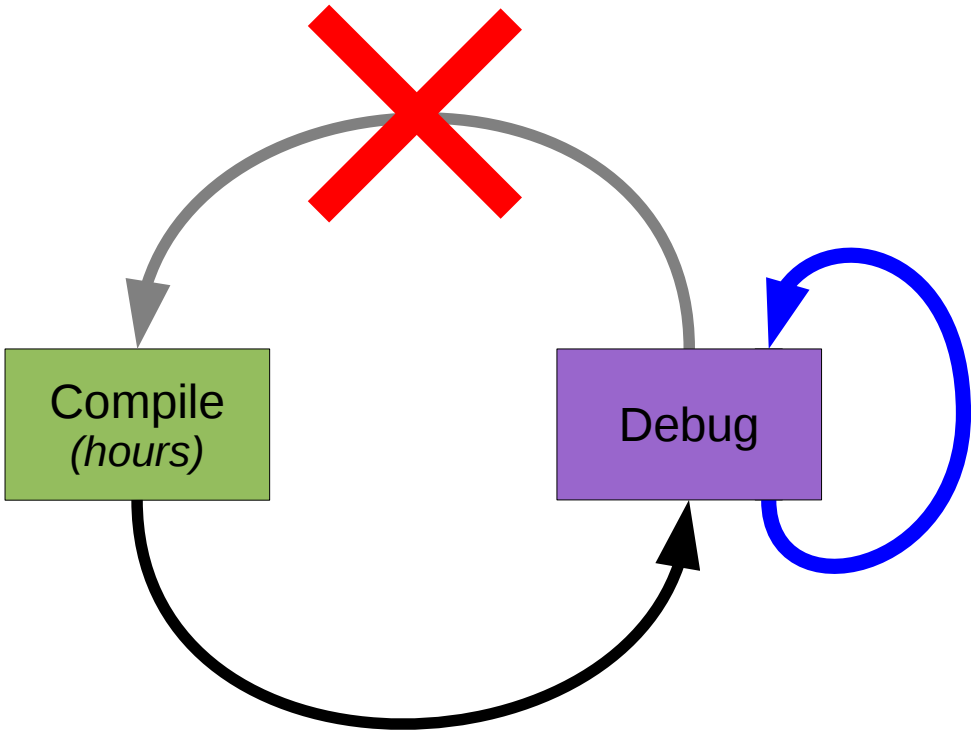
But every time you want to change this subset, must recompile!

Many iterations required to root-cause bugs



Long term vision: Simulator-like Observability

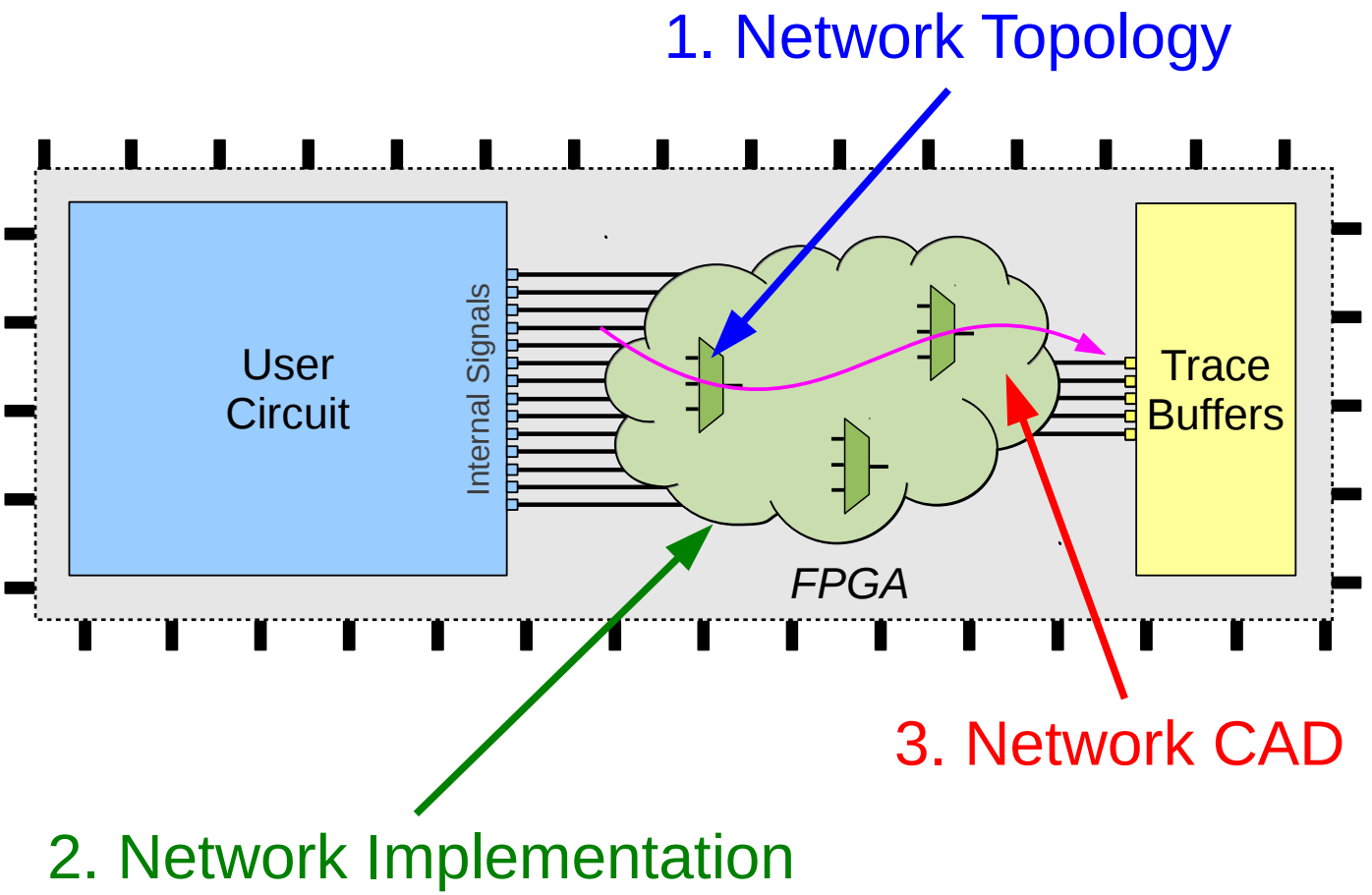
- What do we mean by this?
  - **No more re-compilation** to change signals



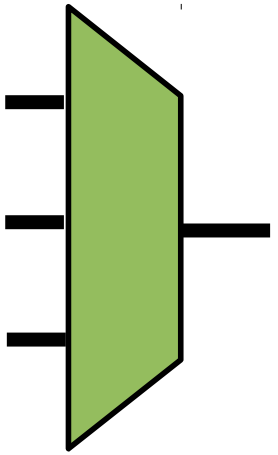
Refine signal selection
(seconds)

Our Solution

- Composed of three novelties:

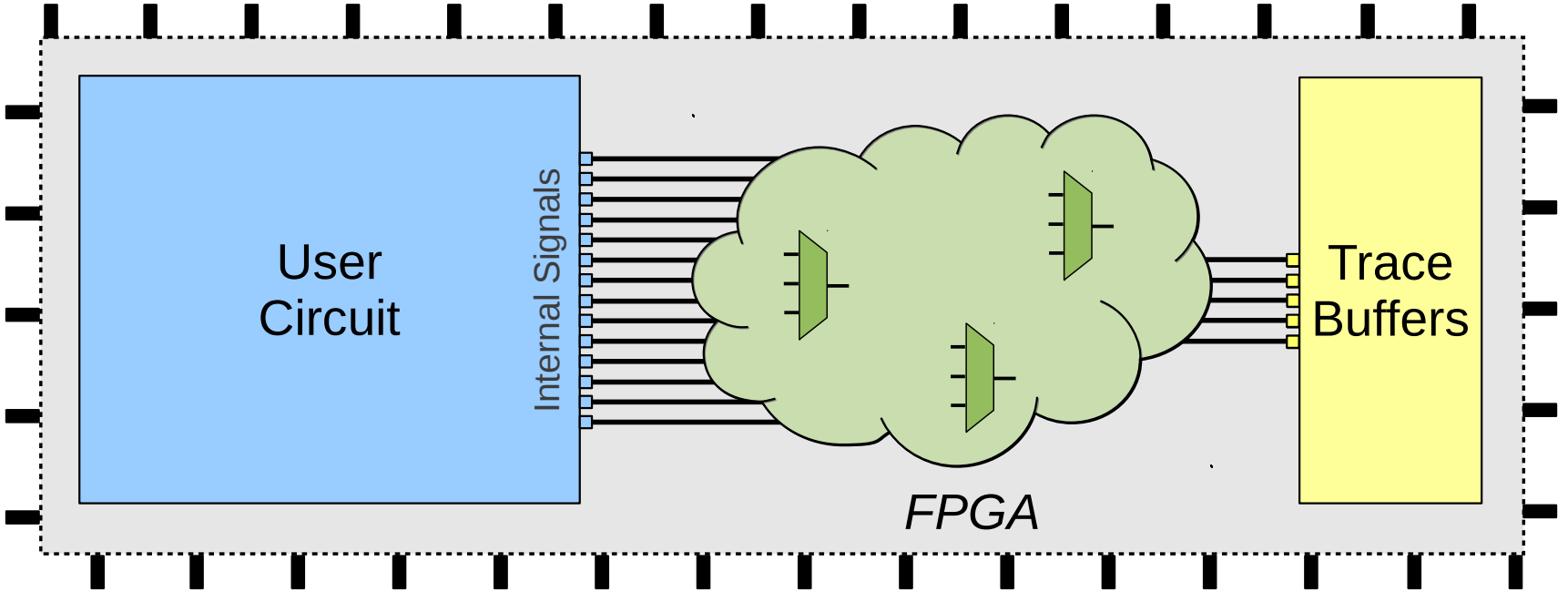


Novelty 1:
Network Topology



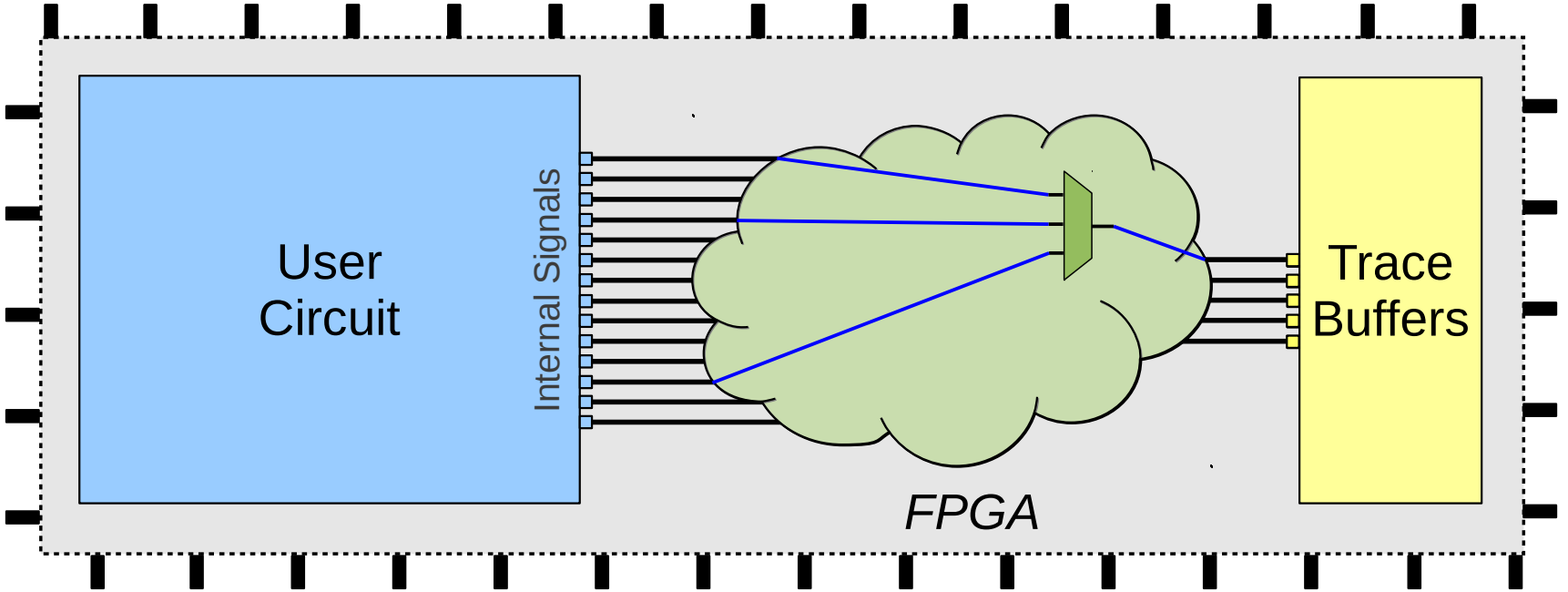
Overlay Network Topology

- Big Idea: build a network which multiplexes all user signals to trace-buffer inputs



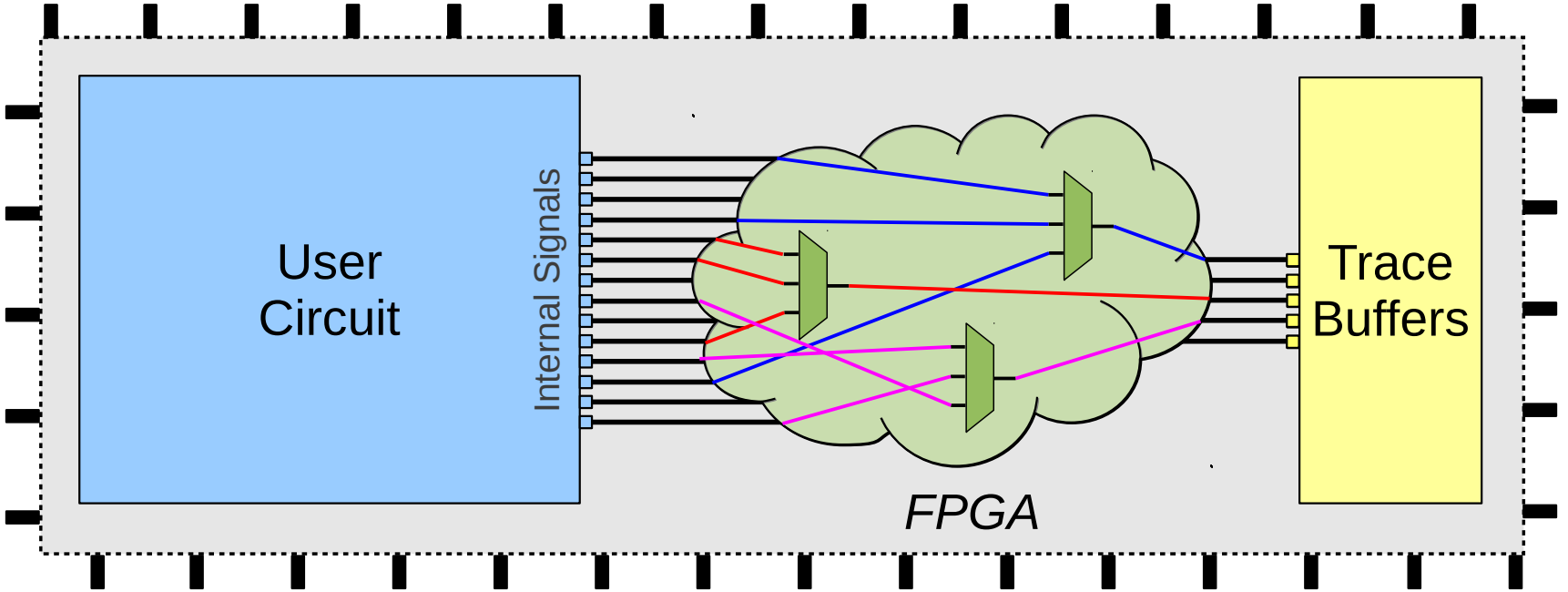
Overlay Network Topology

- Big Idea: build a network which multiplexes all user signals to trace-buffer inputs



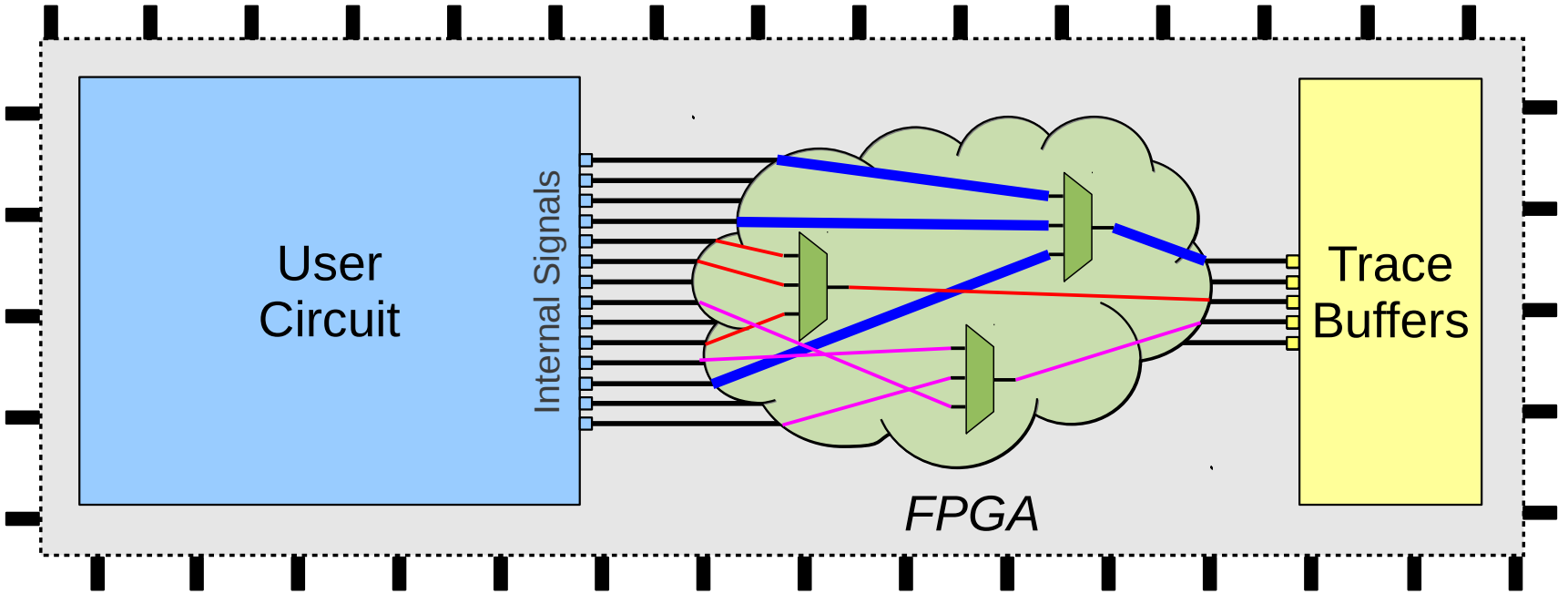
Overlay Network Topology

- Big Idea: build a network which multiplexes all user signals to trace-buffer inputs

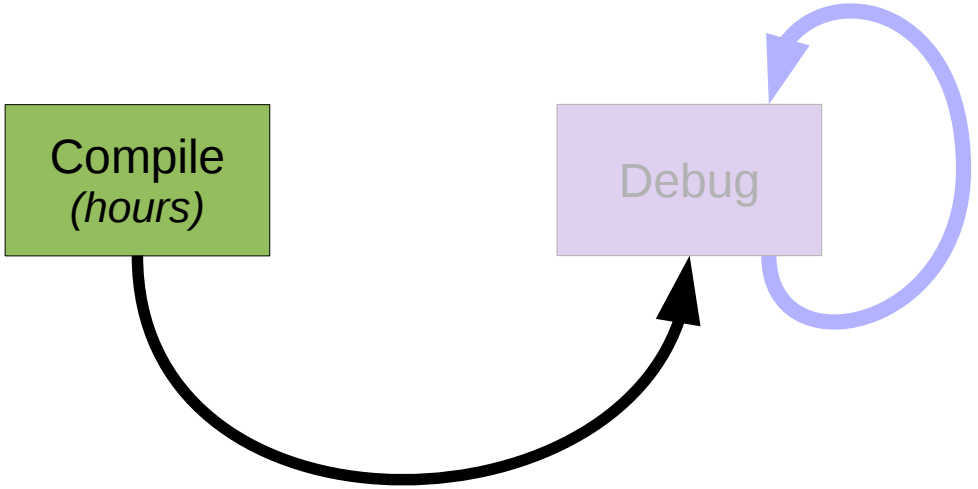


Overlay Network Topology

- Key characteristics:
  - Each trace-buffer input can select signals **independently** of all other trace-inputs
  - Detailed connections of network ⇒ Novelty 2

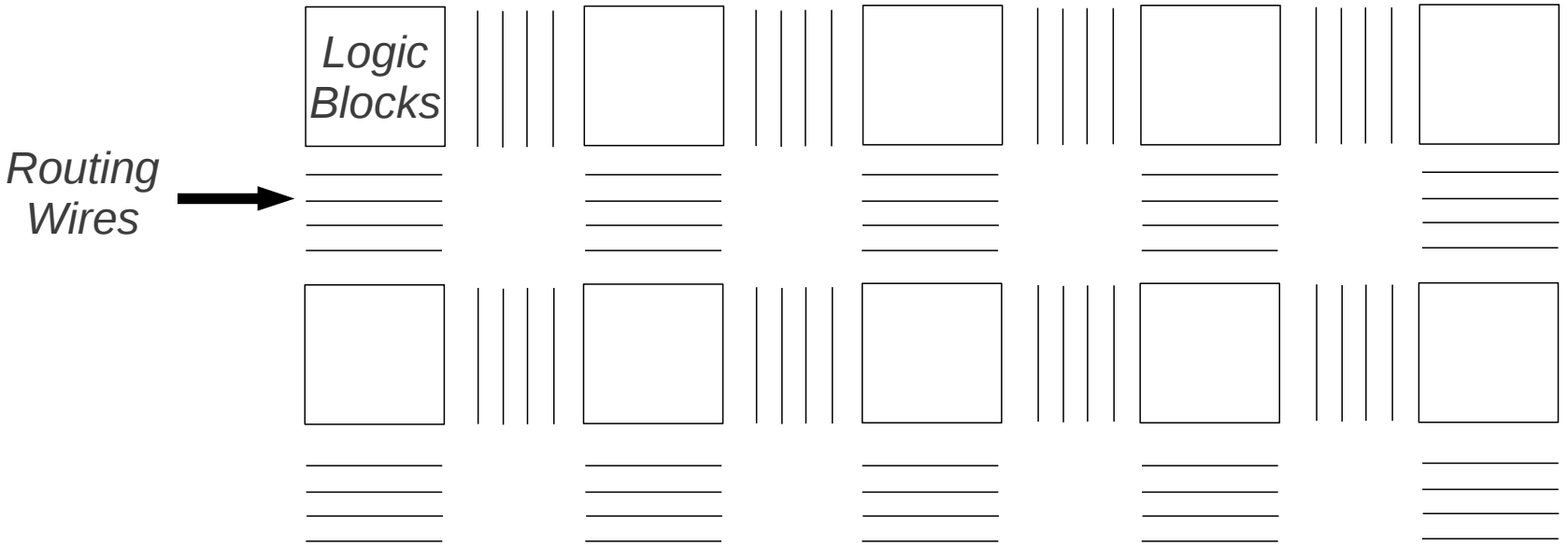


Novelty 2:
Network Implementation



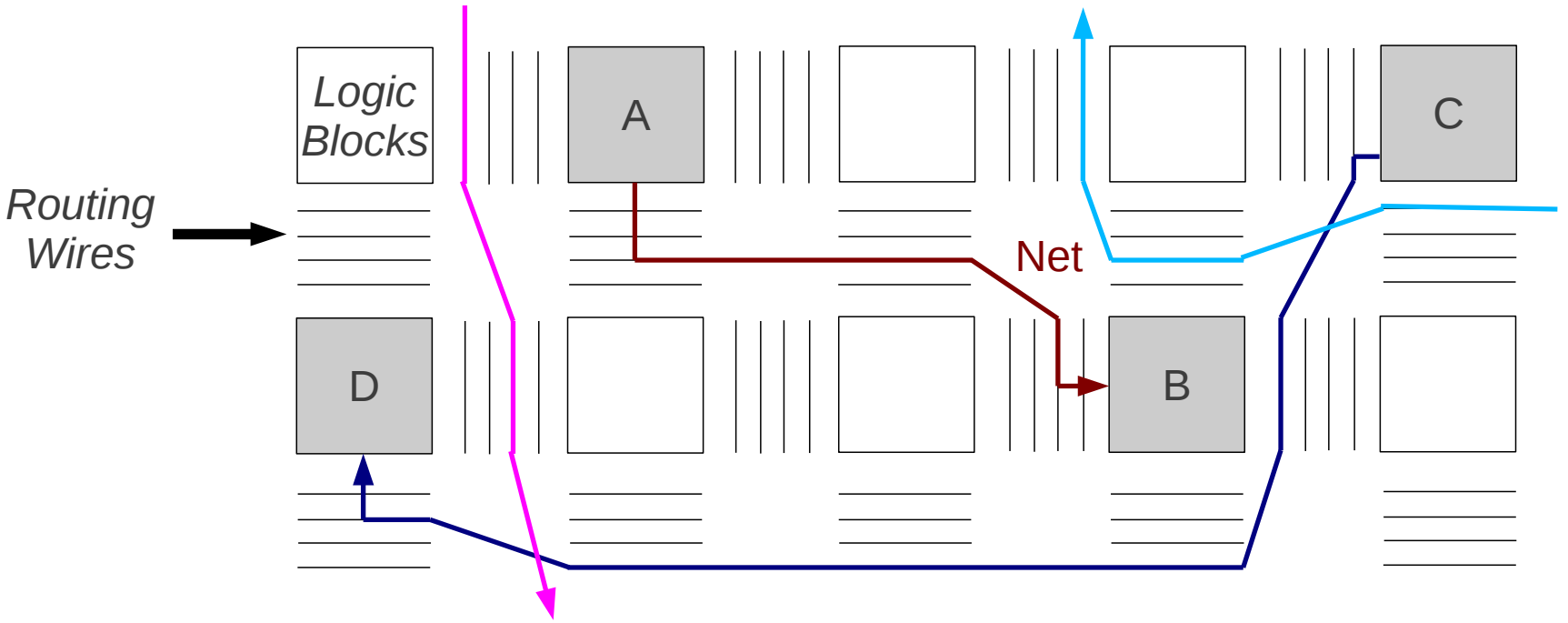
Overlay Network Implementation

- Construct a network connecting **all** on-chip signals to TBs (every used LUT/FF, RAM, DSP)



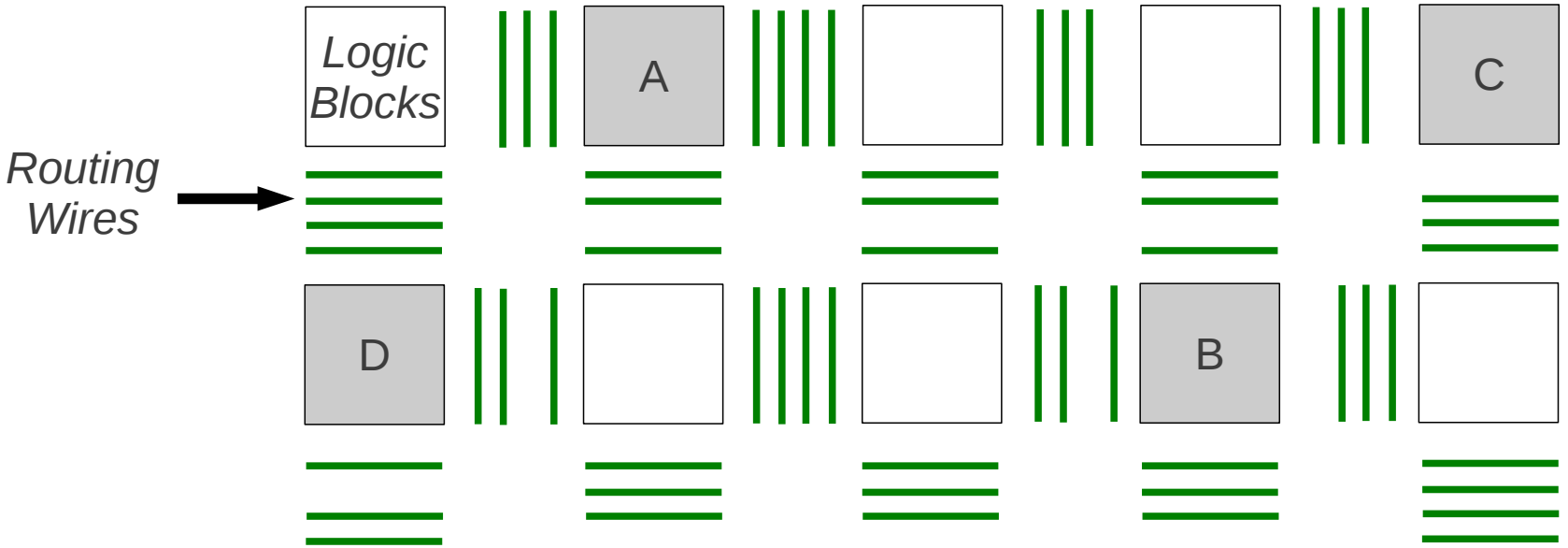
Overlay Network Implementation

- Construct a network connecting **all** on-chip signals to TBs (every used LUT/FF, RAM, DSP)
  - First, compile user circuit as normal



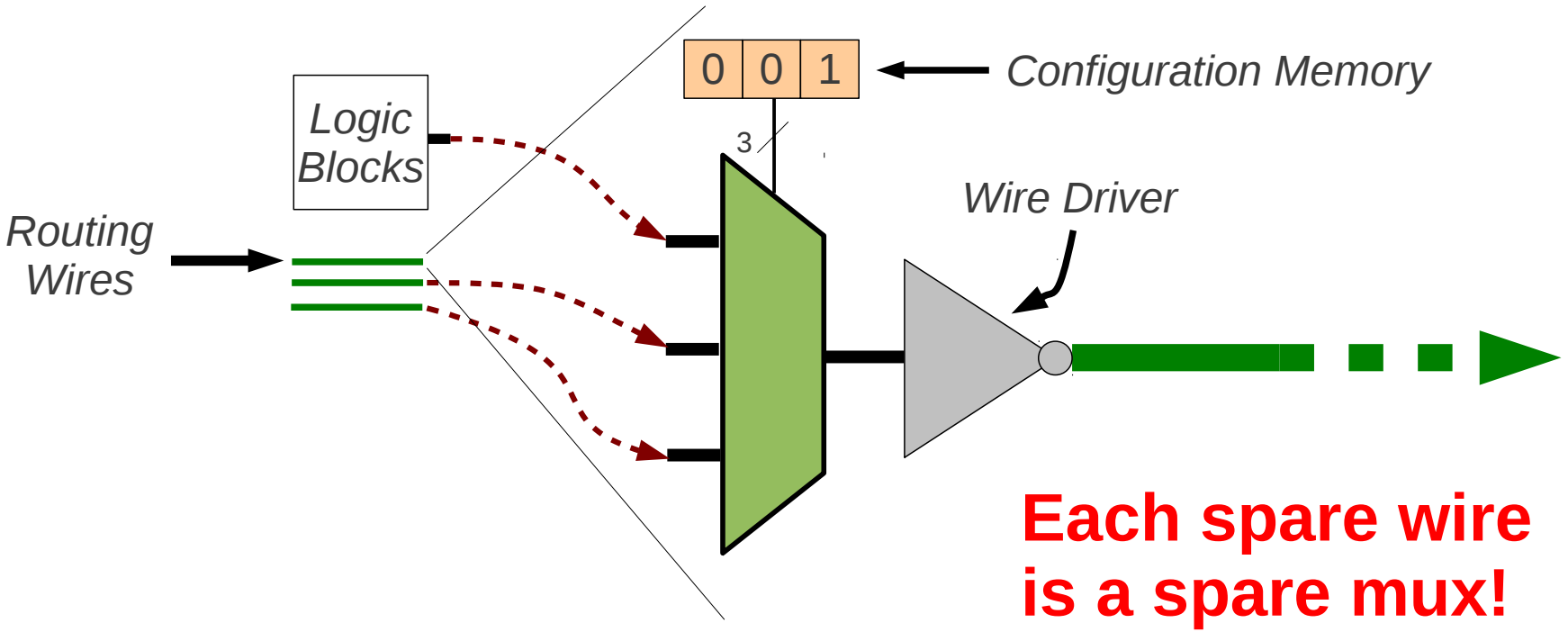
Overlay Network Implementation

- Construct a network connecting **all** on-chip signals to TBs (every used LUT/FF, RAM, DSP)
  - Insert network *incrementally*: without affecting user circuit, utilizing spare routing resources left behind



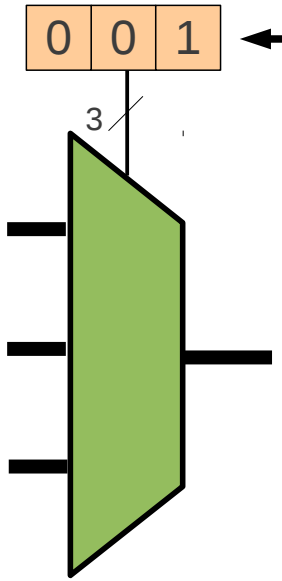
Overlay Network Implementation

- Construct a network connecting all on-chip signals to TBs (every used LUT/FF, RAM, DSP)
  - Insert network *incrementally*: without affecting user circuit, utilizing spare routing resources left behind



Overlay Network Implementation

- Construct a network connecting all on-chip signals to TBs (every used LUT/FF, RAM, DSP)
  - Insert network *incrementally*: without affecting user circuit, utilizing spare routing resources left behind

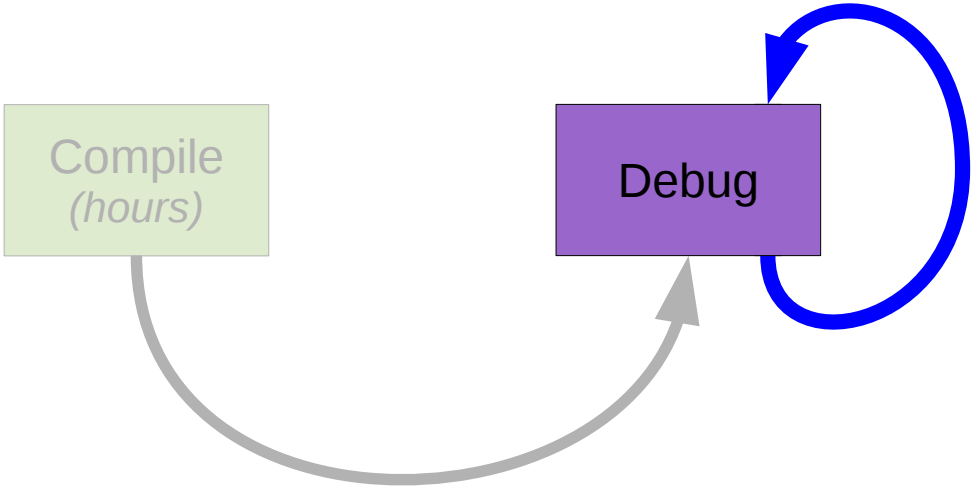


Configuration Memory

Modifiable using:

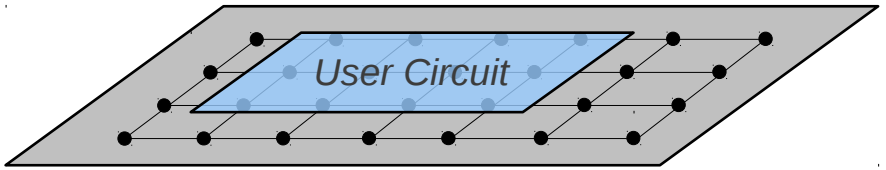
- Dynamic Partial Reconfiguration
  - Altera claims to be able to do this...
- Editing Static Bitstream and Programming
  - Academic tools exist to do this (vendor tools *should* too!)
  - No place-and-route necessary

Novelty 3:
Network CAD



Overlay Network CAD

- Previously...

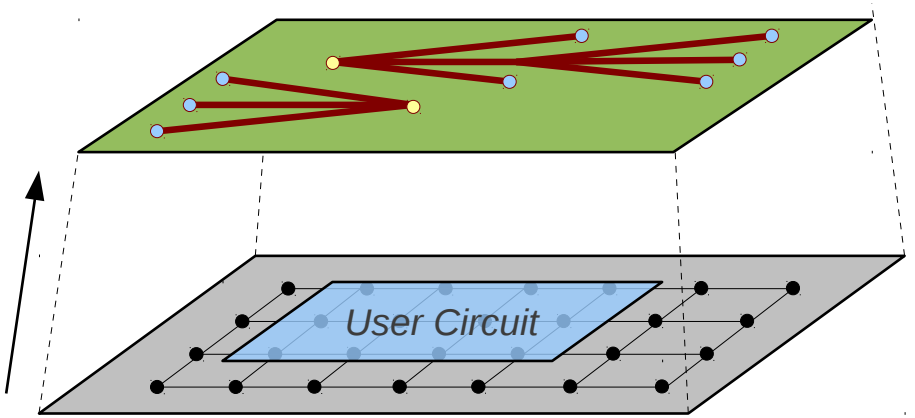


FPGA Routing Resources

Overlay Network CAD

- Previously...

Compile **once**

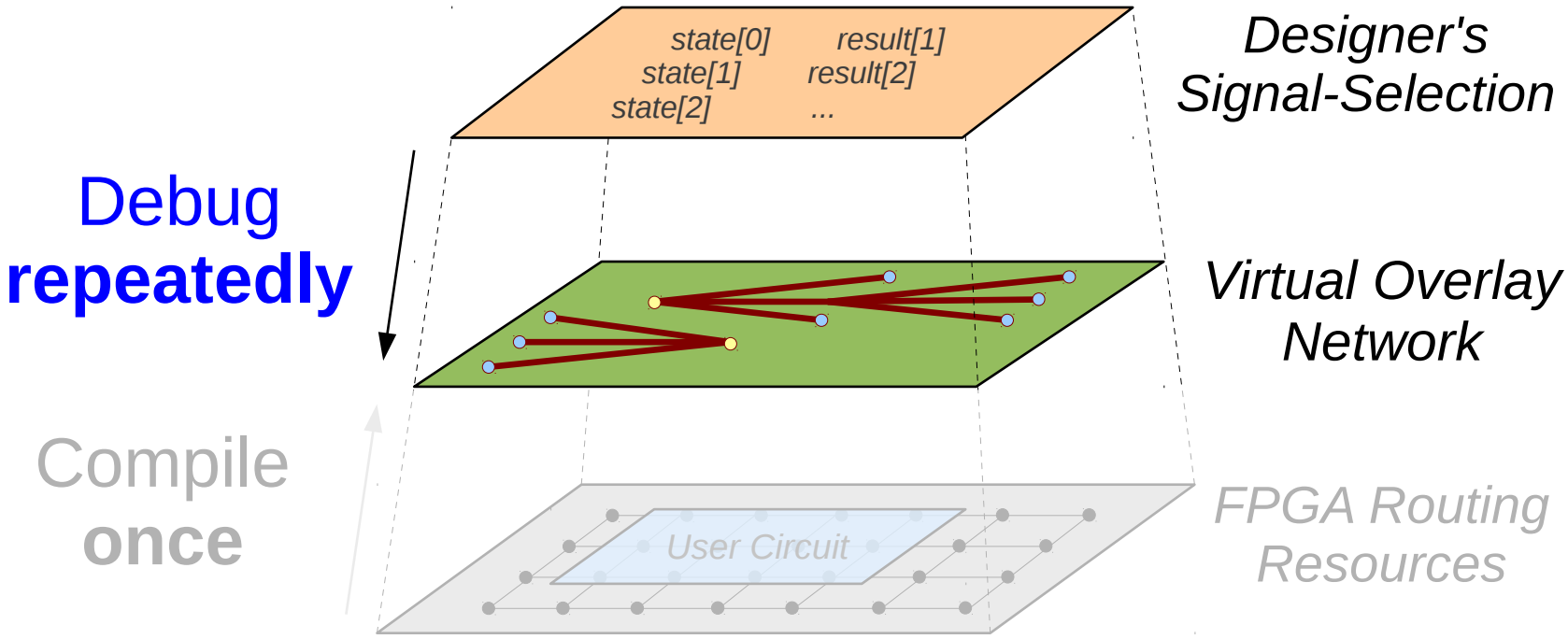


Virtual Overlay Network

FPGA Routing Resources

Overlay Network CAD

- Routing multiplexers must be configured for each new set of trace signals

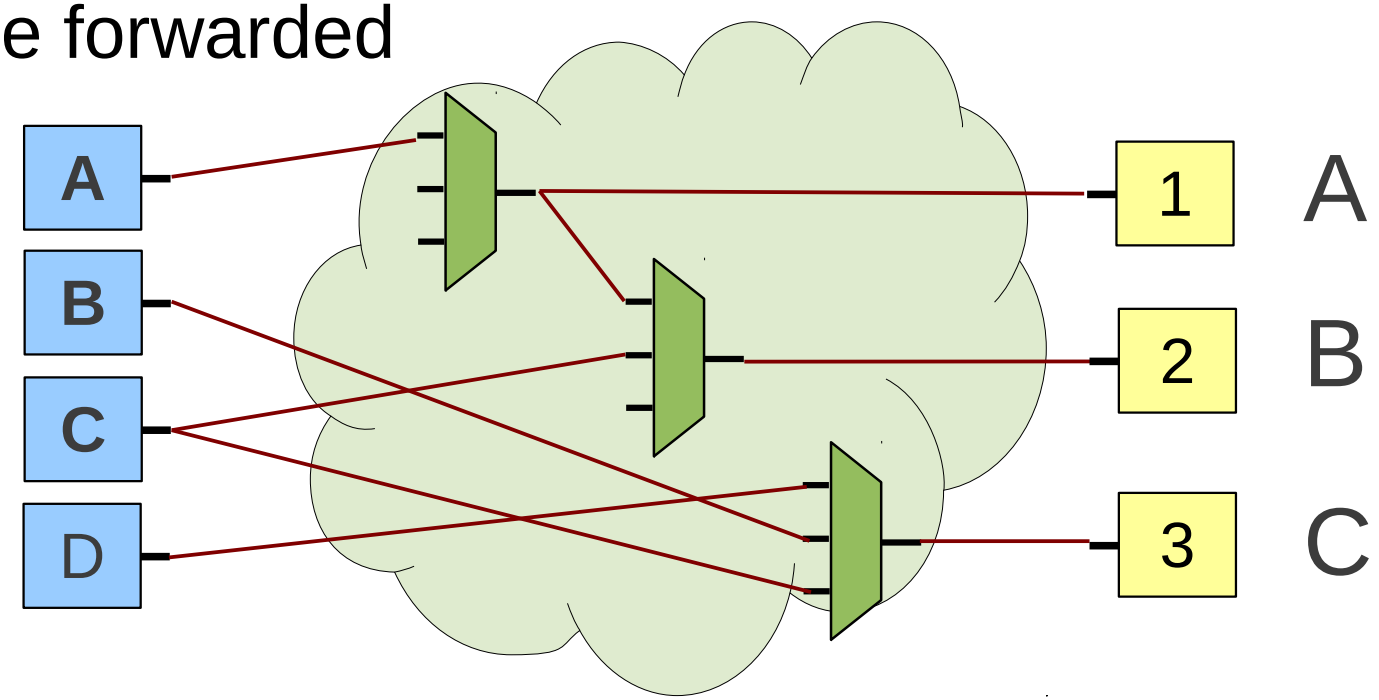


Overlay Network CAD

- Routing multiplexers must be configured for each new set of trace signals
- But Overlay Network is a Blocking Network
  - Unlike crossbar, not all combinations of signals can be forwarded

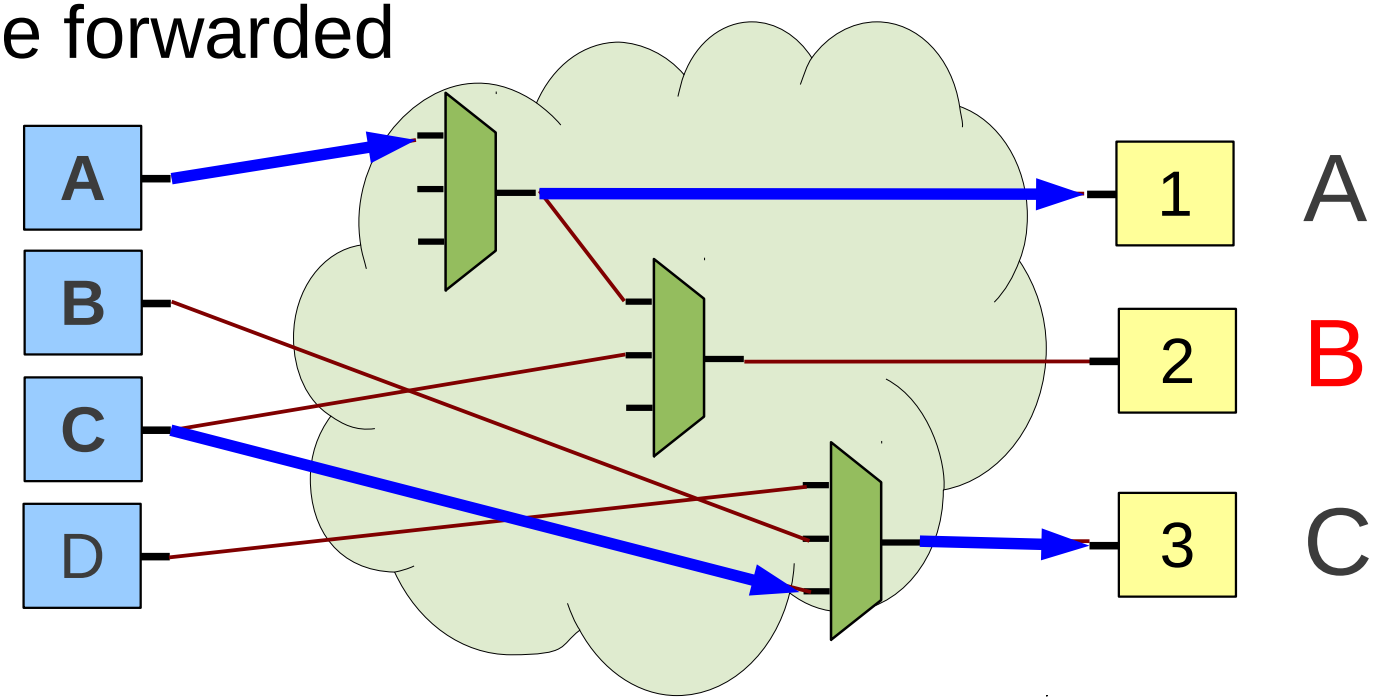
Overlay Network CAD

- Routing multiplexers must be configured for each new set of trace signals
- But Overlay Network is a Blocking Network
  - Unlike crossbar, not all combinations of signals can be forwarded



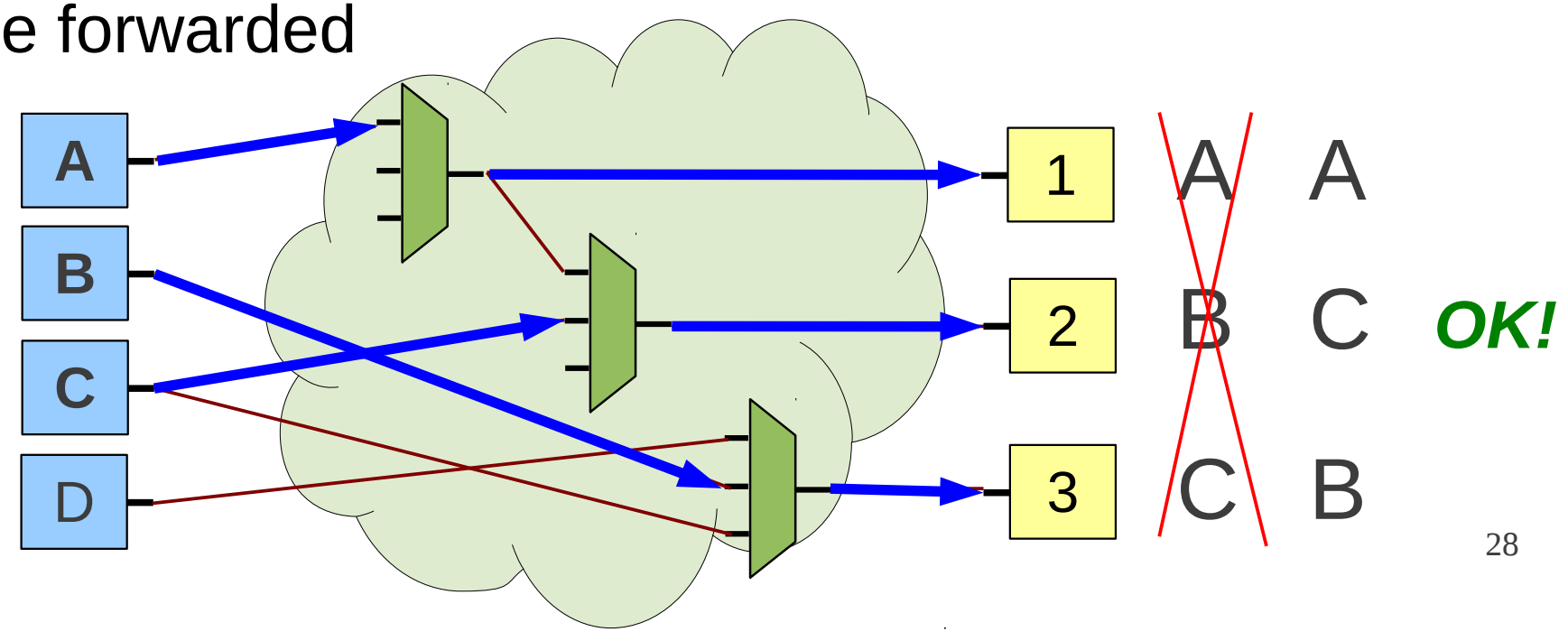
Overlay Network CAD

- Routing multiplexers must be configured for each new set of trace signals
- But Overlay Network is a Blocking Network
  - Unlike crossbar, not all combinations of signals can be forwarded



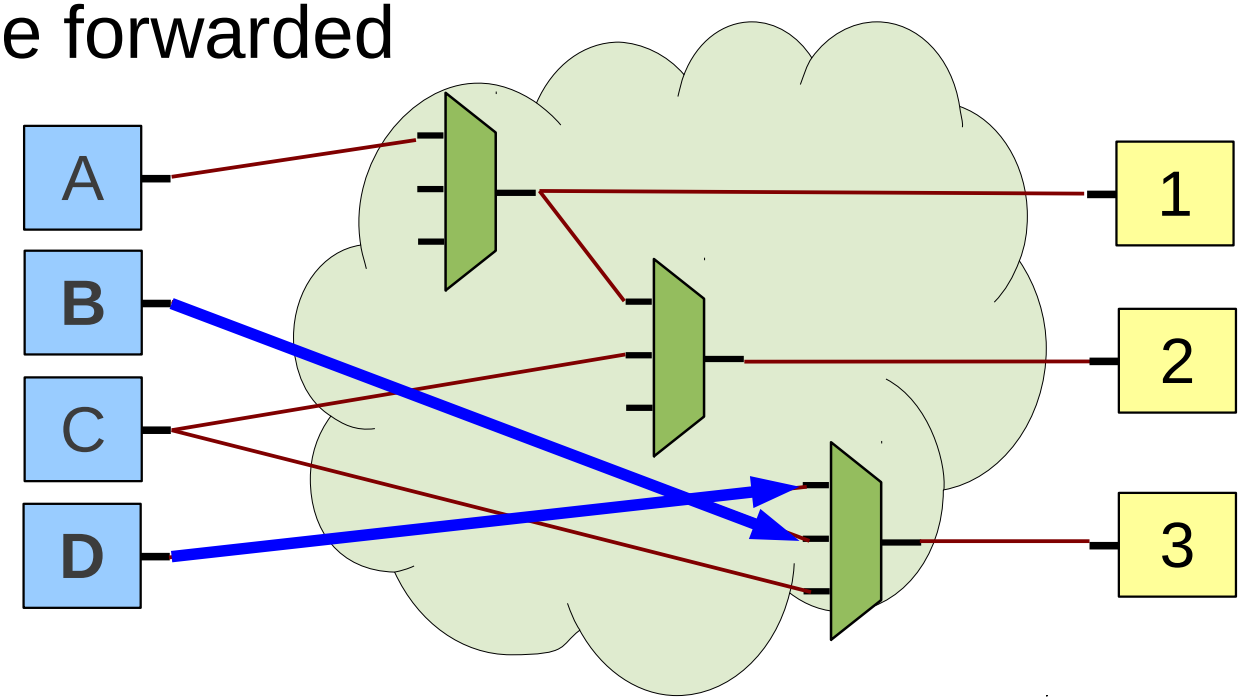
Overlay Network CAD

- Routing multiplexers must be configured for each new set of trace signals
- But Overlay Network is a Blocking Network
  - Unlike crossbar, not all combinations of signals can be forwarded



Overlay Network CAD

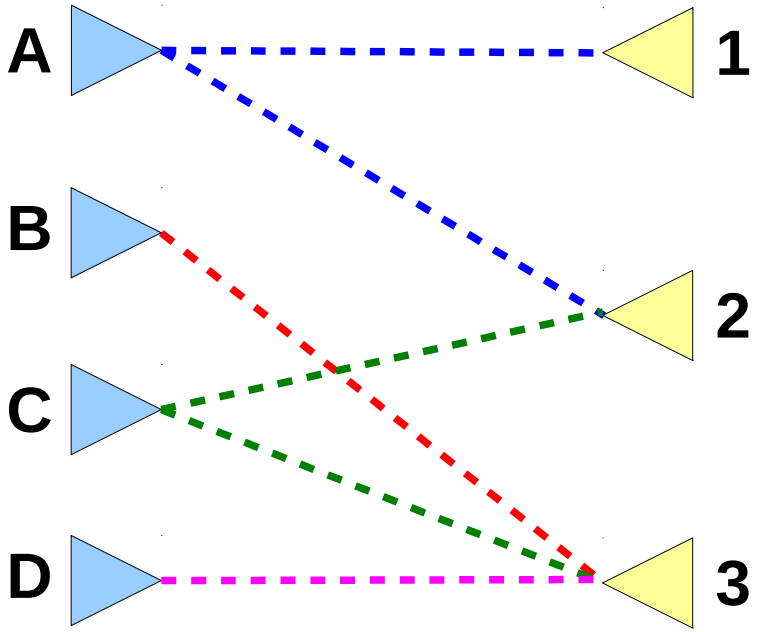
- Routing multiplexers must be configured for each new set of trace signals
- But Overlay Network is a Blocking Network
  - Unlike crossbar, not all combinations of signals can be forwarded



B & D
can never be observed at the same time!

Overlay Network CAD

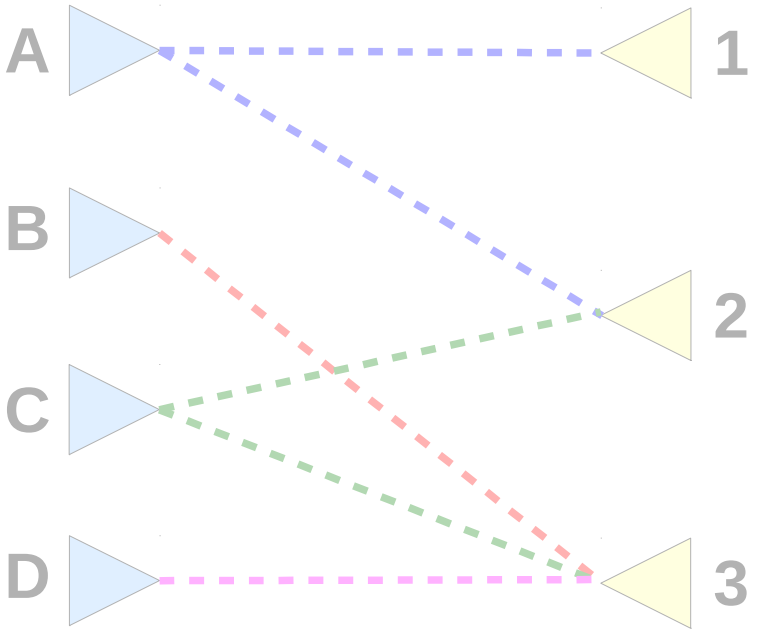
- Routing multiplexers must be configured for each new set of trace signals
  - Solve using bipartite graph maximum-matching!



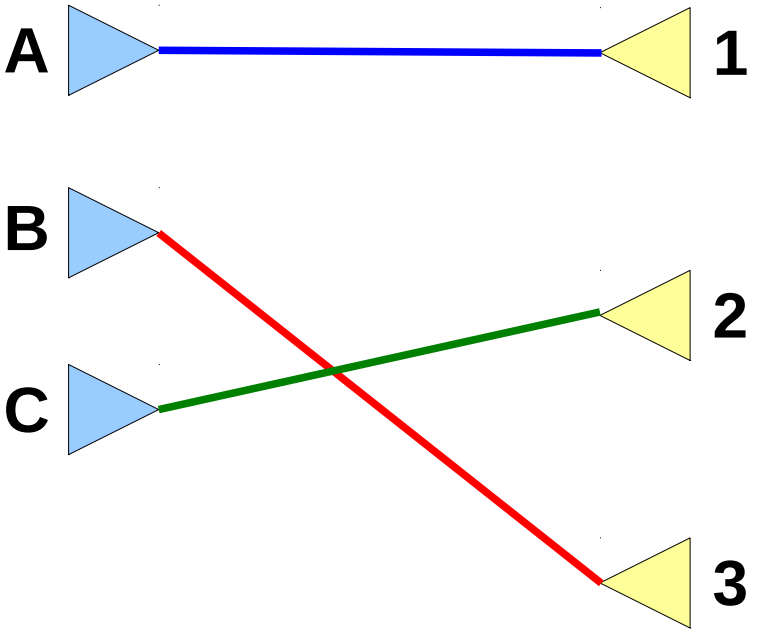
Connectivity Graph

Overlay Network CAD

- Routing multiplexers must be configured for each new set of trace signals
  - Solve using bipartite graph maximum-matching!



Connectivity Graph

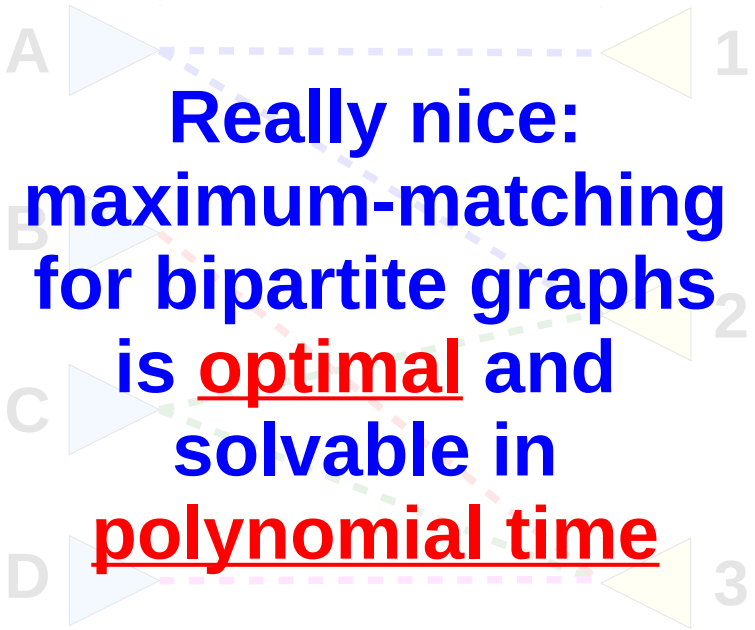


Matched Graph

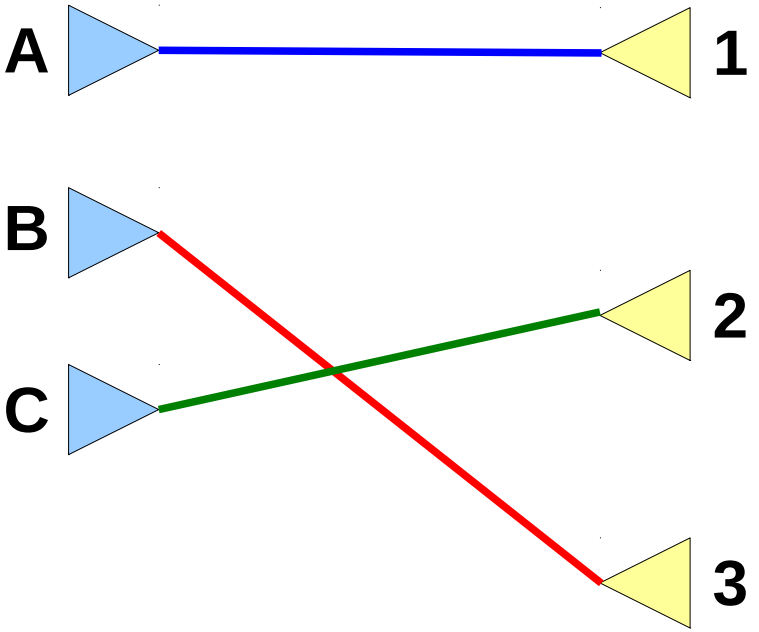
Overlay Network CAD

- Routing multiplexers must be configured for each new set of trace signals
  - Solve using bipartite graph maximum-matching!

Really nice: maximum-matching for bipartite graphs is optimal and solvable in polynomial time



Connectivity Graph

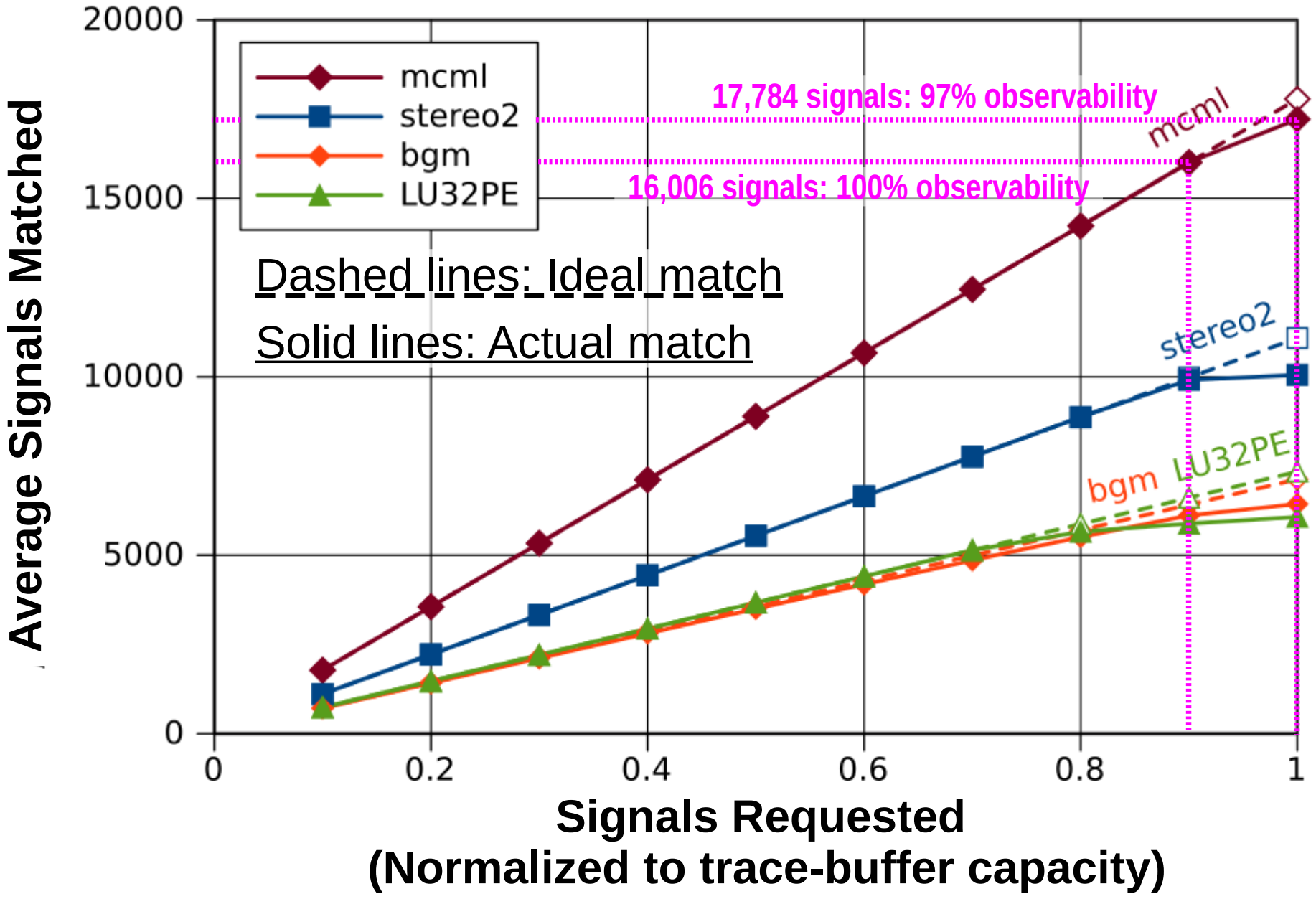


Matched Graph

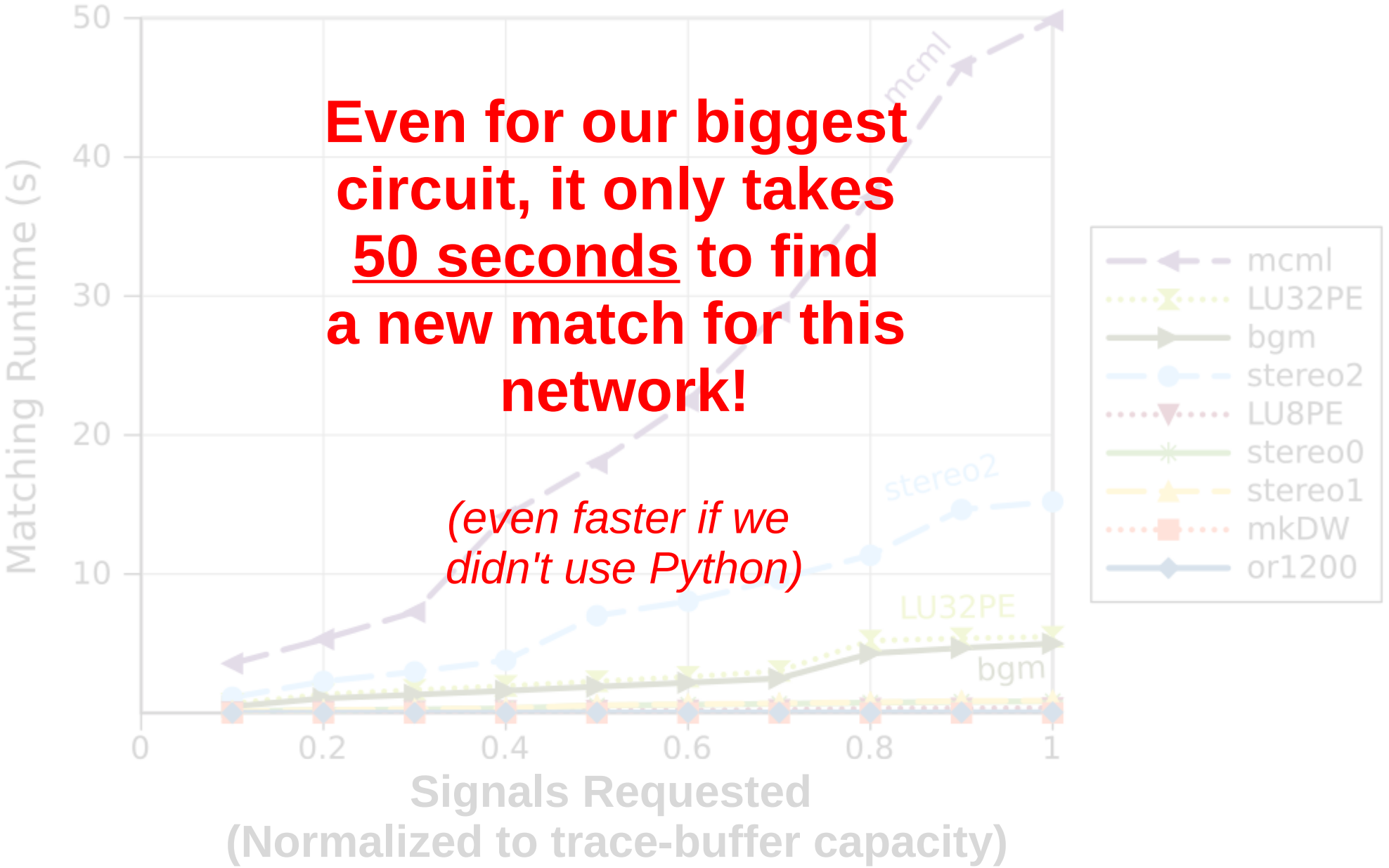
Overlay Network: Results

- Largest benchmark: “mcml”
  - 100,000 LUTs, 30 DSPs, 38 RAMs
  - 119x119 FPGA
  - 247 RAMs free, at 72 data inputs each
    - Total: 17784 trace-buffer inputs

Overlay Network: Results



Overlay Network: Matching Runtime



Overlay Network: Delay

- Circuit with overlay network: +9.0% delay
  - Recall: network installed incrementally
  - Increase solely due to wirelength of network
- But network does not affect user circuit!
  - Original operating frequency can always be restored by disregarding instrumentation

Simulator-like Observability ... are we there yet?

- Trace-buffers have finite memory capacity
  - But even in simulation, is all trace data necessary?
- External triggering assumed
  - Deep trace-buffers can tolerate triggering latency
  - Work-in-progress: On-chip triggering
- Feasible to realize on commercial FPGAs
  - Currently exploring: RapidSmith / GoAhead

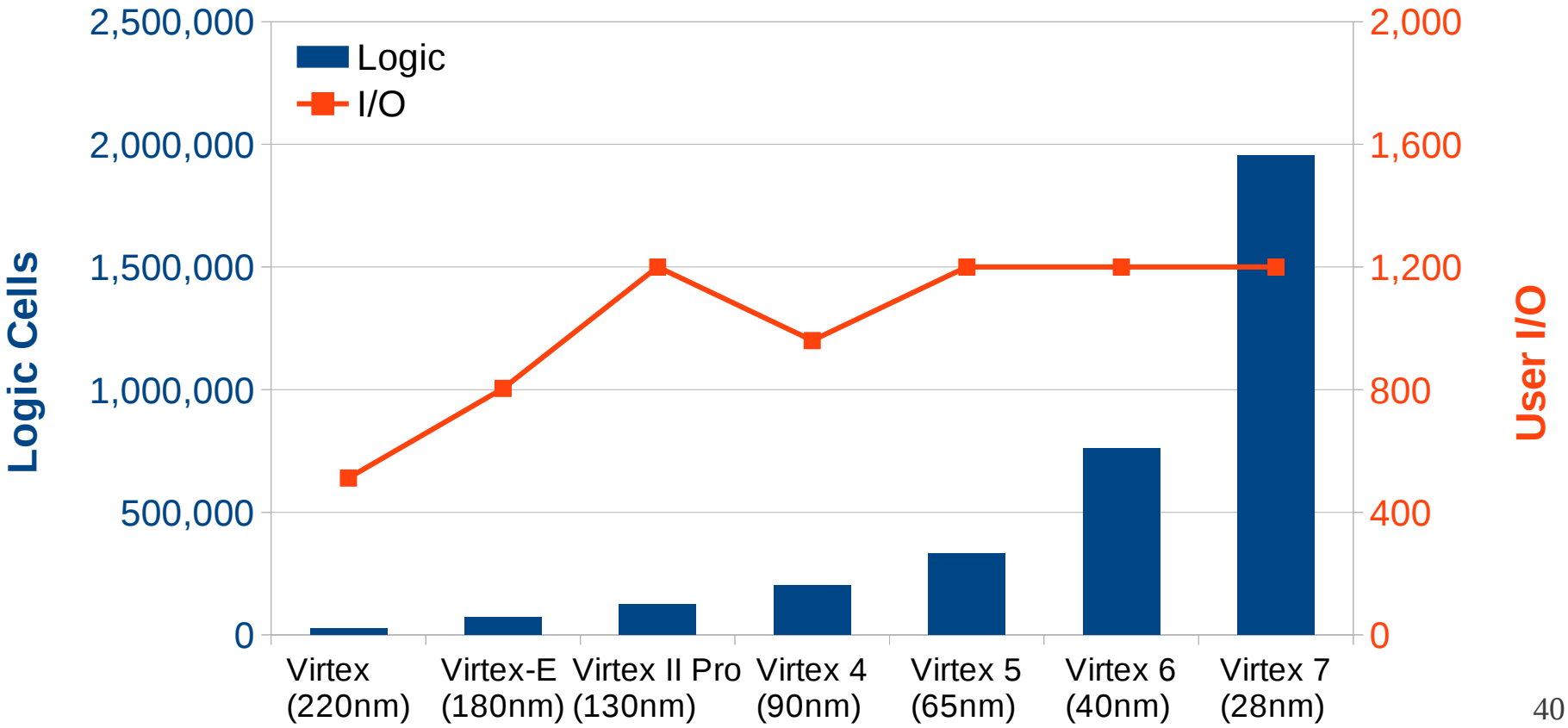
Conclusion

- FPGAs commonly used to prototype ASICs
  - Primary challenge during debug: Observability
- Accepted solution: Trace-Buffers
  - Only records subset of signals, recompile to change
- We propose building an overlay network for TBs
  - Utilizing an efficient network topology
  - Constructed out of reclaimed routing muxes (free!)
  - CAD algorithm to create network match in seconds
- Outcome: 10,000s observed signals with no overhead

Backup Slides

Introduction

- Challenge in post-silicon debug: visibility
  - Limited I/O ⇒ lack of access to internal nodes



Routing Utilization on VPR

- At minimum channel width + 30%

Benchmark	***Before***	***After***
LU8PEEng	51%	76%
stereovision2	38%	68%
bgm	50%	79%
LU32PEEng	41%	74%
mcml	36%	79%

Routing Utilization on Altera Devices

- Uninstrumented circuit

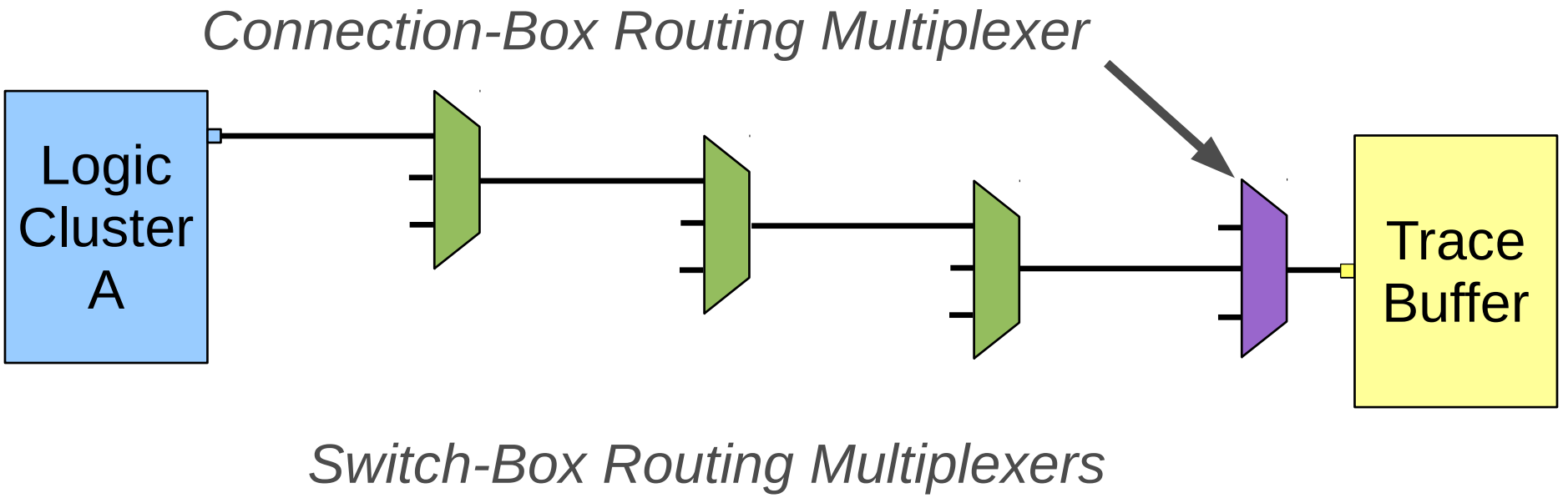
	LU8PE	bgm	mcml
Device	EP4CE30	EP4CE30	EP4SGX110
LAB (logic cluster) utilization	95%	71%	99%
Average routing utilization	25%	21%	35%
Peak routing utilization	62%	36%	47%

TABLE III

ROUTING UTILIZATION WHEN MAPPED ONTO ALTERA DEVICES

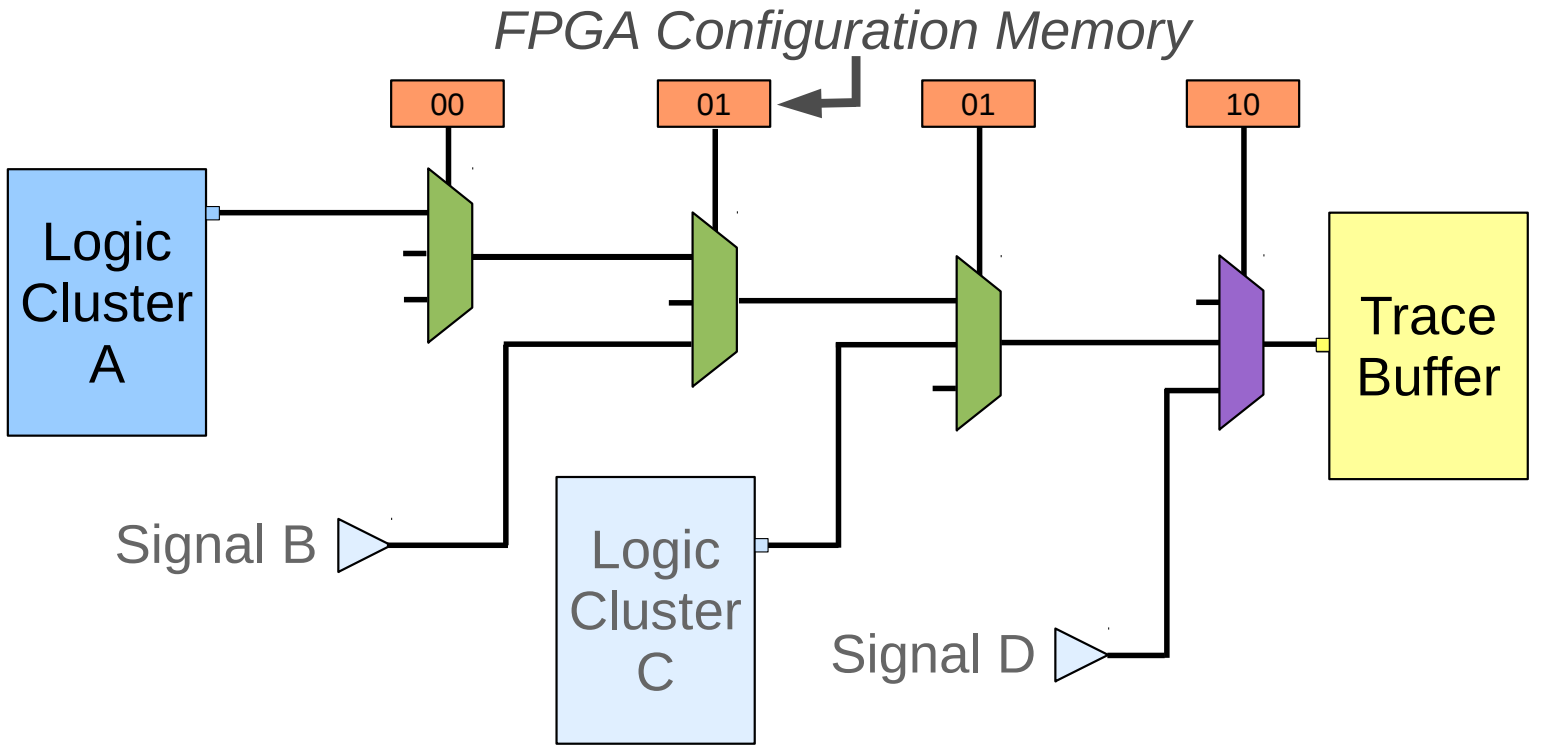
Virtual Overlay Network

- Existing IP (ChipScope, SignalTap) build single point-to-point trace connections for each signal:



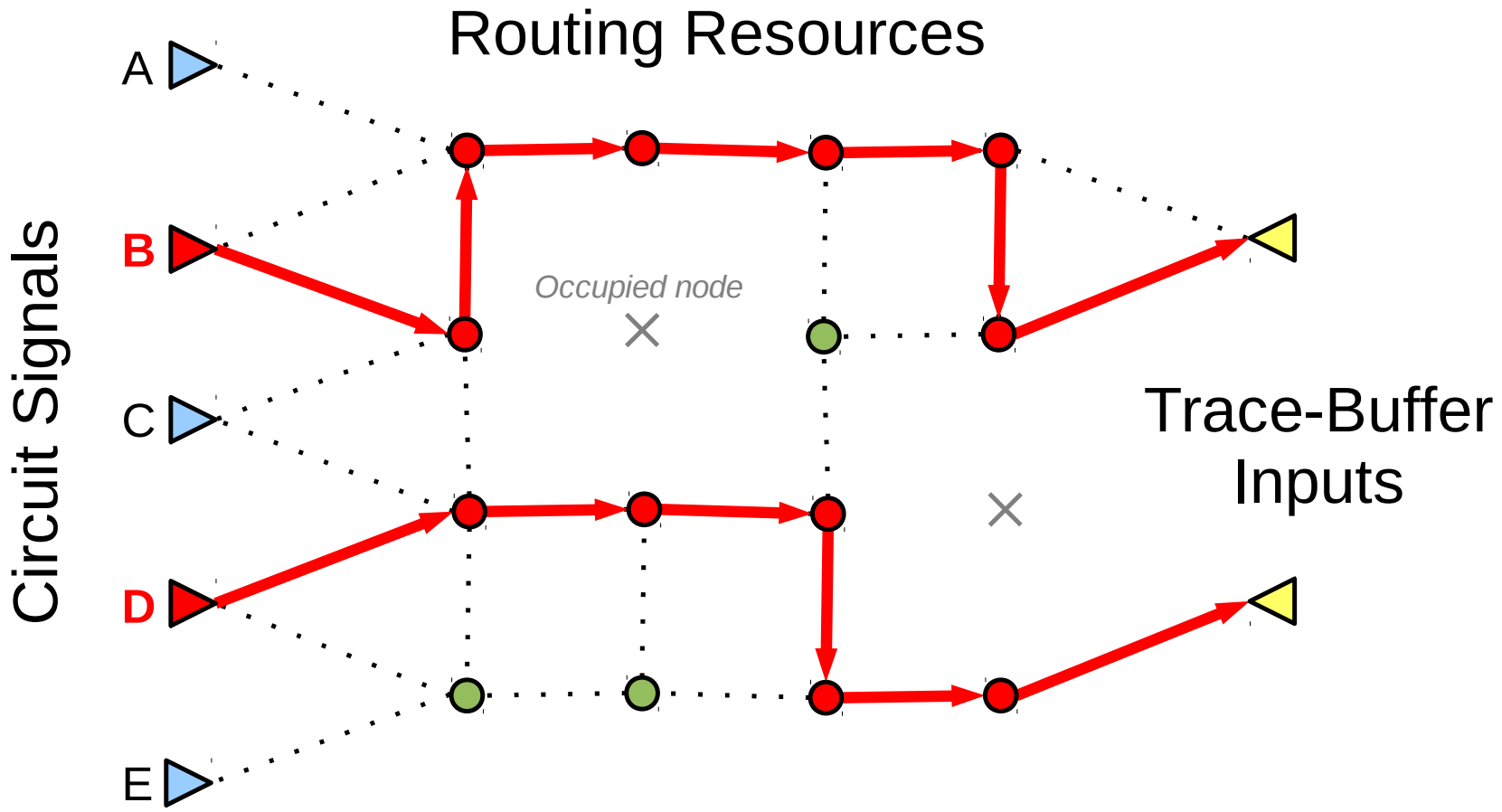
Virtual Overlay Network

- Existing IP (ChipScope, SignalTap) build single point-to-point trace connections for each signal
- We propose building an overlay network out of spare routing muxes to connect multiple signals



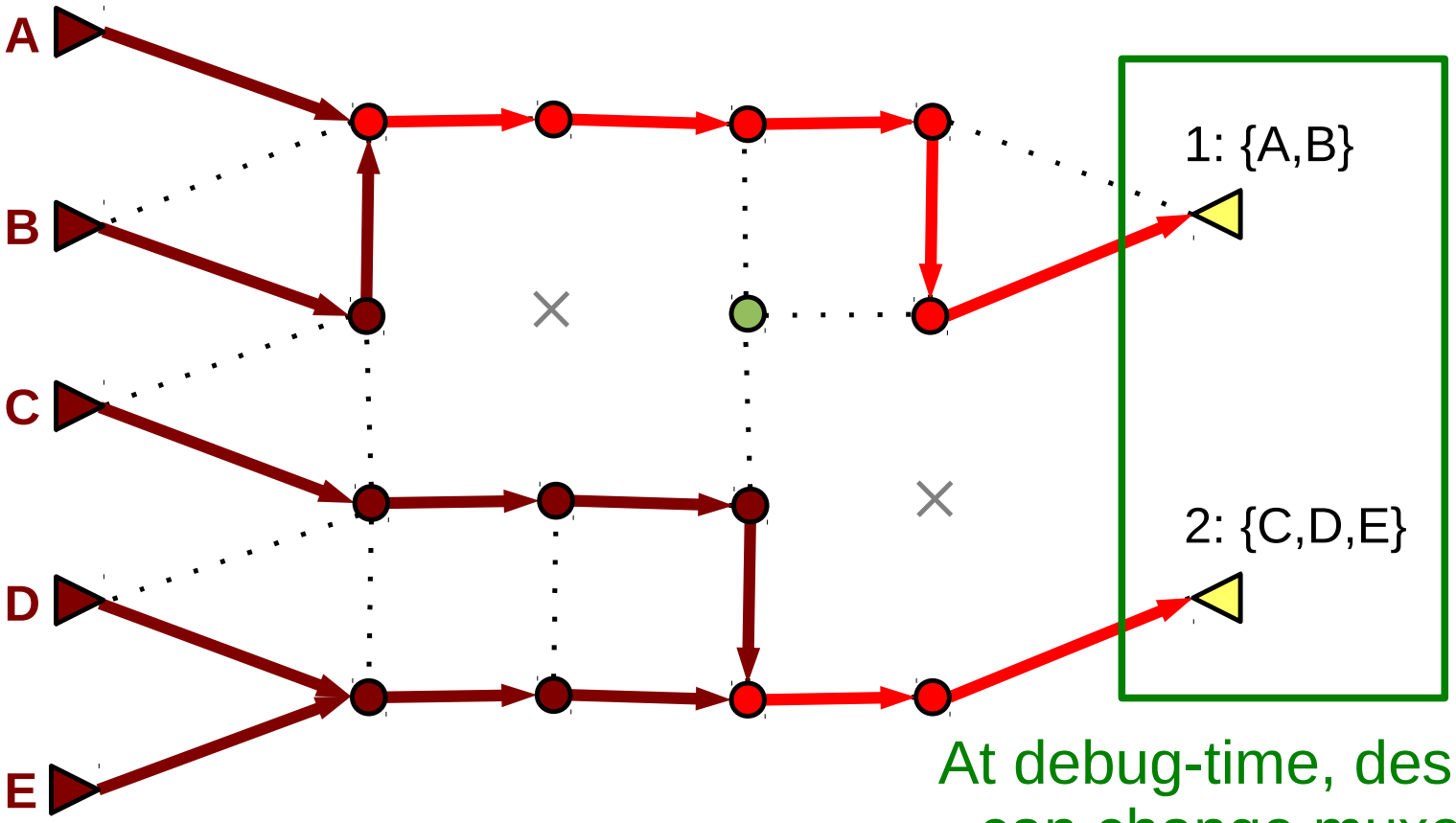
Overlay Network: Compile-time

- Routing Resource Graph: Point-to-Point



Overlay Network: Compile-time

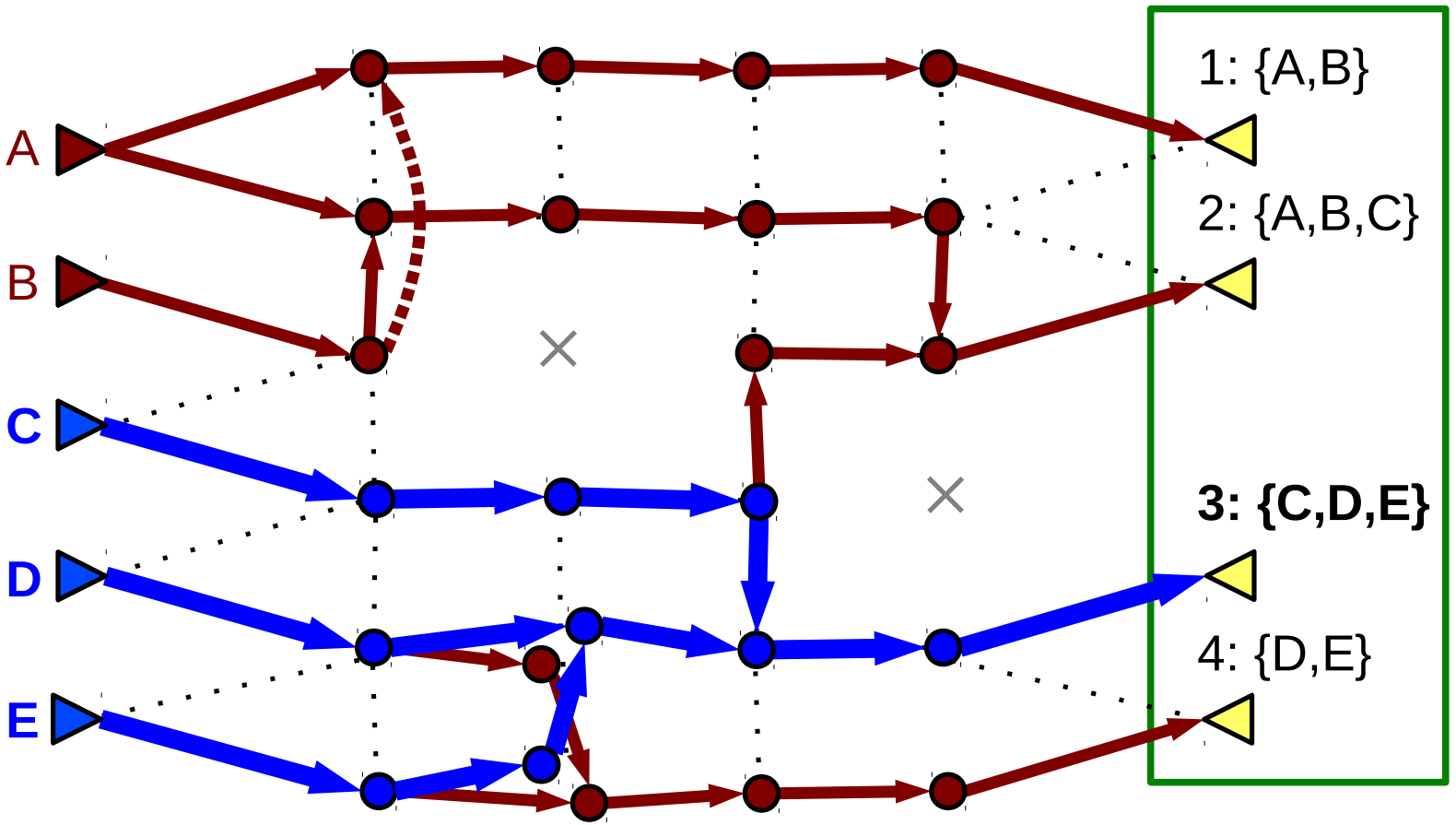
- Our approach: Disjoint Union of Trees



At debug-time, designers can change muxes to look at **any** single signal!

Overlay Network: Compile-time

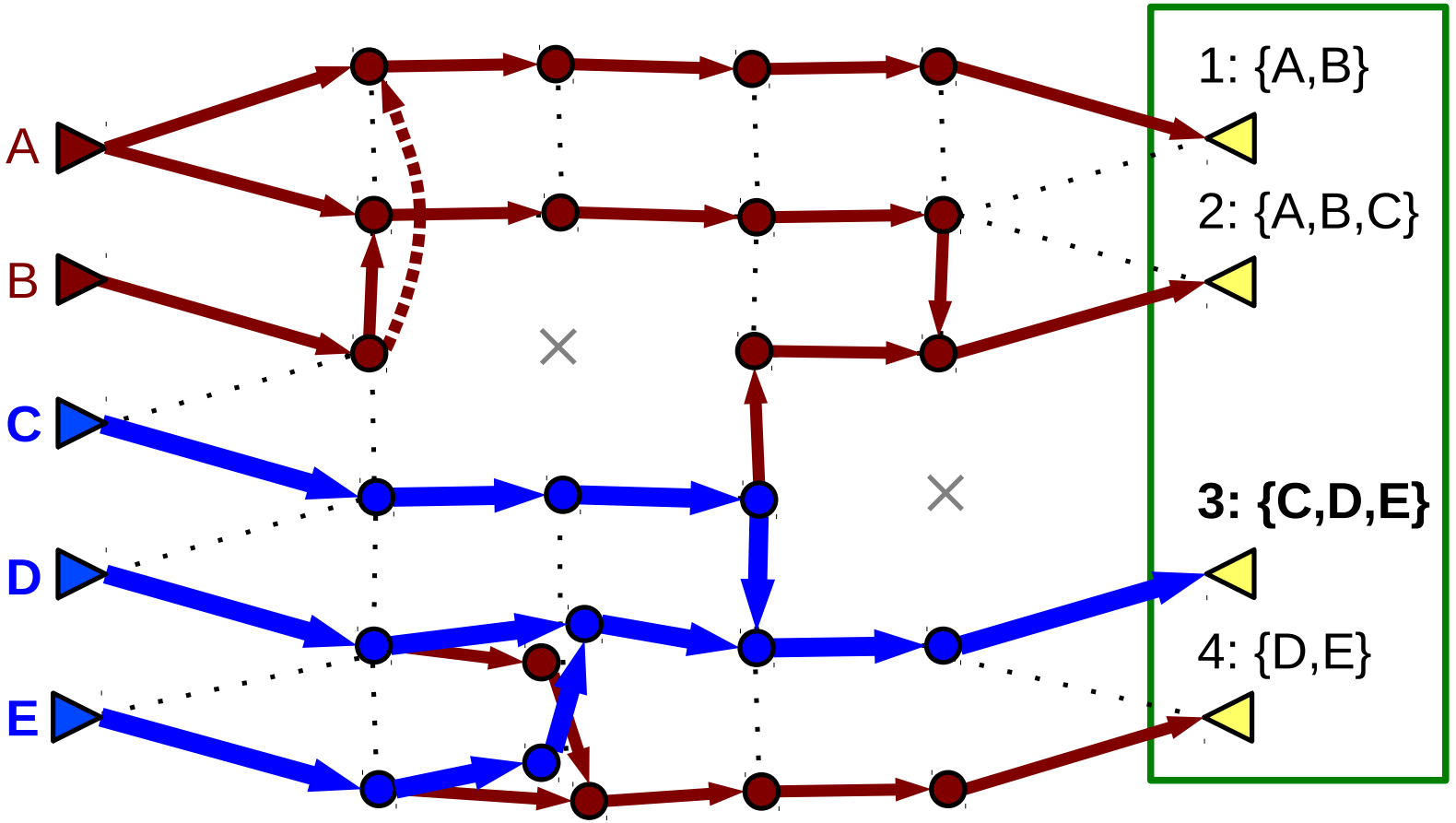
- Virtual Network: Signals as leaves of many trees



At debug-time, designers can change muxes to look at **any comb. of two** signals!

Overlay Network: Compile-time

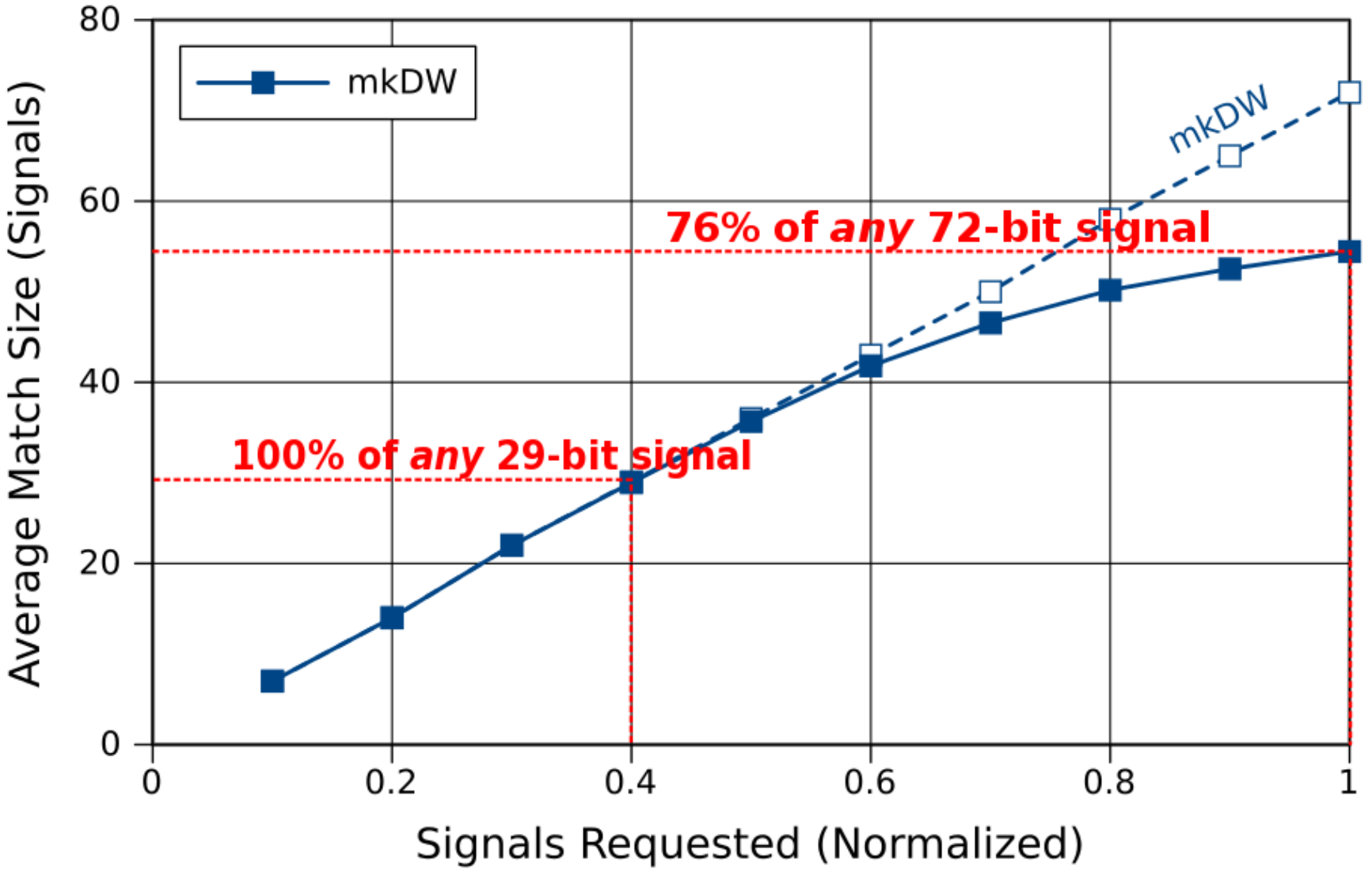
- Virtual Network: Signals as leaves of many trees



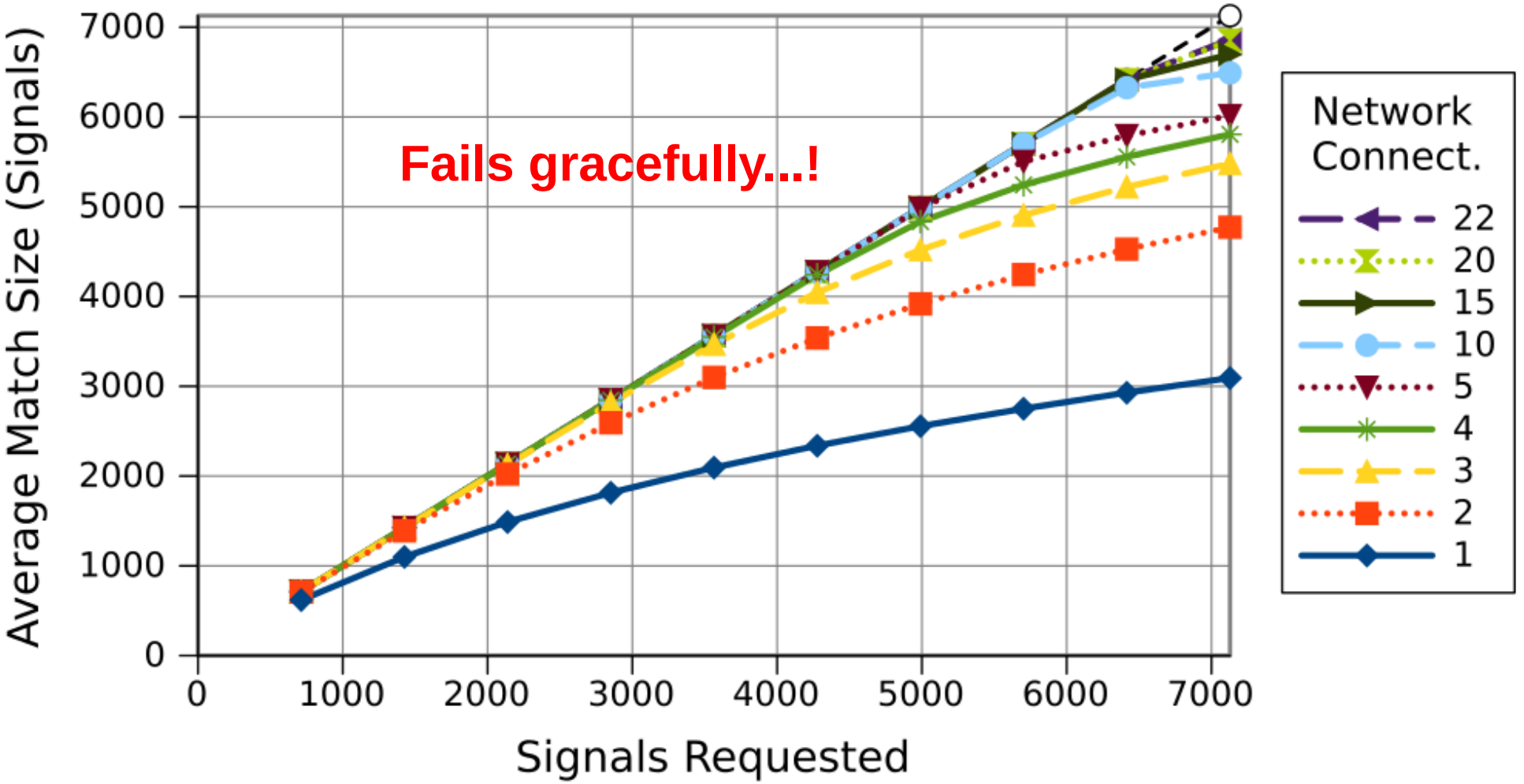
At debug-time, designers can change muxes to look at **any comb. of two** signals!

(actually, for this particular hand example, any comb. of four signals!)

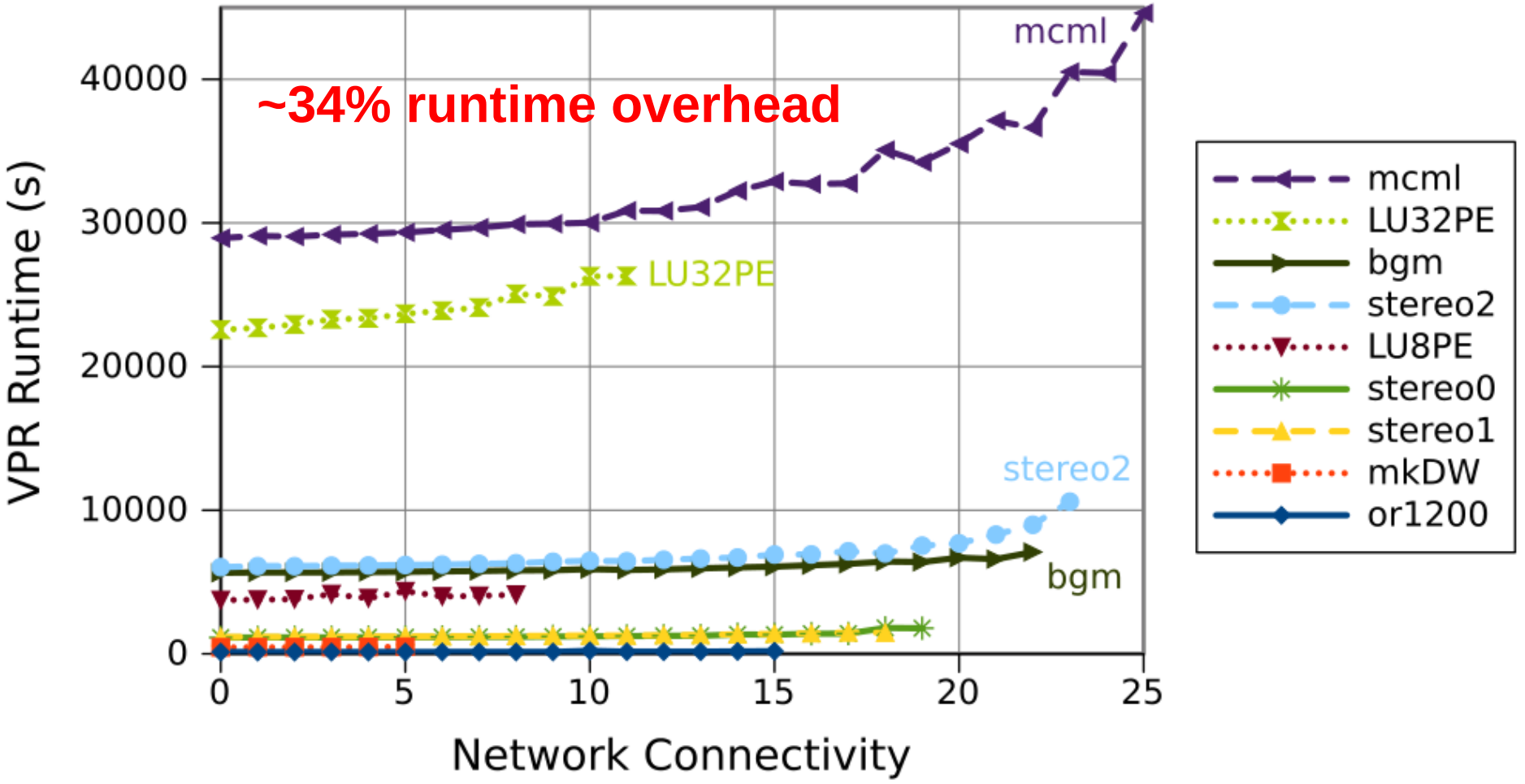
Overlay Network: Reduced BRAM



Overlay Network: Reduced Routing



Overlay Network: CAD Runtime



Overlay Network: Matching Runtime