2015 DESIGN AND VERIFICATION DVCON CONFERENCE AND EXHIBITION UNITED STATES

Navigating The Functional Coverage Black Hole:
Be More Effective At Functional Coverage Modeling

Jason Sprott
Verilab UK Ltd

Presented by:
Paul Marriott
Verilab Canada Inc

Matt Graham
Cadence Design Systems

Aim to avoid this:

Agenda

- Planning Coverage
- Kinds of Coverage
- The MECE Mindset
- Coverage Collection Concerns
- Register Coverage: Blessings and Curses
- SystemVerilog 1800-2012 Coverage Extensions?
- Coverage Best Practice Checklist
- Coverage Implementation Review
- Ten Key Review Checklist Items

Coverage Planning

- Coverage is, unfortunately, a manual process
  - Without AIs to read a spec, ***you*** have to define it!
  - Potentially a huge amount of data
  - Should always be traceable to the specification
- Three key criteria
  1. Accurate
  2. Representative
  3. Complete

Coverage Planning

- Coverage data is not useful unless it is ***accurate***
- We must ensure that we:
  - Use the correct trigger events
  - Use correct sampling events
  - Avoid false positives
  - Avoid creating results that can be misunderstood
  - Avoid multiple paths to the same outcomes

Coverage Planning

- Coverage data has to be ***representative***
  - Two categories here:
    1. Functional
       - Impossible or impractical to hit **all** possible scenarios
    2. Priority
       - Ruthless prioritization is needed to achieve the business goals
         - » Have we hit the core functionality required for the project?
         - » Have we covered enough of covered all the highest risk items?

Coverage Planning

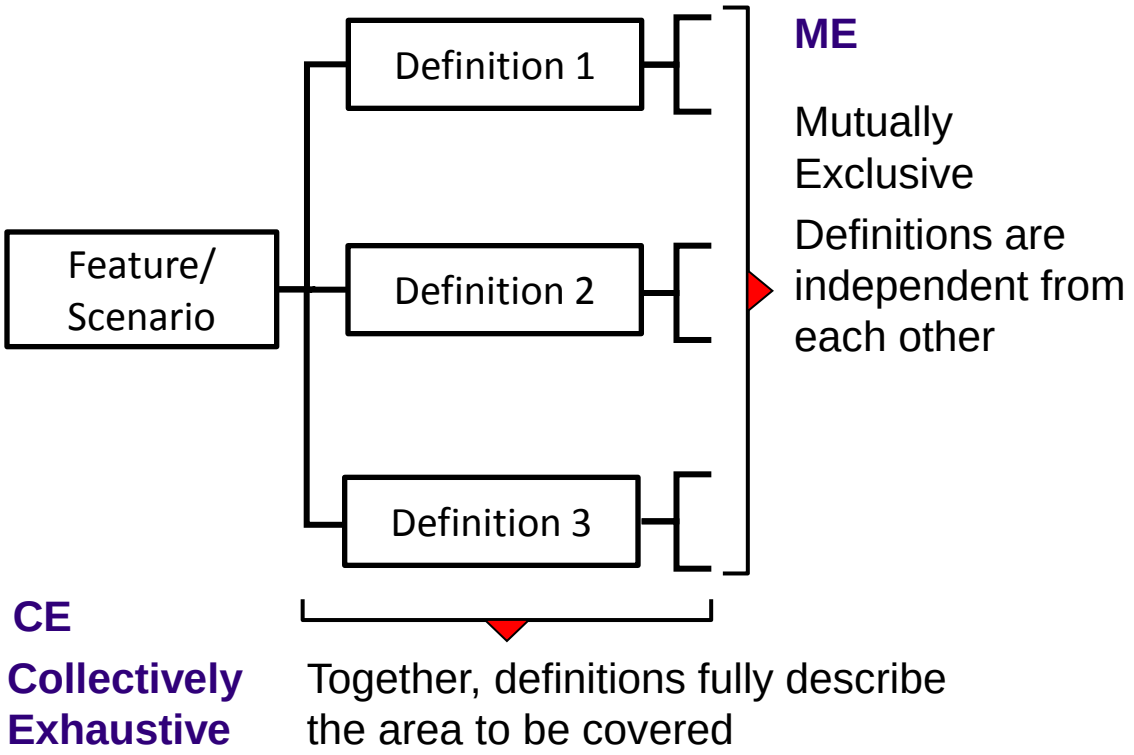
- Completeness
  - Based on scope with a representative subset set of functionality
    - Need to determine ***collectively exhaustive*** feature list
- May not be possible to *implement* all features
- ***Analysis*** is still important
  - Subtle but important items may otherwise be missed

Kinds of Functional Coverage

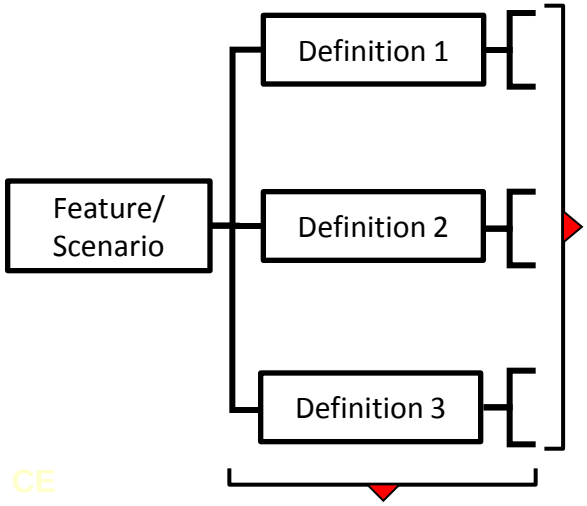
- Use Cases
- Interesting Scenarios
- Register Coverage
- Protocol
- Modes of Operation
- Responses
- Temporal Proximity
- State Machines
- Cross cutting concerns
- Error Injection
- Illegal Conditions
- Analog / Mixed Signal

The MECE Mindset

- Mutually Exclusive Collectively Exhaustive



The MECE Mindset



ME

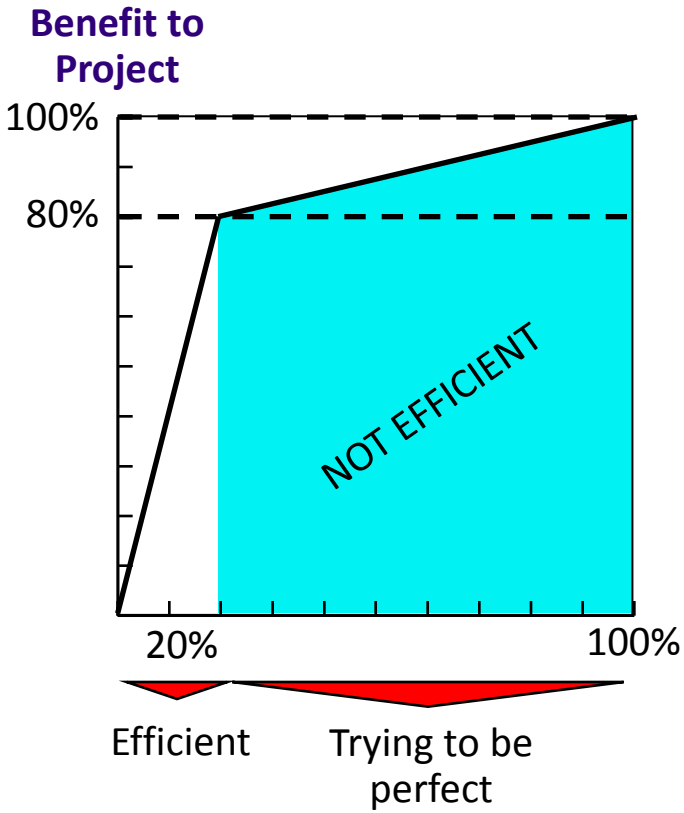
Mutually Exclusive

Definitions are independent from each other

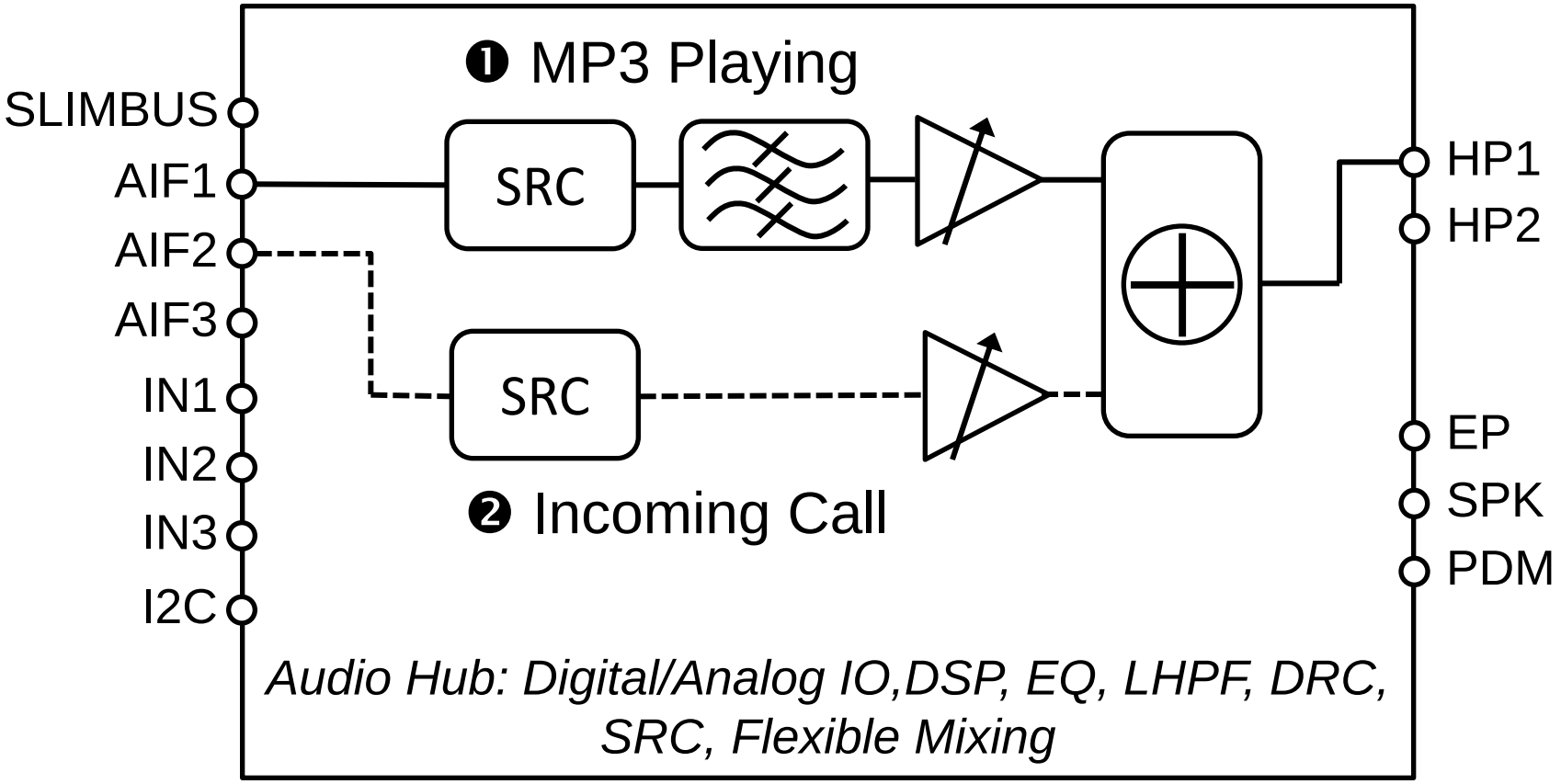
CE

Collectively Exhaustive

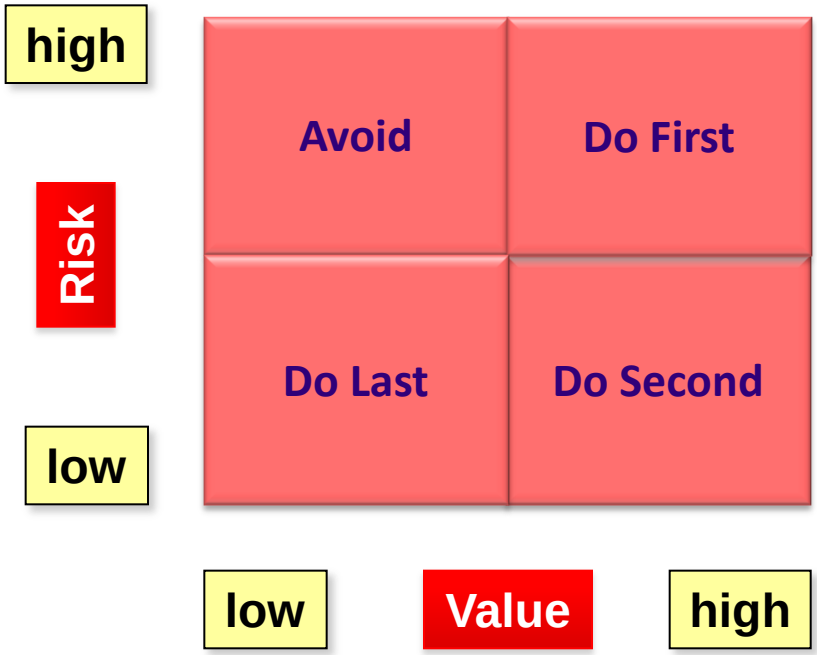
Together, definitions fully describe the area to be covered

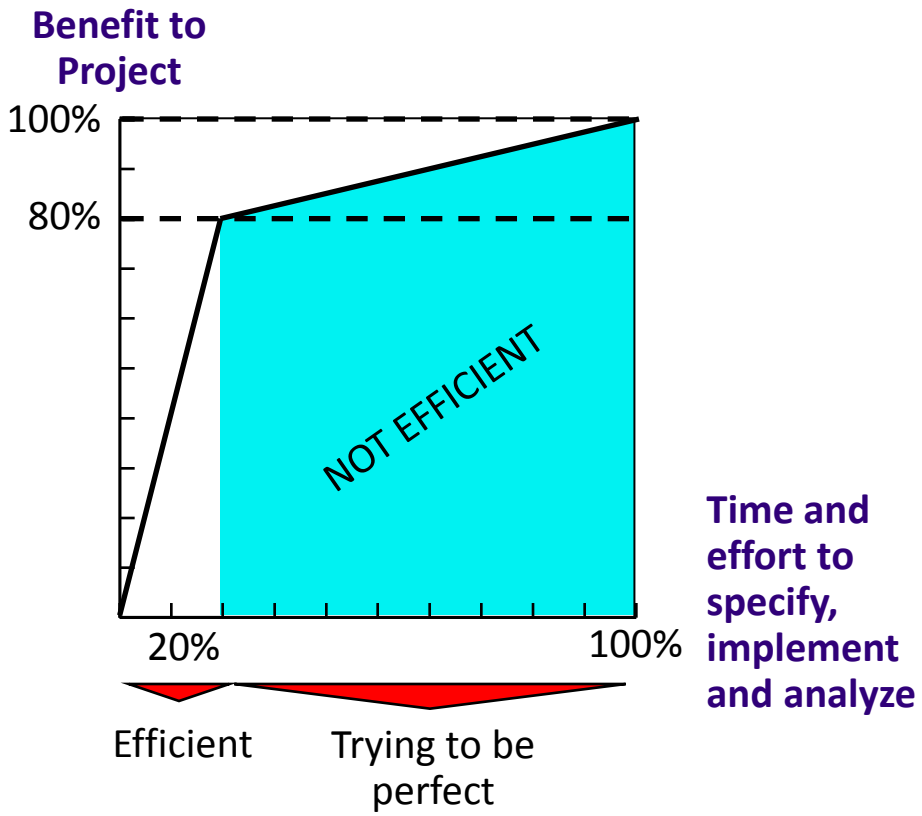


Audio Hub Example

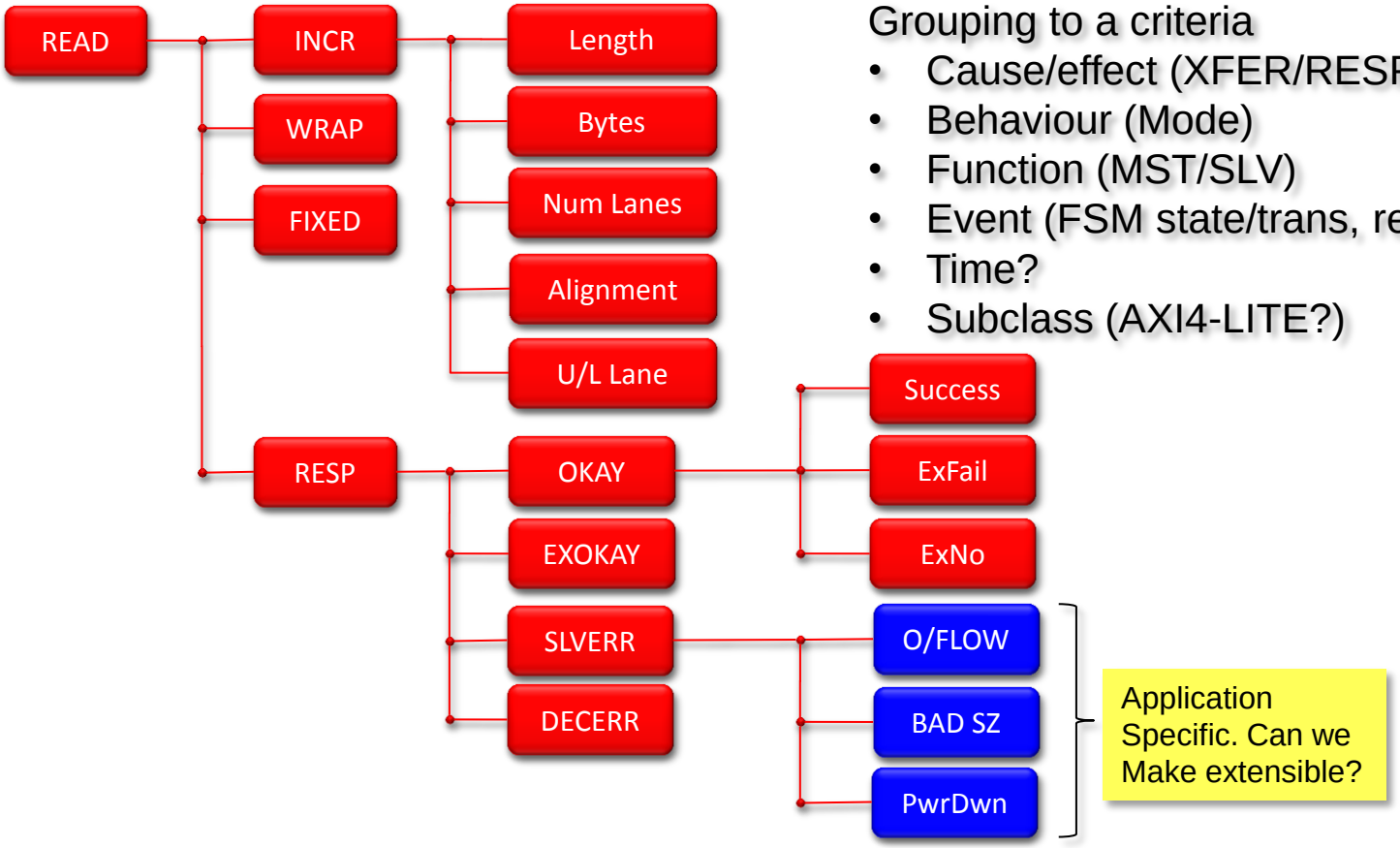


Ruthless Prioritization





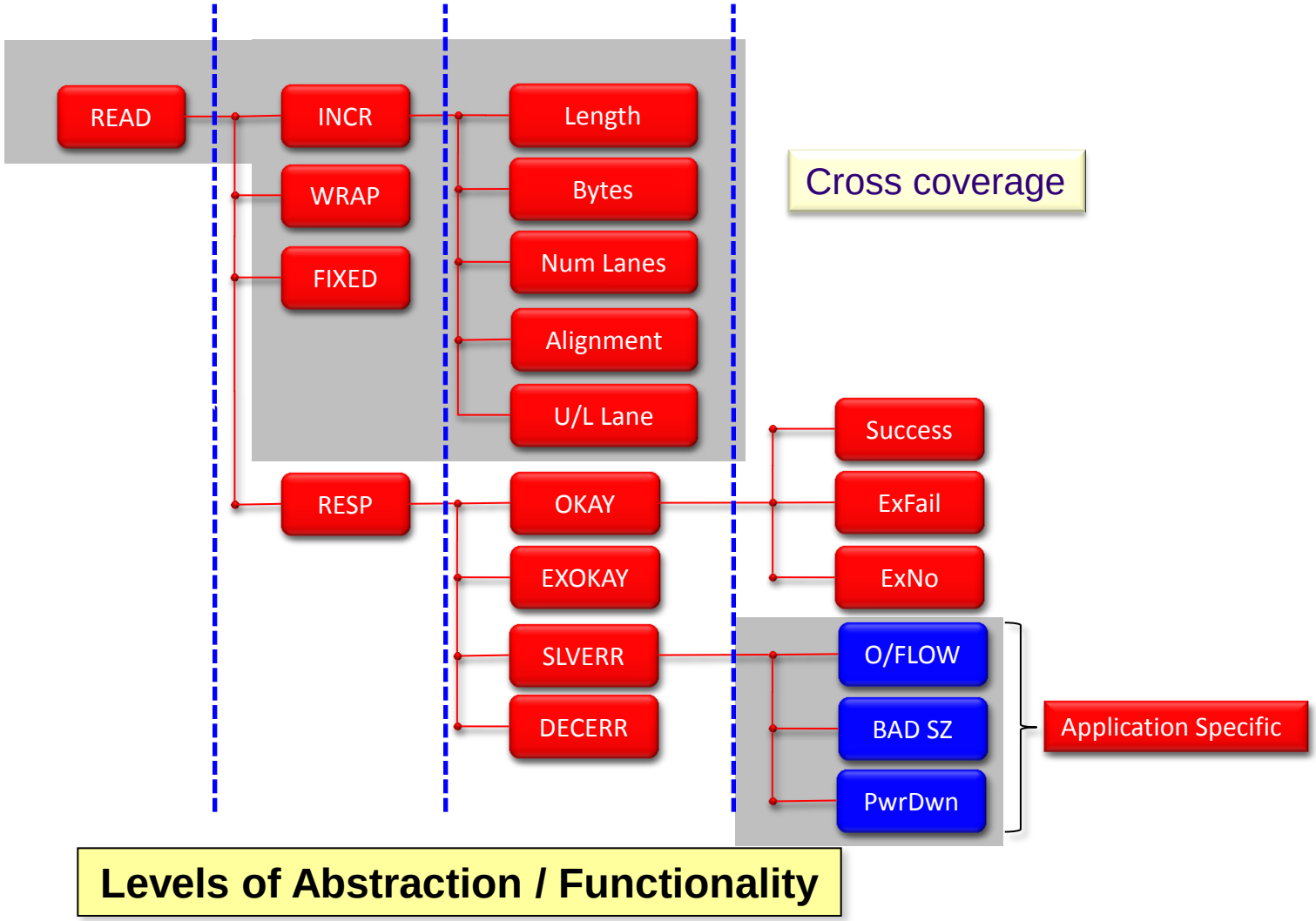
AXI Bus Protocol Example



Grouping to a criteria

- Cause/effect (XFER/RESP)
- Behaviour (Mode)
- Function (MST/SLV)
- Event (FSM state/trans, reset)
- Time?
- Subclass (AXI4-LITE?)

AXI Bus Protocol Example



Coverage Collection Concerns

1. Identify important values
2. Relationships between values
3. Illegal Conditions
4. When to sample / When to Ignore
5. Conditionals
6. Post-sample behaviour
7. Accuracy

- All of the above are useful for an implementation review
  - Together with priorities and targets

Coverage Collection Concerns

- Identify important values
  - Coding covergroups is quite simple(!)
    - Easy to generate large amounts of data
  - Identify normal cases
  - Identify exceptions
  - Group values into ranges, corner cases
    - Think of downstream analysis of holes
  - Beware of input frequency of stimulus

Coverage Collection Concerns

- Relationship between values
  - Direct correlations between values in current sample
  - Relationships that transition state
  - Cause and effect
    - E.g. enable leading to entering/leaving a state
    - Can the enable be changed dynamically?
      - When can the enable change?
      - What adjacent behaviour is implicated?
- Don’t get carried away!
  - But these observations can be very useful

Coverage Collection Concerns

- Illegal conditions
  - Conditions that cannot happen?
    - Can lead to holes that can't be hit
    - What if they do happen? Throw an error?
  - Conditions that should not happen?
    - Trap as errors?
    - Just ignore?
    - What about re-use?
  - Aim is to avoid having to analyze downstream
  - Can reduce the coverage state space though

Coverage Collection Concerns

- When to sample
  - Beware of being greedy or sloppy
    - Data might not be valid in all cases
      - Be especially aware of false positives
  - Avoid oversampling
    - Use lowest frequency sampling event related to functionality
    - Gives false sense of security
  - Avoid undersampling
  - Be aware of measurements that span multiple samples

Coverage Collection Concerns

- When to ignore
  - During certain modes of operation
    - E.g. during reset, clock-gear changing, power transitions
  - Difficulty with reliable sampling
    - Risk/reward payoff – a feature may not be worth the effort to sample correctly
  - Prioritization
  - Low risk and low value items may not be worth the effort
  - Be aware of impact on data and sampling events

Coverage Collection Concerns

- Conditionals
  - Potential source of false positives
    - E.g. (A or B) ➔C
      - If A and B are not covered separately, can't know all the paths into C
      - Usually seen in cover properties
        - » Applies to covergroup sampling events too, though
  - Many forms of conditionals that hide paths and states

Coverage Collection Concerns

- Post-sample behaviour
  - What events can change the validity of a sample after-the-fact?
  - Checkers typically have to handle this
    - Otherwise there could be many false errors
  - Coverage bins may be incremented without end-to-end data validation
    - Essentially gives false positives

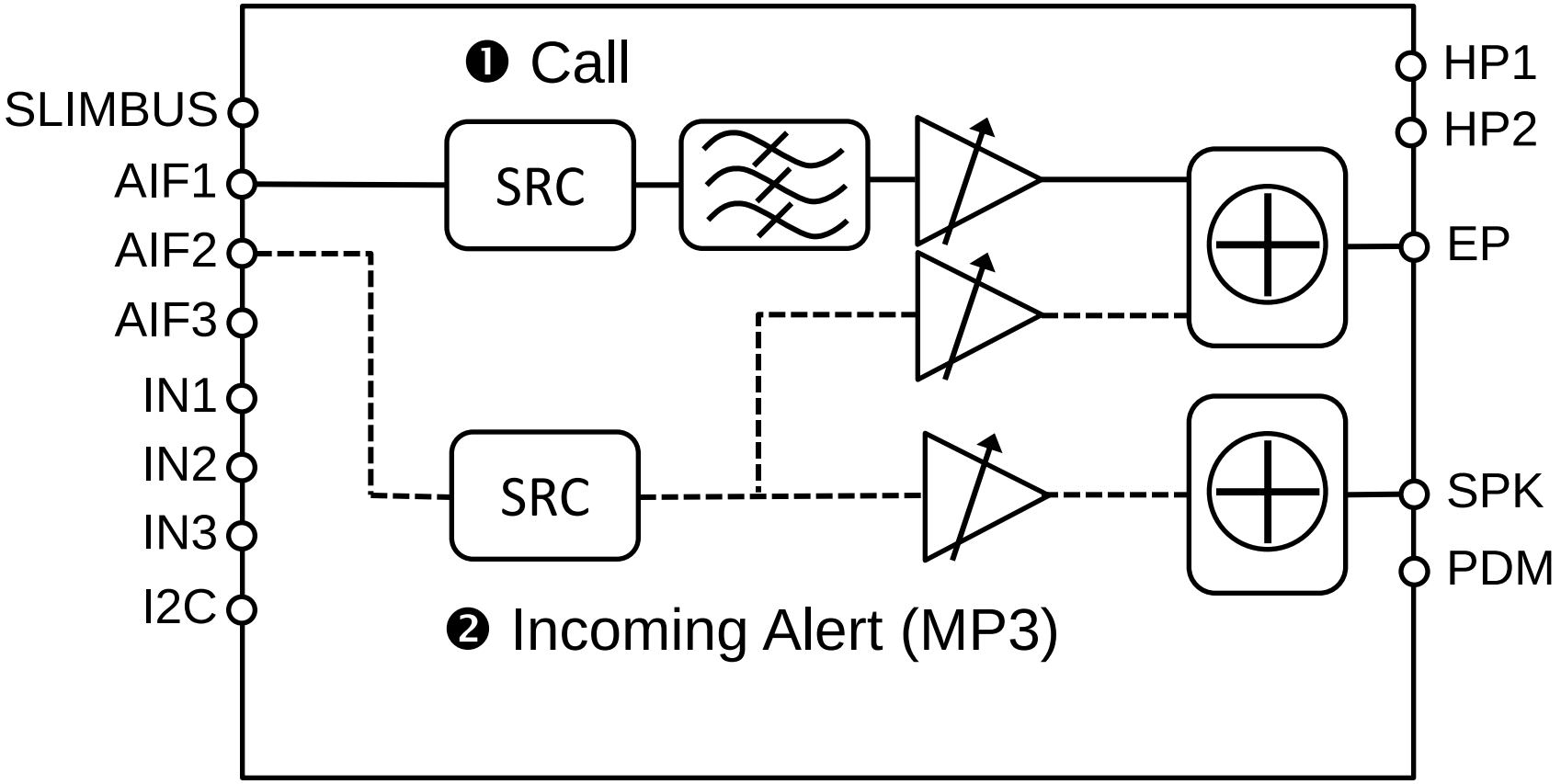
Coverage Collection Concerns

- Accuracy
  - Covergroups tend to group data into transactions
    - May not coincide with actual net activity
    - Modeling can take a lot of effort
      - Some behaviour may be assumed, not observed
  - Never acceptable to assume something is acceptable!
    - Built-in inaccuracy is difficult to spot
      - Same problem as spotting a disabled check
  - If a trade-off is made, make sure it's obvious for review later

Register Coverage: Blessings and Curses

- Register Coverage
  - Often built-in and comes “for free”
    - Some classify this as akin to code toggle coverage
  - However, it’s more useful than nothing
  - Many features can be verified to some extent
  - Can give a false sense of security
  - Typically **not** related to the features of the blocks containing the registers themselves
  - Both sequence and values may be important

Audio Hub Example



Register Coverage: Blessings and Curses

- Register Coverage
  - Sequence of operation of registers is important
    - As a call comes in, datapaths have to be switched
    - Volume control registers ramped down for the MP3 path, ramped up for the call
    - Once the call is over the MP3 path has to resume
  - Extremely unlikely to hit any of the above scenarios using register testing
  - Deep understanding of the use cases is required

Review of SV-2012 Additions

- Working towards making coverage easier to express
  - Coverpoint Variables
  - Coverage *bin … with* expressions
  - Functions in covergroups
    - Clearer than the current *binsof* / *intersect* which are hard to understand
    - A good idea, but very difficult to make extensible
- More details in our paper

Coverage Best Practice Checklist

- Ensure reused coverage is still accurate
  - Interface usage may be different at different integration levels
- Name coverage based on intent
  - Always keep future readers and reviewers in mind
- Groups multiple conditions/threads appropriately
  - Beware of hiding untaken paths
- Use bin ranges to reduce data volume
  - but check are meaningful and don’t hide corner cases

Coverage Best Practice Checklist

- Beware of re-implementing toggle-coverage
  - Focus on functionality and correlations
- Reduce complex crosses to useful values
- Use illegal bins sparingly and provide a disable
  - Can be more useful for debug
- Use “free” register coverage (eg uvm_reg)
  - but check usecases and design features separately
- Split SVA cover statement and covergroups
- Include sequence / scenario coverage

Coverage Implementation Review

- Three broad review points
  1. Style: for inspection, maintenance, and reuse
  2. Sampling events: errors, over/under sampling
  3. Bin triggering: errors, relevance, missing
- If the coverage is misleading, confusing or worthless:

  It will cost more in the long run to analyze than the value it brings to the project
- Value and reliability are **key metrics**

Coverage Review Processes

- Assumptions
  1. The vPlan has been reviewed
  2. Any priorities are valid
  3. Access to checkers is available
- Imported coverage validity
- Deep dive (normal operation)
  1. Have all the checks been reviewed?
  2. Are all coverage concerns addressed?
  3. MECE analysis for quality spot check
- Deep dive (corner cases)

Ten Key Review Checklist Points

1	Review the verification plan	☑	Completeness
2	Review the coverage being analyzed	☑	Accuracy
3	Review coverage not part of the verification plan	☑	Completeness
4	Every coverage item should have a check	☑	Accuracy
5	Check vPlan is mapped to implemented coverage	☑	Representative
6	Check sampling criteria (avoid over and under)	☑	Accuracy
7	Review binning and illegal conditions	☑	Completeness
8	Mapping of the coverage to the specification and business goals	☑	Representative
9	Prioritization aligned to business goals	☑	Representative
10	Involve all stakeholders in the review process	☑	Accuracy