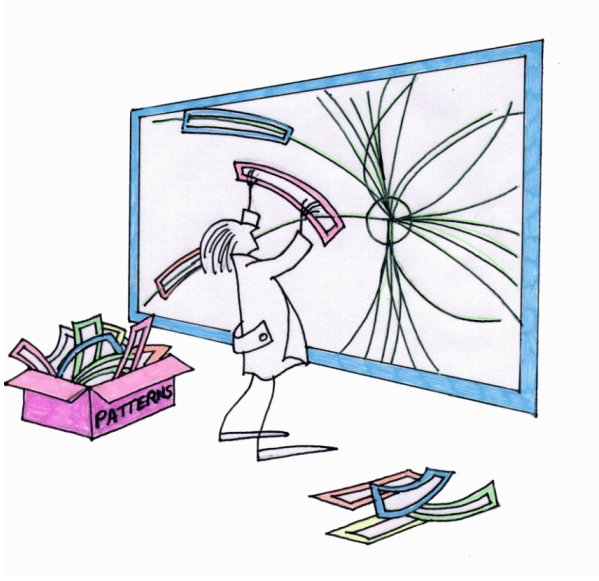
MARIE CURIE IAPP: FAST TRACKER FOR HADRON COLLIDER EXPERIMENTS

1ST SUMMER SCHOOL: VHDL BOOTCAMP

PISA, JULY 2013

Introduction to Simulation

Calliope-Louisa Sotiropoulou
PhD Candidate/Researcher
Aristotle University of Thessaloniki



AUTH e-LAB
Aristotle University of Thessaloniki-Electronics Laboratory

UNIVERSITÀ DI PISA

Introduction to Simulation

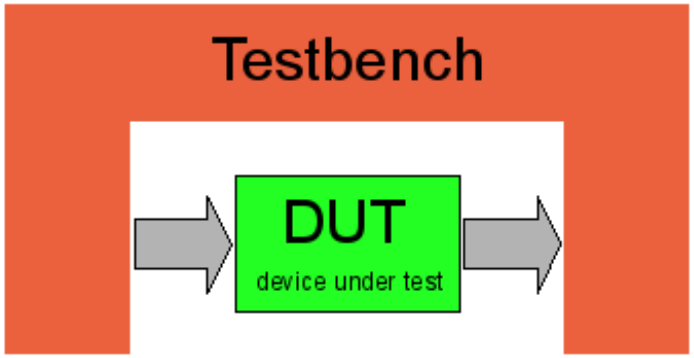
- Introduction
- Methodology
- ISim

Introduction

- **Verification** is a *process* used to demonstrate the functional correctness of a design

- A **testbench** is code that is used to create a pre-determined input sequence to a design
  → Usually to observe the response

- It is a closed system for the Design Under Test (DUT)

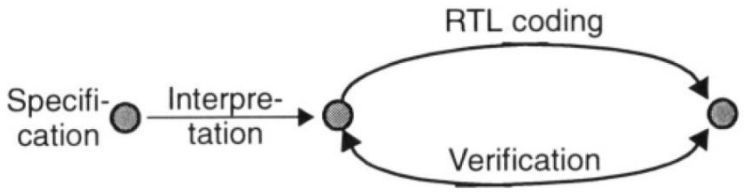


Verification

- The more complicated the design, the more complicated and time consuming its verification
- Verification can reach an estimated total of 70% of a SoC's design effort and 80% of the total code volume
- Expect the unexpected!!!

What are we verifying?

- Ensure that the results of each design is as intended or as expected
- Verify the functionality of each block, from bigger granularity to smaller
- Beware of the Human Factor
  - Introduce automated processes
  - Or simple foolproof steps

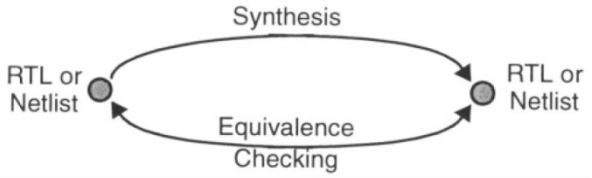


Introduction to Simulation

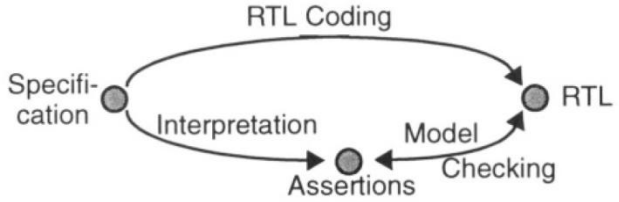
- Introduction
- Methodology
- ISim

Formal Verification

- *Equivalence Checking*
  - *Compare two netlists → Are we 100% that netlist completely implements the original RTL code?*
  - *Detect bugs in the synthesis software*



- *Model Checking*
  - *Assertions or characteristics of a design are formally proven or disproved*



Functional Verification

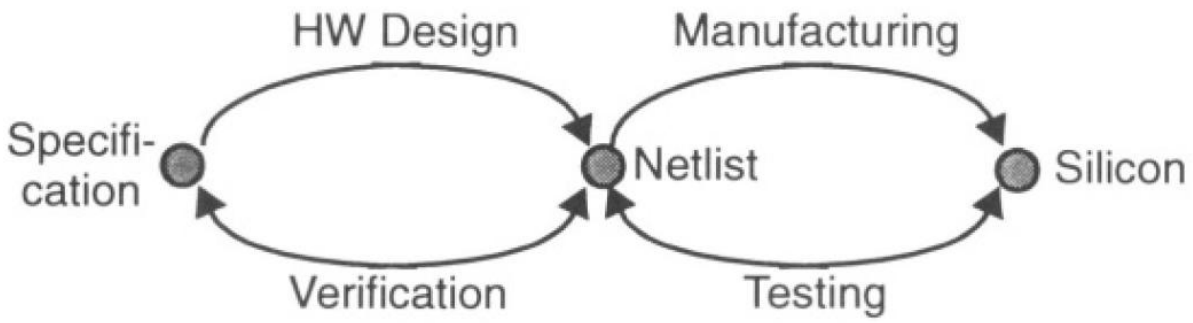
- Ensure that a design implements intended functionality

- **You can prove the presence of bugs, but you cannot prove their absence** (you can only show it)

Testbench Generation

Testing vs Verification

- Testing verifies that the design was manufactured correctly



- Testing is accomplished through test vectors. It verifies that internal nodes can be toggled

Verification Tools

Simulators → Approximating reality

- The physical characteristics of the design are simplified. Some are even ignored.

- Execute a description of the design. The outputs are validated externally, against design intents.

- Methods
  - Event-driven simulation
  - Cycle-based simulation

Verification Tools

Waveform viewers

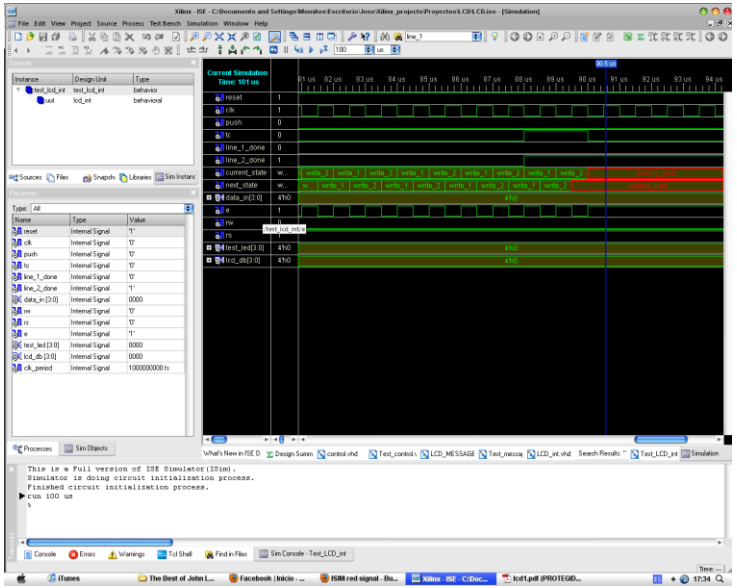
- Used in conjunction with simulators. Means of visualization of multiple signals transition over time.

- Sufficient only for a limited number of signals. Complexity multiplies with the increase of the number of signals to be observed

Verification Tools

- Independent, verification specific tools:
  ModelSim, NCVerilog, MPSim

- Integrated in the general flow:
  Aldec Active-HDL, Xilinx ISE Isim



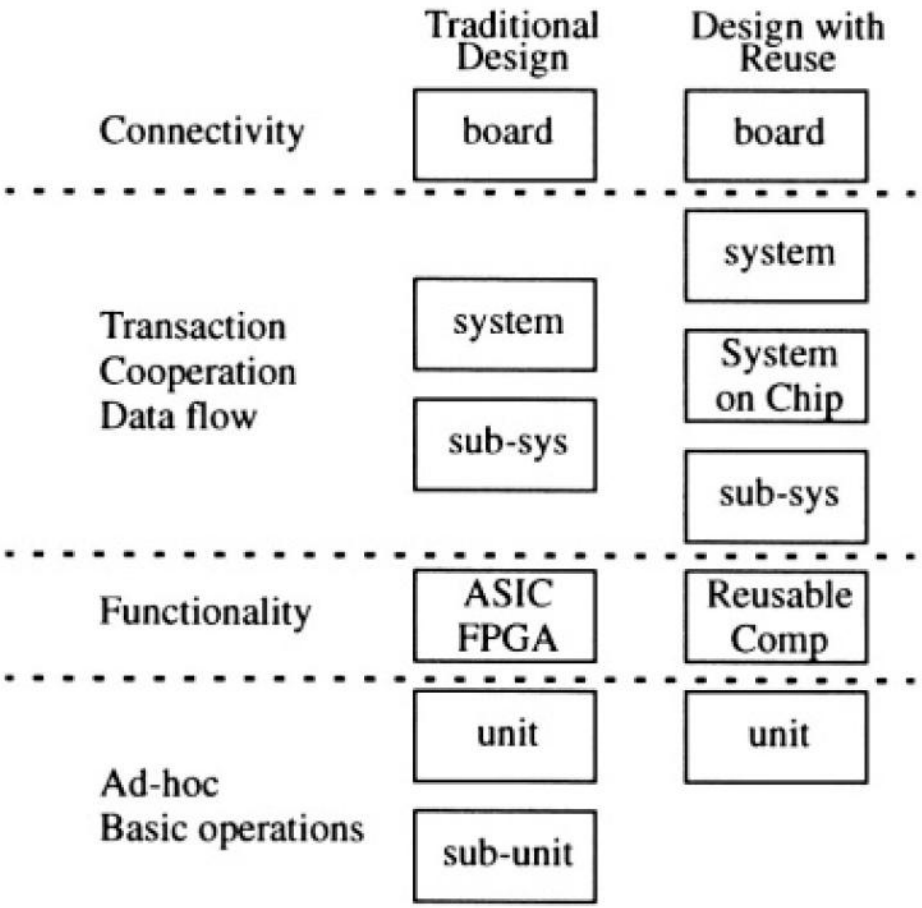
Tricky Points!

- Code coverage
  Must ensure that all possible cases in a code have been tested → Especially in FSMs (Finite State Machines)

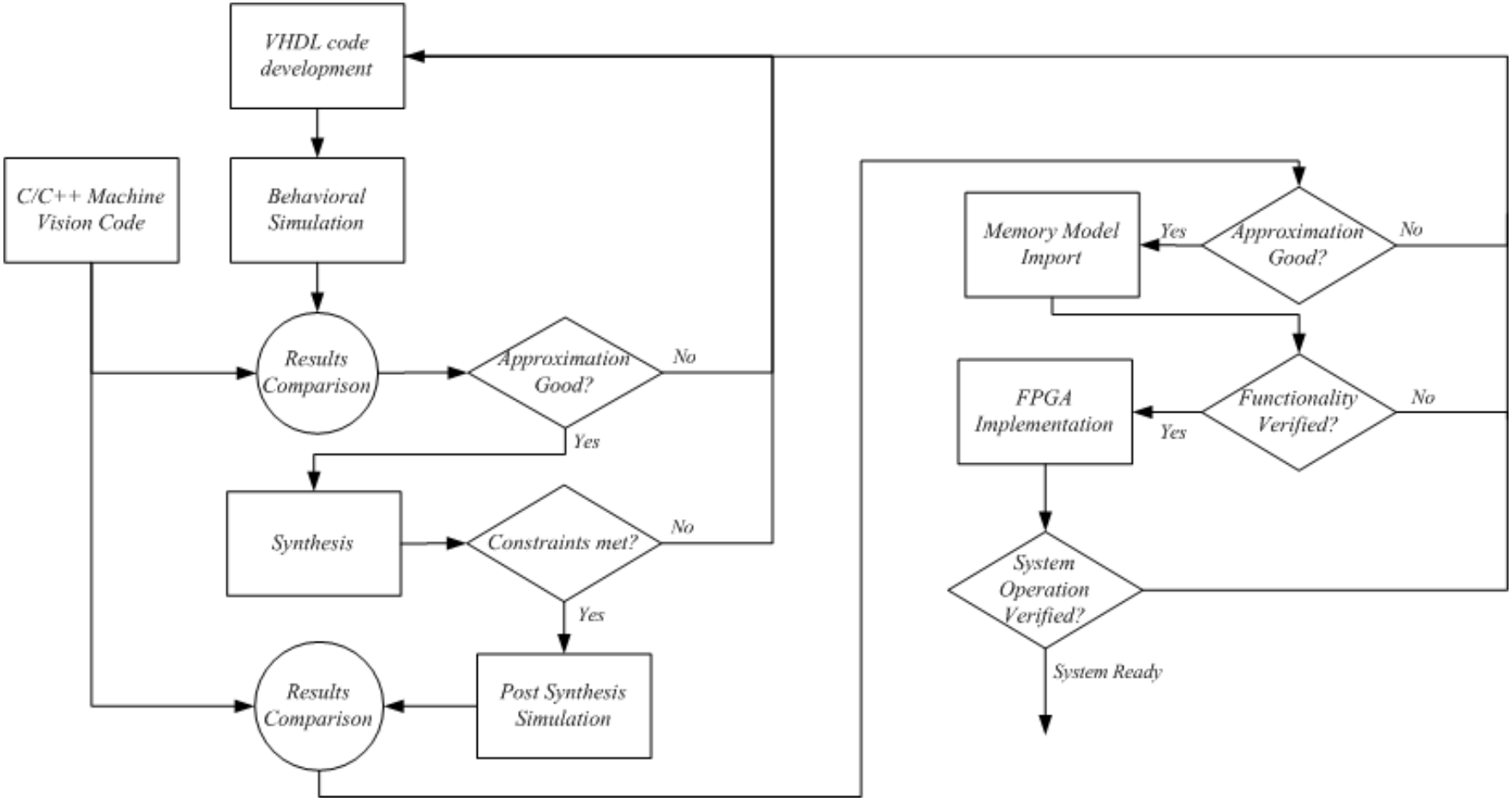
- Revision control
  Make sure everyone is using and testing the same code version (Syntax, functionality etc) → Use of a repository

- Behavioral code vs RTL code

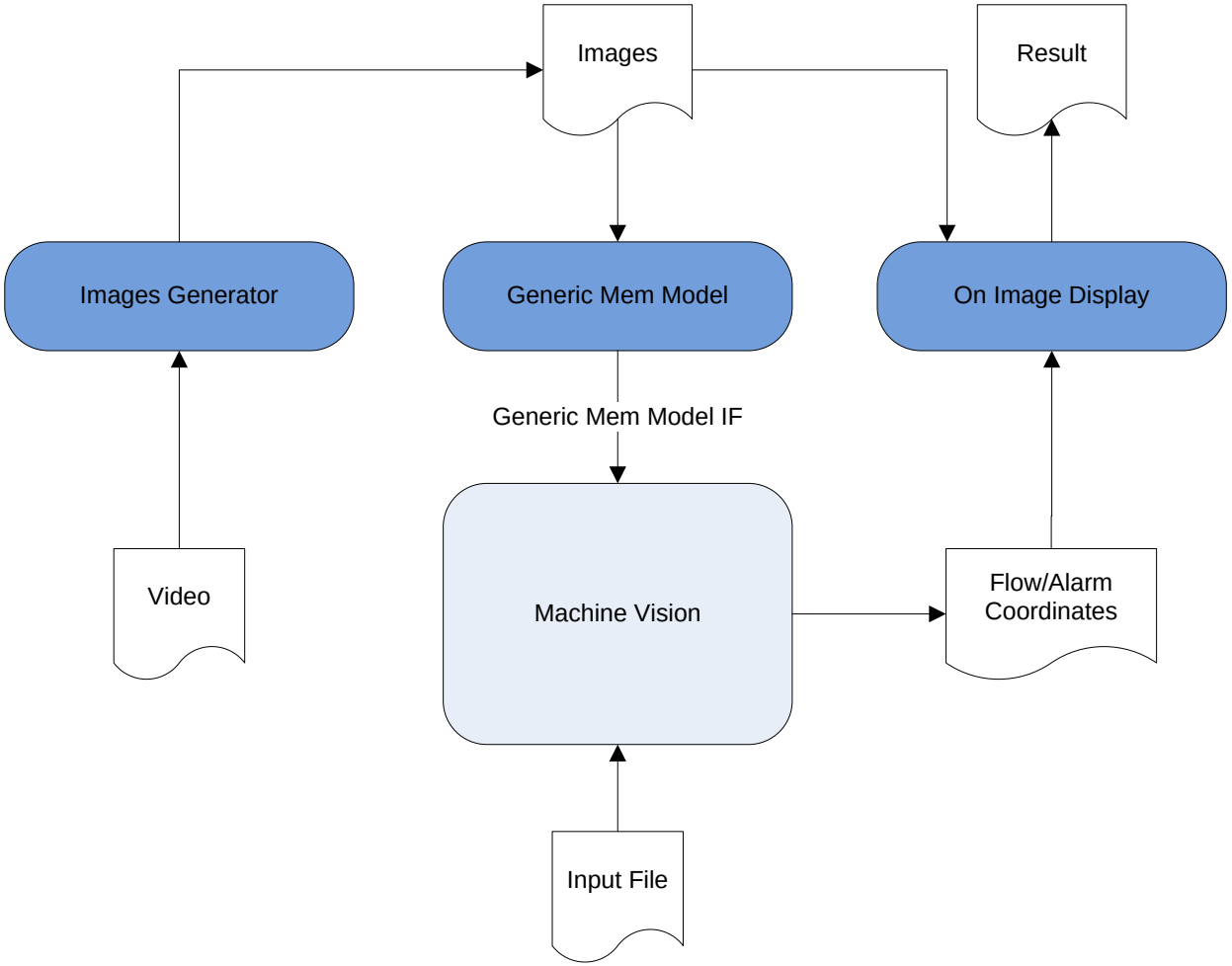
Verification Levels



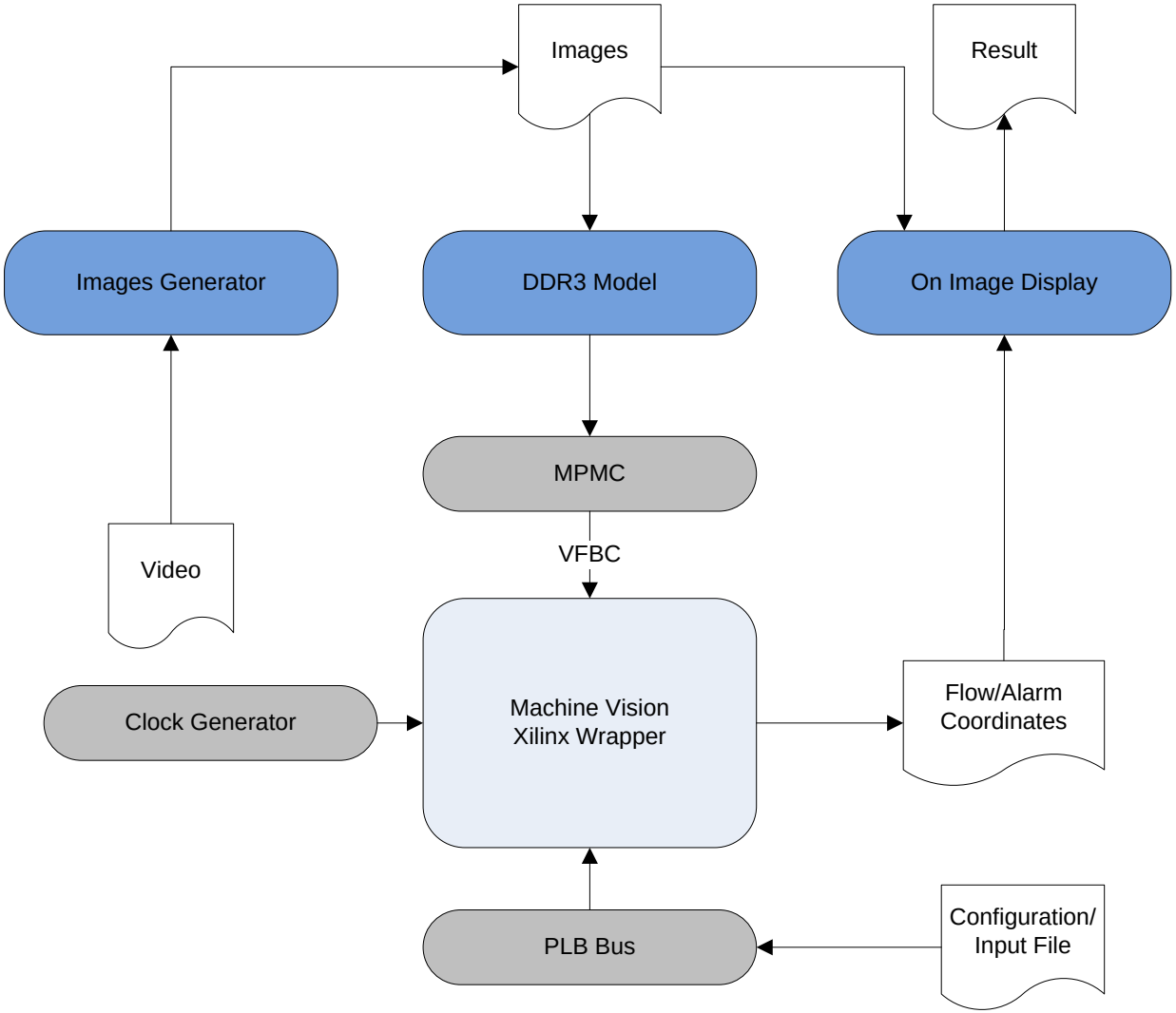
Verification Flow



Verification Flow – Generic memory model



Verification Flow – Bus Functional Model



Introduction to Simulation

- Introduction
- Methodology
- ISim

XILINX ISim

- Xilinx ISim is a Hardware Description Language (HDL) simulator that lets you perform behavioral and timing simulations for VHDL, Verilog, and mixed VHDL/Verilog language designs

- ISim is a separate program integrated to the XILINX Toolchain

XILINX ISim

- Isim can be invoked at different staged of the design development to perform:
  - Behavioral Simulation
  - Post Synthesis Simulation
  - Post Map Simulation
  - Post Place and Route Simulation