Use of Verification Point Tools

When (and whether!) to use them

Michael Thompson, Rianta Solutions
Thomas Zboril, Huawei Technologies Canada

April 21, 2017
Canada

Agenda

Goals for this Presentation.

Context for using point tools: a Generic Development Process.

Discussion of Three Point Tools for Verification.

Conclusions.

What is a “Point Tool”?

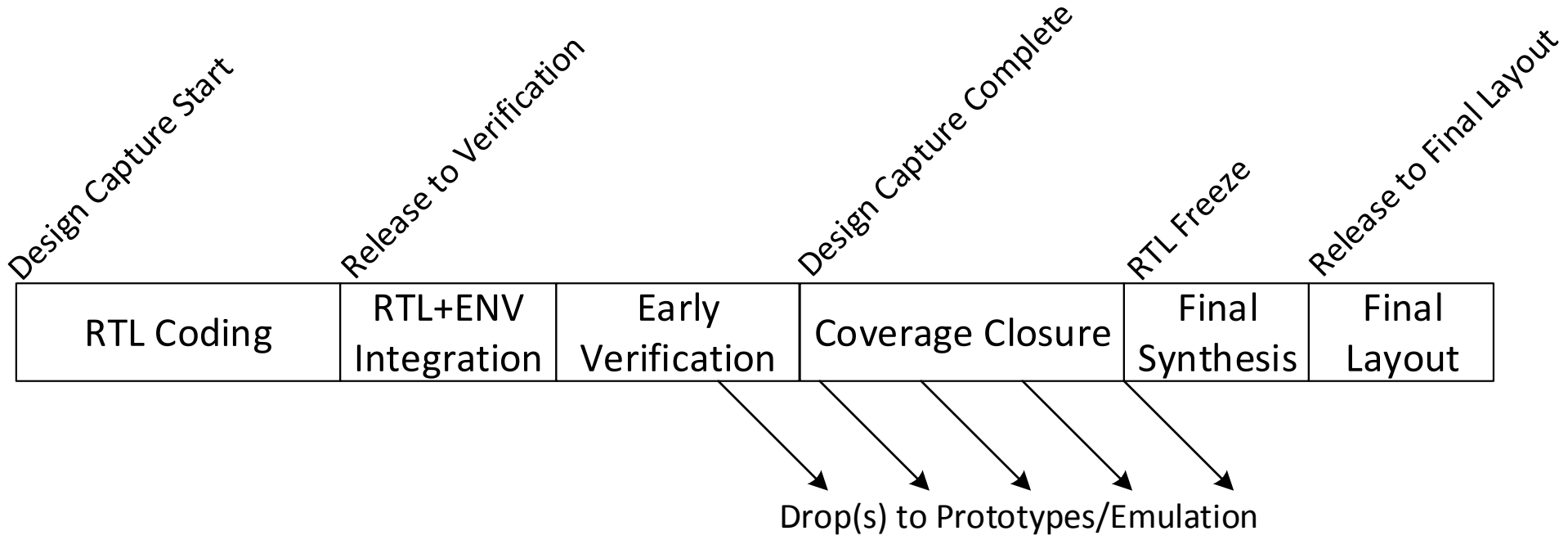
- It’s a made up word! ☺
- In this discussion a point tool shall mean any tool that is used at one or more specific short intervals (points) in the ASIC/SoC/IC/FPGA development process.
  - Contrasts with a simulator that is used throughout the whole process.
- We will look at three point tools:
  - XPROP
  - FCA
  - Certitude

Goals for this Presentation

- This Presentation is not a tutorial about tools.
  - Your Synopsys CAE will do a better job of that than we ever could.
- Our goal is to assist you in determining the best way to apply each of these tools to your project.
- This primarily translates to ***when*** to use them:
  - Using too early can waste time.
  - Using too late may mean unnecessary delays to your schedule.
- It is also important to consider the ***cost*** of using point-tools:
  - In this context, cost means time and effort.

A Generic Development Process

- To get the most from a point-tool, it should be used at the right time.
- Here, we introduce the concept of a generic development process:
  - We will use it to define where each of these point-tools fits in that process.
  - Mapping from this generic process to your specific process is an exercise for the reader…



Three Verification Point Tools

- XPROP:
  - XPROP is a simulation tool that adds “X-pessimism” to RTL simulations.
  - Useful for finding reset and initialization issues that do not always manifest themselves in RTL simulations, but do exist at the gate level.

- Formal Coverage Analyzer:
  - FCA is a static tool that looks for unreachable code in the RTL and creates code coverage exclusions for the unreachable code.

- Certitude:
  - A fault-injection tool that can be used to measure the quality of your Verification Environment.
  - Works by injecting faults into your RTL and determining whether or not the Verification Environment is capable of detecting them.

Point Tool #1:

XPROP

What is XPROP?

- A simulation tool to add X-pessimism to RTL simulations.
- Simulation results will be close to, but not exactly the same as gates:
  - Xmerge is overly pessimistic.
  - Tmerge behavior is close to gates.

```
// SEL is Unknown (X)
always @* begin

  if (SEL) begin
    OUT <= A;
  end
  else begin
    OUT <= B;
  end
end
```

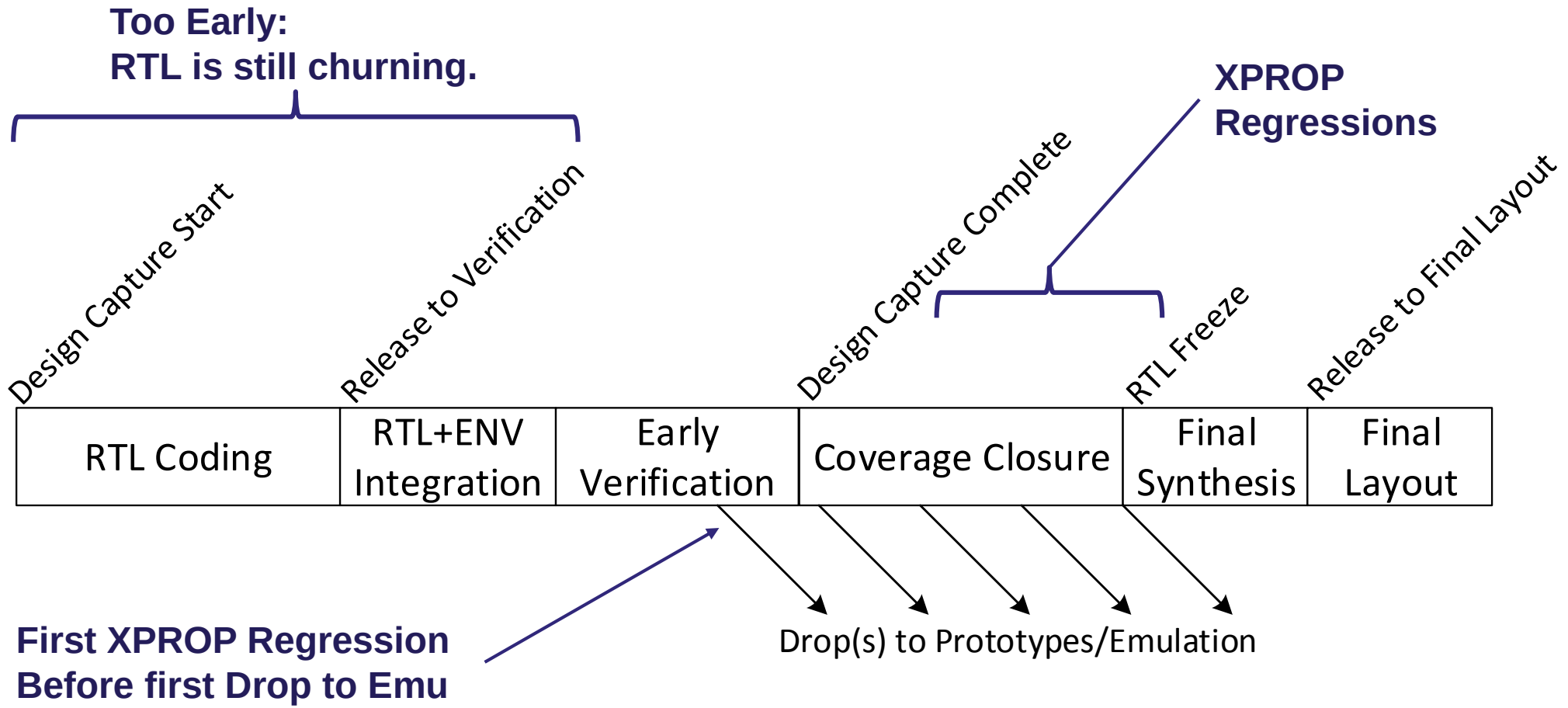
In an RTL sim, OUT will always follow B when SEL is unknown.

SEL	A	B	RTL	tmerge	xmerge
X	0	0	0	0	X
X	0	1	1	X	X
X	1	0	0	X	X
X	1	1	1	1	X

The Cost of Using XPROP

- XPROP is now a mature technology.
  - Of the three tools discussed here, the value of XPROP is easiest to recognize.
- Set-up is trivial:
  - Create a (typically very simple) configuration file.
  - Add **–xprop=[*cfg_file*]** to the VCS command line.
- Run-time impact is on the order of 15% to 20%.
- Licensed separately from VCS.

When to Use XPROP



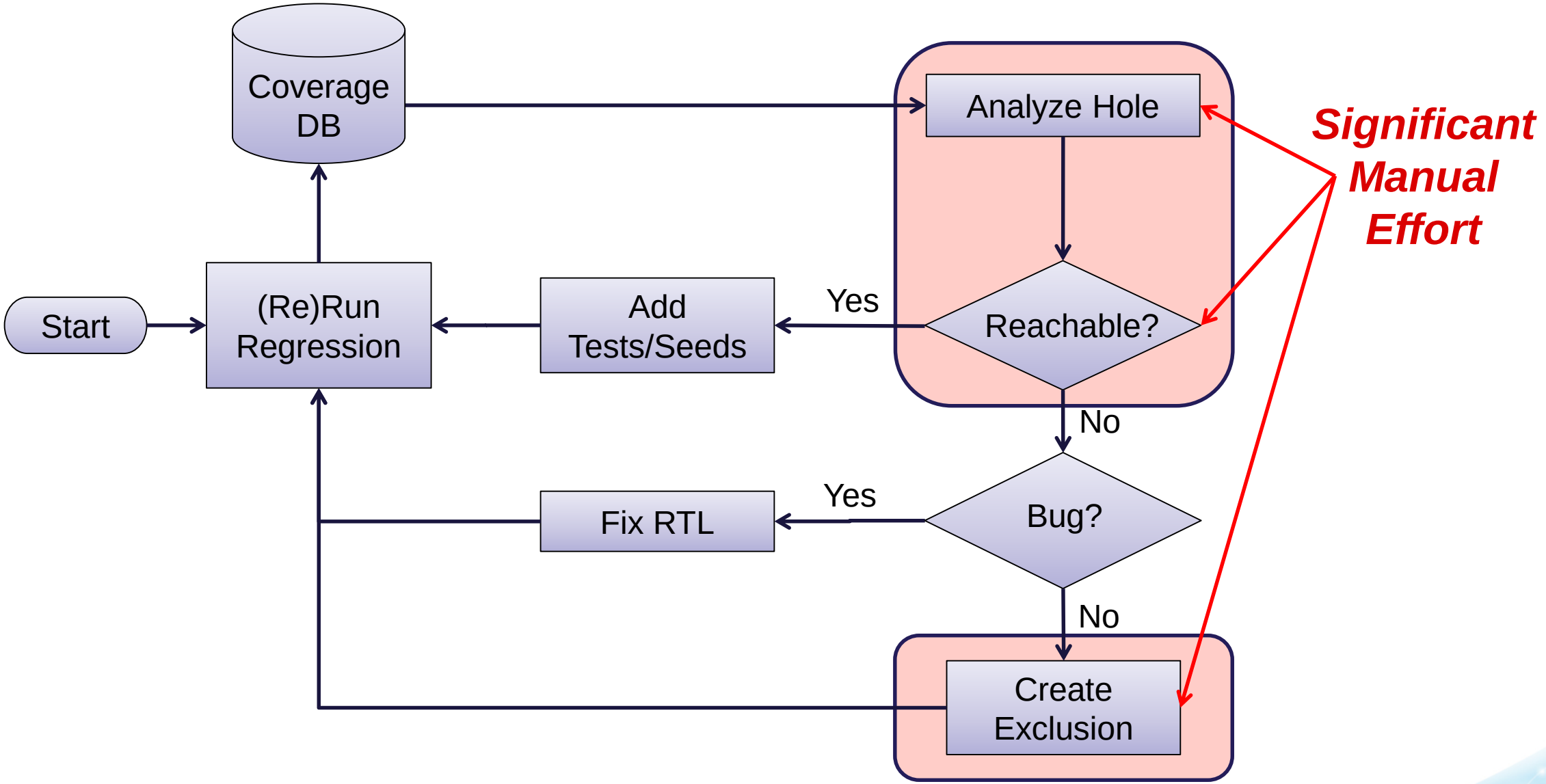
Specific XPROP Recommendations

- We recommend that XPROP be used in virtually all ASIC development process.
- Start running XPROP close to Design Capture Complete.
- If your flow involves prototyping and/or emulation, XPROP should be used prior to the first ‘drop’ to prototype/emulation.
- Run a full XPROP regression at regular intervals (at least bi-weekly).
- XPROP should not be used as a substitute for gate-level simulations.

Point Tool #2:

Formal Coverage Analyzer

What is FCA?


Coverage DB
Analyze Hole
Significant Manual Effort
Start
(Re)Run Regression
Add Tests/Seeds
Yes
Reachable?
No
Yes
Fix RTL
Bug?
No
Create Exclusion

The Cost of Using FCA

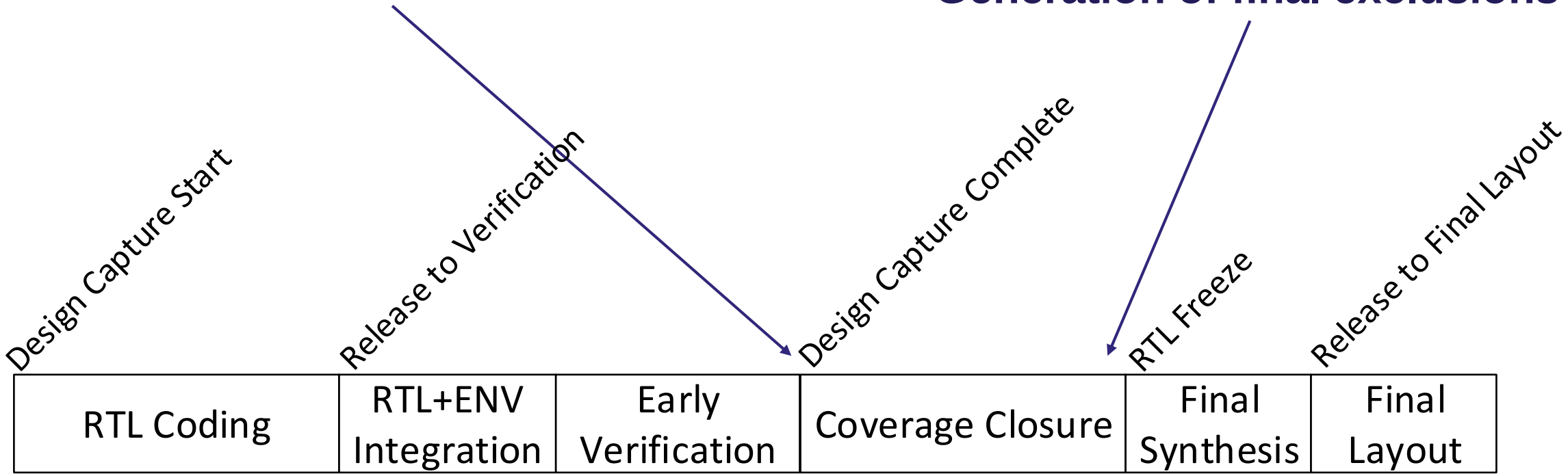
- The cost of using a tool:
  - The set-up and run-time effort of using FCA.
  - The effort required to review the input constraints used by FCA to generate the exclusions.
  - The effort required to review the actual exclusions generated by the tool.
- The cost of human effort:
  - The effort required to manually review the RTL and manually generate the exclusions without the aid of FCA.

- Use FCA when the cost (time and effort) of using the tool is deemed to be less than the cost of using humans.

When to Use FCA

First Application of FCA:
Pipe cleaning of set-up & constraints

Second Application of FCA:
Generation of final exclusions



Specific FCA Recommendations

- FCA is a fit for those teams that use code-coverage as a primary sign-off criterion.
- FCA could be applied as soon as possible after Design Capture Complete, and again as close as possible to RTL Freeze.
- Review your FCA input constraints as part of the code-coverage sign-off criteria.
- Double check: all code identified as dead-code by FCA should be reviewed to ensure that the Designer agrees the code is eligible for exclusion from code-coverage.

Point Tool #3:

Certitude

What is Certitude?

- Certitude works by inducing structural faults (or mutations) in the RTL code:
  - For example, a statement **a = b || c;** could be mutated to **a = b && c;**
- Certitude then runs your regression to determine whether the verification environment can detect the fault (mutation).
- Certitude arranges faults/mutations into nine “fault classes”:

Class Name	Faults In Design	Faults In Report	Non-Activated	Non-Propagated	Detected	Non-Detected	Disabled By Certitude	Disabled By User	Dropped	Not Yet Qualified
TopOutputsConnectivity	15	15	0	0	15	0	0	0	0	0
ResetConditionTrue	251	251	0	4	120	1	105	3	18	0
InternalConnectivity	7173	7017	14	346	4060	0	2180	2	415	0
SynchronousControlFlow	1597	1502	8	25	340	3	735	3	388	0
SynchronousDeadAssign	490	490	0	16	193	0	168	1	112	0
ComboLogicControlFlow	595	595	0	29	155	8	138	8	257	0
SynchronousLogic	1840	1831	7	37	184	1	935	3	664	0
ComboLogic	7363	7362	7	175	2975	5	2286	9	1905	0
OtherFaults	3139	0	0	0	0	0	0	0	0	0
All Fault Classes (9)	**22463**	**19063**	**36**	**632**	**8042**	**18**	**6547**	**29**	**3759**	**0**

An Example Certitude Fault: ResetConditionTrue

```
// Original RTL
always @(posedge clk) begin
  if (!reset_n) begin
    grant <= 0;
  end
  else begin
    grant <= rdy && msk;
  end
end
```

```
// Mutated RTL
always @(posedge clk) begin
  if (1'b1) begin
    grant <= 0;
  end
  else begin
    grant <= rdy && msk;
  end
end
```

The Cost of Using Certitude

- Setup time.
- Fault-Inserted Regressions.
- Results Analysis.

```
…
parameter THREESTAGE = 1;
…

always @(posedge clk) begin
  if (!reset_n) begin
   s1  <= 1'b0;
   s2  <= 1'b0;
   out <= 1'b0;
  end
  else begin
   s1 <= in;
   s2 <= s1;
   if(THREESTAGE) begin
     out <= s2;
   else
     out <= s1;
   end
  end
end
```

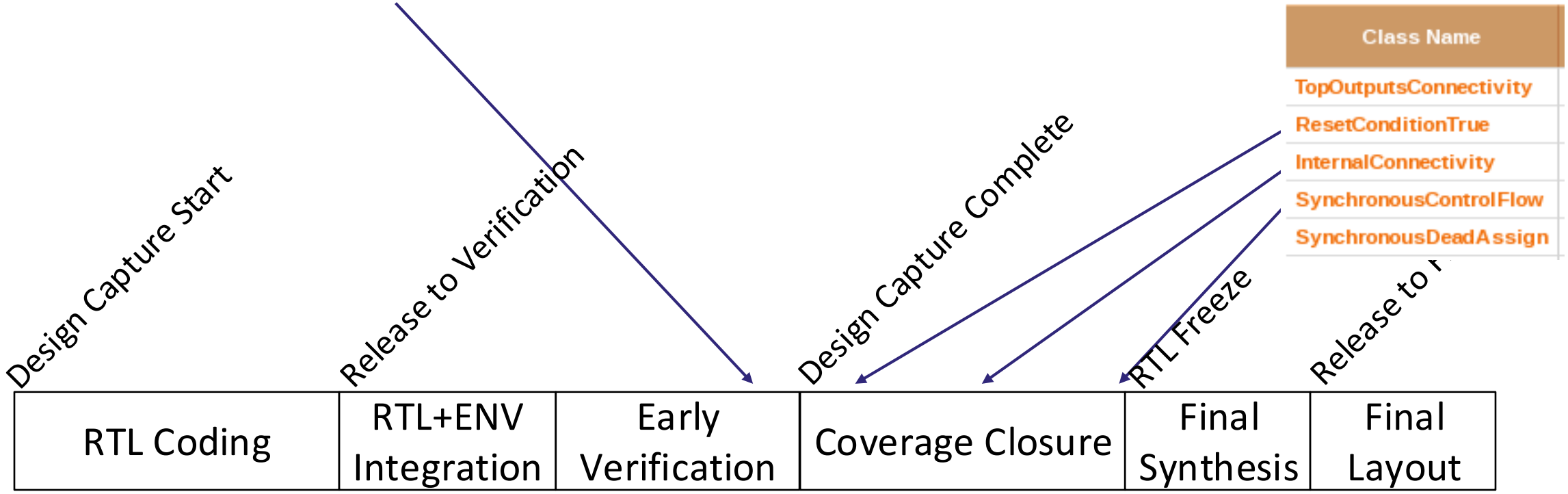
```
…
parameter THREESTAGE = 1;
…

always @(posedge clk) begin
  if (!reset_n) begin
   s1  <= 1'b0;
   s2  <= 1'b0;
   out <= 1'b0;
  end
  else begin
   s1 <= in;
   s2 <= s1;
   if(1'b0) begin
     out <= s2;
   else
     out <= s1;
   end
  end
end
```

Is this a detectable fault when THREESTAGE set to 0?

When to Use Certitude

First Application of Certitude:
TopOutputsConnectivity faults only.

Step-Wise Application:
Iterate on successive fault classes



Specific Certitude Recommendations

- Consider deploying Certitude for Designs with very high sensitivity to functional defects in silicon.

- A pre-deployment review will save you a lot of time in the end:
  - Consider the types of faults that are important to your project.
  - Exclude faults on areas that are known to have low code coverage (e.g. dead-code as identified by FCA or human review).

- Use a step-wise approach to perform multiple iterations of Certitude regressions.

- Use code coverage to determine whether or not your Environment is thorough and complete enough to detect the expected faults at each classification level.

Conclusions

Its all about Cost and Timing

Point Tool	Cost	Timing
XPROP	Low	• Start using soon after Design Capture Complete. • Apply before each drop to Emulation or Prototyping. • XPROP regressions at regular intervals until RTL Freeze.
FCA	Moderate	• Use as soon as possible after Design Capture Complete. • As close as possible to RTL Freeze.
Certitude	High	• Start using soon after Design Capture Complete. • Employ a stepped approach of Fault Classes. • Use Code Coverage data to guide decision to start next step.

Understanding your Project's tolerance to functional bugs in silicon will inform your decision to use these point-tools.

Making intelligent choices about when to deploy these point-tools will increase their value to your Project.

Thank You