Using Virtual Coverage to Hit Hard-to-Reach Events

Laurent Fournier and Avi Ziv

Verification and Services Technologies Department
IBM Haifa Research Lab

Prelude

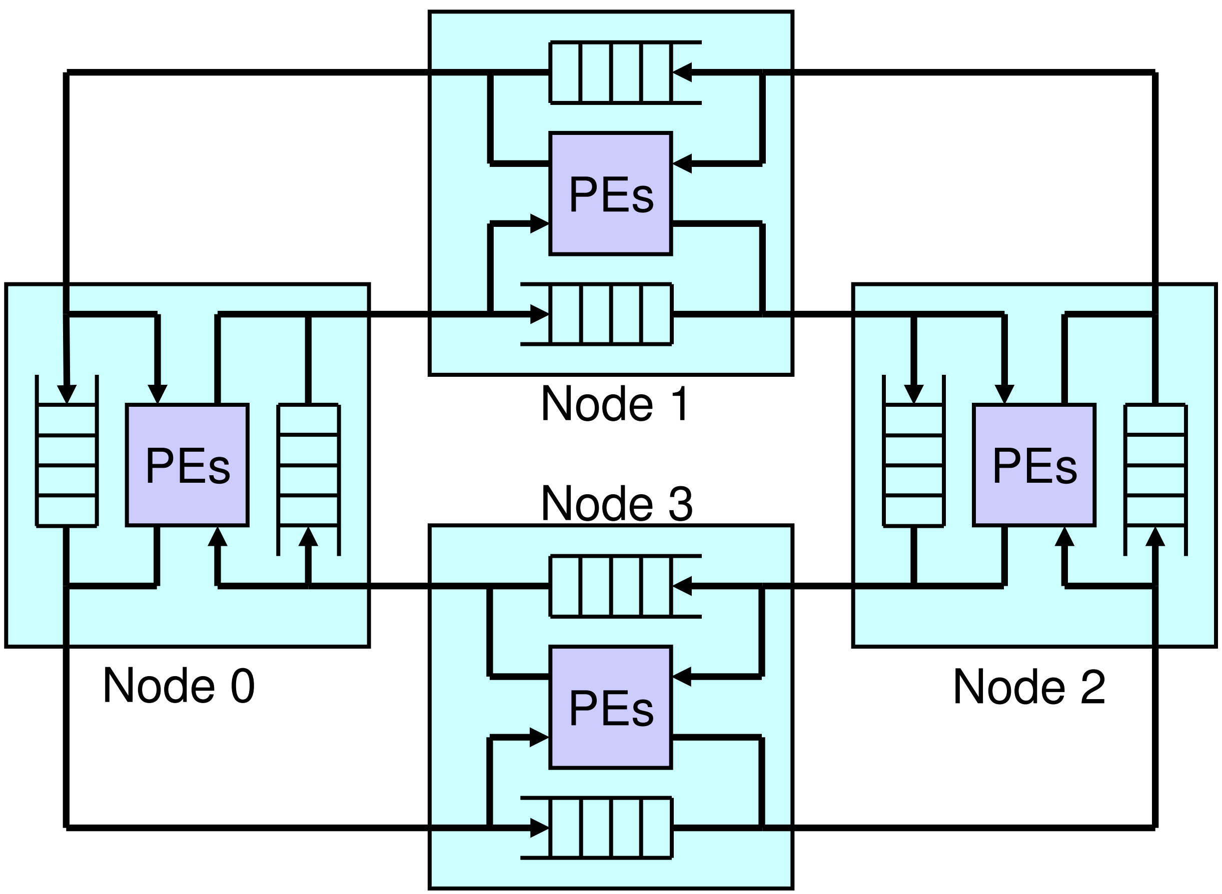
- Bug in a complex multiprocessor system – found in the lab
  - Overflow (that was not supposed to happen) in an internal buffer caused the system to hang
- Attempts to recreate the bug in simulation failed
  - The necessary conditions for the bug were identified
    - But even reaching them in simulation was difficult
- Even so, the bug was fixed and the bug fix is (apparently) correct
- Still, the design and verification teams want reassurance that the bug does not exist in current and future versions

… and here we enter the picture

Outline

- The problem
  - Description of the target system
  - Description of the coverage goal (bug)
  - Manual solution
- Coverage Directed Generation
  - Concept of operation
  - Where and why it works
  - Why it does not work in our case
- Virtual coverage
  - What it is
  - Using it in our case
  - Results
- Summary and conclusions

The Target System



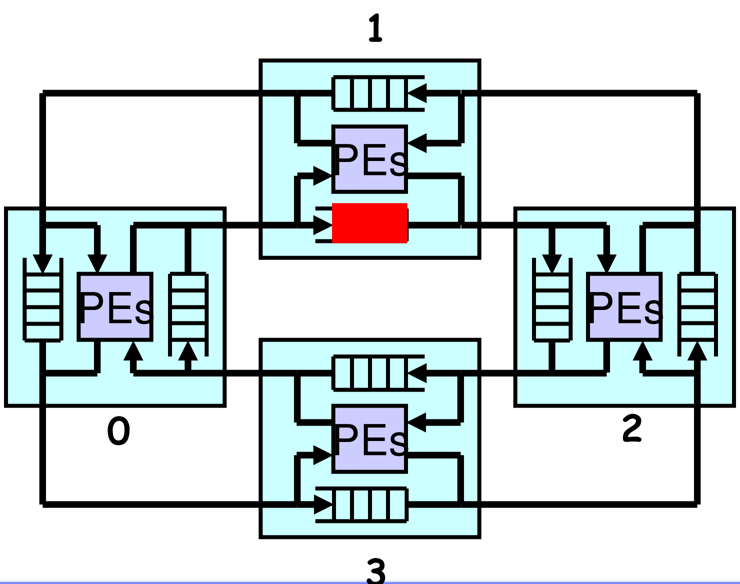
- Four compute nodes
  - 8 processors
  - I/O
  - Memory
  - Caches
- Two unidirectional rings for communication
  - Used mostly for accessing memory on remote nodes
- Small flow-through buffers for transient transactions
- Two modes of operation
  - Closed ring
  - Open ring

The Coverage Goal

- The bug caused an overflow in one of the flow-through buffers, which in turn caused the system to hang

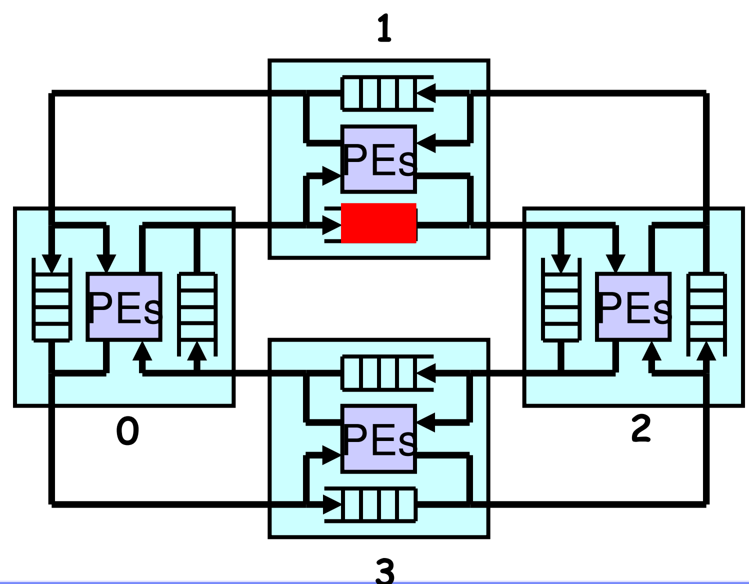
- Necessary condition for the bug: The flow-through buffer is full for long periods of time

- The target coverage event:
  Fill a specific flow-through buffer for more than 50 cycles

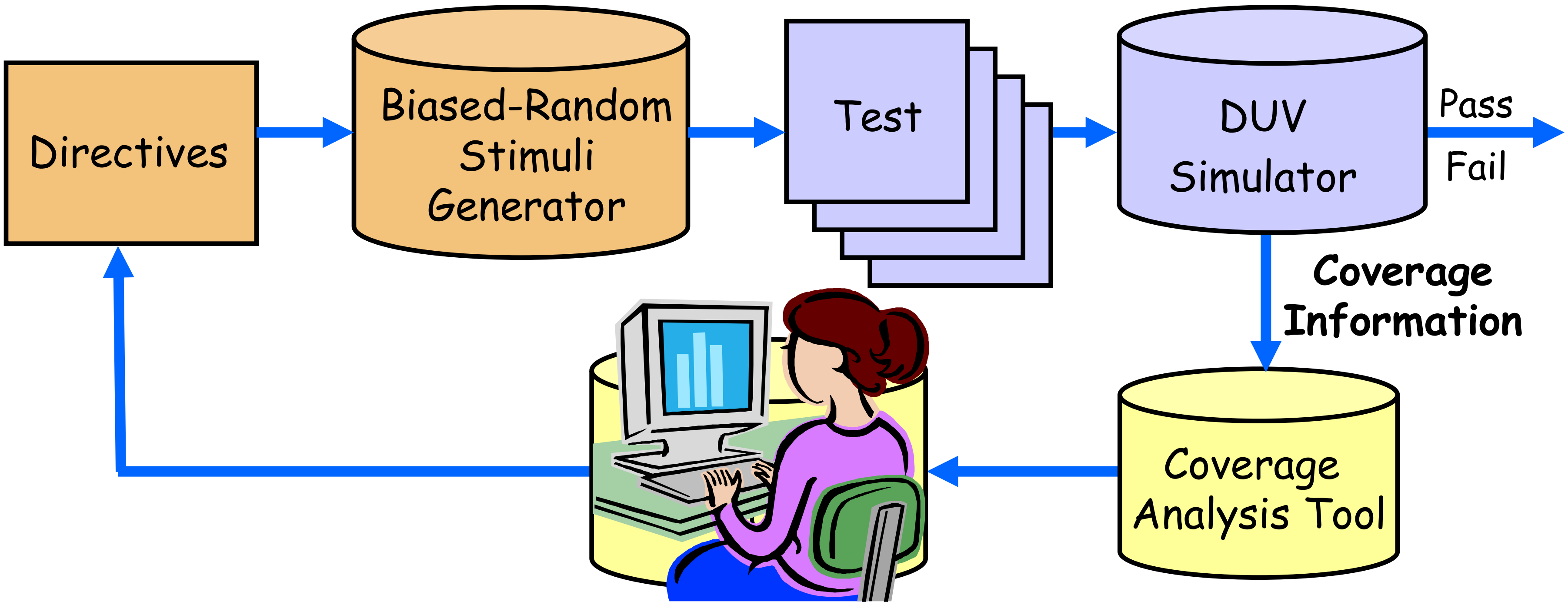


Manual Solution

- Work in open ring mode
  - Increase traffic using relevant buffer
- Create directive files with parameters that lead toward the target
  - More transactions using the buffer
    - Push from Node 0, pull from Node 2
    - Processor and I/O transactions
      - Many types
  - Cause transactions to stay longer
- Simulate the system with the directive files
- Analyze the results
  - If target not reached, repeat



Data-Driven Coverage Directed Generation



- Reduce the bottleneck of closing the loop from coverage to generation
- Replace human expert with automatic reasoning
- Leads to faster and better coverage with fewer resources

How Data-Driven CDG Works

- The CDG engine is fed pairs of inputs (directives) and outputs (coverage data)
  - These pairs are often called training data
- The CDG engine "understands" the relations between inputs and outputs and can answer queries about the relations
  - What directive can lead to a requested coverage event?
- Two levels of understanding
  - Memorizing
  - Generalization
- In CDG we are usually interested in pairs not seen in the training data
  - Specifically, how to reach uncovered events

→ Generalization is the key to success

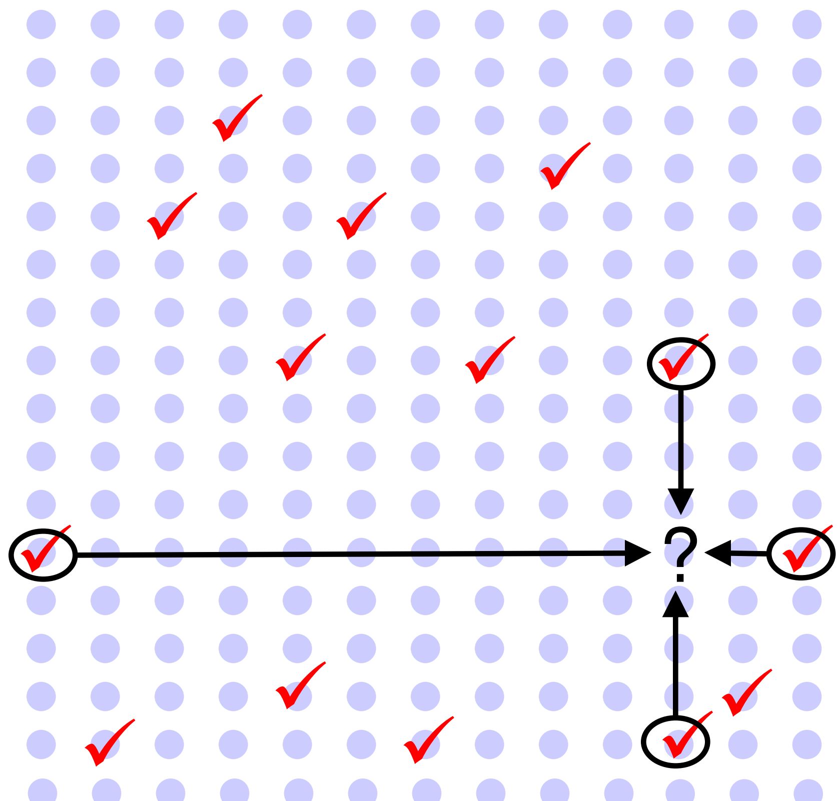
How to Generalize

- Need to know the relations between items in the output space
  - And similarly in the input space

- Example – ordering rules (<, >, =)
- Example – similarity
  - Usually means breaking the item into sub-items

Cross-Product Coverage and Generalization

- Cross-product coverage is a natural form for generalization in the coverage space
  - Break up the output space along the attribute's axis
  - Understand the input-output relations for each attribute
  - Generalize by combining the understandings
- But life is not that simple
  - Attributes are related
  - Conflicting understanding
  - Randomness
  - …

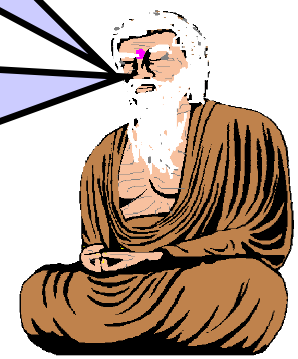


Back to the Flow-Through

But how can CDG generalize from a singular event?

CDG can automatically reach coverage events

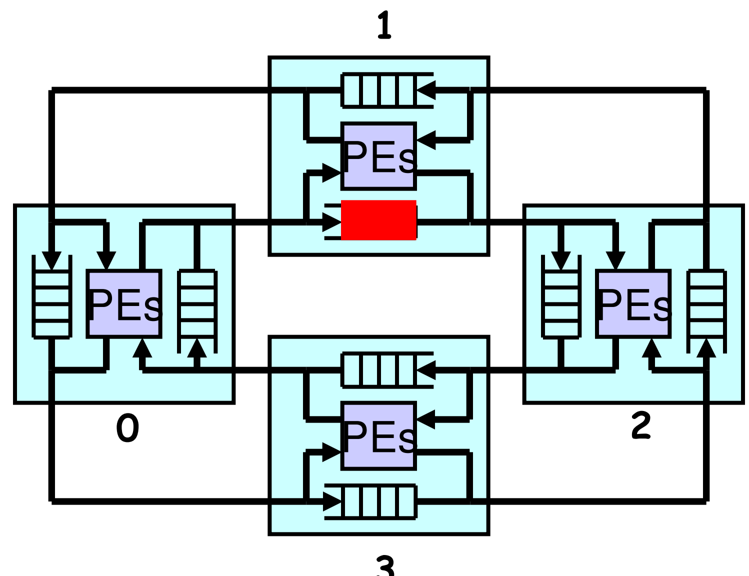
So build a structured coverage model around it and use this model to generalize



Virtual coverage model is a structured coverage model defined around the target coverage event

- ◈ The target event is one of the points in the virtual coverage model
- ◈ The model is defined to help CDG exploit its structure to learn how to reach the target event

Covering the virtual model is not a goal of the verification plan

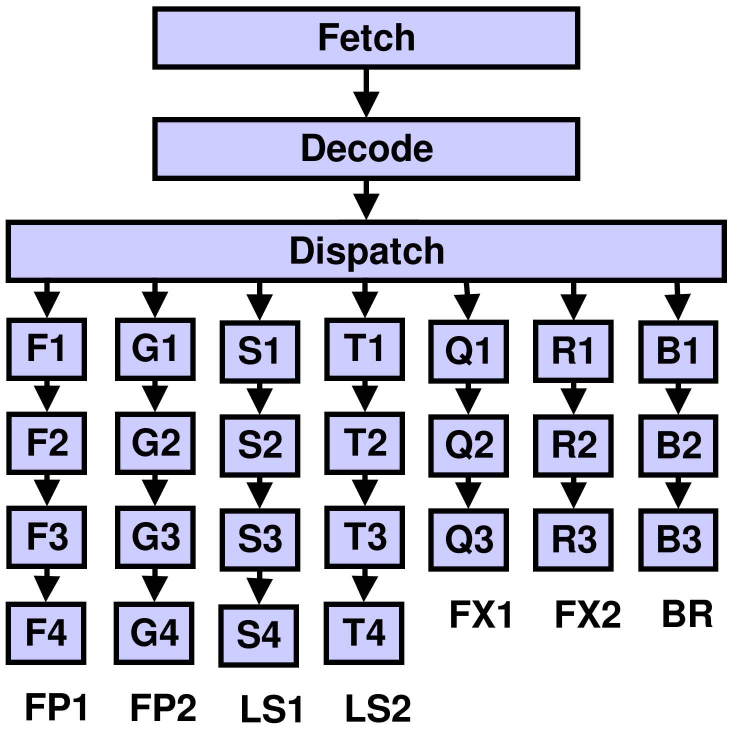


How to Define a Virtual Coverage Model

- Break the target event into a set of sub-events
  - The sub-events can be implicit
  - Dependency between the sub-events is possible
- Each sub-event is an attribute in a cross-product model with its own set of possible values

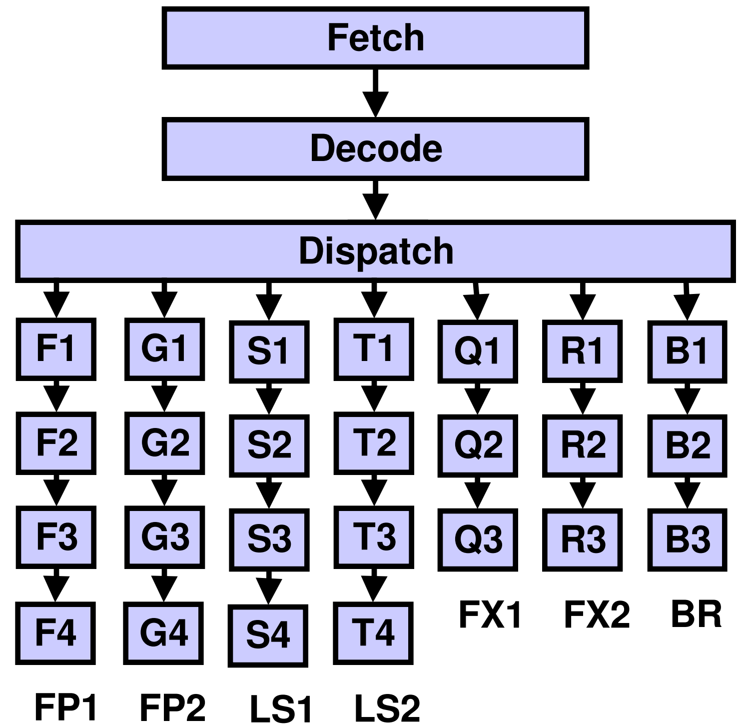
Dispatch is blocked from below ⇔
cannot dispatch to F1 and cannot dispatch to G1 and … and cannot dispatch to B1 ⇨

Dispatch to F1 (Y/N) N
Dispatch to G1 (Y/N) N
…
Dispatch to B1 (Y/N) N



How to Define a Virtual Coverage Model (II)

- Augment the basic virtual model with additional information
  - Add values to attributes representing sub-events
  - New attributes that do not directly belong to the target event
- Augmenting the model increases its size



Dispatch is blocked from below ⇨

State of F1 (free, busy, on hold, …) on hold

…

State of B1 (free, busy, on hold, …) on hold

State of F2 (free, executing, finished)

…

State of B2 (free, executing, finished)

Virtual Coverage and the Flow-Through Buffer

- The target event is to fill the flow-through buffer and keep it full for more than 50 cycles
- The basic virtual coverage model
  - Flow-through buffer full (Y/N) Y
  - More than 50 cycles from arrival to next departure (Y/N) Y
- After adding values to the attributes
  - Flow-through buffer utilization (0, 1, 2, …, 7, 8) 8
  - Time from arrival to next departure
    (0-10, 11-20, …, 50-100, 100-) 50-100 and 100-
- Adding new attributes
  - Time in buffer
  - Arrival rate
  - Transaction command

CDG and the Virtual Coverage Model

- Identify relevant parameters in the test generator directive files
  - Start with parameters identified by a domain expert
  - Use sensitivity analysis to ensure that the parameters are relevant and which coverage attributes they influence

- Identified parameters included
  - CPU initiated transactions
    - Instruction fetch, data fetch, data store
  - I/O initiated transactions
    - Data read and write
  - Addresses of transactions

CDG and the Virtual Coverage Model (II)

- Build the CDG engine for the specific virtual coverage model and relevant parameters
  - In our case, the CDG engine is based on Bayesian Networks
  - Building the CDG engine comprised
    - Manual definition of the Bayesian Network structure
    - Automatic training of the network
- Use the CDG engine to generate directives file with highest probability of reaching the target event

CDG Results

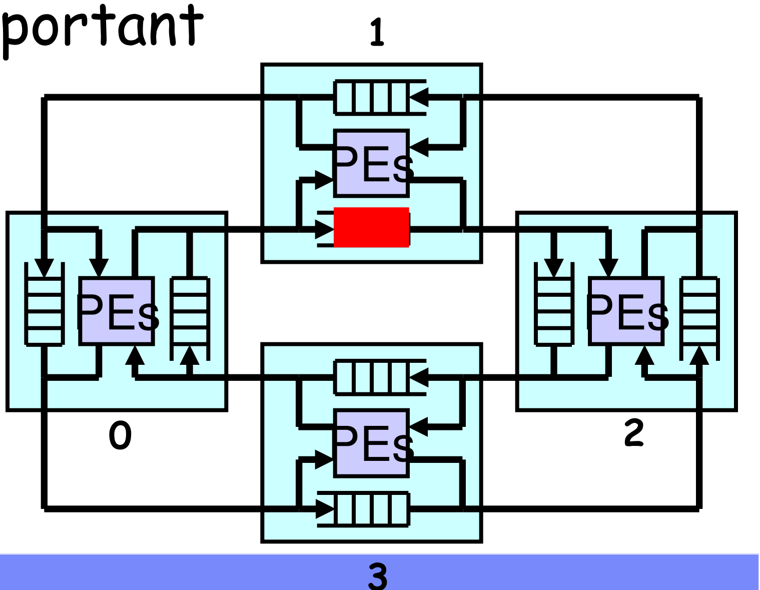
- We reached the target event … and more
  - Buffer remains full for very long periods of time
  - Works in both open and closed ring modes
    - With better results in the open ring mode
  - Good control of the transactions that fill the buffer

	Open ring mode	Close ring mode
Probability of reaching the event	50%	30%
Maximal full time	200 cycles	140 cycles

Analysis of Results

- In general, no big surprises
  - Node 0 pushes data to Node 2
  - Node 2 pulls data from Node 0
  - Node 1 utilizes the ring

- CDG is better than manual directive files because
  - Node 1 utilization of the ring is very important
  - CPU transactions are more important than believed
  - Fine tuning the mix of transaction sources and types



Flow-Through Buffer Summary

- ☺ We reached the target event … and more
- ☹ Overall effort – three person months, four calendar months
  - This effort is much too high for "normal" coverage events
- ☹ We did not find any bug during that process
  - But apparently, there are no related bugs in the design
  - We did not try our solution on the buggy design
- ☺ We increased the confidence of the design and verification teams that such a bug does not exist

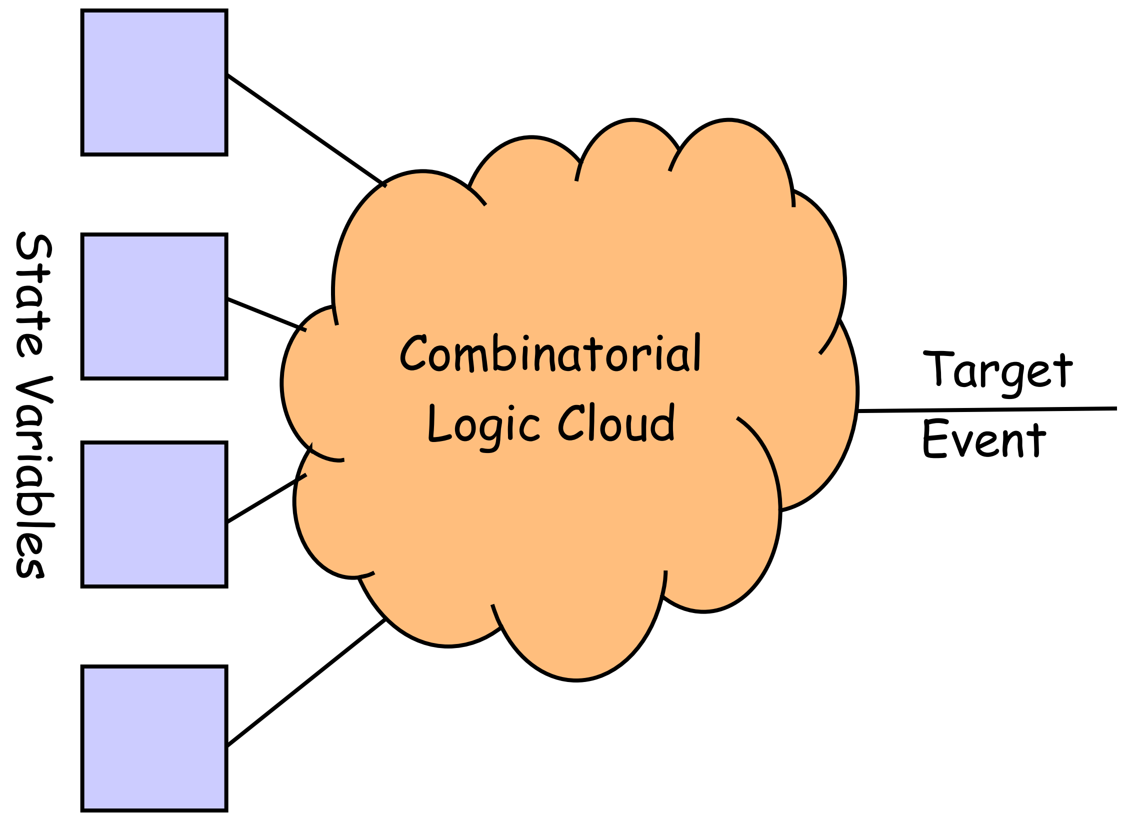
Conclusions

"The paper gives the impression that this was a somewhat ad-hoc approach where a specific example drove the work and afterwards a paper got written around it."

- Started with two ideas for solutions to a difficult problem
  - Use CDG to reach a hard-to-reach coverage event
  - Learn how to reach the event by building a grid of events around it
- Built a concept around it
  - Gave it a catchy name "virtual coverage"
  - Some guidelines on how to define the coverage model
  - Identified strength and weakness in concept
- Working to address weaknesses
  - Reduce effort by increased automation in virtual coverage definition

Increased Automation in Virtual Coverage Definition

- Syntactically break the target event into sub-events based on state variables in cone-of-influence
  - Each variable is an attribute in the virtual model
- Initial experiments show promising results




Thank you for your attention