FUNCTIONAL COVERAGE IN CHISEL

DTU - Department of Applied Mathematics and Computer Science

OUTLINE

- What is functional coverage?
- Our solution for implementing Functional Coverage in Chisel/Scala.
- What's left to add from SystemVerilog?

WHAT IS FUNCTIONAL COVERAGE?

FUNCTIONAL COVERAGE: WHAT IS IT?

- Line coverage = **quantitative** approach to getting the verification progress.
    - How much code have we tested?

- Functional coverage = **qualitative** approach to getting the verification progress.
    - How many features have we tested?

FUNCTIONAL COVERAGE: WHAT IS IT?

- Coverage based off of a **verification plan** defined by:
    - **CoverGroups**: set of DUT ports that will be sampled together, named CoverPoints.
    - **CoverPoints**: Set of values that a port is expected to have to verify a feature, those constraints are called bins.
    - **Bins**: Value ranges or transition, that a port is expected to have to verify a feature.

- Verification plan should be representative of the DUT's expected features.

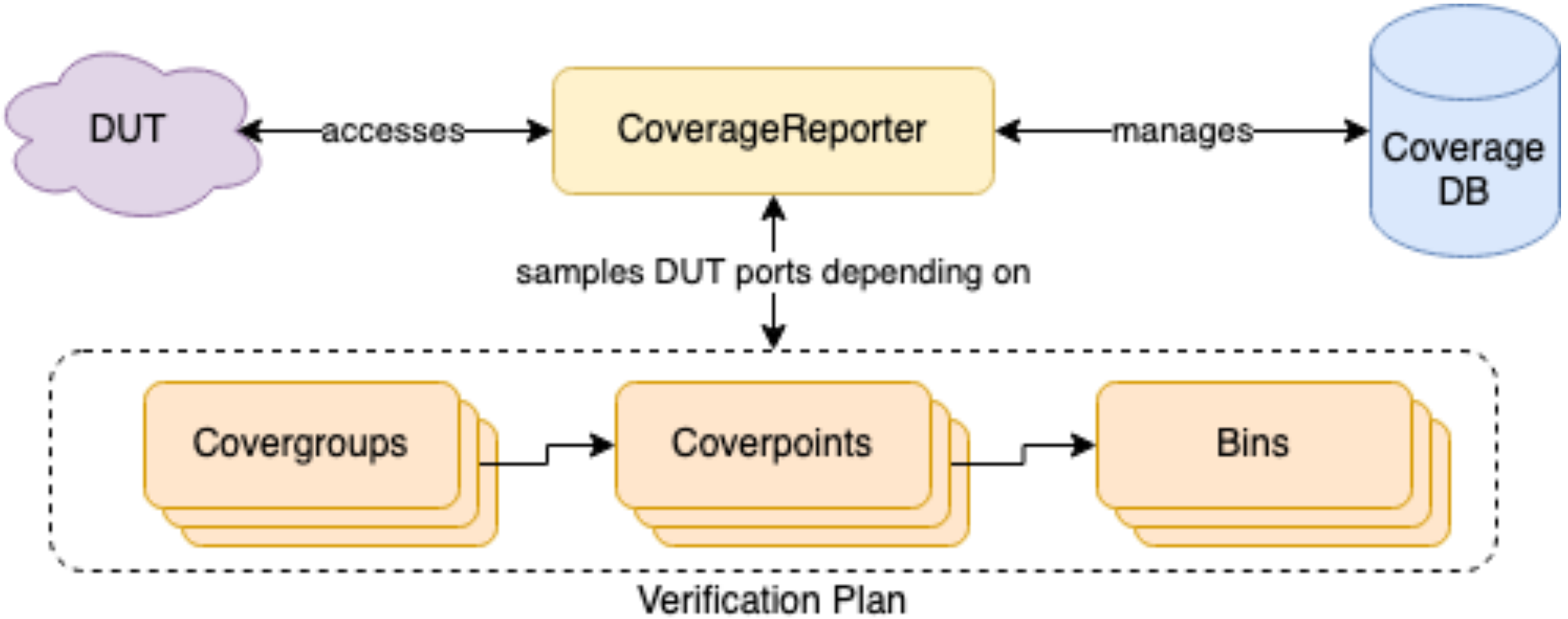
FUNCTIONAL COVERAGE: CROSS COVERAGE

- Most useful part of functional coverage:
  - *Do my signals cover these two values simultaneously?*

- Cross between two coverpoints:
  - Once two cover points are defined, we can defined an extra point that is the **cross** of the two points.
  - We can then define two ranges (one for each point) and check how many value pairs within this range have been hit during a test suite.

IMPLEMENTING FUNCTIONAL COVERAGE IN CHISEL/SCALA

IMPLEMENTING FUNCTIONAL COVERAGE: OVERVIEW



IMPLEMENTING FUNCTIONAL COVERAGE: THE METHOD

- The idea is to create something similar to SystemVerilog's *CoverGroup, CoverPoint* and *Bins* constructs.

- **CoverageDB:** DataBase that maintains the values gathered for all of the bins across multiple tests in a test suite.

- **Coverage Reporter:** Handles the registration of CoverPoints and Bins to the DB, samples the bin values and creates the coverage report.
    - This is used to create the verification plan.

IMPLEMENTING FUNCTIONAL COVERAGE: USE EXAMPLE

- Create the coverage reporter and verification plan.

```
val cr = new CoverageReporter
cr.register(
    //Declare CoverPoints
    CoverPoint(dut.io.accu , "accu", //CoverPoint 1
        Bins("lo10", 0 to 10)::Bins("First100", 0 to 100)::Nil)::
    CoverPoint(dut.io.test, "test", //CoverPoint 2
        Bins("testLo10", 0 to 10)::Nil)::
  Nil,
  //Declare cross points
  Cross("accuAndTest", "accu", "test",
      CrossBin("both1", 1 to 1, 1 to 1)::Nil)::
  Nil)
```

- Sample the CoverPoints inside of the test.

```
cr.sample()
```

RESULT: FUNCTIONAL COVERAGE REPORT

- Create and print the coverage report

```
//Print coverage report
cr.printReport()
```

- Result:

```
============ COVERAGE REPORT ============
============== GROUP ID: 1 ==============
COVER_POINT PORT NAME: accu
BIN lo10 COVERING Range 0 to 10 HAS 8 HIT(S)
BIN First100 COVERING Range 0 to 100 HAS 9 HIT(S)
=========================================
COVER_POINT PORT NAME: test
BIN testLo10 COVERING Range 0 to 10 HAS 8 HIT(S)
=========================================
CROSS_POINT accuAndTest FOR POINTS accu AND test
BIN both1 COVERING Range 1 to 1 CROSS Range 1 to 1 has 1 HIT(S)
=========================================
```

WHAT'S LEFT?

WHAT'S LEFT?

- Bins only cover value ranges, should add **transitions** as well

- SystemVerilog(SV) has **conditional coverage**: When sampling, only consider bins that verify an extra predicate.

- SV also has **auto-sampling**: Sample automatically when a predicate is verified.

- SV has **auto-bins**: Cover all possible values for a port automatically if no bins were specified.

REFERENCES

- Chris Spear, 2006, “SystemVerilog for Verification: A Guide to Learning the Testbench Language Features”, Synopsys Inc., 377 Simarano Drive Marlboro MA 01752 USA

- Current Project repository:
https://github.com/chisel-uvm/chisel-verify

QUESTIONS?