2022
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
EUROPE
MUNICH, GERMANY
DECEMBER 6 - 7, 2022

Agile Approaches to ASIC Verification (A3V) – A Novel Agile Flow in Functional Verification

Adithya Rangan CK, Vidyasagar Kantamneni, Vishal Dalal

Infineon Technologies, Bangalore, India

infineon

accellera
SYSTEMS INITIATIVE

Contents

- Introduction
- Problem Statement
- Agile Methodology
- Agile approaches to ASIC verification (A3V) flow
- Use-case example – Memory IP (MIP)
- Results
- Opportunities versus challenges
- Conclusion

Introduction

Rapid evolution of ASIC development – design complexity & re-use

Design verification (DV) strategies

- Must provide best quality, optimum cost/time/resources

Conventional DV approaches

- Waterfall Model, Requirements-driven-flow (RDF) etc.
- Less flexible for changing verification requirements
  - Priorities of task execution are already baselined
- May delay sign-off and increase overall cost-factor

Problem Statement

Challenges posed by conventional methods

- Needs stable environment and limited alterations to DV
- Require experienced personnel
- Limited user involvement at various stages
- Increased cost, time and resources effort for even small changes
- Progress at end of cycle may be invalidated and discarded

Strong need to define verification work-flow to address above challenges

This will inherently drive verification to be agile

Agile Methodology

- Characteristics of being agile
  - Task division into small phases
  - Prioritization of task execution
  - Continuous collaboration
  - Periodic assessments & reviews
  - Iterative & incremental value addition
  - Visible metrics

- Agile Frameworks: Scrum, Extreme Programming (XP), Kanban etc.

Agile Approaches To ASIC Verification – A3V Flow

A3V

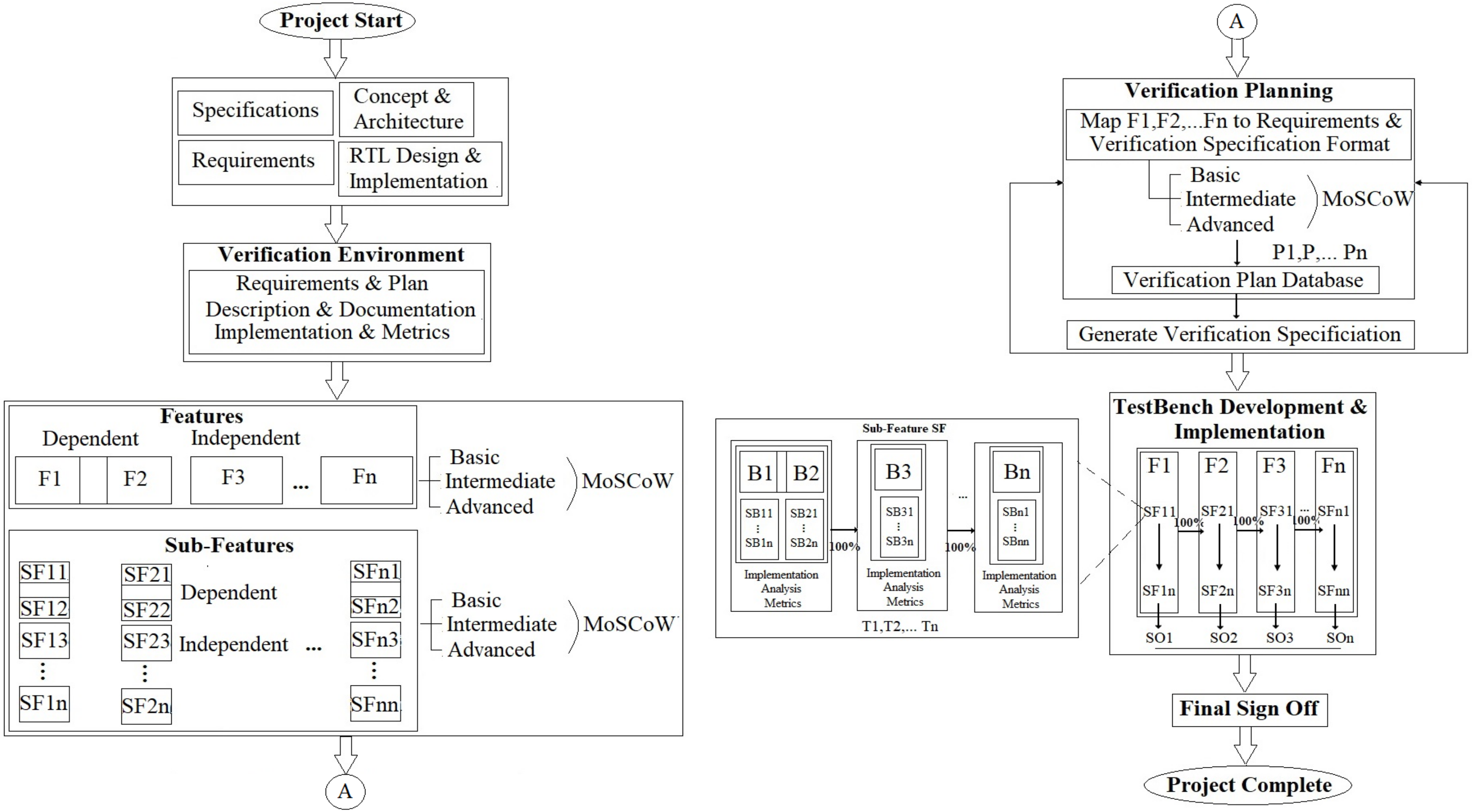
Combination of various agile frameworks

- Identify characteristics of DV
- Select agile methods based on individual advantages
- Actuate agile techniques as per scope, requirements & resources

Constituent frameworks of A3V

- Scrum, Kanban, Extreme-Programming (XP)
- Feature-Driven-Development (FDD)
- Adaptive-Software-Development (ASD)
- Behavioural-Driven-Development (BDD)

A3V Flow



Project Start

Specifications

Concept & Architecture

Requirements

RTL Design & Implementation

Verification Environment

Requirements & Plan

Description & Documentation

Implementation & Metrics

Features

Dependent

Independent

F1 F2 F3 ... Fn

Basic

Intermediate

Advanced

MoSCoW

Sub-Features

SF11 SF12 SF13 ⋮ SF1n

SF21 SF22 SF23 ⋮ SF2n

Dependent

Independent ...

SFn1 SFn2 SFn3 ⋮ SFnn

Basic

Intermediate

Advanced

MoSCoW

A

A

Verification Planning

Map F1,F2,...Fn to Requirements & Verification Specification Format

Basic

Intermediate

Advanced

MoSCoW

P1,P,... Pn

Verification Plan Database

Generate Verification Specification

Sub-Feature SF

B1 B2

SB11 ⋮ SB1n

SB21 ⋮ SB2n

Implementation Analysis Metrics

100%

B3

SB31 ⋮ SB3n

Implementation Analysis Metrics

...

100%

Bn

SBn1 ⋮ SBnn

Implementation Analysis Metrics

T1,T2,... Tn

TestBench Development & Implementation

F1 F2 F3 Fn

SF11 100% SF21 100% SF31 ... 100% SFn1

SF1n SF2n SF3n SFnn

SO1 SO2 SO3 SOn

Final Sign Off

Project Complete

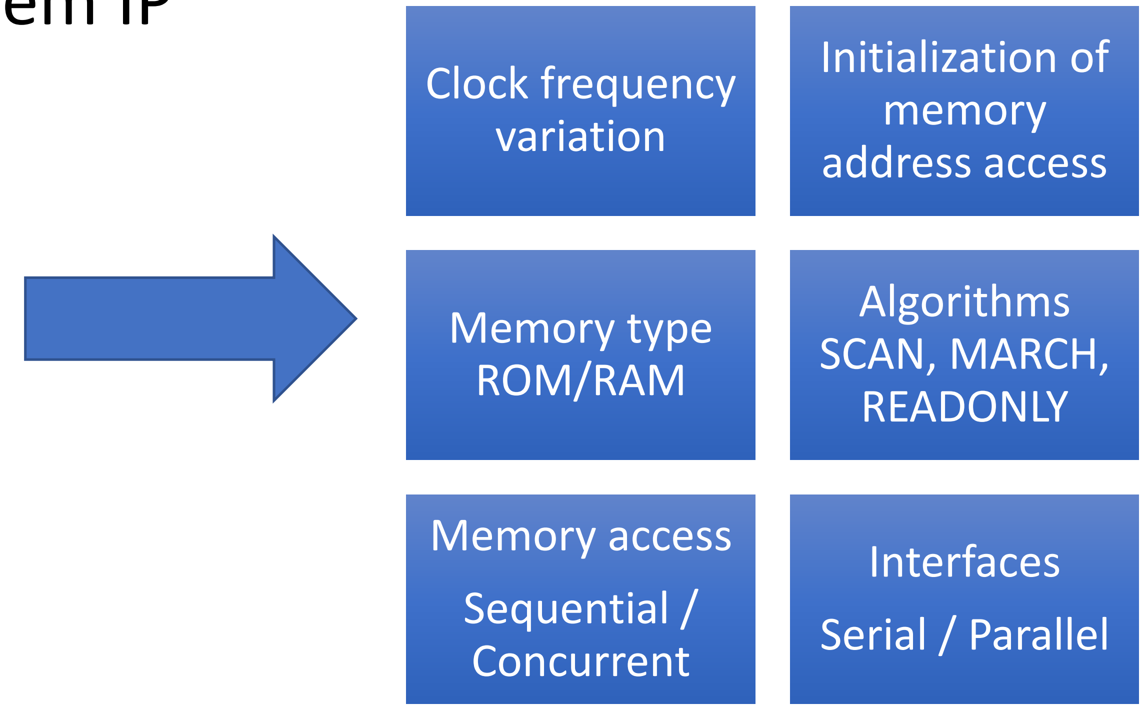
DV Flow Comparison

Conventional Flow	A3V Flow
Plan-driven & requirements-oriented	Agile & feature-oriented mapped onto requirements
Fixed task priorities throughout execution cycle	Ad-hoc task priorities at every stage of execution cycle
Concurrent execution of blocks with specific features (multi-block basis)	Sequential execution of blocks covering all features (block-by-block basis)
Limited visibility & clarity due to isolation of blocks/tasks	Full visibility due to sharing of blocks/tasks & constant reviews
Frequency of status alignment meetings is low initially and increases towards the sign-off	Frequency of status alignment meetings is high initially and decreases to constant value towards sign-off
Non-iterative and non-linear progress which can be concluded only at the final stage of project	Iterative and incremental progress at every stage ensures continuous assessment of project status

Use-case example – Memory IP (MIP)

- Development and verification of a user-interface for highly configurable and scalable memory subsystem IP

- Configurable features of MIP



Clock frequency variation

Initialization of memory address access

Memory type ROM/RAM

Algorithms SCAN, MARCH, READONLY

Memory access

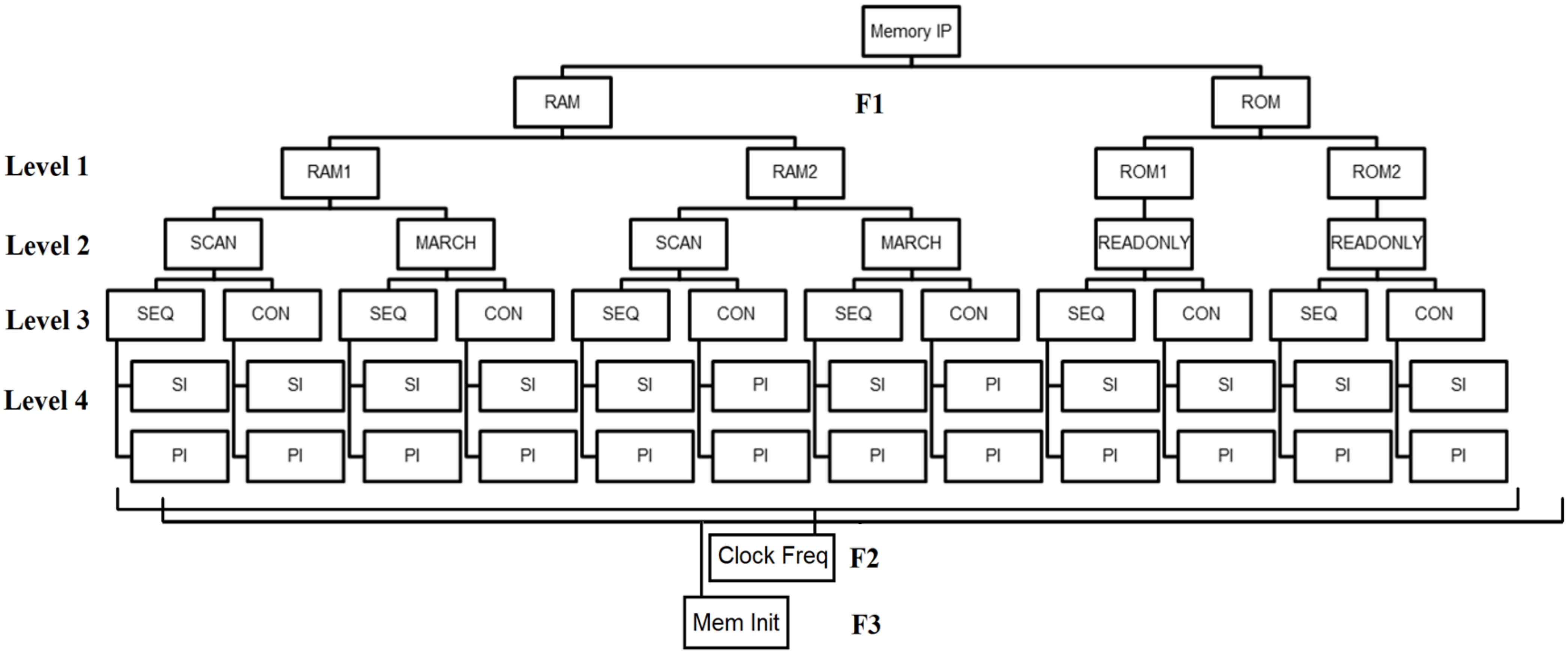
Sequential / Concurrent

Interfaces

Serial / Parallel

- Levels are assigned to each of above features

Use-case example – Memory IP (Contd…)



Results

Overall DV schedule – 20 work weeks

Each stage has individual sign-off SO1, SO2… SO(Final)

SOx → Resource & cost optimization

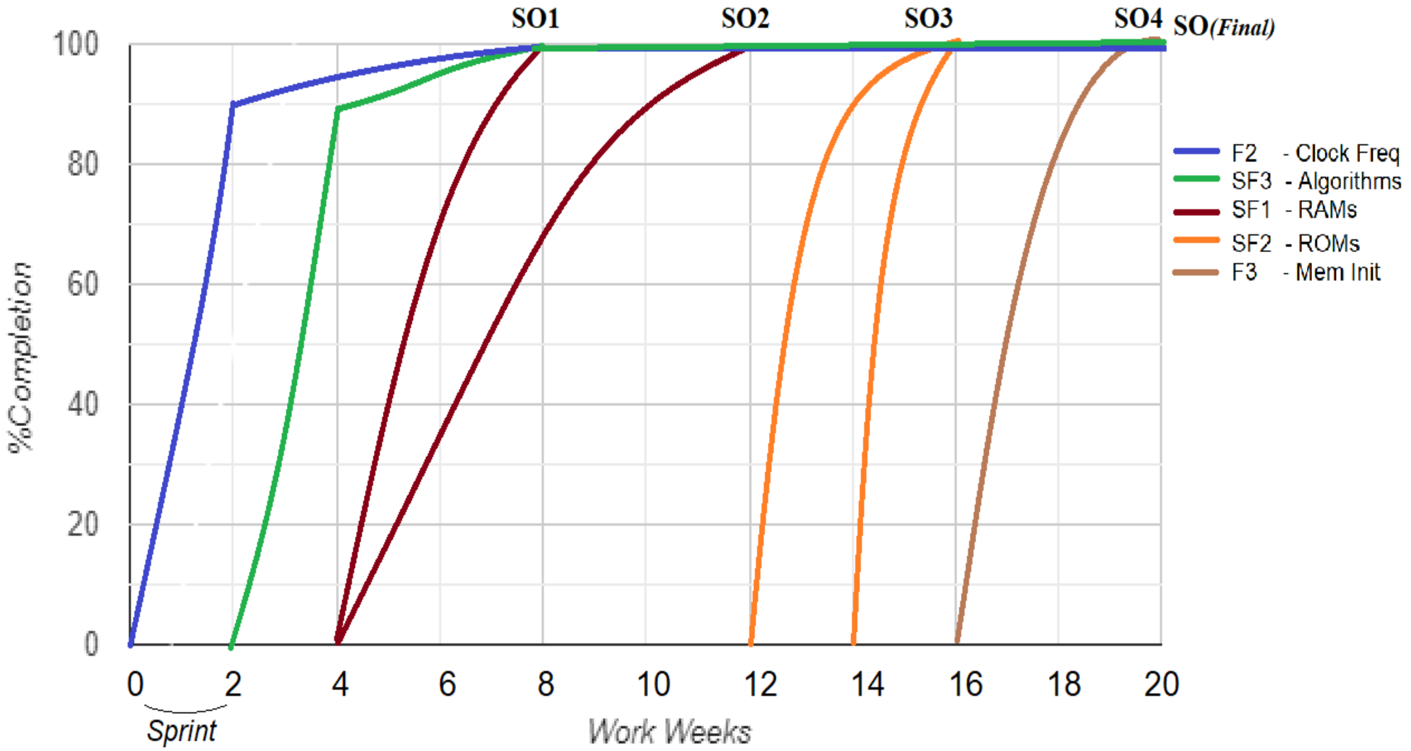
Effective work-load sharing

Less waiting time

Focused efforts → Improved quality

Efficient DV re-use during SOx execution

Sprint of 2 weeks for each feature/sub-feature



Opportunities Versus Challenges

Opportunities

- Optimum resource utilization
- Code optimization – peer-reviews & work-sharing
- Adaptable execution
- Tangible deliverables
- High level of confidence

Challenges

- Requires agile friendly design
- Need to resolve dependencies during planning phase
- Accurate and precise feature classification
- Agile flow awareness

Conclusion

A3V Flow – novel strategy for design verification

Combination of agile frameworks based on individual merits

Feature classification and task prioritization is the key

Encouraging results for a highly configurable and scalable MIP

Optimum execution cost

Enhanced overall quality of verification

References

[1] Shinobu Komai, Hiroshi Nakanishi and Hamdani Saidi, “Guidelines for selecting agile development method in system requirements definition”, 7th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), November 2017, ISBN:978-1-5386-3898-9

[2] Marco Kuhrmann, Paolo Tell, and Regina Hebig, “What makes agile software development agile,” IEEE Transactions on Software Engineering, vol. 1, pp. 1-1, July 2021, ISSN: 1939-3520

[3] Shi Zhong, Chen Liping, and Chen Tian-en, “Agile planning and development methods”, IEEE 3rd International Conference on Computer Research and Development, March 2011, ISBN:978-1-61284-840-2

[4] Wrike, “Agile Methodology Basics - Project Management Guide”, https://www.wrike.com/project-management-guide/agilemethodology-basics, 2009

[5] Mike Beedle, Arie van Bennekum, et. al, "Agile Manifesto and Principles", https://agilemanifesto.org, February 2001

[6] F. Paetsch, A. Eberlein, and F. Maurer, “Requirements engineering and agile software development,” WET ICE Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, June 2003,ISBN:0-7695-1963-6

[7] Yunsup Lee, Andrew Waterman, Henry Cook et. al, “An Agile Approach to building RISC-V Microprocessors IEEE Micro, Vol. 36, Issue.2, March 2016, ISSN: 1937-4143

[8] Luke Collins, “Applying agile techniques to IC design,” https://www.techdesignforums.com/practice/technique/agile-ic-methodology/, June 2015

[9] Neil Johnson and Byran Morris, “A Giant, Baby Step Forward: Agile Techniques for Hardware Design,” http://www.synopsys.com/news/pubs/snug/boston2009/mc3_johnson_paper.pdf, 2009

[10] Paul Cunningham, "Agile Approach to SoC Design Verification", Cadence Design Systems, https://www.eetasia.com/agile-approachto-soc-design-verification, June 2021

[11] Md Shamsur Rahim, AZM Ehtesham Chowdhury, Dip Nandi, Mashiour Rahman, "ScrumFall: A Hybrid Software Process Model", I.J. Information Technology and Computer Science, vol 12, pp 41-48, December 2018

[12] Sergio Marchese, "Formal verification enables Agile RTL development", https://www.techdesignforums.com/practice/technique, January 2014

THANK YOU
QUESTIONS?