VERIFICATION IN HARDWARE (ASIC/FPGA)

Sadat Rahman (E-mail)
Digital ASIC & FPGA Design
Ericsson Lund, 2016-10-04

LECTURE CONTENTS

- Introduction – 15 minutes
  - Ericsson (Who we are and what we do)
  - Hardware Development in Ericsson and Ericsson Lund.
- General Topics – 20 minutes
  - ASIC & FPGA Development Flow
  - Importance of Verification in Reality

Break (5 minutes)

- Technical Deep Dive – 30 minutes
  - Verification in different stages
  - Front-end Verification types and the process
- Misc. – 15 minutes
  - Skillset for a design & Verification Engineer
  - Days of Verification Engineer.
  - Some observation; Q & A

ABOUT ME

- Sadat Rahman
  - Originally from Bangladesh, lived in 3 countries so far
  - Bangladesh 19 years, Japan 12 years and Sweden last 4 years.
- Education:
  - Bachelor & Masters in Electrical & Electronics Engineering from Tokyo.
  - Specialized in Digital Signal Processing for Wireless Communication, Adhoc Networking, IVC (Inter Vehicle Communication) system.
    - 2 Technical Papers, 2 International Conference (2005-2007)
- Interest: Travelling, History
- Language: Japanese, Bangla, English, Swedish (very basic)

MY JOURNEY WITH ERICSSON

- Started with Ericsson in 2008 as HW Design Engineer
  - Why this Role:
    - Had interest how algorithms are implemented in real hardware.
    - I was focused on what I want to do while I was in 4th year of Bachelor course.
  - Why Ericsson:
    - Preferred Ericsson over Sony, Intel, NTT as I wanted to be in wireless domain and liked more the European Work Culture.
    - True global company (presence in almost every country in the world)
    - Vast opportunities available (depends on what the individual wants to explore)
  - 2008-2012 involved in 3G/4G Modem ASIC design, verification
    - Cross site product development (Sweden, Japan, Germany) but worked as a single unit.

- Moved to Sweden in 2012 and Joined Ericsson Modem
  - Verification Engineer, Verification Project Manager, Verification Architect/Methodology

THIS IS ERICSSON

Our Vision

NETWORKED SOCIETY

1980’s

1990’s

2000’s

2009

2020

28 billion devices

New services

OUR BUSINESS

OUR FOCUS AREAS

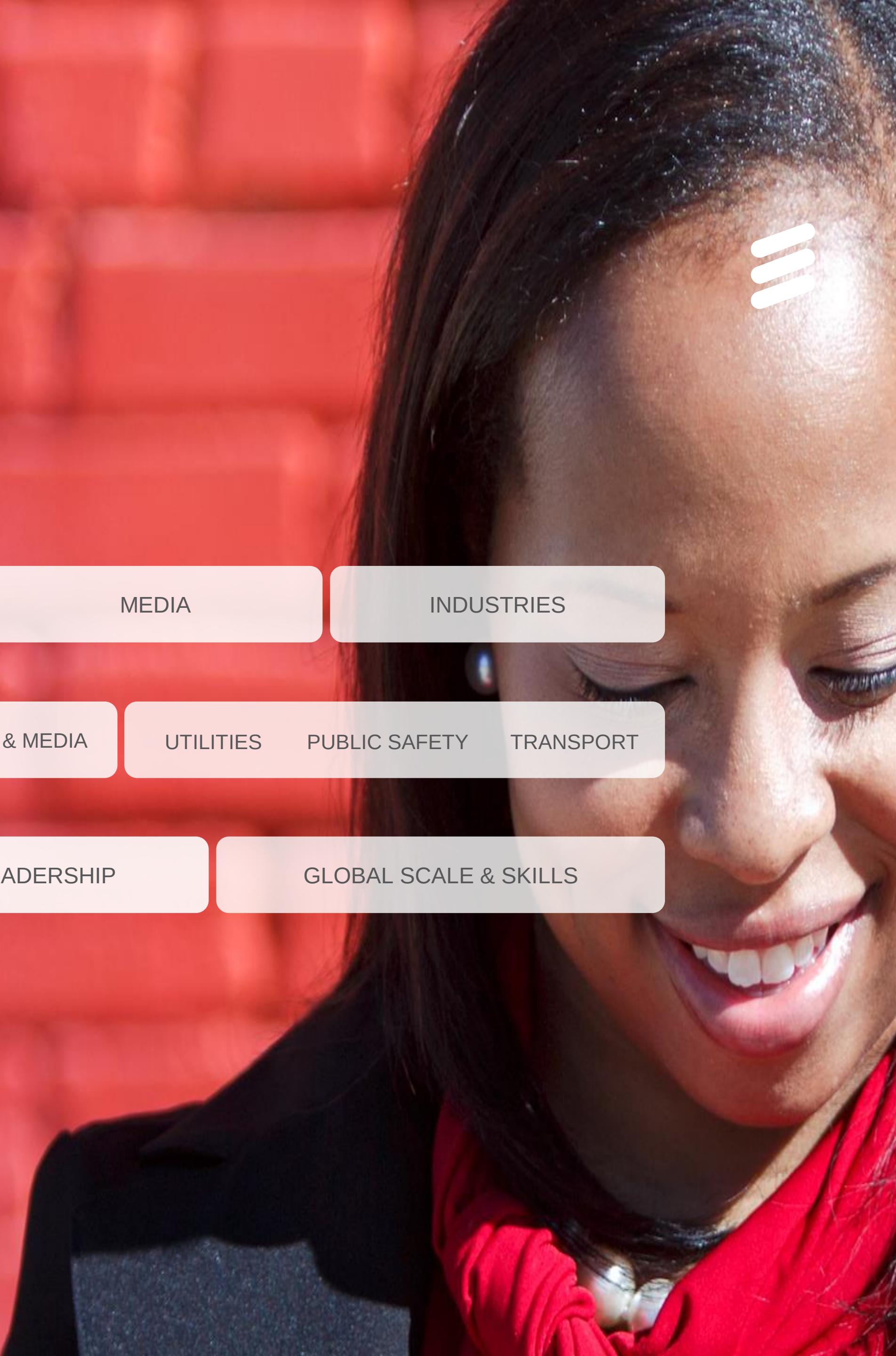
NETWORKS | IT | MEDIA | INDUSTRIES

OUR CUSTOMERS

OPERATORS | TV & MEDIA | UTILITIES | PUBLIC SAFETY | TRANSPORT

OUR ASSETS

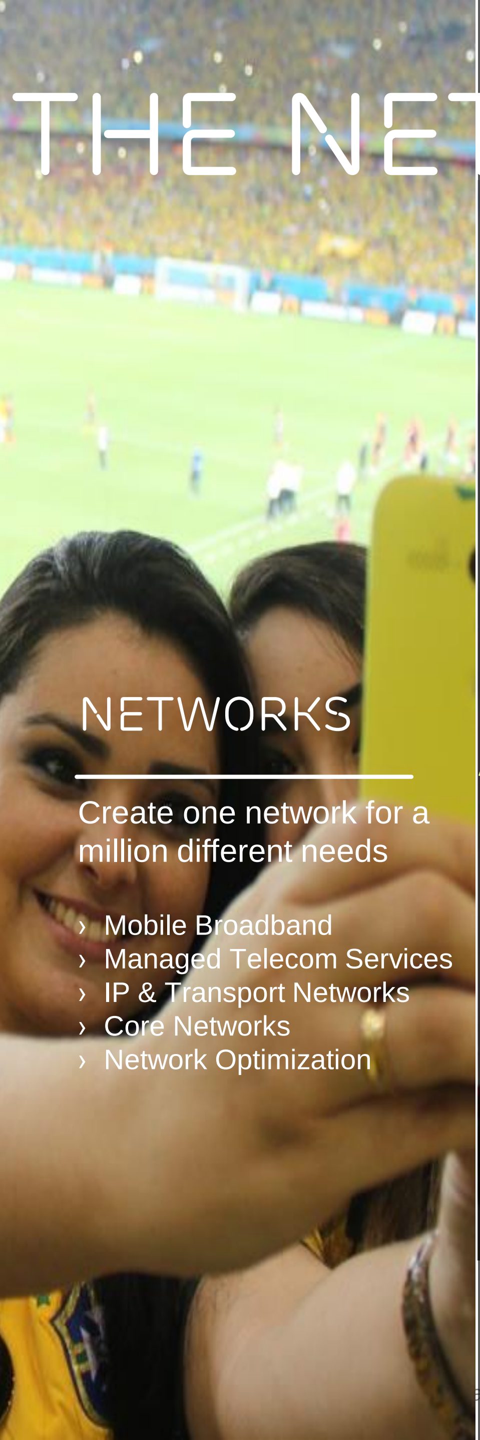
TECHNOLOGY LEADERSHIP | SERVICES LEADERSHIP | GLOBAL SCALE & SKILLS



THE NETWORK BUSINESS

Create one network for a million different needs

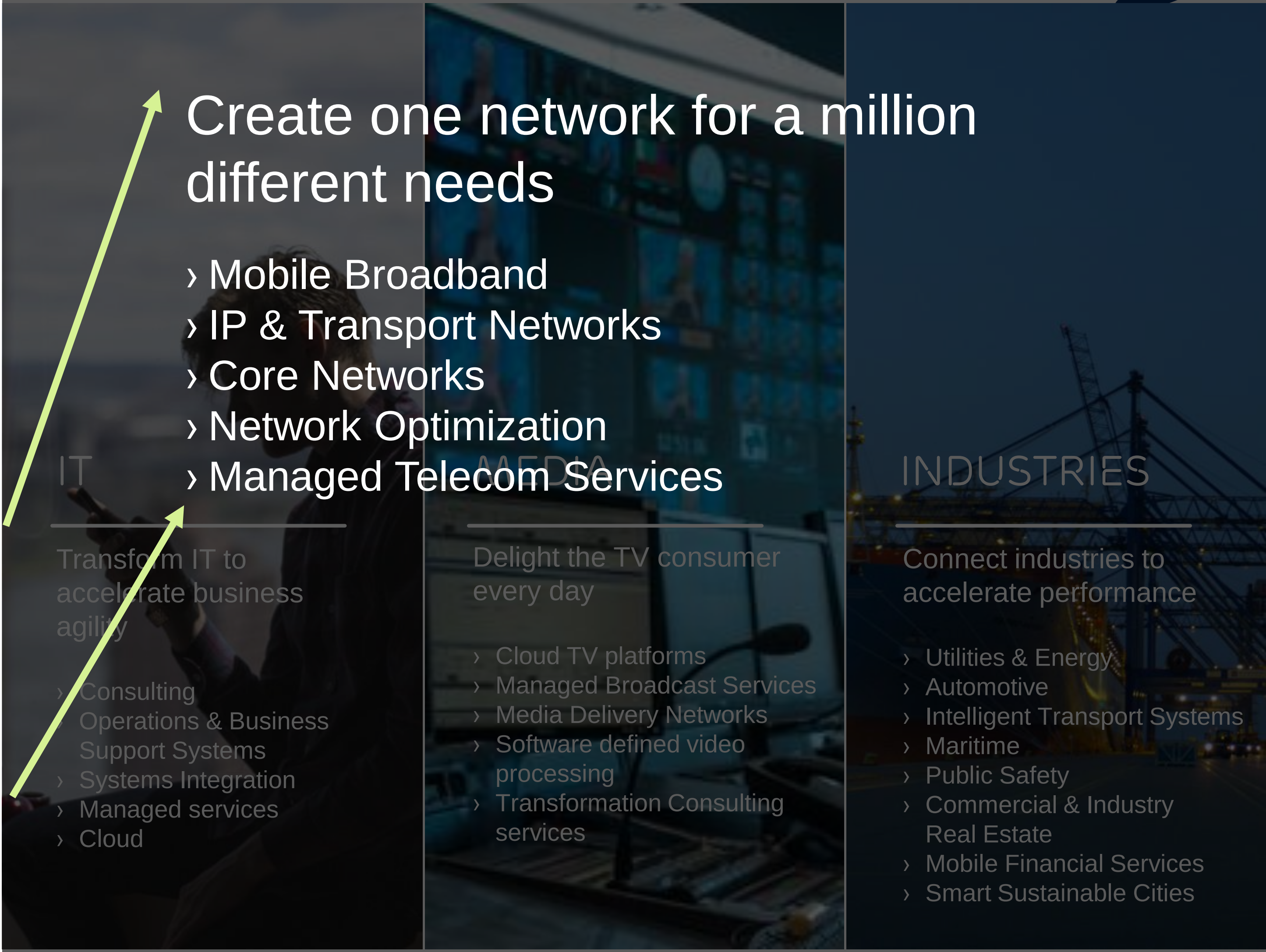
- Mobile Broadband
- IP & Transport Networks
- Core Networks
- Network Optimization
- Managed Telecom Services



NETWORKS

Create one network for a million different needs

- Mobile Broadband
- Managed Telecom Services
- IP & Transport Networks
- Core Networks
- Network Optimization



IT

Transform IT to accelerate business agility

- Consulting
- Operations & Business Support Systems
- Systems Integration
- Managed services
- Cloud

MEDIA

Delight the TV consumer every day

- Cloud TV platforms
- Managed Broadcast Services
- Media Delivery Networks
- Software defined video processing
- Transformation Consulting services

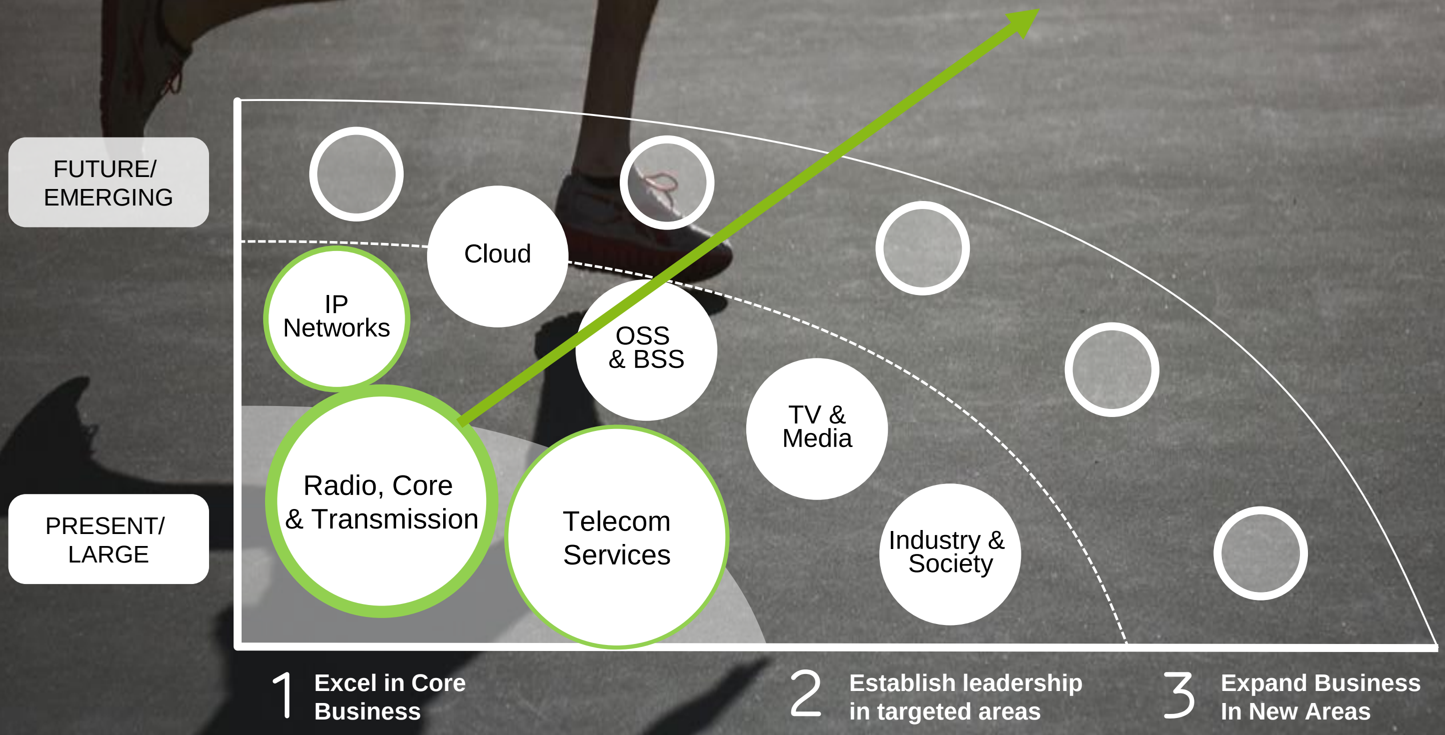
INDUSTRIES

Connect industries to accelerate performance

- Utilities & Energy
- Automotive
- Intelligent Transport Systems
- Maritime
- Public Safety
- Commercial & Industry Real Estate
- Mobile Financial Services
- Smart Sustainable Cities

OUR STRATEGIC DIRECTION

Radio & Transmission Products are in the core of Ericsson's network business

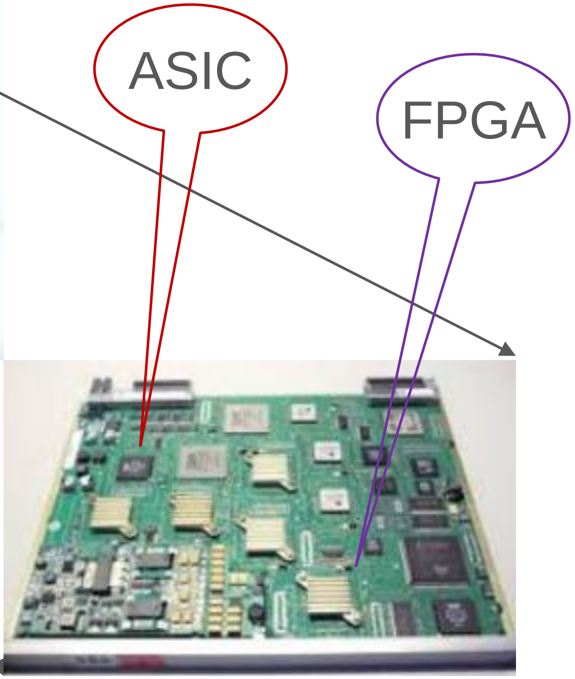


ERICSSON
RADIO
SYSTEM

JOURNEY TO THE 5G

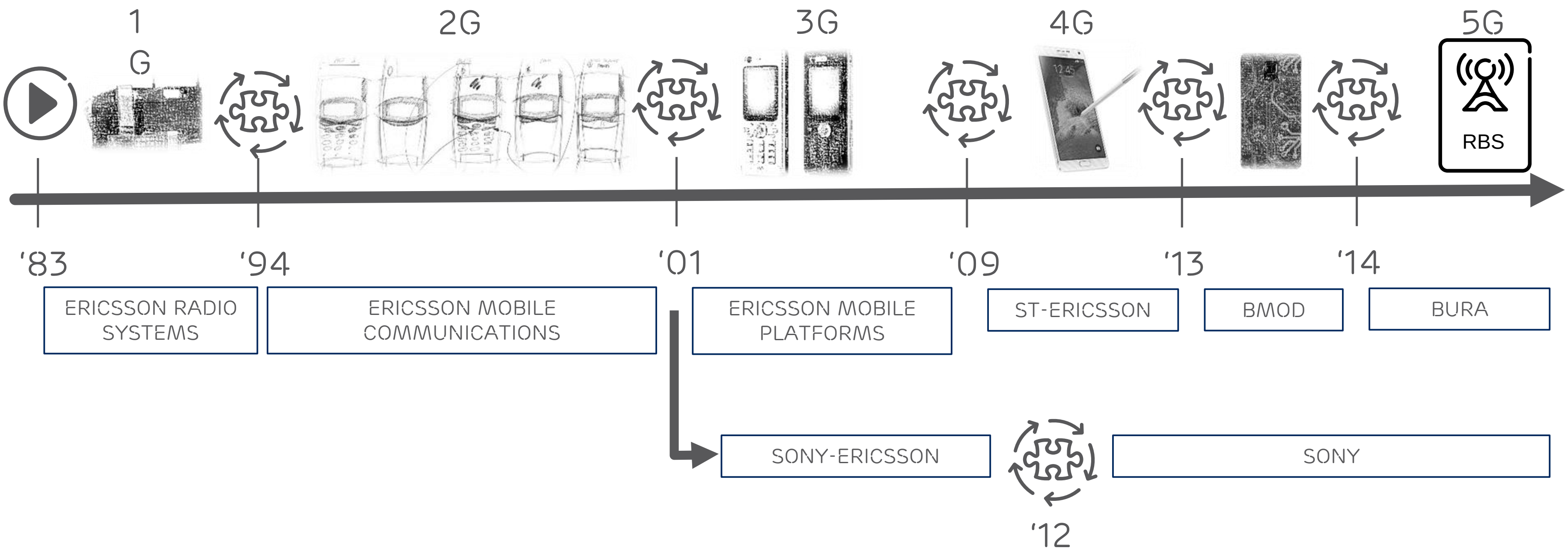
RADIO HW – THE CORE OF THE CORE

WHERE DOES HARDWARE CONTRIBUTE



Hardware boards (Analog & Digital Components, ASIC/FPGA) for Transceivers (both Radio and Baseband units)

LUND SITE – TIMELINE



1G

2G

3G

4G

5G

RBS

'83 ERICSSON RADIO SYSTEMS

'94 ERICSSON MOBILE COMMUNICATIONS

'01 ERICSSON MOBILE PLATFORMS

'09 ST-ERICSSON

'13 BMOD

'14 BURA

SONY-ERICSSON

'12

SONY

RBS: **R**adio **B**ase **S**tation

BMOD: **B**usiness **U**nit **Mod**em (2G/3G/4G Chipset)

BURA: **B**usiness **U**nit **Ra**dio

LUND SITE AND BURA

- Since 2014, Lund is a competence center for radio network development
- Activities span from HW and SW product development to research and standardization
- Approximately 600 employees
  - 450 in Business Unit Radio
    - 65 people working in Hardware domain.
  - 75 in Ericsson Research
  - 40 in other units

RADIO PRODUCTS & VARIANTS
- WHO WE ARE & WHAT WE DO

- Senior employees
  - 10-30 years experience
- Strengths
  - Innovative spirit (Numerous HW patents from this site)
  - Digital ASIC
  - RF ASIC
  - IP and system development
  - Holistic view on entire systems
  - Low power design

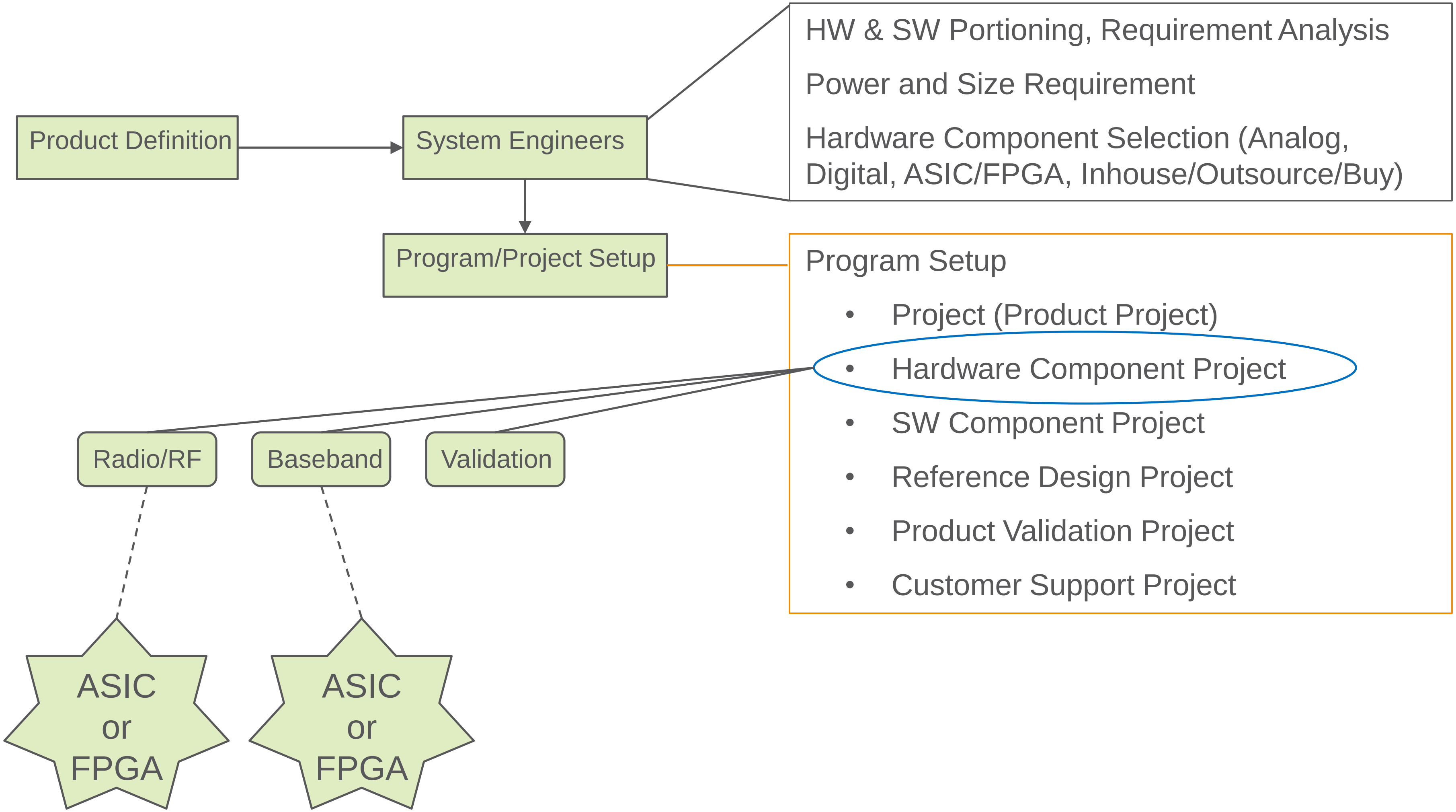
- Currently ongoing
  - Digital ASIC & FPGA development for 5G Product line up
  - Radio Product Component (such as Filter, Power Amplifier, Antenna Controlling Unit) Development

GENERAL TOPIC:

→ ASIC/FPGA DEVELOPMENT FLOW

→ IMPORTANCE OF VERIFICATION

PRODUCT DEVELOPMENT PROCESS ASIC/FPGA (SIMPLIFIED)



HW & SW Portioning, Requirement Analysis

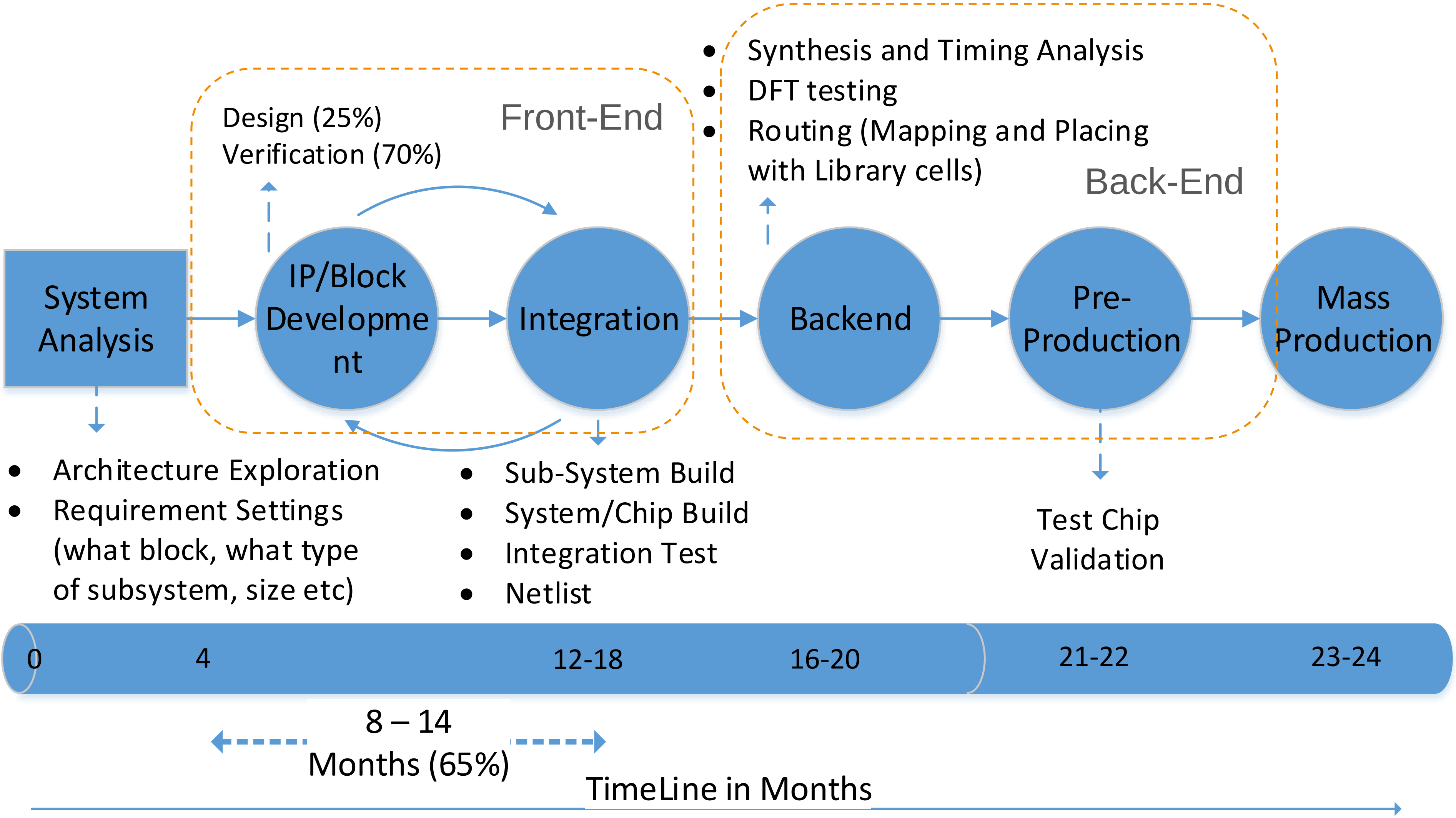
Power and Size Requirement

Hardware Component Selection (Analog, Digital, ASIC/FPGA, Inhouse/Outsource/Buy)

Program Setup

- Project (Product Project)
- Hardware Component Project
- SW Component Project
- Reference Design Project
- Product Validation Project
- Customer Support Project

ASIC DEVELOPMENT FLOW



Front-End

Design (25%)
Verification (70%)

Back-End

- Synthesis and Timing Analysis
- DFT testing
- Routing (Mapping and Placing with Library cells)

System Analysis → IP/Block Development → Integration → Backend → Pre-Production → Mass Production

- Architecture Exploration
- Requirement Settings (what block, what type of subsystem, size etc)

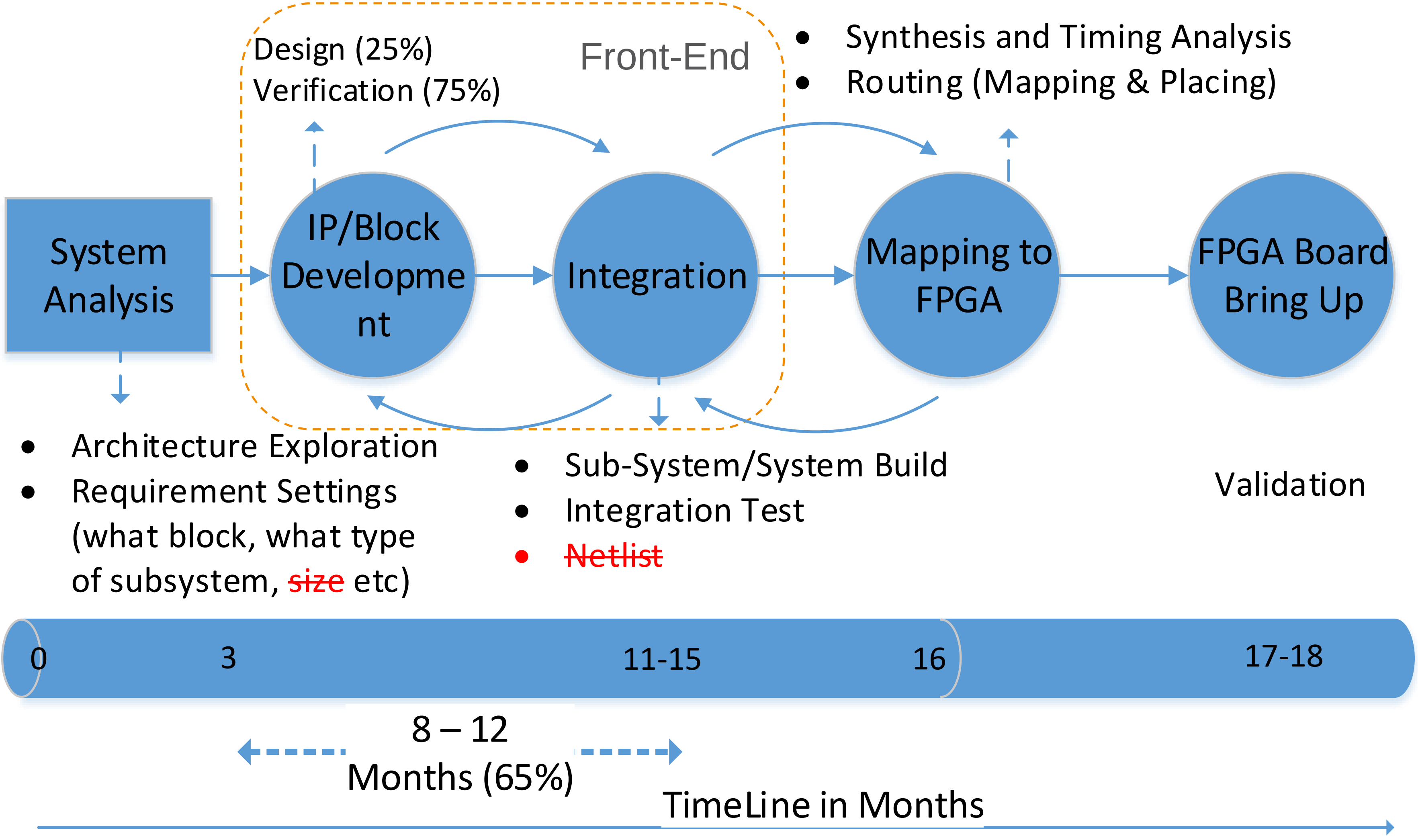
- Sub-System Build
- System/Chip Build
- Integration Test
- Netlist

Test Chip Validation

8 – 14
Months (65%)

TimeLine in Months

FPGA DEVELOPMENT FLOW



- Synthesis and Timing Analysis
- Routing (Mapping & Placing)

- Architecture Exploration
- Requirement Settings (what block, what type of subsystem, ~~size~~ etc)

- Sub-System/System Build
- Integration Test
- ~~Netlist~~

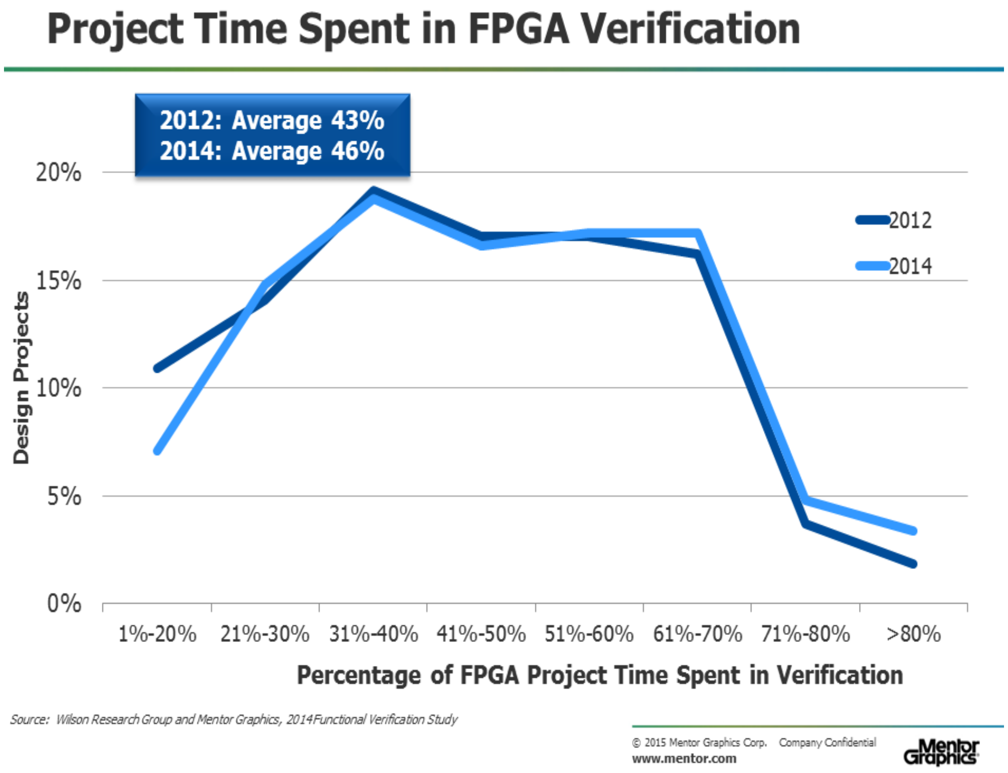
Validation

WHAT IS YOUR OBSERVATION ON THESE TWO PROCESS???

MOST TIME CONSUMED BY VERIFICATION

--→ VERIFICATION IS CRITICAL

WHAT THE MARKET SAYS

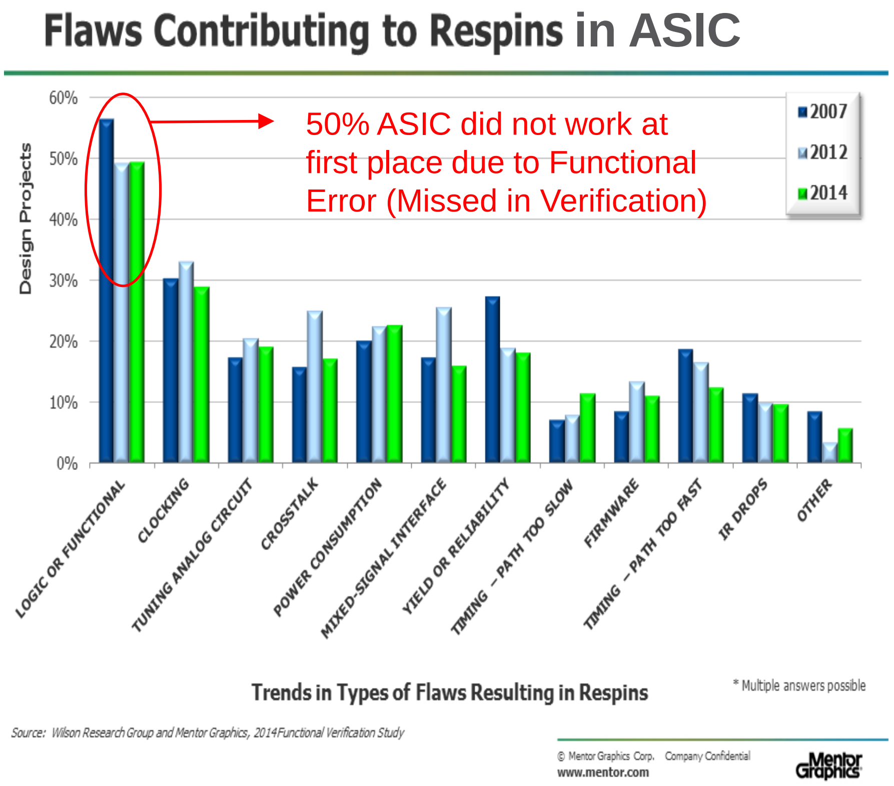


Project Time Spent in FPGA Verification

2012: Average 43%
2014: Average 46%

Source: *Wilson Research Group and Mentor Graphics, 2014 Functional Verification Study*

© 2015 Mentor Graphics Corp. Company Confidential
www.mentor.com



Flaws Contributing to Respins in ASIC

50% ASIC did not work at first place due to Functional Error (Missed in Verification)

Trends in Types of Flaws Resulting in Respins

* Multiple answers possible

Source: *Wilson Research Group and Mentor Graphics, 2014 Functional Verification Study*

© Mentor Graphics Corp. Company Confidential
www.mentor.com

ASIC verification is biggest challenge in digital hardware development.

Post Silicon Bug can cause respin → Costly fix → delay in TTM

TIME FOR A TEASER

- What is the difference between Testing and Verification?
  - According to the dictionary these are synonyms to each other, but are these same????

SW TESTING VS HW VERIFICATION

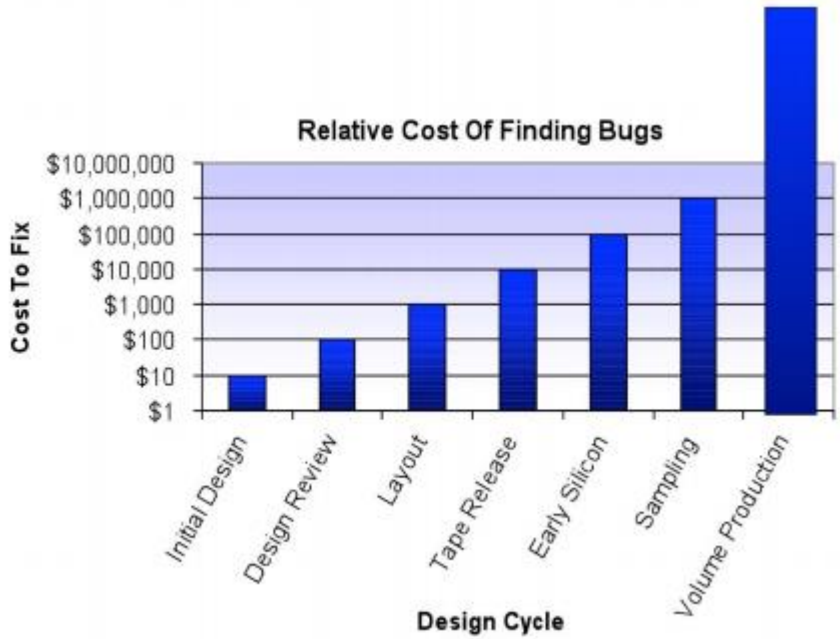
Comparison Factor	SW Testing	HW Verification
Meaning	Make sure requirements are fulfilled	Make sure to validate the hardware
Granularity level	High level requirement verification	Low level requirement verification
Aspect	Testing targets that SW requirements wish list are tick off that matches with HW performance. SW upgrade is possible anytime.	Think what may happen in a real hardware, e.g. thermal condition, latency, load difference, failsafe etc. Assume that HW should never fail.
Clock	No Clocking concept. Very Limited time domain scenario	Clocking concept is pre-requisite in HW verification.
Physical Pattern	No physical requirements such as size/power	Need for Size & power consumption analysis.

IMPORTANCE OF VERIFICATION IN HW

- Product does not work when there is even a minor failure.
  - Who would buy a car if there is breaking problem at high speed?
  - An ASIC that does not work is nothing but a stone.
    - In FPGA you have the opportunity to fix the error though. BUT not ASIC
  - Validate the requirements from not only the stakeholders/product owners but also verify unseen scenarios.
    - Question the requirements → Be inquisitive

- Verification is biggest challenge in hardware development.
  - Post Silicon/production Bug can cause respin → Costly fix → delay in TTM (time to market)

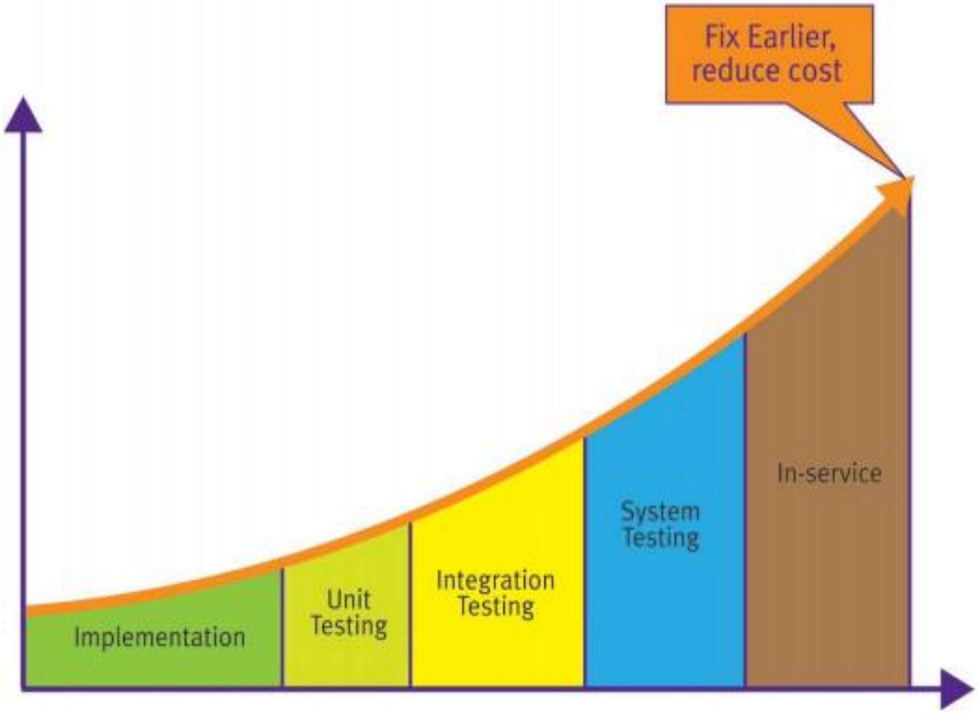
THE REALITY



Silicon Debug, Doug Josephson and Bob Gottlieb, (Paul Ryan)

D. Gizopoulos (ed.), Advances in Electronic Testing: Challenges and Methodologies, Springer, 2006

← Depending on the size/complexity of ASIC it may cost 10M USD in respin



Better in Verification → Catch Bugs Early → The sooner the better

TECHNICAL DEEP DIVE

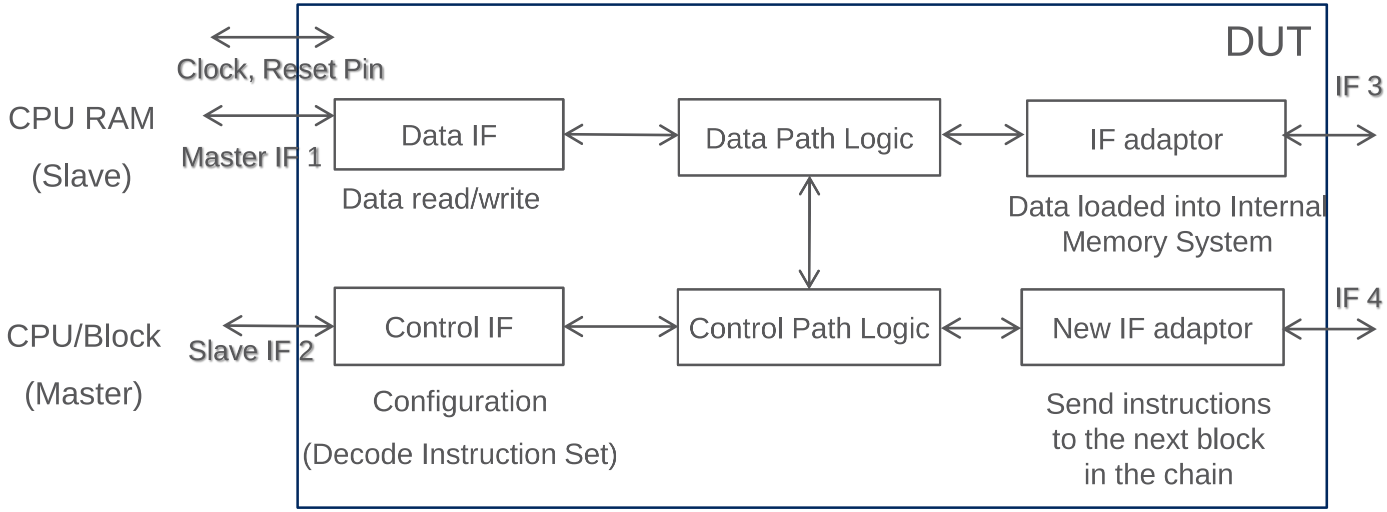
→ VERIFICATION IN DIFFERENT STAGES

→ VERIFICATION TYPES & THE PROCESS

SOME BASIC KNOWLEDGE

- When talking about hardware
  - Think about clock and think about reset
  - We talk about “event” → When some action happens at certain clock cycle.
  - We often talk about transaction
    - Transaction == data struct (like C) with some fields/members for the tragetted design
    - Transaction == Sequence Item/Stimulus
- Different Activities are ongoing in a test/real HW
  - Unit activity is called ”Sequence”
    - Human behavior like ”a day in the school” → Multiple big actions, like lectures, lab test → Each lecture is like a sequence.
  - In HW basic configuration is like a sequence.

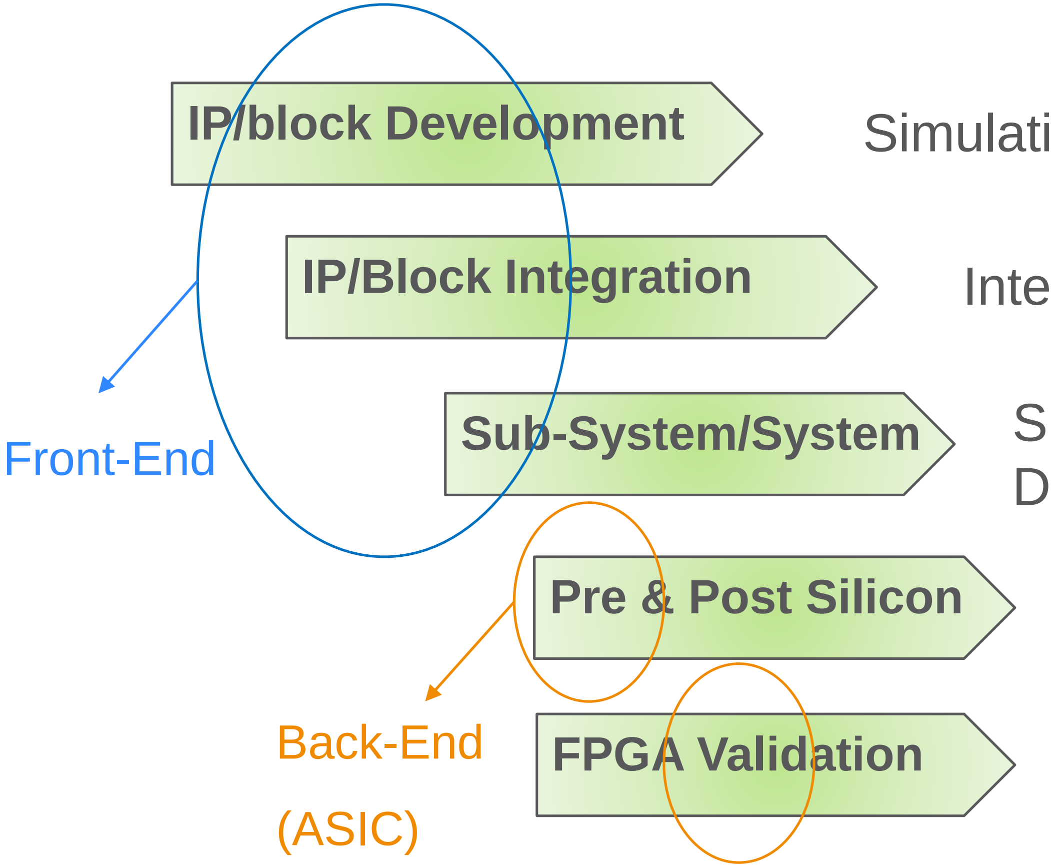
PRE-VERIFICATION ASPECTS



- To Verify the Design
  - What are the functionalities, How does it work in the system chain, The Traffic Flow
  - What are the Control IF, Data IF.
  - What the are critical path/use cases/difficult to verify. → Test Case Definition
  - How to measure the progress, when are we done? → Coverage Metrics
- Verification & Verification environment is built considering all these aspects.

VERIFICATION STAGES

- ASIC/FPGA Development flow & corresponding verifications



IP/block Development — Simulation, Formal (Functional)

IP/Block Integration — Integration Test (Connectivity)

Sub-System/System — Subsystem/System Functional Test, DFT test

Pre & Post Silicon — Backend Test, SoC level test (SW driven)

FPGA Validation — Board Bring up/Validation test (No Backend in FPGA)

Front-End

Back-End

(ASIC)

FRONT END VERIFICATION

- Two main categories
  - Dynamic Functional Verification (Simulation)

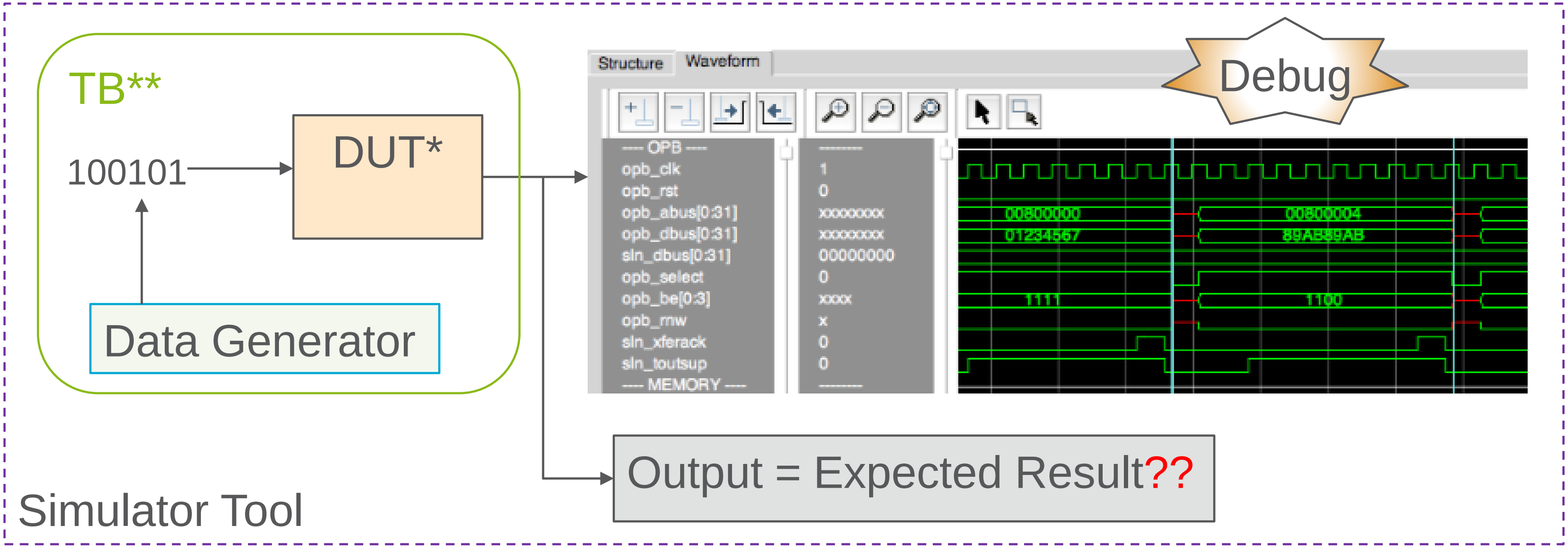
    Main Trend

    - User simulates the real life use cases
    - Event or Transaction based real time verification
    - A well defined/architected simulation environment is necessary
    - Programming Skill needed (Object Oriented, Functional)
  - Static Functional Verification (Formal)

    New Trend

    - Observes activity based on defined behavior.
    - Abstraction based verification, not the real time .
    - No environment needed but needs high logical & analytical skills
    - Limited programming skill but high HW knowledge needed

WHAT IS SIMULATION



TB**

100101

DUT*

Data Generator

Debug

Output = Expected Result??

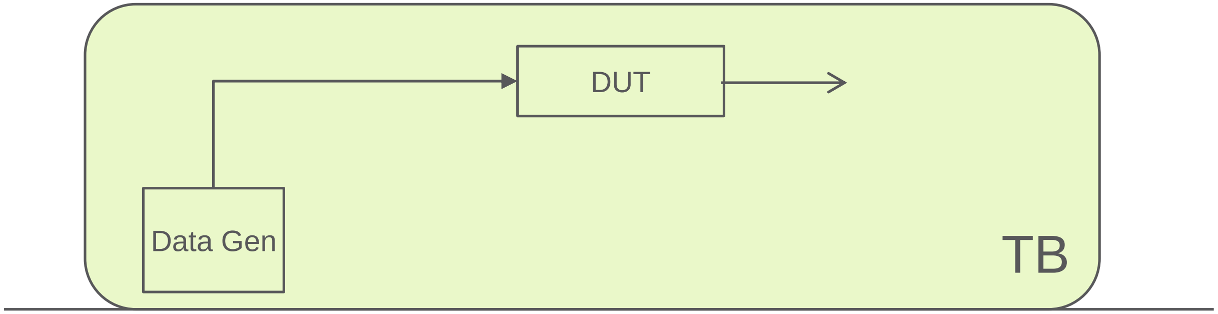
Simulator Tool

- In a Test Environment User generates valid input to the design over certain period
  - − If result == expected data → Test Success (Requirement Fulfilled)
  - − If result does not match to expected → Test Fails
- How does the TB look like??
  - − Relation to the DUT, let us explain part by part.

**DUT: Design Under Test*

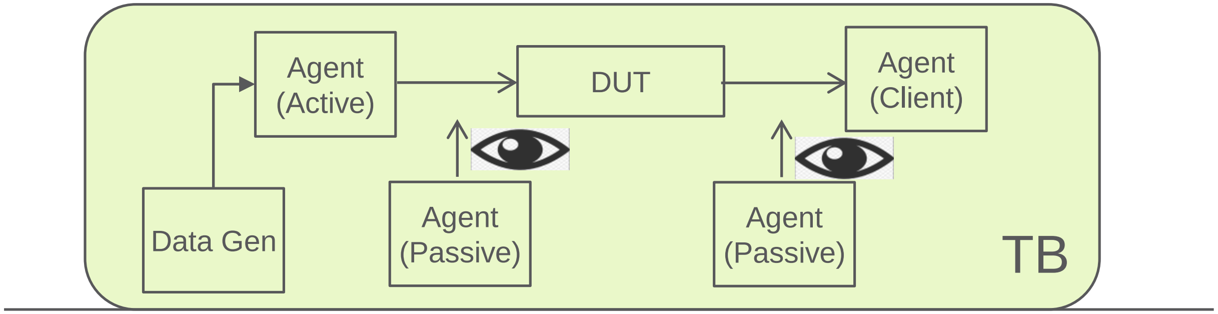
***TB: Tetsbench (Test Environment)*

TB DEFINITIONS (1/3)



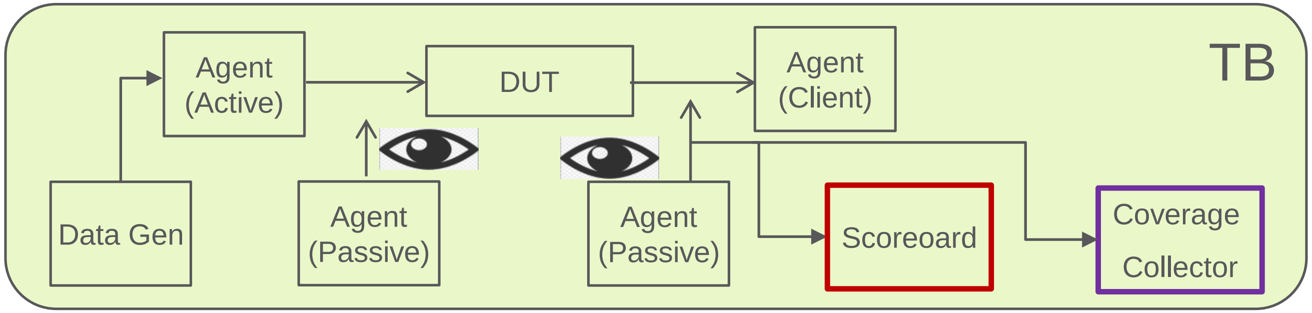
- DUT/Design receives data in a certain format
  - Data Sturucture depends on the functional requirement
  - Data means both
    - Configuration Parameters that is written in Registers
    - Data Packets/Frames/Payload that is processed by Design.
- Data Generator (Data Gen) Component takes care of this task.

TB DEFINITIONS (2/3)



- To Drive & Receive Configuration & Data we need verification components.
  - Verification **Agent**
    - Driving ability → Active Agent
    - No driving ability, only monitors activity → Passive Agent
    - Response to Master’s request, driving ability → Client Agent
    - Serves Data to multiple Master’s request → Server Agent
  - Active Agent receives data from Data Gen Component & sends to DUT

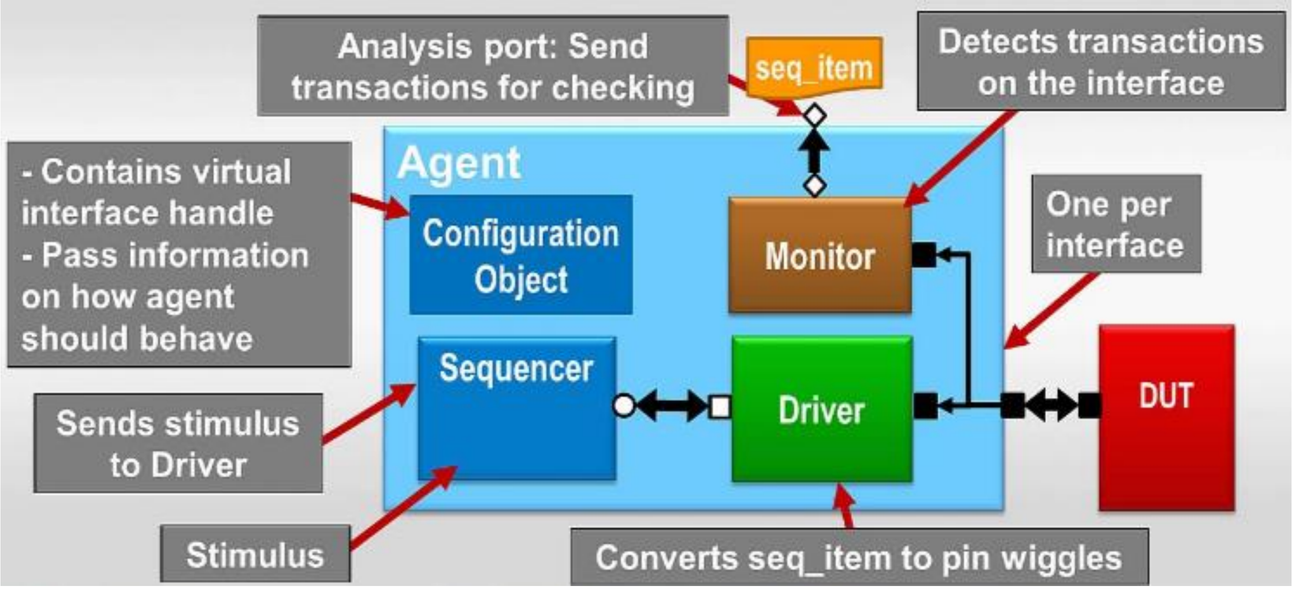
TB DEFINITIONS (3/3)



- We need to receive & process the output data.
  - Are we getting the correct data?? Comparator required.
    - ”Scoreboard” component.
    - Output is compared against reference data.
- How much are we progressing??
  - ”Coverage Collector”
    - Define & Collect the metrics.
  - Coverage Types (Functional Coverage, Code Coverage etc)
    - Broad topic, need another session.

INSIDE THE AGENT

Heart of a TB


Analysis port: Send transactions for checking
seq_item
Detects transactions on the interface
- Contains virtual interface handle
- Pass information on how agent should behave
Agent
Configuration Object
Monitor
One per interface
Sequencer
Driver
DUT
Sends stimulus to Driver
Stimulus
Converts seq_item to pin wiggles

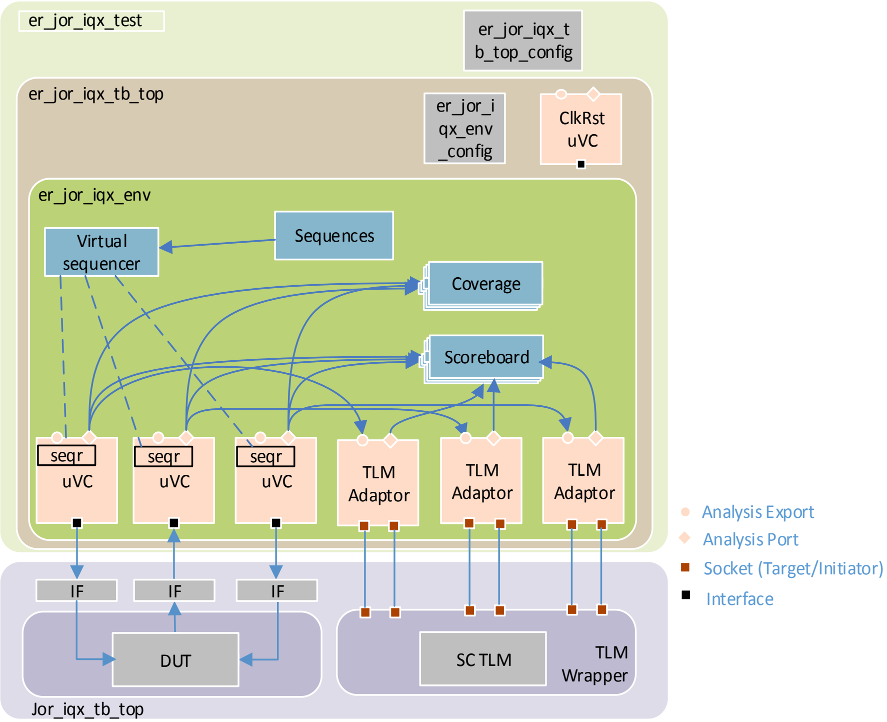
- ”Sequencer”
  - Generates traffic, handles traffic requests from user
- ”Driver”
  - Receives traffic request from Sequencer
  - Converts to DUT PIN level activity
- ”Monitor”
  - Watches ongoing activity in IF
  - Translates into Transaction item
  - Notifies subscribers if something happens.
- Configuration object
  - What is the configuration mode

SIMULATION – TEST ENVIRONMENT

- A real Example of semi-complex TB environment.
  - UVM library based
  - Object Oriented Programming

- Challenges
  - Smart Strategy to reuse the TB at higher level.
  - Complex setup
  - Bring up takes long time
  - Flexible enough for future project

JOR IQ Switch TB Environment Example

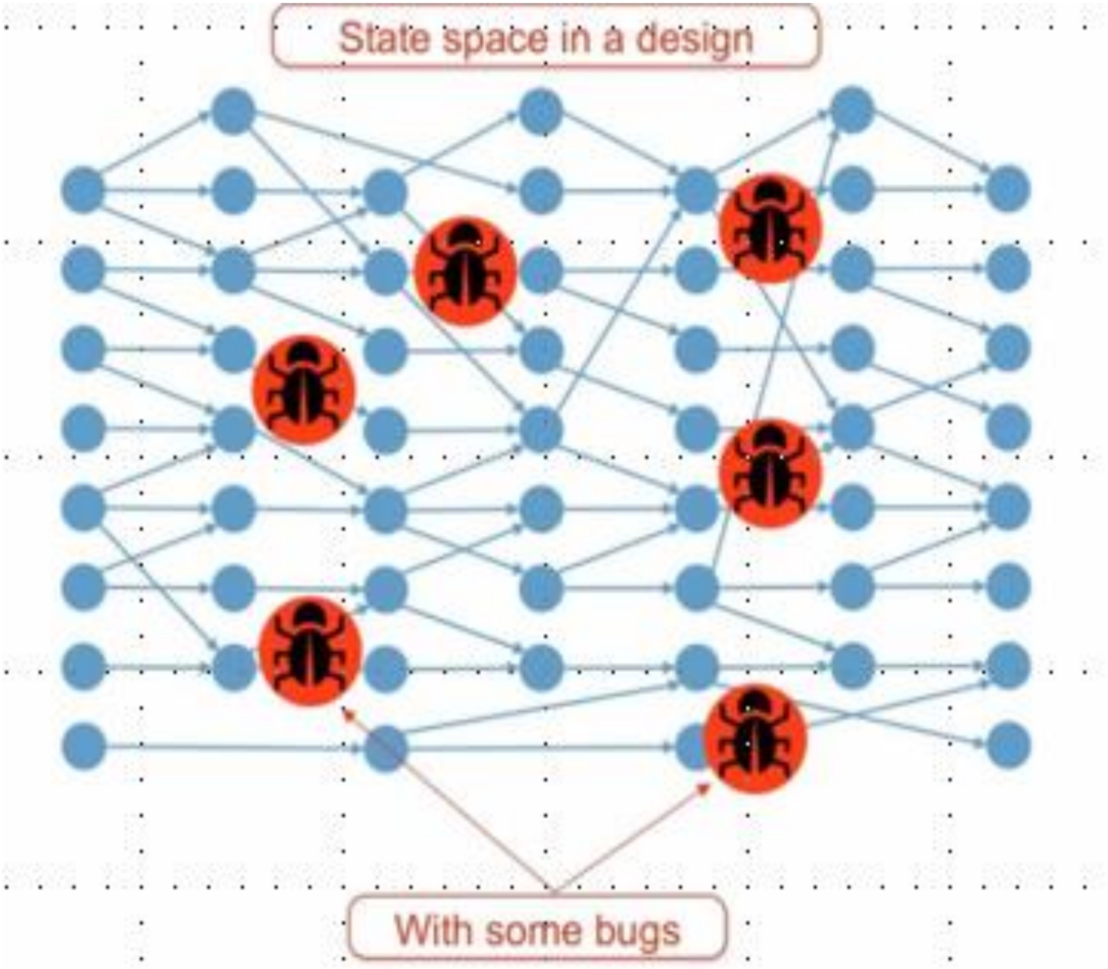


FRONTEND :

FORMAL VERIFICATION

STATE SPACE IN DESIGN

- Bugs are hidden somewhere in state space in the design.
- Large design → More States → More bugs

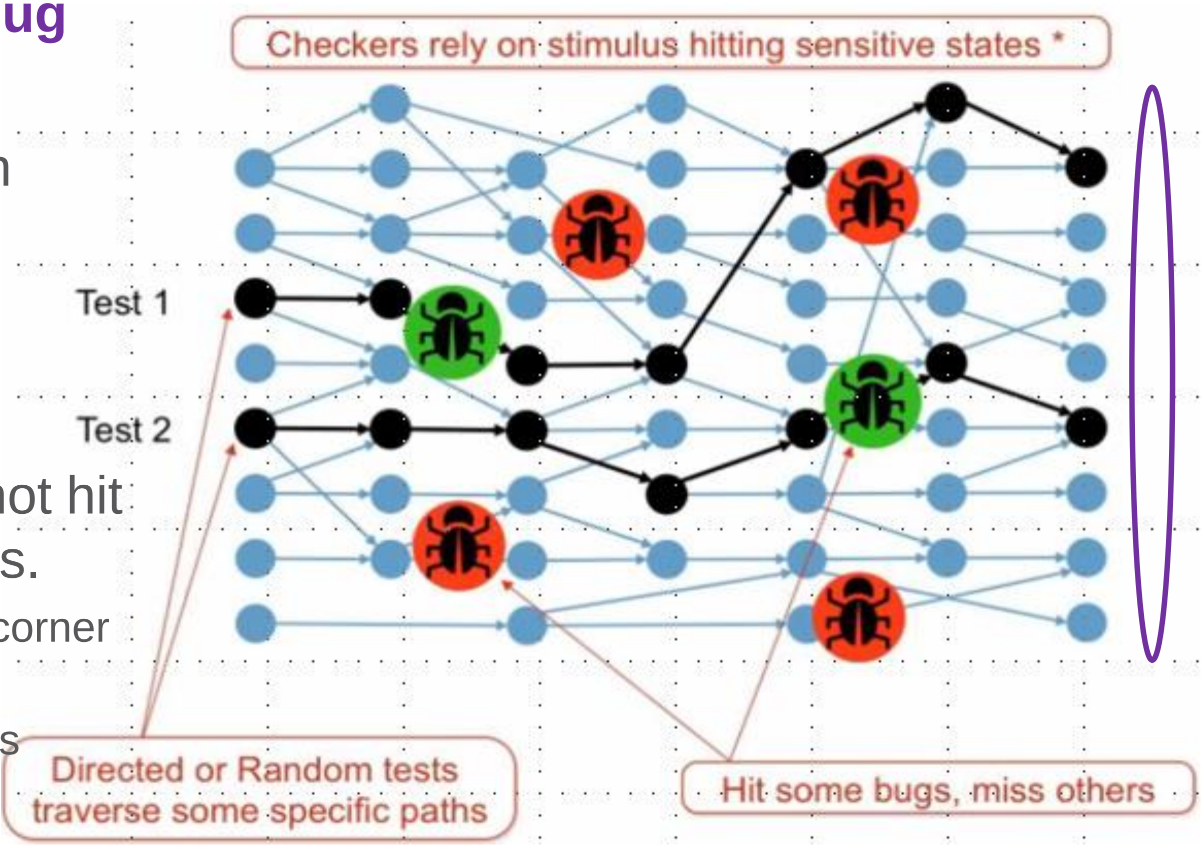


SIMULATION AT WORK

- **Checkers to detect the bug**

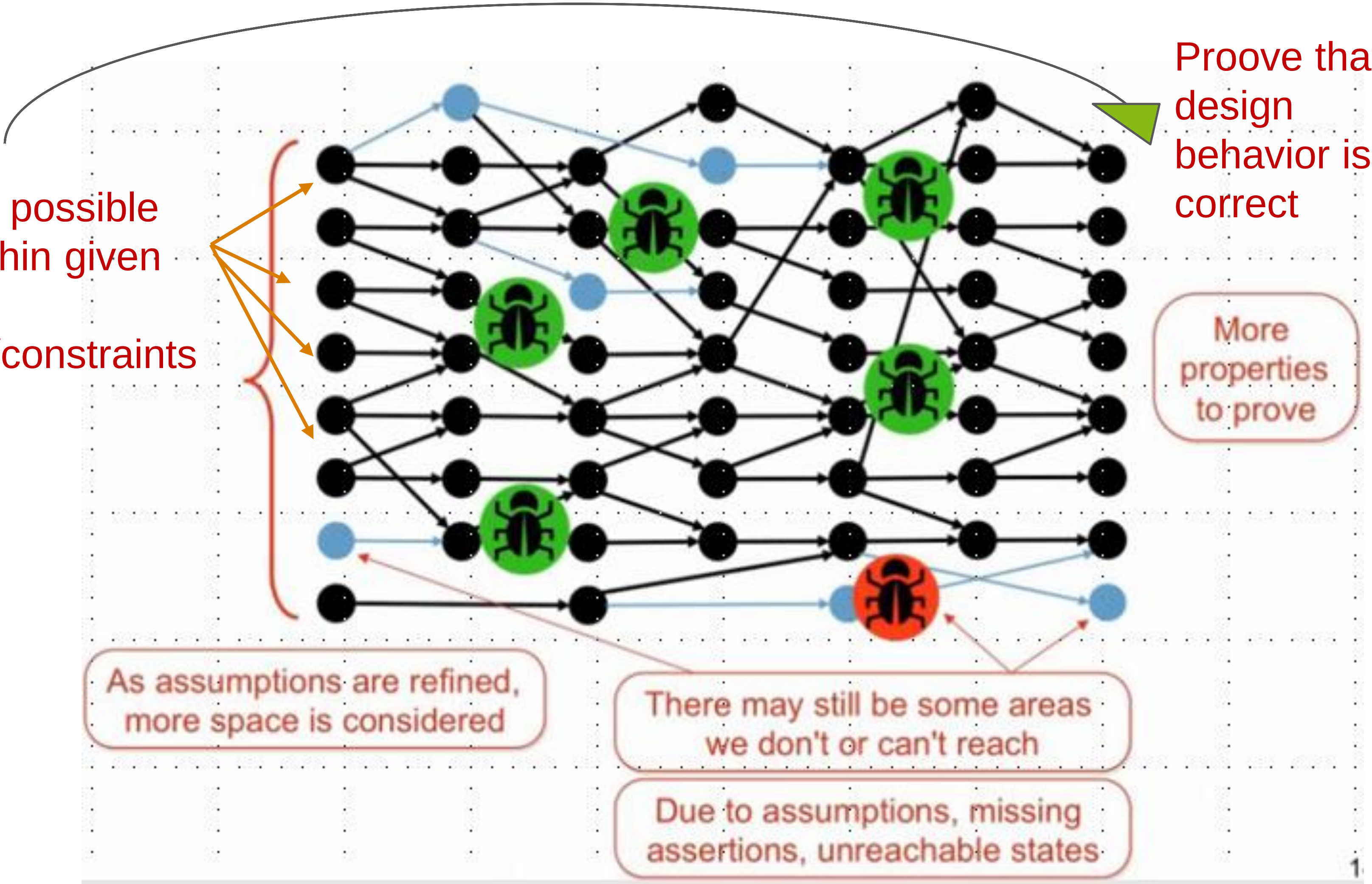
- Bug Detection depends on
  - How good stimuli pattern generation is.
  - How smart the checkers are.

- Smart Tests may or may not hit state spaces where BUG is.
  - Missing bugs in deep state (corner case)
  - Bug in a state where debug is difficult


Checkers rely on stimulus hitting sensitive states *
Test 1
Test 2
Directed or Random tests traverse some specific paths
Hit some bugs, miss others

FORMAL VERIFICATION

Injects all possible inputs within given design limitation/constraints


More properties to prove
As assumptions are refined, more space is considered
There may still be some areas we don't or can't reach
Due to assumptions, missing assertions, unreachable states

Proove that design behavior is correct

- Unreachable states are detected by Formal Tools.

FUNDAMENTALS OF FORMAL VERIFICATION

Specification

If A =0, then Y = 0

If A =1, then Y = B

Verify Design Properties

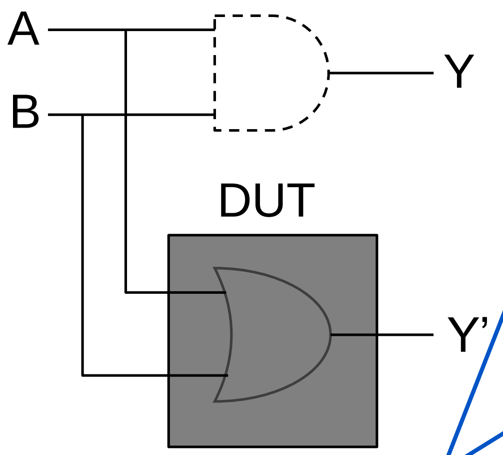
Reference

!A |-> !Y

A |-> Y == B

or Y == A && B

Design Constraints/Assumptions

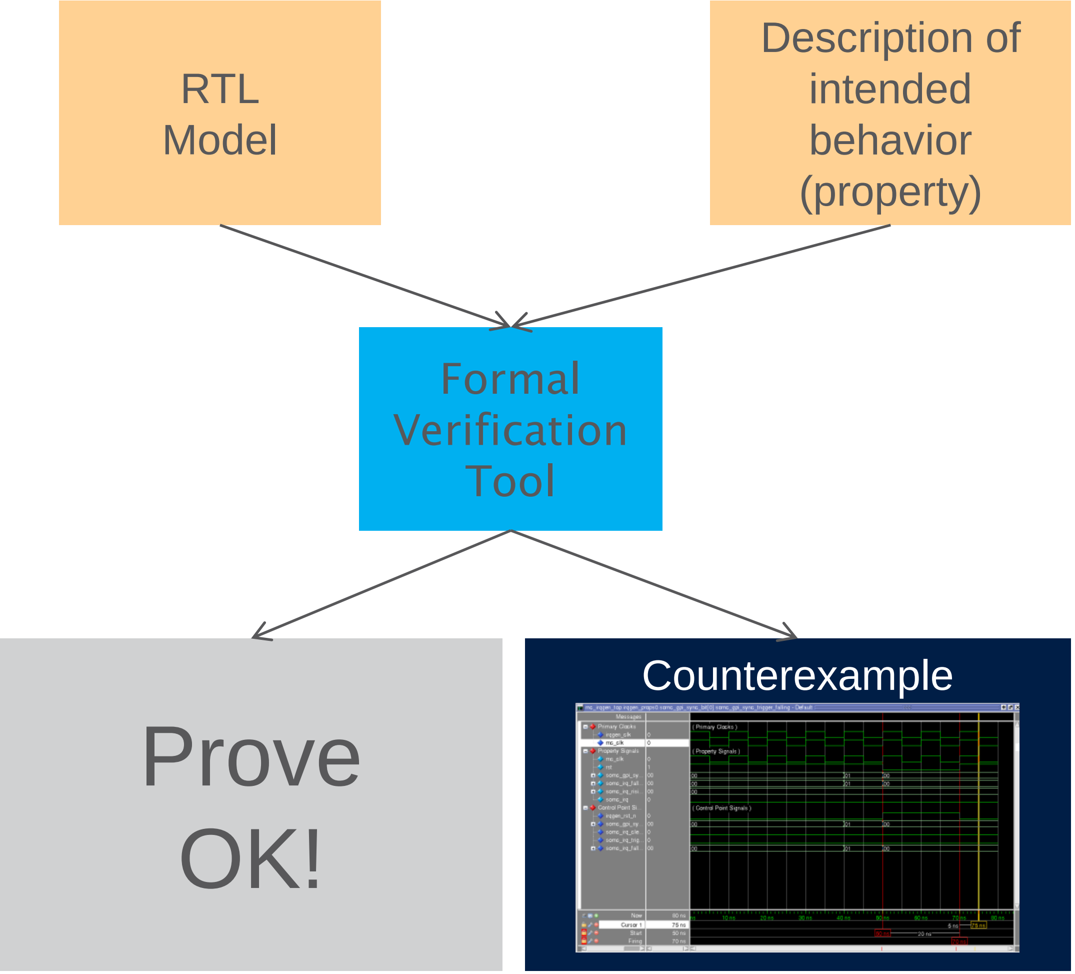


A	B	Y	Y’
0	0	0	0
	1	0	1
1	0	0	1
	1	1	1

Simulation :
Only limited combinations are examined.
=> bug oversight possibility

Formal :
All possible combinations are examined against the reference

FORMAL VERIFICATION USAGE



RTL Model

Description of intended behavior (property)

Formal Verification Tool

Prove OK!

Counterexample

FORMAL VERIFICATION VS SIMULATION

- **Simulation:**
  A huge number of real life scenarios are applied to DUT and the simulated response is compared against a golden result (in whatever form).

  A misbehavior results in a comparison mismatch!

- **Formal Verification:**
  Intended behavior of DUT is described with *properties* & the formal tool checks, that the model of the DUT obeys this behavior in **every possible** way. This can be considered as doing all possible simulations and filter out traces which do not satisfy the proof.

  A misbehavior results in a counterexample!

FORMAL VERIFICATION VS SIMULATION (CONT)

Criterion	Formal	Simulation
Testbench Creation	no testbench needed, just write constraints and properties	complex UVM testbench
Runtime	Typically faster	Typically slower
Exactness	• **100% coverage** feasible; • formal finds complex issues, which were not thought of	• Depends on skills of verifier, complexity of DUT and runtime spent • Bug detection subject to random
Applications	Applications possible, which are difficult to realize with simulation, e.g. • Connectivity check • check that an access to one register does not affect another one • cover point reachability	
Capacity	restrictions w.r.t. design complexity AND trace length	almost unlimited

FORMAL VERIFICATION VS SIMULATION (CONT)

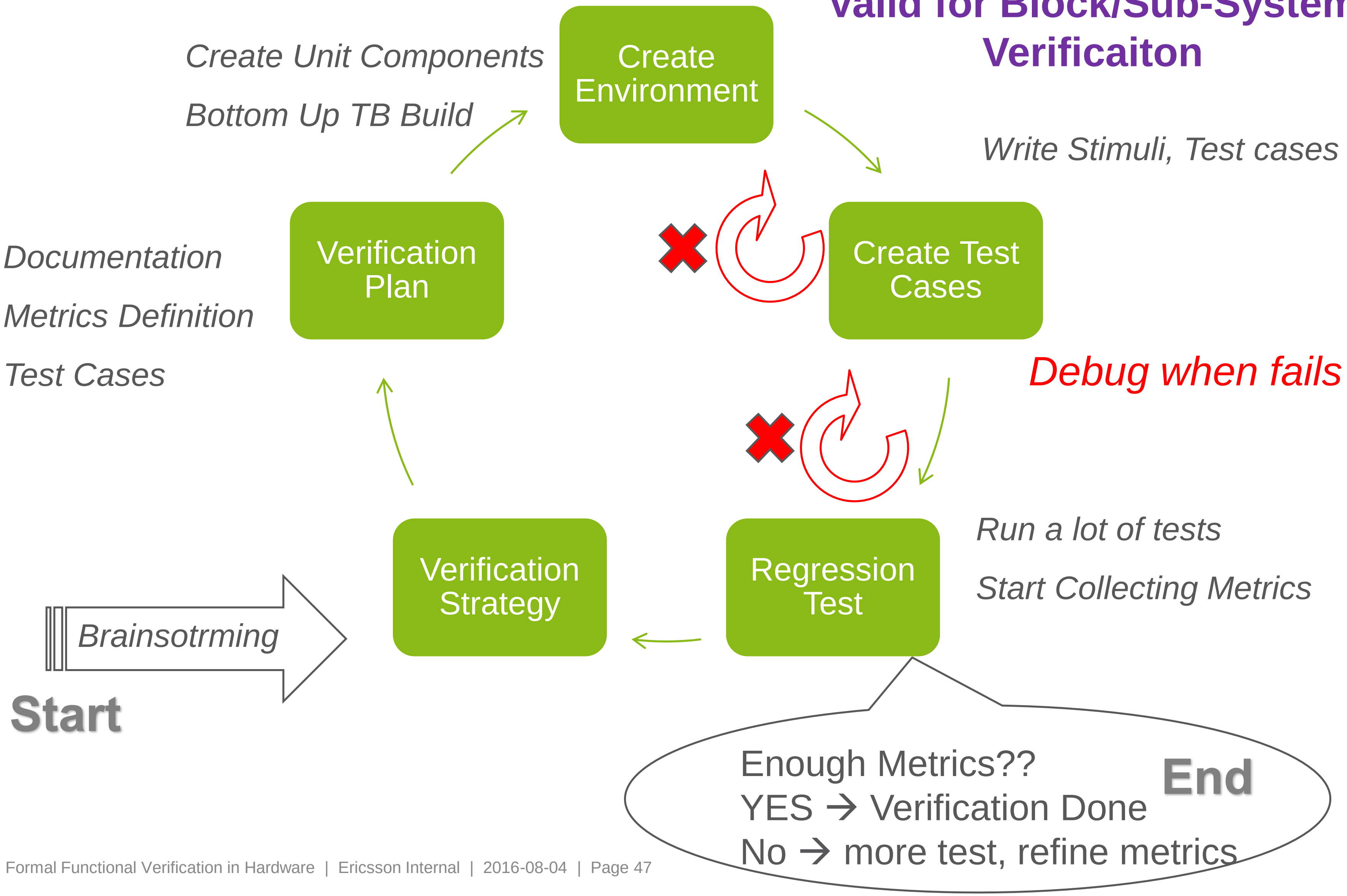
Criterion	Formal	Simulation
Predictability	It may happen that it shows after some effort, that formal is not applicable for a problem => experience needed to judge beforehand	• almost no limitations • high predictability
Skills needed	• *Thorough* design understanding to decipher counterexamples • SVA, PSL • formal tool handling • strong logical thinking *“your brain has to compete against a mathematical tool trying to outsmart you in all possible ways”*	• Design undestanding • Simulator handling • SV • UVM • Constraint Random
Costs	Formal tool license typically more expensive than simulator	Cheaper than Formal tool

Conclusion: both verification techniques have their weaknesses and strengths. The best approach is to leverage them wherever they fit better

=> **potential users need to be able to judge which technique to use for a particular job**

HW VERIFICATION PROCESS

Valid for Block/Sub-System Verificaiton



Brainsotrming

Start

Verification Strategy

Documentation

Metrics Definition

Test Cases

Verification Plan

Create Unit Components

Bottom Up TB Build

Create Environment

Write Stimuli, Test cases

Create Test Cases

Debug when fails

Regression Test

Run a lot of tests

Start Collecting Metrics

Enough Metrics??
YES → Verification Done
No → more test, refine metrics

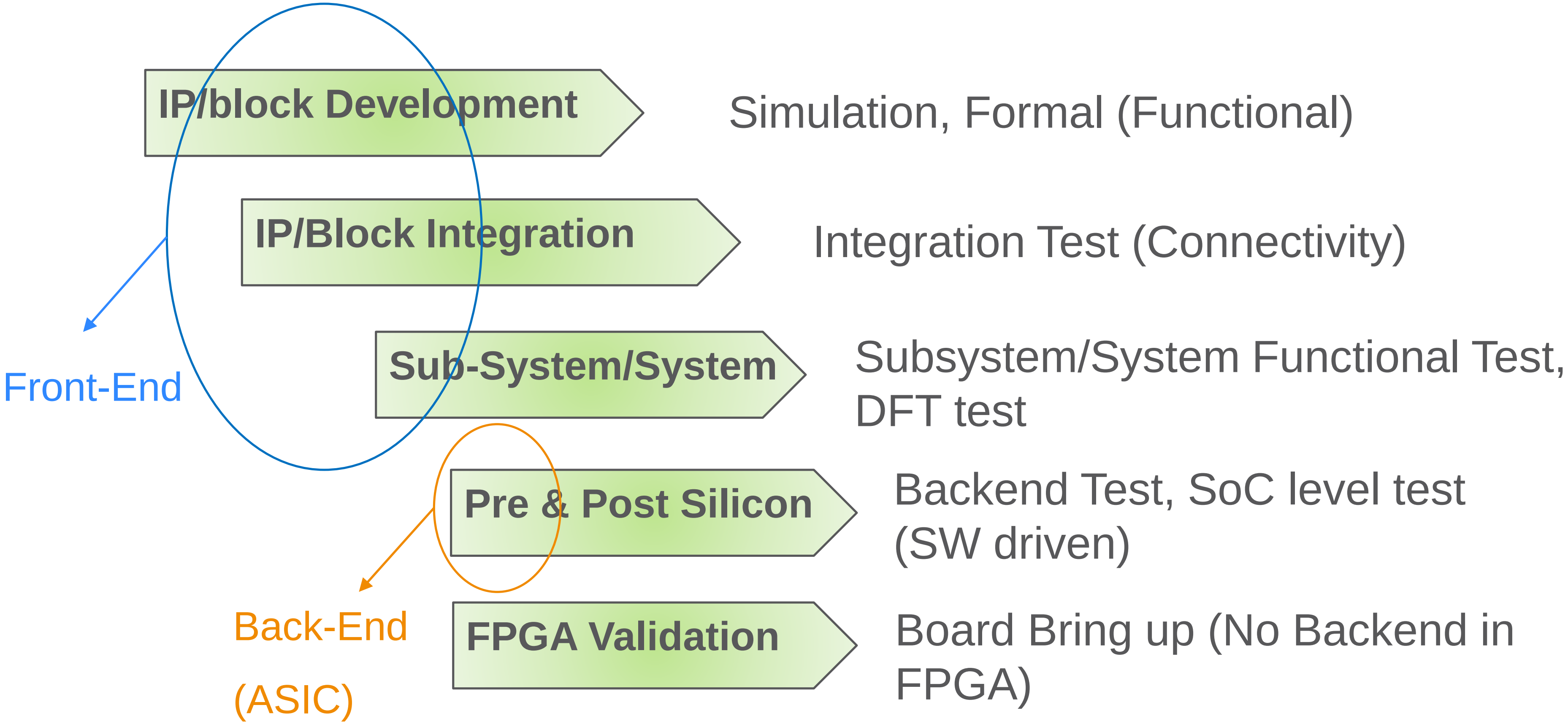
End

RECAP

THE VERIFICATION STAGES

VERIFICATION STAGES

› ASIC/FPA Development flow & corresponding verifications

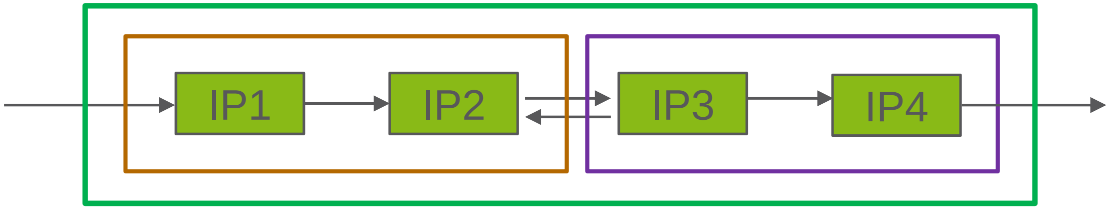

IP/block Development
Simulation, Formal (Functional)
IP/Block Integration
Integration Test (Connectivity)
Sub-System/System
Subsystem/System Functional Test, DFT test
Front-End
Pre & Post Silicon
Backend Test, SoC level test (SW driven)
Back-End
(ASIC)
FPGA Validation
Board Bring up (No Backend in FPGA)

INTEGRATION TEST

› Verify that each block/IP is connected in the system as it is defined → Connectivity Check

  – source PIN is connected to destination PIN

  – IP1 → IP2 → Subsystem 1 → Subsystem 2 (IP 3 → IP 4) → System/SOC



› Done in either Simulation or Formal

SYSTEM LEVEL FUNCTIONAL TEST

- Simulate the higher level scenario in real use case.
  - System Boot up
  - Program the registers in HW blocks
  - Clock/Reset Test etc
- Done in SW like test environment.
  - Massive environment (whole Chip), need emulation tool to run the test faster.
  - SW driven Test, e.g. build SW like test driver. (unlike HW IP level test where we consider too many details)
- System Knowledge required
  - birds eye view needed.

HW VERIFICATION PROCESS RECAP

- A lot of collaboration needed
  - Between stakeholders (Designer, Verification Architect, System Architect)
  - Communication is important
- Long Term Impact Analysis
  - Try to predict consequence of the action → Example, no quick fix
  - Do not complicate things too much.
- RISK management (Will we finish on right time, good quality)
  - Time == money (Good Estimation, Deadline, Estimated vs Actual Time)
  - Quality (No late bug should be found)

MISCELLANEOUS

DESIGN & VERIFICATION ENGINEER PROFILE (1/2)

- Technical Aspects
  - Overview of embedded architecture
  - Overview of Analog and digital signal processing
  - Good logical and analytical skill
  - Good at SW Programming concept
  - Very Good at one programming language
- Non-technical Aspects
  - Endurance (do not give up)
  - Intention to have the Birds eye view
  - Very good at communication & colaboration skill
  - Good at self management
  - Knows how to handle pressure
  - Always learning attitude, embrace changes

DESIGN & VERIFICATION ENGINEER PROFILE (2/2)

- Programming Skill (As a fresh graduate)
  - C (Any Functional Programming skill, very good at it) → Must
  - Object Oriented Programming (OOP like, C++/JAVA) → Good to have.
  - VHDL/Verilog (Beginner level) → Must
  - System Verilog (Basic) → Good to have
  - UVM, e.g. Universal Verification Methodology → Wish, definitely a plus point when you know the overview.
- Any Skill can be developed if individual has the right attitude and passion to win.

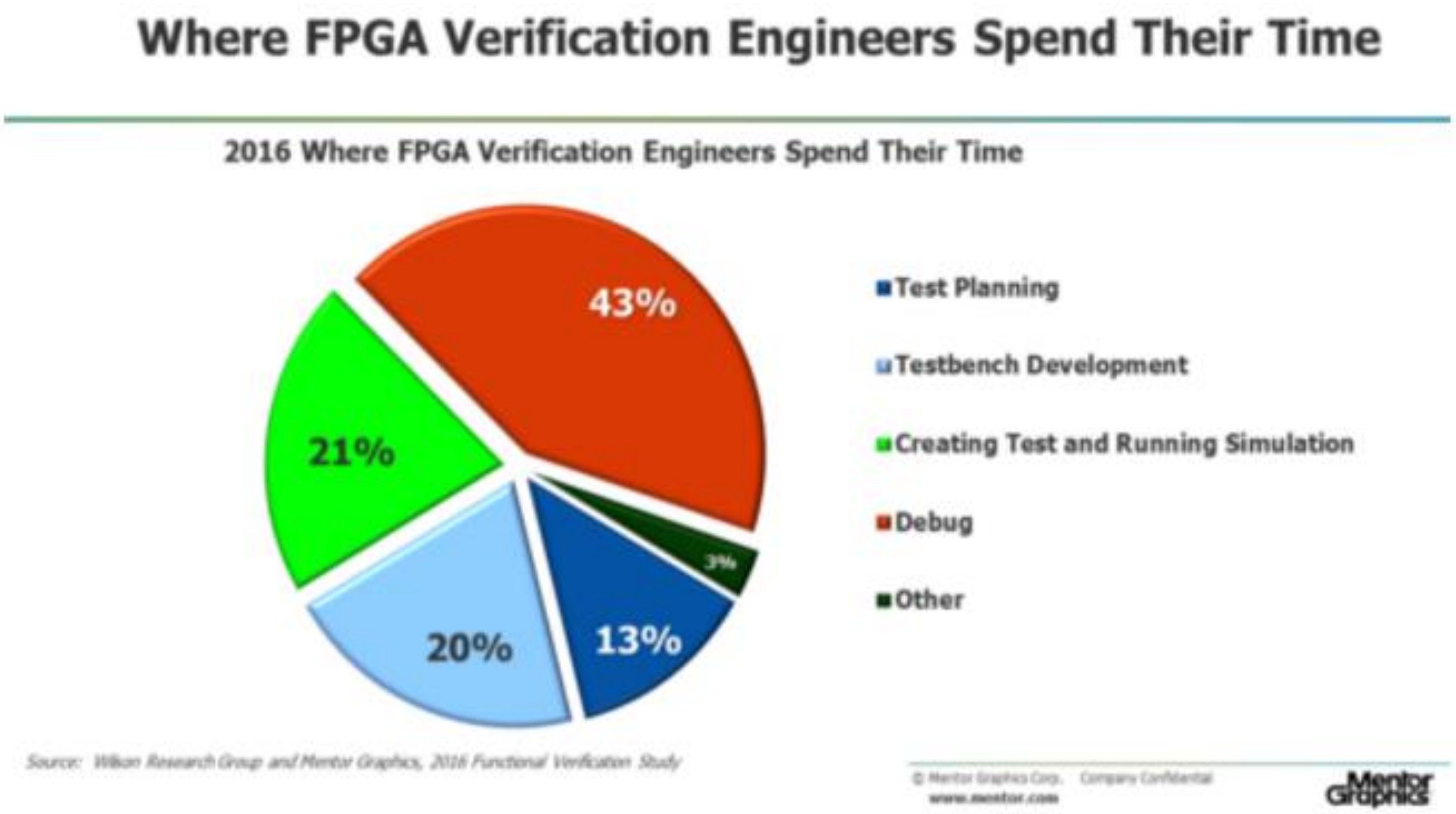
DAYS OF VERIFICATION ENGINEER (1/2)

- Initial Phase (Beginning of IP/Block Development)
  - Prestudy the Implementation Proposal, System Documentation
  - Meeting with designers, system architects
    - Extract the system requirements, block requirements
  - Block Verification Strategy Proposal
    - Review & Refine
- Environment Build Phase
  - Verification Specification finalize
  - Start Coding Verification Components
  - Hook up the design into the TB

DAYS OF VERIFICATION ENGINEER (2/2)

- Execution Phase (Beginning of IP/Block Development)
  - Create Test Case & Debug
    - Failure → Cross check with Designer, System Architect
    - Pass → Create regression suit.
  - Add more test cases (according the the test plan)
    - Test Cases passing → Coverage Model create & hook up
- Closing Phase
  - Run Regression & Debug
  - Collect Coverage Metrics
    - When staisfactory metrics achieved → test done
  - Create Verification Report and Review

WHERE MOST TIME IS SPENT??



Where FPGA Verification Engineers Spend Their Time

2016 Where FPGA Verification Engineers Spend Their Time

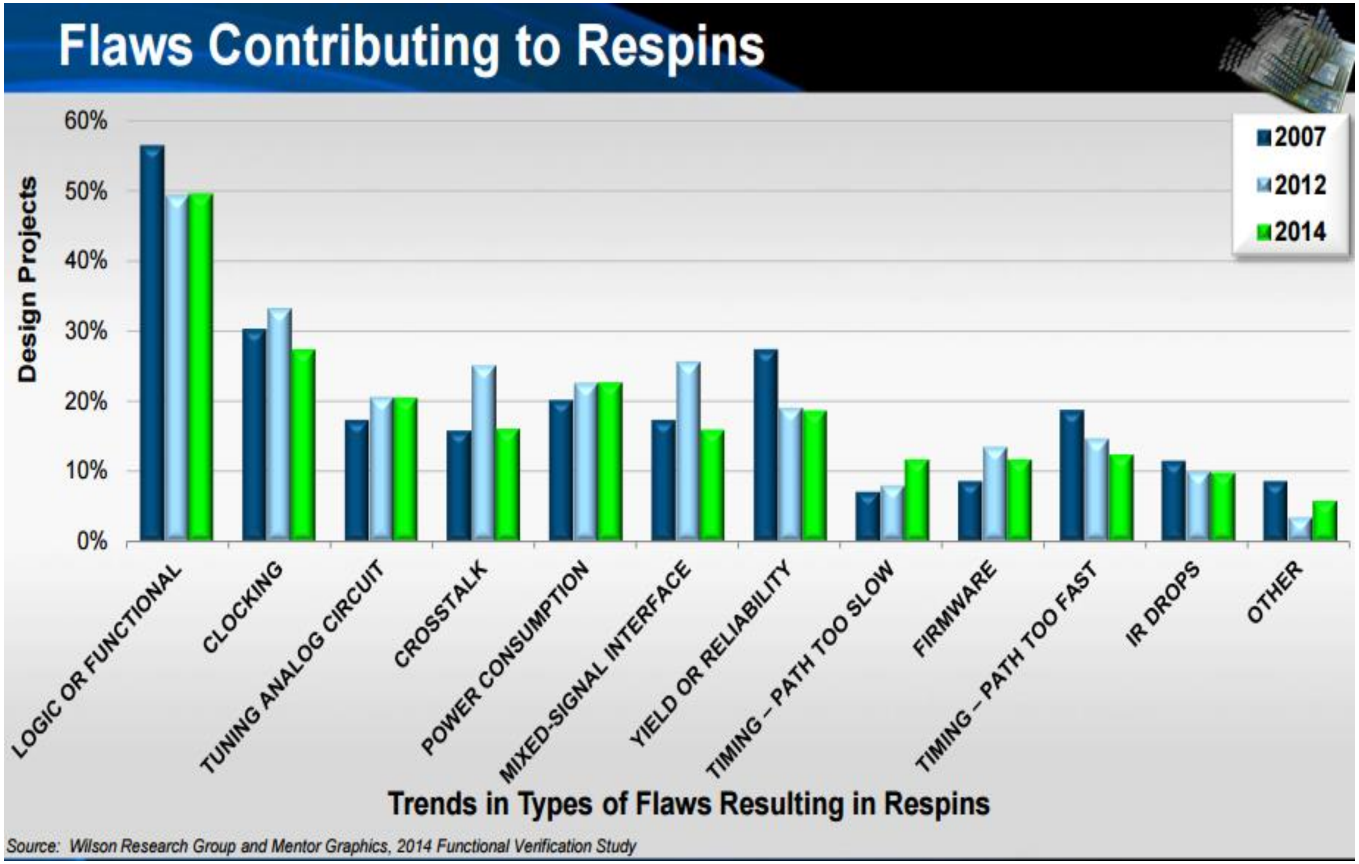
Source: Wilson Research Group and Mentor Graphics, 2016 Functional Verification Study

© Mentor Graphics Corp. Company Confidential
www.mentor.com

Mentor Graphics

- Debug is the critical & time consuming task.
  - Debug if Error is caused by Design or by the TB itself.

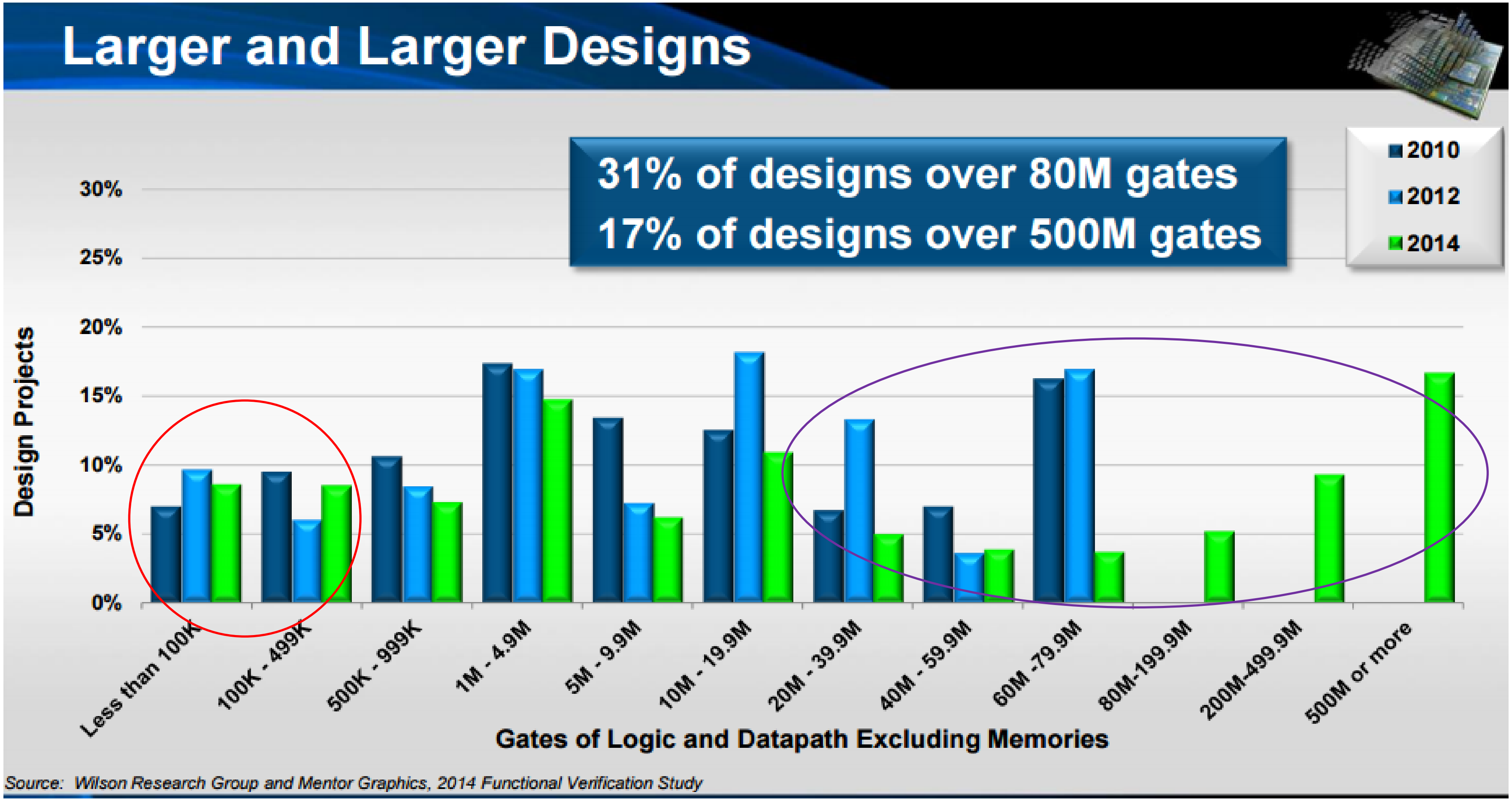
FUTURE IN VERIFICATION



Source: Wilson Research Group and Mentor Graphics, 2014 Functional Verification Study

- 50% Error comes from Functional/logic error
  - BUG escape to production
  - Verification hole (Scope of Verification Improvement)

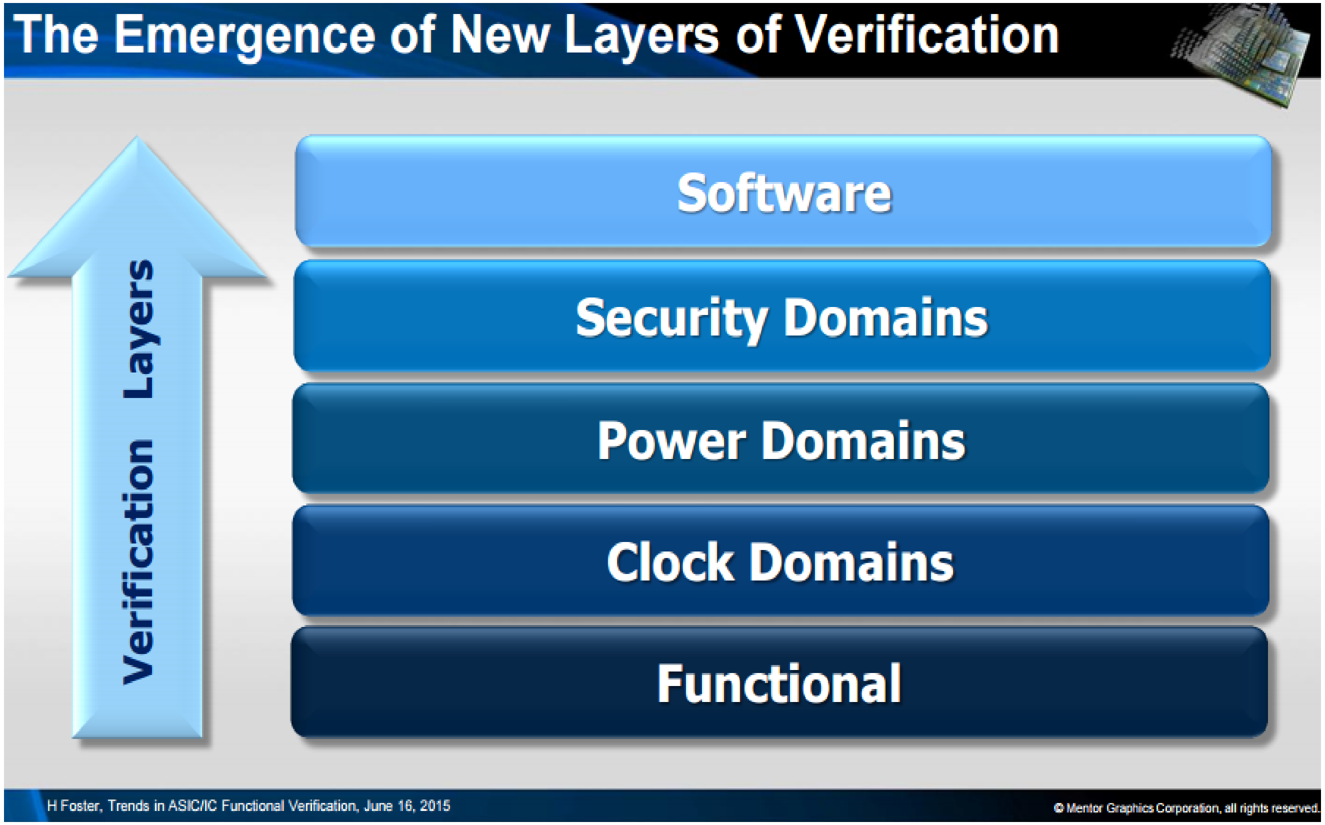
BIG DESIGNS, MORE TASK



Source: Wilson Research Group and Mentor Graphics, 2014 Functional Verification Study

- Big Design –> More bugs → More verification
- Small design → More integration testing in IOT

MY OBSERVATION (1/2)



The Emergence of New Layers of Verification

H Foster, Trends in ASIC/IC Functional Verification, June 16, 2015

© Mentor Graphics Corporation, all rights reserved.

- New verification domains are emerging due to
  - Security (prevent from hacking)
  - Low Power Consumption
  - More functionality but verify early

Verification will keep growing in next 15-20 years

MY OBSERVATION (2/2)

- As a verification engineer you can transform career in different directions
  - Technical
    - Verification Methodologist/Architect
    - System Architec (Having the brids eye view)
  - Embedded SW development
    - Strong background in HW development always a plus point
  - Non-Technical
    - Project Management
      - complexity handling ability will help you a lot
      - Communication & Collaboration skill will help

HW CAREER IN ERICSSON

- Why to choose ericsson over others
  - Communication industry will grow further
    - More Robust & complicated infrastructure will be needed than ever before.
      - New services, new HW → More new developement
  - We know how to build competence from scratch.
    - Strong in methodology, we have our tailored Way of Working.
  - True global company (no others like us in this region)
    - Presence in 180 countries.
  - Healthy work culture, open communication.

Q & A

REFERENCE (GOOD READING)

- Verification Academy
  - Know the ins and outs on verification, training
    - https://verificationacademy.com/

- Open source simulator with verification code compiler, debugger
  - https://www.edaplayground.com/
  - Supports most of the language and verification framework

- Introdution to UVM (Universal Verification Methodology)
  - http://www.doulos.com/knowhow/sysverilog/uvm/tutorial_0/
  - https://verificationacademy.com/courses/introduction-to-the-uvm

ERICSSON