1. Introduction

The workshop aimed to explore the role of software engineering within the Integrated Circuit (IC) design cycle. This is an interdisciplinary field where hardware design and software engineering intersect to create functional, efficient, and reliable ICs. Participants gained insights into the methodologies, tools, and best practices employed in the development and testing of software solutions used in IC design.

2. Objectives of the Workshop

The main objectives of the workshop were:

- To understand the various stages of the IC design cycle.
- To explore the integration of software engineering practices within each stage.
- To analyse the tools and technologies that bridge software and hardware in IC development.
- To investigate the challenges and best practices in software development for hardware design.

S.E.A College of Engineering and Technology

Affliated to VTU, Belagavi and Approved by AICTE, New Delhi and Accreditated by NAAC

Ekta Nagar, Near Ayyappa Nagar Circle, Devasandra Main Road, K.R.Puram ,Bangalore-49

Department of Computer Science & Engineering

INSTITUTION'S INNOVATION COUNCIL (Ministry of HRD Initiative)

Organizes

Workshop on

"SOFTWARE ENGINEER IN IC DESIGN CYCLE"

Speakert : MAHENDRA B M

TECH LEAD VLSI WING UST GLOBAL ,MALAYSIA

In Association with IQAC, IIC

On 9th November 2024

at 9.30AM

3.Overview of IC Design Cycle

The IC design cycle is a complex and multi-step process, involving both hardware and software components. The typical stages are:

1. **Specification and Design Requirements**:
   At this stage, the specifications for the IC are defined. Software tools such as specification management systems help capture and validate these requirements.
2. **High-Level Design**:
   High-level algorithms and architectures are designed for the IC. Simulation software (e.g., MATLAB, Simulink) helps software engineers model and test these designs before moving forward.
3. **RTL (Register Transfer Level) Design**:
   At this level, designers create a hardware description of the IC using hardware description languages like Verilog or VHDL. Software tools are used to write, test, and simulate these descriptions.
4. **Synthesis and Verification**:
   Synthesis involves converting RTL into gate-level representations. Software tools like Synopsys Design Compiler and Cadence Genus are used for synthesis. Verification tools ensure that the design behaves as expected.
5. **Physical Design**:
   The layout of the IC is created, and physical constraints are considered. Software plays a major role in designing and verifying the physical structure of the IC.
6. **Fabrication and Testing**:
   After the design is finalized, the IC is fabricated. Post-fabrication, testing tools help validate the IC's functionality and performance.
7. **Production and Deployment**:
   The IC is ready for mass production and deployment. Software tools ensure the production process is efficient and consistent.

Day 1: Introduction to IC Design and Software Engineering Integration

- **Session 1: Overview of the IC Design Cycle**
  - Key stages of IC design (from specification to production).
  - Overview of hardware and software roles in IC development.
  - Basic terminology in hardware design (RTL, VHDL, Verilog, SoC, etc.).
- **Session 2: The Role of Software Engineering in IC Design**
  - Software tools used at each stage of the IC design cycle (specification, simulation, verification, testing).
  - Software engineering's impact on hardware design through automation and validation.
  - Case studies on how software engineers work alongside hardware engineers.
- **Session 3: Introduction to Key Software Tools in IC Design**
  - Overview of CAD tools (e.g., Synopsys, Cadence, Mentor Graphics).
  - Simulation software for hardware validation (e.g., SPICE, ModelSim, Vivado).
  - Embedded software tools (e.g., IAR Embedded Workbench, Keil).
  - Hands-on demonstration of a simulation or design tool.
- **Session 4: Design and Verification with Software-Driven Approaches**
  - Introduction to hardware description languages (e.g., Verilog, VHDL).
  - Basic design examples in RTL and using software for validation.
  - Introduction to verification methodologies (SystemVerilog, UVM)

Day 2: Advanced Topics in IC Design and Software Integration

- **Session 1: High-Level Design and System Modeling with Software**
  - Introduction to high-level design tools like MATLAB/Simulink.
  - Software modeling for complex systems (e.g., digital, analog, mixed-signal designs).
  - Simulation-driven development: How software engineers model and test ICs before hardware implementation.
  - Hands-on activity: Creating a basic system model in Simulink.
- **Session 2: Synthesis and Optimization of IC Designs Using Software Tools**
  - Understanding the synthesis process (converting RTL to gate-level design).
  - Using software tools to optimize IC design for power, area, and timing (e.g., Synopsys Design Compiler).
  - Hands-on demonstration: Running an RTL design through a synthesis tool.
- **Session 3: Physical Design and Layout with Software Support**
  - Introduction to physical design tools (e.g., Cadence Virtuoso, Synopsys IC Compiler).
  - The role of software in floor planning, routing, and chip layout.
  - Physical verification and checking for design rule violations.

- Case study: Optimizing the physical design for a specific chip.
- **Session 4: Software and Hardware Co-Design**
    - Importance of co-design in System-on-Chip (SoC) development.
    - How software engineers and hardware designers collaborate on co-designing software and hardware for performance and efficiency.
    - Tools and techniques for hardware-software co-simulation (e.g., Co-simulation, FPGA prototyping).
    - Interactive exercise: Designing a simple embedded system with both hardware and software component

Day 3: Testing, Validation, and Future Directions

- **Session 1: Verification and Validation in IC Design**
    - Techniques for verifying hardware designs using software (functional verification, timing verification).
    - Introduction to automated testing frameworks (e.g., UVM, formal verification).
    - Post-silicon validation: How software engineers test the functionality of fabricated ICs.
    - Hands-on: Running a simple verification test using UVM/System Verilog.
- **Session 2: Challenges in Post-Silicon Testing and Debugging**
    - The complexities of post-silicon testing and how software engineers address them.
    - Using software-driven tools to debug hardware issues (e.g., JTAG, boundary scan).
    - Challenges in signal integrity, timing, and debugging issues.
    - Case study: Post-silicon debugging of a complex chip.
- **Session 3: Real-World Applications of Software in IC Design**
    - Exploring modern trends in IC design and the role of software engineering (e.g., AI/ML chips, 5G, automotive ICs).
    - Software's role in the design of cutting-edge technologies (e.g., IoT, autonomous systems).
    - How software tools are evolving to meet the needs of modern IC designs.
    - Guest speaker: Industry expert discussing the future of IC design and software engineering.
- **Session 4: Future Trends and Closing Remarks**
    - The future of software engineering in IC design: AI, machine learning, and automation in hardware design.
    - Emerging trends in hardware-software co-design and integrated system modeling.
    - Final thoughts on skill-building for software engineers in the IC design space.
    - Networking and closing remarks.

Key Learnings & Conclusion

Workshop is conducted for 5th semester students ,The workshop underscored the importance of software engineering in the IC design cycle. As IC designs become more complex, the role of software engineering continues to expand, especially in simulation, verification, and optimization. The integration of advanced software tools into the IC design cycle is essential for achieving high performance, efficiency, and scalability in modern ICs.

Students left the workshop with a deeper understanding of how software engineering complements hardware design and the key tools and methodologies used to ensure a successful IC design process.

Glimpse