2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SAN JOSE, CA, USA
MARCH 4-7, 2024

Functional Verification from Chaos to Order:

Using Continuous Integration for Hardware Functional Verification

Kirolos Mikhael, Abdelouahab Ayari

Siemens EDA

SIEMENS

accellera SYSTEMS INITIATIVE

Agenda

- Introduction
- Continuous Integration Framework (Concepts)
- CI Infrastructure
- Choose CI platform
- Requirements for EDA vendors
- Experiment
- Results & Conclusion

2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SAN JOSE, CA, USA
MARCH 4-7, 2024

Introduction

Basic concepts for Continuous integration and delivery in hardware flow

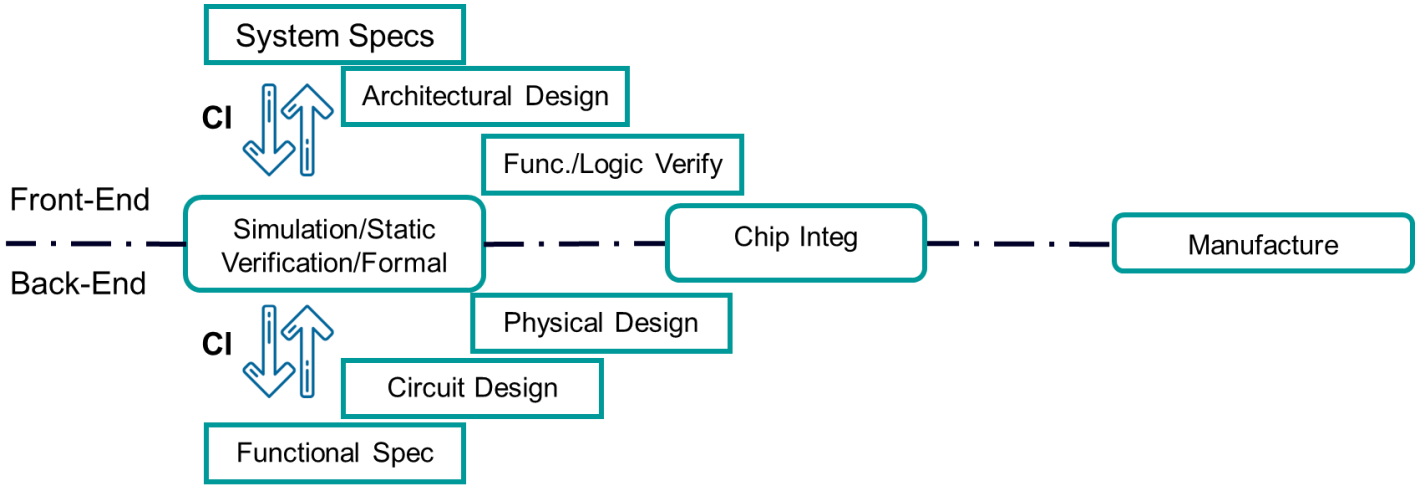
accellera SYSTEMS INITIATIVE

Introduction

- History
  - Used extensively in software development in late 90's
  - Starts to have more usage in hardware design and verification
  - Used for projects with various contributors (design, verification, and system/concept engineers)
- Goals
  - Traditionally, several check-ins over several days are verified very late (usually in weekend regressions)
  - Failures/inconsistencies introduced in new check-ins become complex and time-costly to fix
  - CI enables the frequent and reliable release of new features
  - Code changes are integrated in a way that is
    - Automated
    - Efficient
    - Correct
    - Transparent

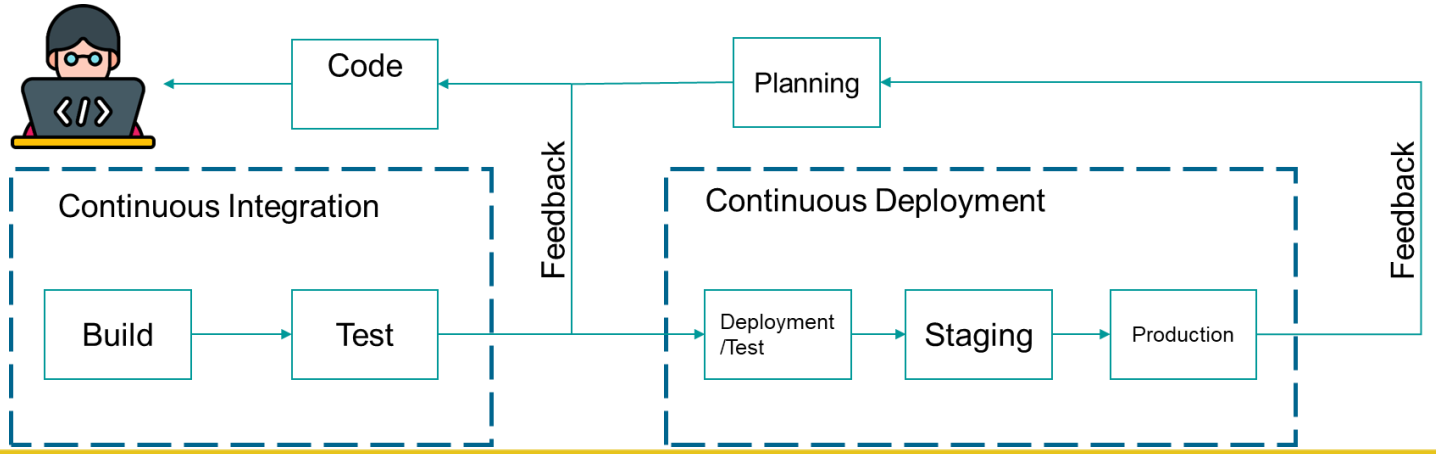
Introduction

- Benefits of using CI flow in hardware
  - Reduce build and test time
  - Increase the visibility and awareness of the build results
  - Support automated testing flow
  - Support working in parallel for different teams



Introduction

- What are the differences between Continous Integration and Continuous Delivery
  - Continous Integration:
    This contains the stages of building and testing the code automatically.
  - Continous Delivery
    This describes the process of automatic production. This may be a little bit risky but it can be deployed after gaining enough confidence in the pipeline.



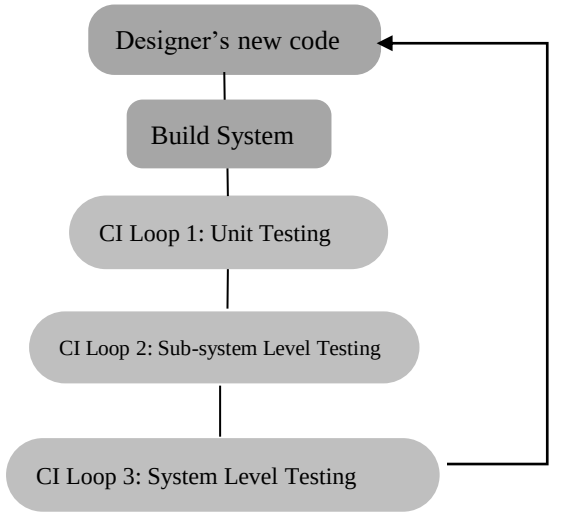
2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

Continuous Integration Framework

accellera
SYSTEMS INITIATIVE

Continuous Integration Framework

- Continuous Integration (CI) Framework Overview:
  - Designed for automated code testing.
  - Implemented at both subsystem and chip levels in modern SoCs.
  - Enables integration of code into larger subsystems and systems.
- CI Framework Structure:
  - Involves multiple loops, each representing a larger subsystem.
  - Code passes through these loops until it achieves stability for integration into the main chip/SoC streamline.



Continuous Integration Framework

- Unit Testing:
  - Validates the smallest parts of the RTL.
  - Benefits include early bug detection, improved code quality, support for refactoring, accelerated development, and enhanced collaboration.
- Sub-system Level Testing:
  - Involves integrating units into larger sub-systems.
  - Example: Processor design with units like Address Unit (AU), Execution Unit (EU), Bus Unit (BU), and Instruction Unit (IU).
  - Unit testing on IU decoder, followed by integration into IU for sub-system testing.
  - Subsequent integration with all other units for system-level testing.
- Extended Testing Loop:
  - If the processor is part of another system, extend the testing loop
  - Automation enhances testing efficiency, providing faster results early in the design stage.
  - Mitigates challenges associated with integration difficulties, commonly known as "Integration Hell."

Continuous Integration Framework

- Looking on the algorithmic part of the framework:
  - Build: is the basic step to make sure that the code is ok to go. (Compile & Lint)
  - Unit Testing
  - Sub-system
  - System
- To have a full integration all the steps should pass successfully

ALGORITHM 1: CONTINUOUS INTEGRATION FOR DESIGNER'S CODE

Input: *DesignerCode (The new code provided by the designer)*
Output: *Status of the CI system*
function ***BUILD****(DesignerCode)*
  return DesignerCode is ***valid*** *? Compile and Test () :* ***Error***
end function

function ***UNIT_TEST****(DesignerCode)*
  for each ***unit*** *in it the DesignerCode: if not* ***test****(unit) then return* ***Failure*** *else return* ***Success***
end function

function ***SUBSYSTEM_TEST****(DesignerCode)*
  for each submodule in System that contains the DesignerCode: if not ***test(sub)*** *then return* ***Failure*** *else* ***Success.***
end function

function ***SYSTEM_TEST****(DesignerCode)*
  For the DesignerCode integrated in the full system; if not test(system) then return ***Failure*** *else* ***Success***
end function

if ***BUILD****(DesignerCode) is* ***Error****: Output "****Build failed****";* ***Stop***
if ***UNIT_TEST****(DesignerCode) is* ***Failure****: Output "Unit test failed";* ***Stop***
if ***SUBSYSTEM_TEST****(DesignerCode in SubSystem) is* ***Failure****: Output "Sub-system test failed";* ***Stop***
if ***SYSTEM_TEST****(DesignerCode in FullSystem) is* ***Failure****: Output "System test failed";* ***Stop***
Output ***"All CI steps passed!"***

2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

CI System Infrastructure

accellera SYSTEMS INITIATIVE

CI System Infrastructure

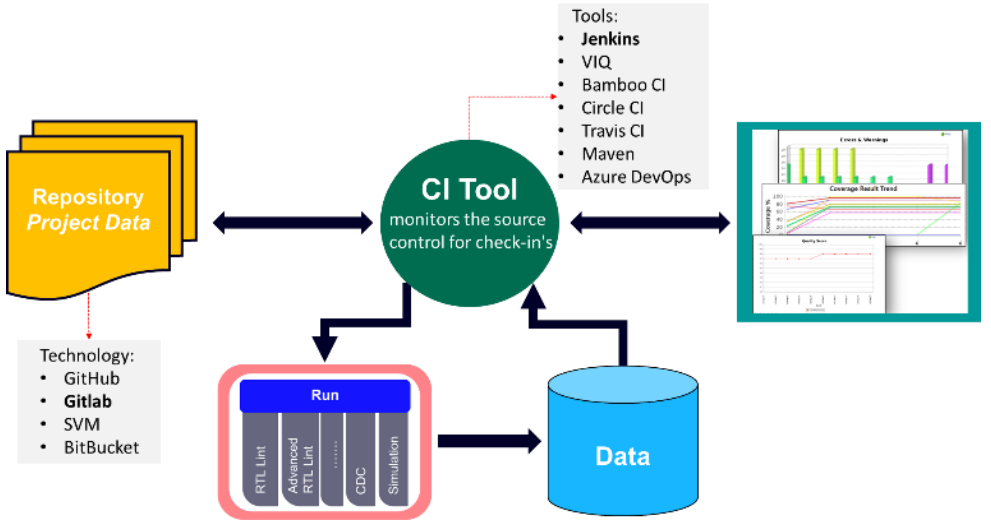
Functional Units in CI System:

Repository Project Data:

- Overseen by version control system.
- Essential for code changes and tracking.

CI Tool:

- The functional brain of the CI system.
- Interfaces with the run process and repository.



Run Process:

- Involves Digital Verification process (static, formal, and simulation verification procedures).
- Delivers pass/fail metrics as results.

Database:

- Generated from the analysis tools and contains the analysis results.

Data Visualization:

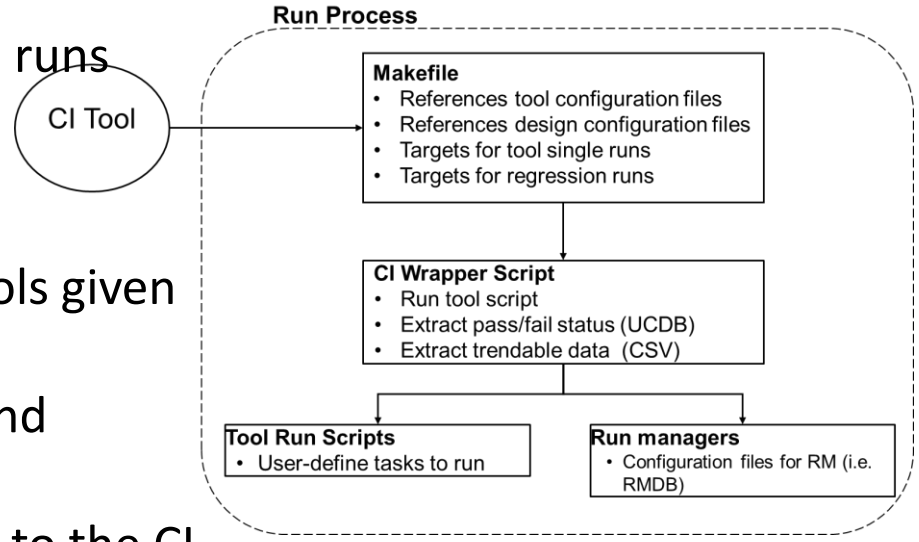
- Method for visualizing CI system data.
- Enhances understanding and transparency.

CI System infrastructure

- Communication Pathways in CI System (Black Arrows):
  - Data flow in CI system (Repo->CI Tool-> Run Process -> Data -> CI Tool -> Visuals).
- Source Code Management (SCM) and CI Tool Connection:
  - Utilizes APIs provided by the SCM.
  - "git" SCM allows straightforward webhook implementation.
  - Webhooks detect code modifications and trigger associated tests on CI tool.
- CI Tool and Visualization Connection:
  - Execution phase yields varied outputs from processes.
  - Data produced in formats like CSV, UCDB, XML, and JSON.
  - APIs of functional verification tools utilized to generate these types of reports.
  - Visualization through plugins or specialized tools (i.e. Questa VIQ)

CI System Infrastructure

- Run Process communication line
  - CI tool activates Makefile in the run process.
  - Makefile is where we configure the tools run options.
  - Makefile targets contains the configurations for the regression runs
- CI Wrapper Script
  - Through wrapper script we added a level of abstraction
  - Wrapper script will invoke Questa Design Solutions, Formal tools given the TCL configuration.
  - The CI wrapper script will handle Pass/Fail status generation and trendable data generation.
  - The CI wrapper will use APIs to generate data and send it back to the CI tool.
  - This level of configurability enables scalability and configurability.



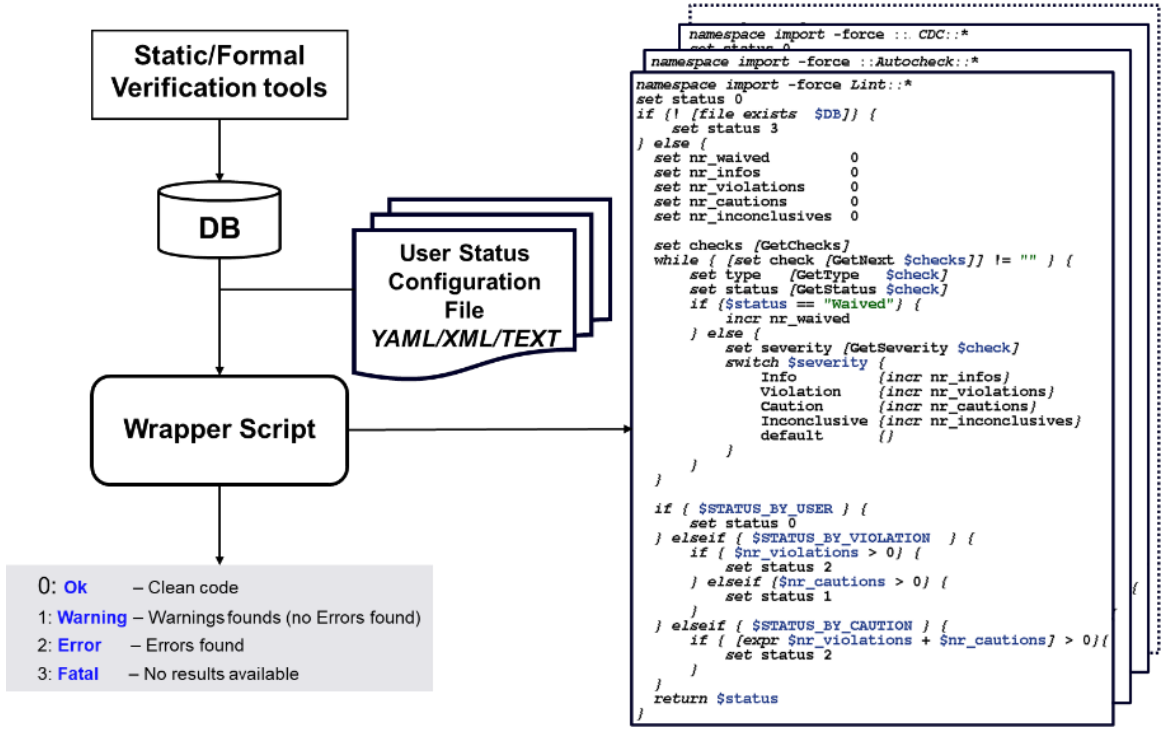
2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

Requirements for EDA Vendors

accellera
SYSTEMS INITIATIVE

Requirements for EDA Vendors

- A. Pass/Fail Status API Mechanism
  - In the CI integration system, the test status is crucial for exchanging information between the CI modules.
  - In the CI integration system, the test status is crucial for exchanging information between the CI modules
  - The status generation is configurable



Requirements for EDA Vendors

- B. Common Output Data Format to be Processed by CI Tools
  - It is essential to have a standardized way of outputting data in a CI system because it integrates multiple tools
  - The CSV format is a suitable way to exchange data, as it is simple and can be integrated with various tools.
  - UCDB is more integrated with simulation results
  - So, EDA vendors should add support for different metrics report format (CSV, Json, and UCDBs)

Requirements for EDA Vendors

- C. Fine Granularity of Tool Configurations
  - The EDA vendor should introduce the full support for different tool modes
  - Full functional mode, light functional mode, and user mode
  - These modes helps in adding a lot of configurability to the pipeline and broaden the use model

2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

Choose CI Platform

accellera
SYSTEMS INITIATIVE

Choose CI Platform

Criteria	Jenkins	Travis CI	CircleCI	GitLab CI/CD
Ease of Setup	Flexible, but manual	Quick and straightforward	Quick and straightforward	Integrated with GitLab
Configuration as Code	Yes (Jenkinsfile)	Yes (.travis.yml)	Yes (.circleci/config)	Yes (gitlab-ci.yml)
Community Support	Large and active	Good	Active	Good
Extensibility	Huge variety of plugins	Limited, but extensible	Rich ecosystem	Built-in features
Container Support	Docker and Kubernetes	Docker	Docker	Docker
Parallel Builds	Supported	Limited	Supported	Supported
Ease of Maintenance	Requires upkeep	Low maintenance	Low maintenance	Integrated with GitLab
Scalability	Scalable	Limited scalability	Scalable	Scalable
Pricing (Cloud-based)	N/A	Freemium	Freemium	Included with GitLab
Integration with Other Tools	Extensive	Limited	Good	Integrated with GitLab
Security Features	Requires configuration	Good	Good	Integrated with GitLab

2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

Experiment

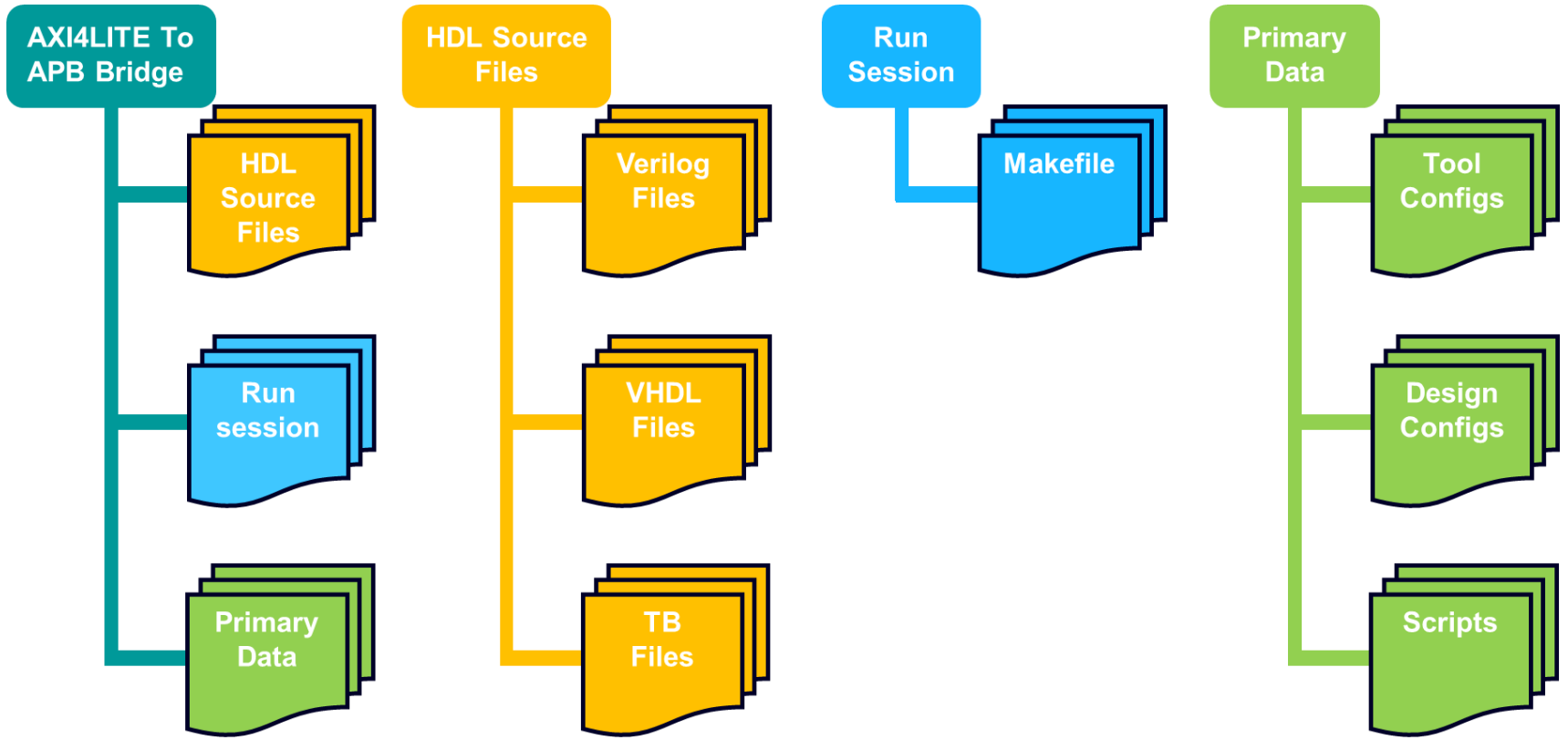
accellera
SYSTEMS INITIATIVE

Experiment

- Design
  - Has 2 primary modes of operation
    - Single-clock mode
    - The AXI4Lite and the APB4 will use the same clock
  - The multi-clock mode
    - The AXI4Lite and the APB4 run with different frequency (use two asynchronous clocks)
    - An APB interface is used to configure the design
- Verification
    - Simulation environment (Questa Sim)
    - Design Solutions (Static analysis): Lint, AutoCheck, Xcheck, CDC, RDC
    - Formal Solutions: PropCheck, CoverCheck

Experiment

- Experiment files



Experiment

- Experiment implemented features
  - Source Control Integration with GitLab
  - Configurable runs based on the design
  - Smart analysis based on the RTL changes
  - Archiving results and analysis based on design history
  - Visualization of the results
  - Send notification for the results
  - Integration with Verification Run Manager

2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

Results & Conclusion

accellera
SYSTEMS INITIATIVE

Results

- Running all tools sequentially for the AXI4Lite to APB4 bridge example leads to 95s execution time. However, using parallel execution in the pipeline reduces this duration to 48s which means ~2x of reduction in the analysis time
- The experiment shows significant time savings using conditional runs.
- Smart tricks save up to 50% of analysis time.
- Tests are re-usable
- Small-sized databases provide efficient archiving, better insights, and support integration with more sophisticated database techniques.

Conclusion

- Achieve close integration and provide essential metrics for verification tracking.
- Applicable across various stages in the digital design flow, from initial to sign-off stages.
- Early bug detection with an optimized flow.
- Time savings, enhanced analysis precision, and adaptability with diverse metric generation and visualizations
- Results showed great value in accelerating analysis time and increasing productivity on an industrial scale.

Questions

- Finalize slide set with questions slide