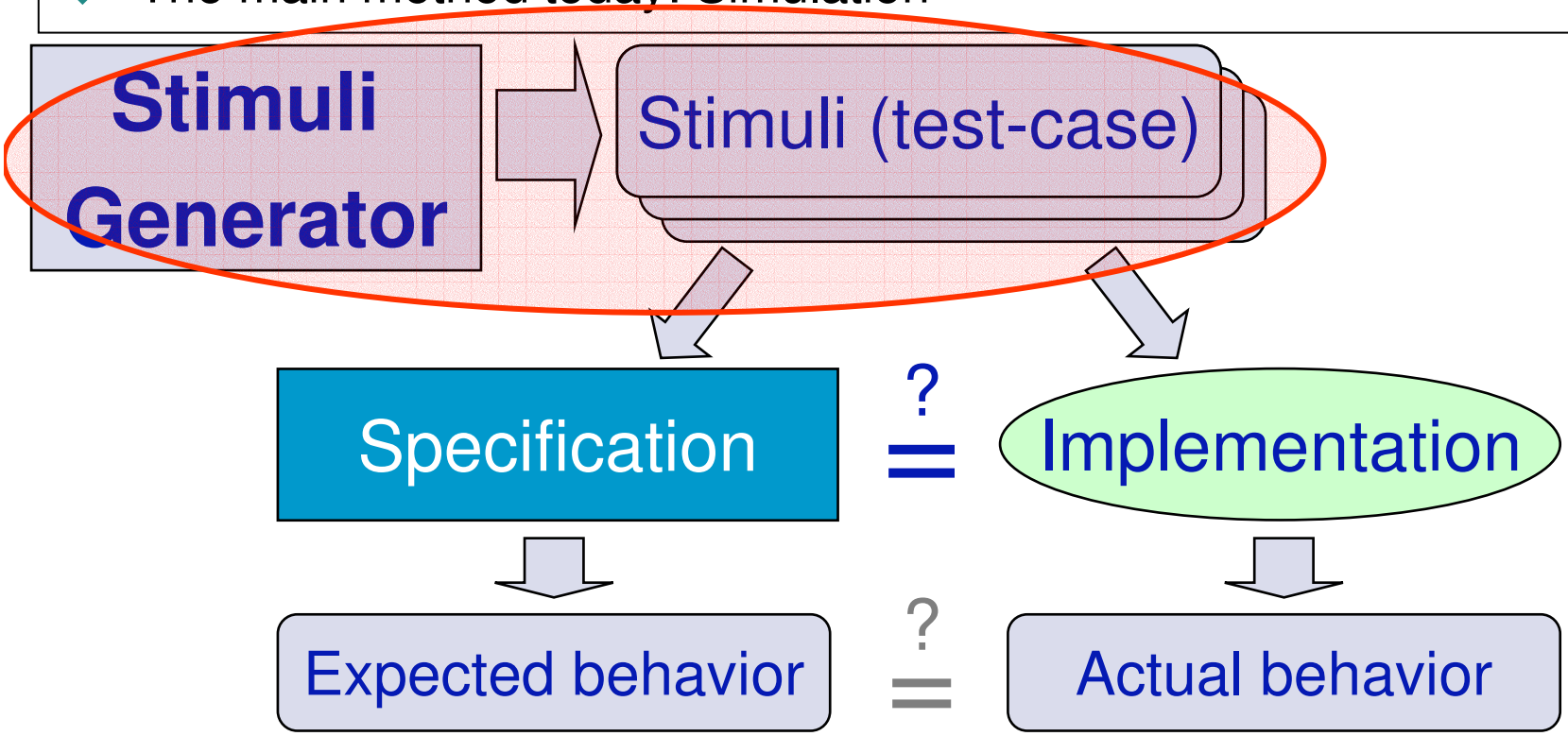
IBM

Quality Improvement Methods for System-level Stimuli Generation

Roy Emek (emek@il.ibm.com)
Simulation-based Verification Technologies

Simulation-based functional verification

- Verification: Show that a design (implementation) conforms to its specification
- The main method today: Simulation



Overview

- Systems and system verification
- The concept of Testing Knowledge
- Testing Knowledge mechanisms: three examples
- Testing Knowledge as a component in a verification methodology
- Experience
  - Functional coverage
  - A sample bug
- Implementation: constraints in Constraint Satisfaction Problems
- I too added a few slides …

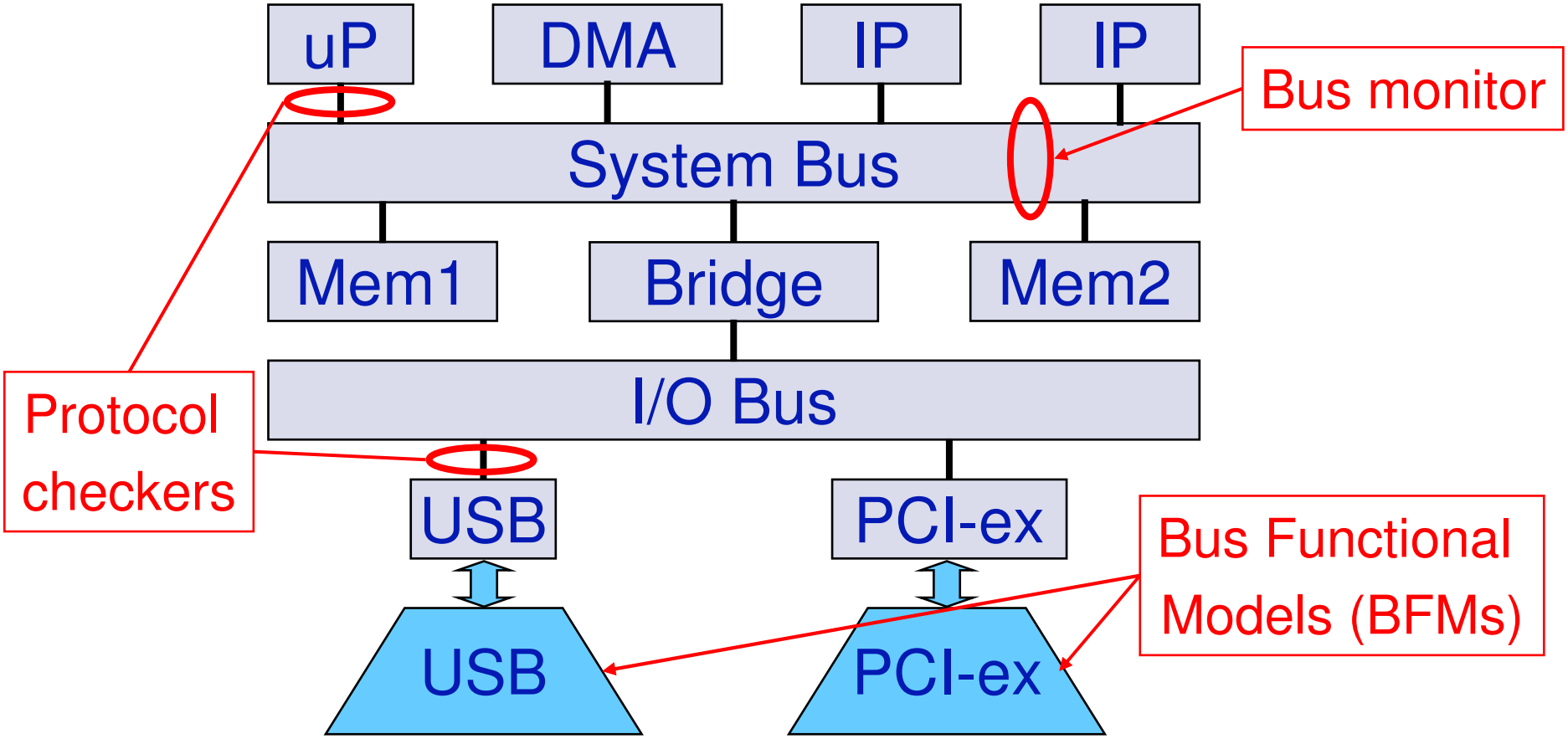
Systems and system verification

I'm speaking about HW

- A system:
  - A configuration of various components
    - E.g., processors, memories, bridges, encoders, interconnect, …
  - Capable of interacting with each other and with the outside world
    - E.g., DMA of 1K bytes, decoding three MPEG frames, …
- System verification: Verifying the integration of pre-verified components
- Challenges
  - Large designs
  - Complex specifications
  - Limited resources, specifically tight schedules
  - Remoteness (physical, in time) from the actual logic designers

➔ "Verification consumes ~70% of the design effort"

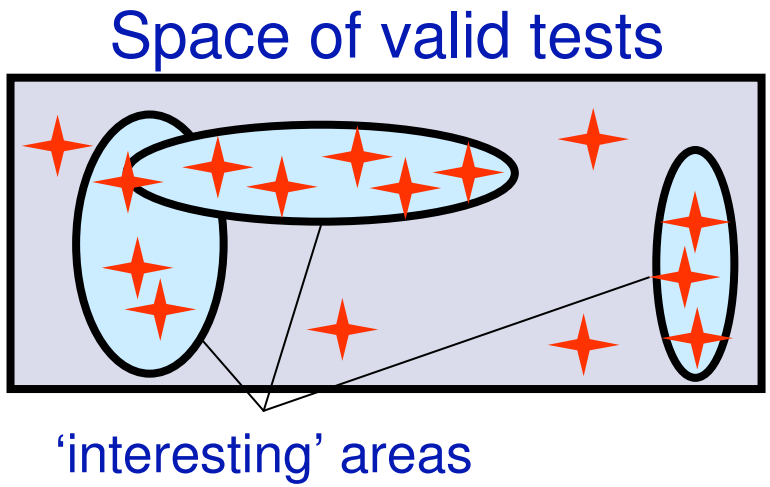
A major solution direction: reuse



TK represents another form of reuse

The concept of system-level testing knowledge

- Testing knowledge (TK): A set of mechanisms that aim at improving test-case quality
- Capitalize on recurring architectural concepts, such as:
  - Caches
  - Address translation and translation tables
  - Multiple instances of the same component type
- The basic mechanism: non-uniform random choice
  - Bias towards ‘interesting’ areas
- Affects all generated test cases
  - But can be controlled by the users of the stimuli generator



X-Gen

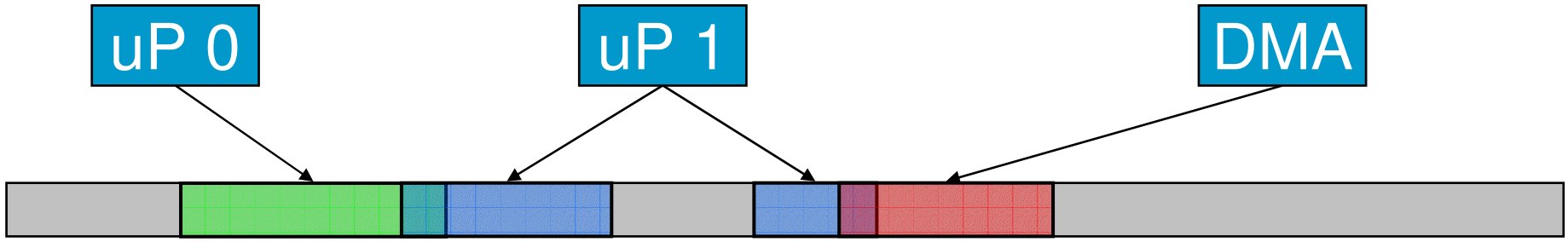
- X-Gen: a system-level test-case generator
  - An in-house tool
  - Used for the verification of several high-end systems in IBM
- The ideas presented here were developed during our work on X-Gen
  - Influenced by ideas from the processor verification domain
  - I.e. Genesys

Overview

- Systems and system verification
- The concept of Testing Knowledge
- Testing Knowledge mechanisms: three examples
- Testing Knowledge as a component in a verification methodology
- Experience
  - Functional coverage
  - A sample bug
- Implementation note: Constraint Satisfaction Problems

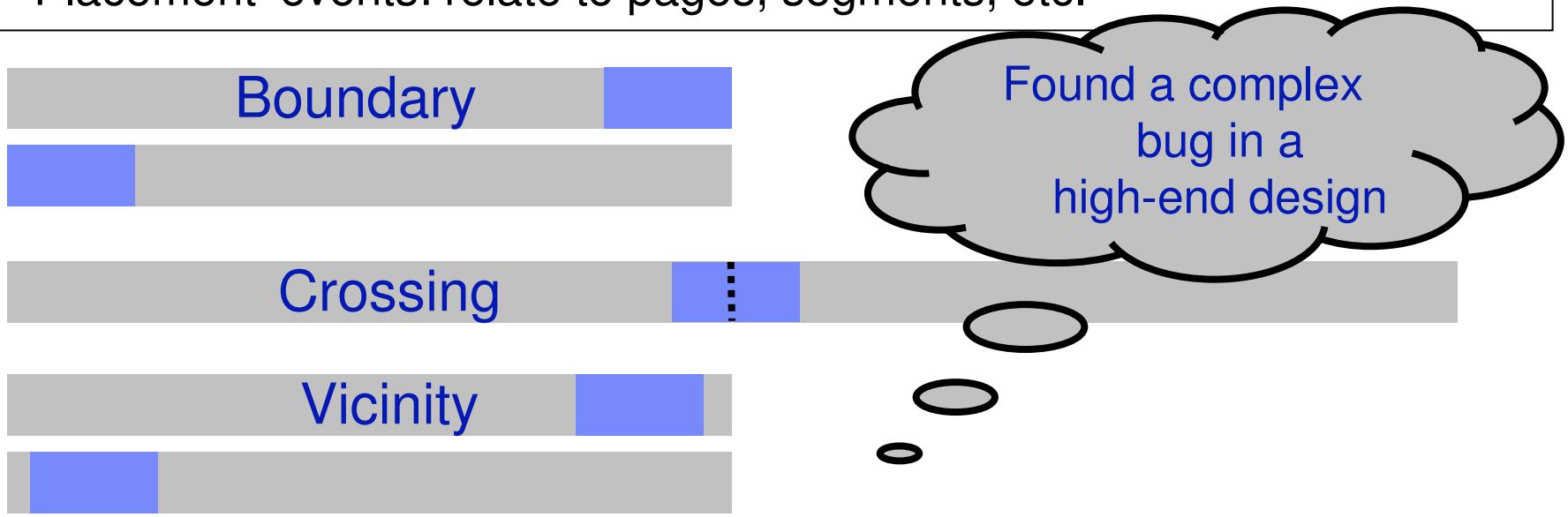
Testing knowledge Example #1: Resource contention

- System level ➔ multiple components ➔ parallelism
- Resource contention is a frequent cause of system-level bugs
  - Example: cache coherency and consistency
- Resource collision TK mechanism
  - Maintain a queue of recently accessed resources
  - With probability X, use one of the resources in the queue
- System-address is a typical system-level resource identifier
  - Resource contention ≈ address collision

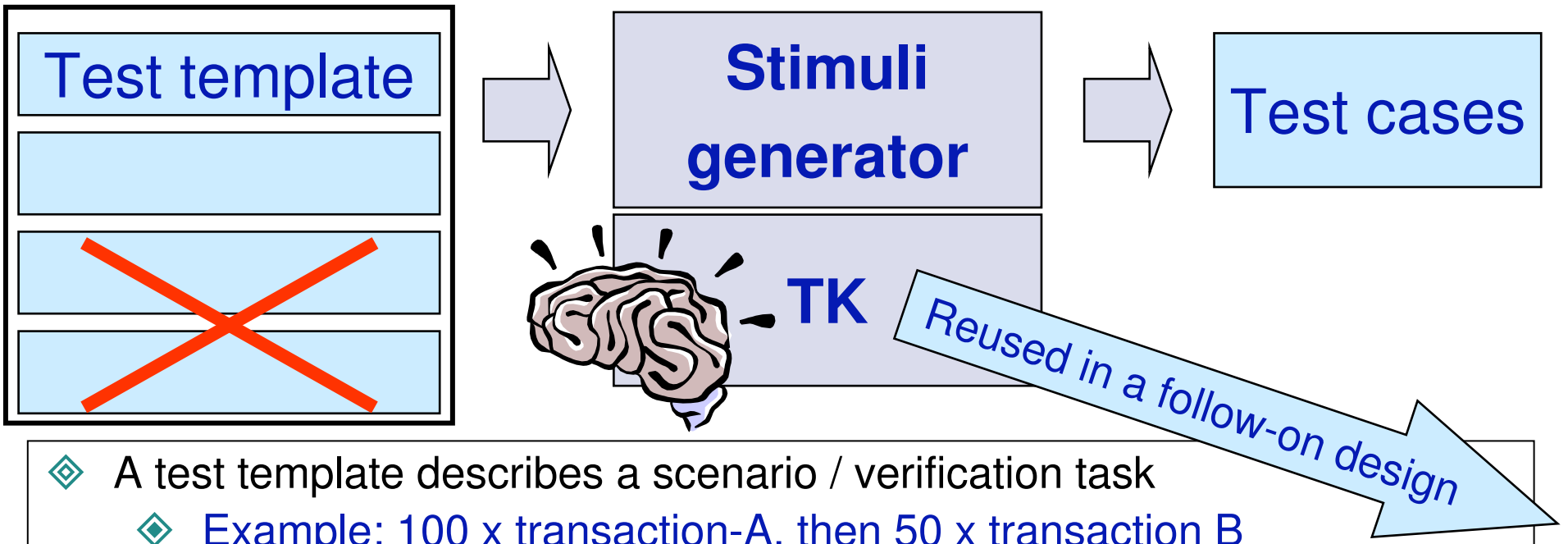


Testing knowledge Example #2: Address translation

- Address translation is a means for decoupling
  - Processors: virtual vs. physical
  - System address space vs. I/O address space
  - High-end interconnect
- 'Placement' events: relate to pages, segments, etc.

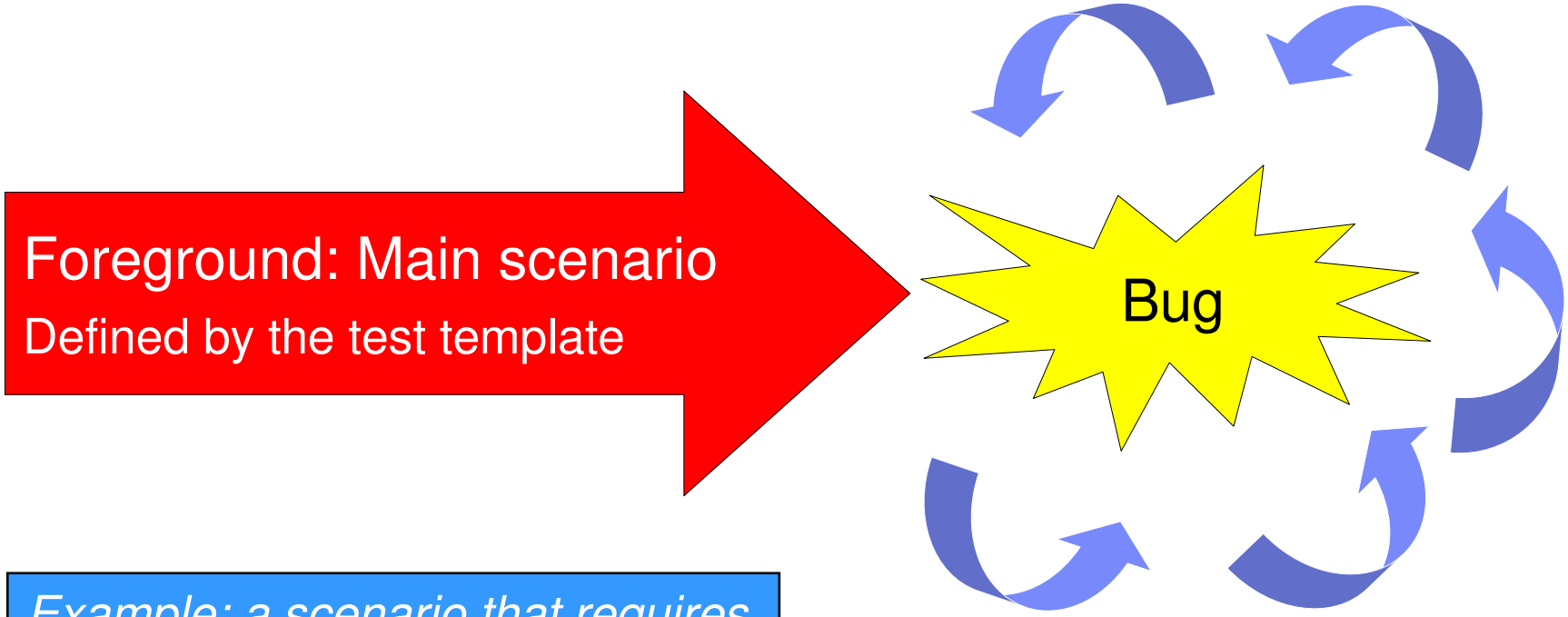


Testing knowledge as a form of reuse



- A test template describes a scenario / verification task
  - Example: 100 x transaction-A, then 50 x transaction B
- The same TK is reused across all the test templates
  - Thus reducing the number of test templates
- Similar testing knowledge can be used across multiple systems

A foreground / background methodology



Example: a scenario that requires cache-misses would reduce the probability of address-collision

Background: testing knowledge
Intelligent random noise
Can be directed by the test-template

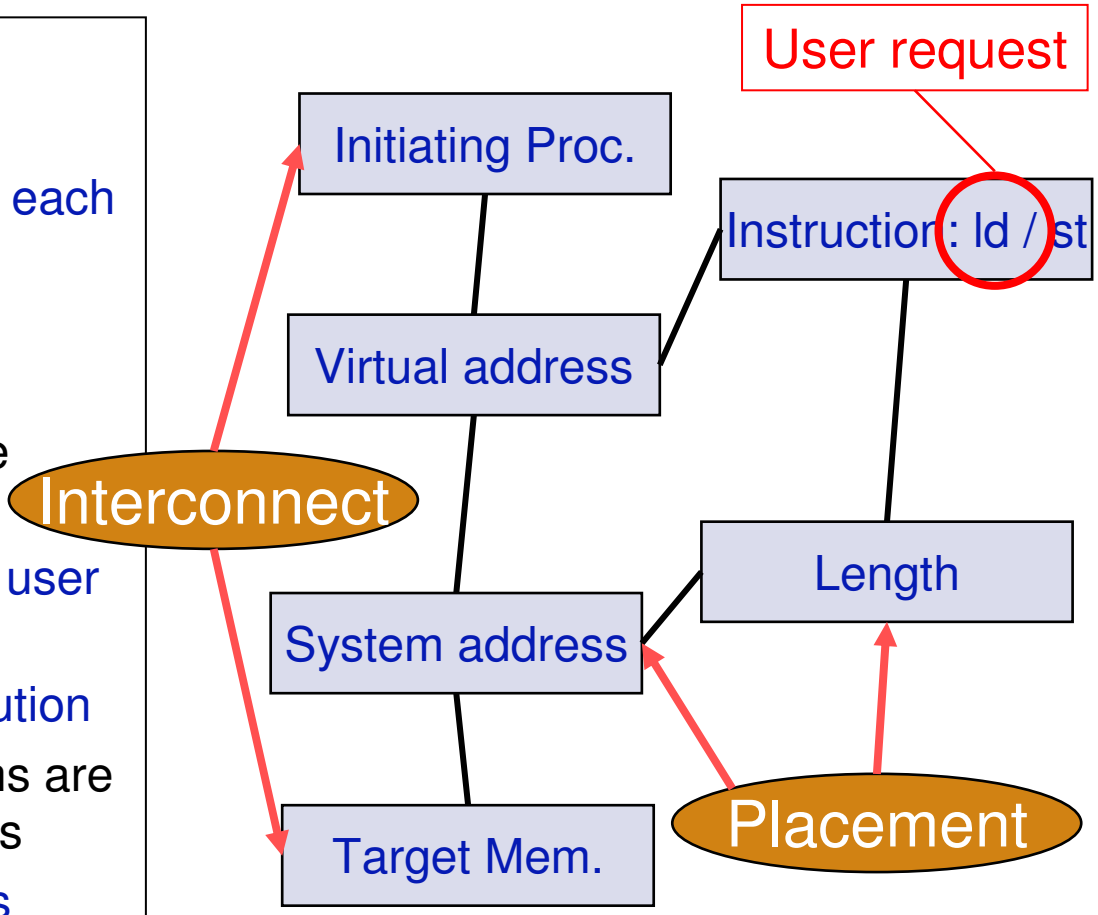
Usage experience – Power4+ based system

Category	TK based tool: X-Gen	Previous tool
No. of request files*	737	7168
Simulation cycles (normalized)	x1	x4.8
Coverage Model #1	40.57%	37.10%
Coverage Model #2	43.84%	26.88%
Coverage Model #3	74.28%	63.80%
Coverage Model #4	61.14%	59.17%

* Rough measurement of the human effort

Implementation note: Constraint Satisfaction Problems (CSP)

- CSP definition
  - A set of variables
  - A domain of valid values for each variable
  - Constraints define valid combinations of values
- The basis for modern test-case generation
  - Constraints impose validity, user requests, aim towards quality
  - A test-case is a random solution
- Testing knowledge mechanisms are implemented as soft constraints
  - Reused for multiple systems



Summary

- Testing knowledge: Directing stimuli generation to ‘interesting’ areas
  - Expanding coverage
  - Increasing the chances of hitting a bug
- Capitalize on recurring architectural concepts
- Examples: resource contention, placement, interconnect

- Reduces the cost of implementing a verification plan
  - Reuse of knowledge between test templates
  - Reuse of knowledge (and technology) between different systems
- And at the same time influences the verification plan

Thank You