The Next Generation of IP & SoC Development Platform

Bertrand Blanc (bblanc@altera.com) & Fergus Slorach (fergus@methodics.com)

Towards a Dynamic IP& SoC Integration Platform

Ad-hoc Release System

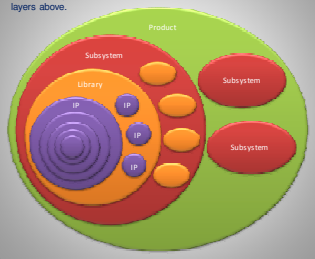
- x Ad-hoc heterogeneous methodologies
- x Poor performance & scalability
- x Poor data traceability
- x Workspace population coherency issues
- x Release paralysis
- x A release is a tag of the entire design hierarchy
- x Tags not immutable, trying to hit a moving target
- x Frequent resource conflicts
- x Weak links to Issue Tracking
- x No IP re-use methodology
- x Bound to underlying legacy DM system
- x Poor ad-hoc data validation
- x File system access control
- x Manual notification

MDX: correct-by-construction

- ✓ Formal & Engineered homogeneous methodology
- ✓ High performance, highly scalable
- ✓ Persistent components database with full history
- ✓ Workspaces built from a known good dataset
- ✓ Incremental release & propagation capability
- ✓ Aggregated releases thru the component hierarchy
- ✓ Immutable releases tags
- ✓ Prevent resource conflicts by design
- ✓ Tightly-coupled with Issue Tracking
- ✓ Database of IP facilitates re-use
- ✓ DM agnostic, supports multiple DM systems
- ✓ Built-in validation of data on release
- ✓ Database-driven access controls
- ✓ Integrated Notification System

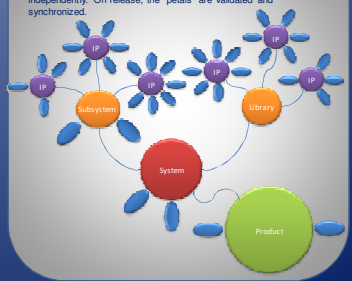
The "Onion" Approach

A released layer wraps the layers beneath. Any modification in a layer will require an entire re-release, and verification, of all the layers above.



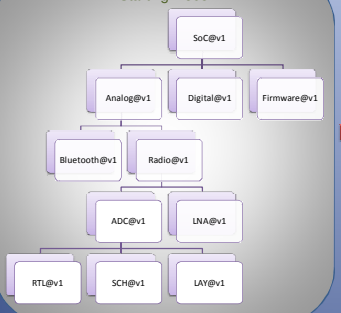
The "Flower" Model

A Component is broken up into Deliverables & Component Resources. These "petals" can be modified and released independently. On release, the "petals" are validated and synchronized.



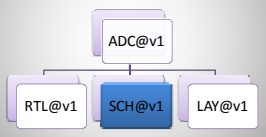
Continuous Dynamic Integration in a nutshell

Starting Node



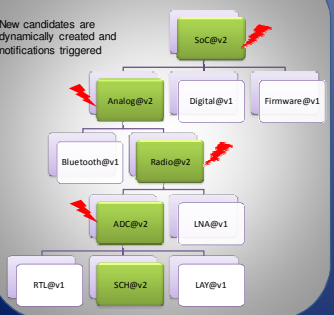
Create a workspace & Edit the data

- Load ADC@v1 into a workspace
- Modify SCH data in a schematic editor



Propagate throughout the hierarchy

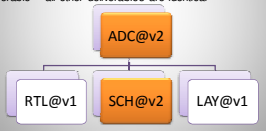
New candidates are dynamically created and notifications triggered



Store the changes, Check & Release

1. SCH@v2 is a new, unverified stored Deliverable
2. SCH@v2 is checked: type check, data check
3. SCH@v2 is checked in the context of ADC@v1
4. Verified SCH@v2 is released towards ADC@v2

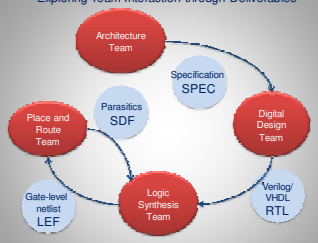
The only changes from ADC@v1 to ADC@v2 are in the SCH deliverable – all other deliverables are identical



Deliverable-based Release Methodology Concepts

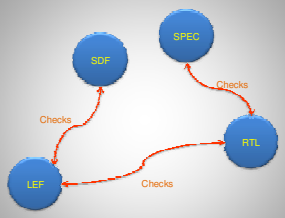
Defining The Corporate Ecosystem

Exploring Team Interaction through Deliverables



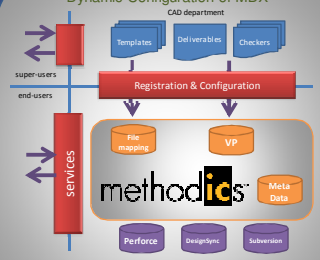
- Identify the teams
- Identify the channels of communication
- Identify the deliverables transiting in the channels

Deliverable Dependencies



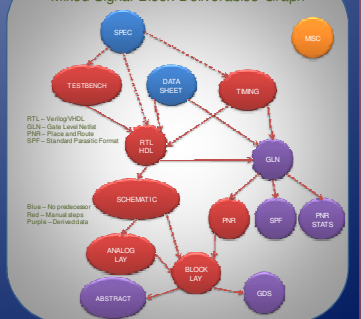
- Precedence relationships between deliverables define required release checks
- Only need to check consistency with direct "neighboring" deliverables

Dynamic Configuration of MDX



- A Template defines the collection of files for each deliverable
- A Type Check and Data Check is created for each deliverable
- A Context Check is created for each relationship between deliverables

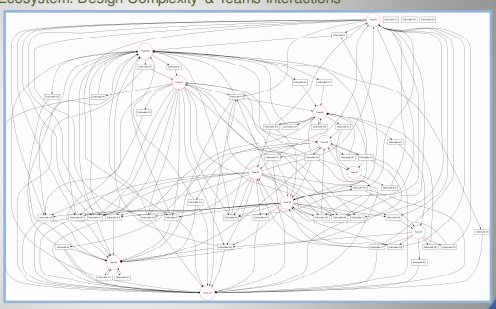
Mixed Signal Block Deliverables Graph



Hands-on Illustration: Altera Ecosystem in Practice

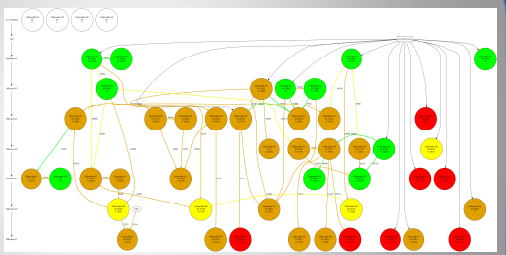
A Corporate Ecosystem: Design Complexity & Teams Interactions

- 11 teams
- 45 unique deliverables
- 201 channels of communication

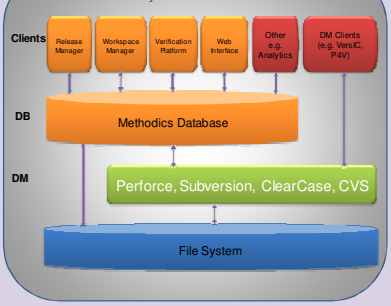


Deliverables Dependencies

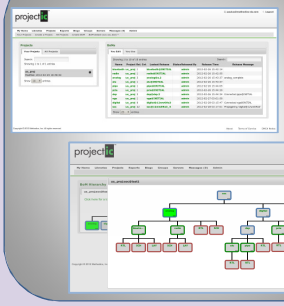
- 42 manifests & root data checks
- 47 context checks



System Overview



Web Interface



Command Line Interface

- Create resources
- Connect resources to build a design hierarchy
- Load workspaces from a known configuration
- Edit only the deliverable(s) you need
- Check your changes
- Release your changes to create a new verified configuration
- Report status from the database
- Set properties on database objects

Extensions

- Roadmap tool for project planning and to provide visibility into project status
- Integrate with regression testing
- Add quality metrics to IP
- Data-mining to get insight into IP Usage, design churn, resource bottlenecks, etc
- APIs allow for customer-developed extensions
- Generic Digital Asset Release Management – not just IC Design