Introduction to the **Design Verification Unit** **COMS30026**

(COMS30025 with coursework)

Kerstin Eder

Trustworthy Systems Lab

University of BRISTOL

Department of COMPUTER SCIENCE

Welcome to Design Verification

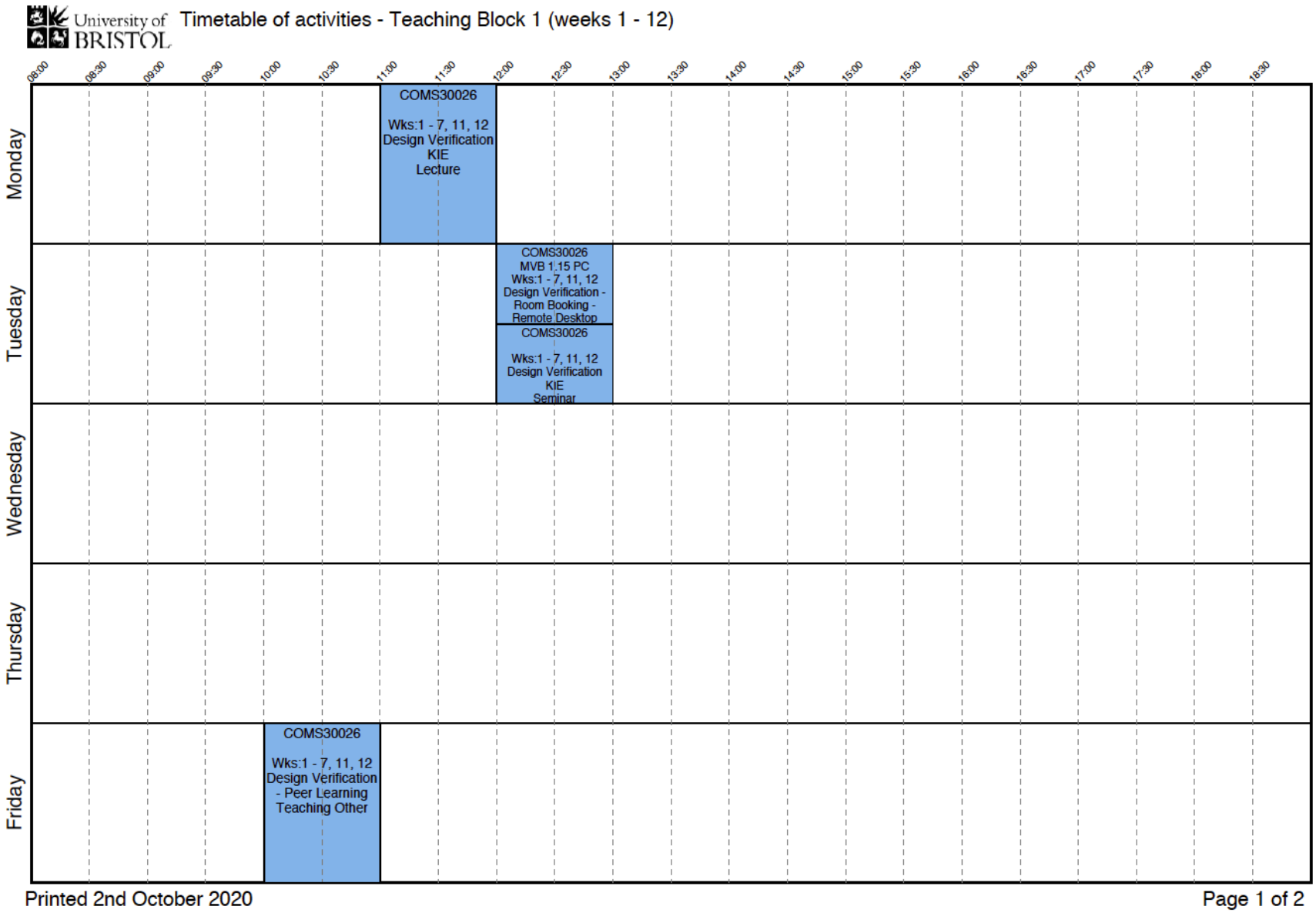
- Lecturer and Unit Director
  - Kerstin EDER
  - Department of Computer Science
- Lecture notes, exercises and additional information are available at uobdv.github.io/Design-Verification/
- Recordings of the lectures will be made available on a weekly basis via Blackboard
- Comments and feedback are always welcome

Design Verification Unit Details

- Lecture times (weeks 1-7 and weeks 11-12)
  - Please check your timetable, at the moment:
  - Monday (1h) 11:00 online synchronous session (Lecture)
  - Fridays (1h) 10:00 Peer learning sessions

- Practical Work (weeks 1-7 and weeks 11-12)
  - Tuesday (1h) 12:00 online lab-based seminar with remote desktop bookings
  - Teaching Assistant: Xuan Zheng

Timetable


University of BRISTOL
Timetable of activities - Teaching Block 1 (weeks 1 - 12)
08:00
08:30
09:00
09:30
10:00
10:30
11:00
11:30
12:00
12:30
13:00
13:30
14:00
14:30
15:00
15:30
16:00
16:30
17:00
17:30
18:00
18:30
Monday
COMS30026
Wks:1 - 7, 11, 12
Design Verification
KIE
Lecture
Tuesday
COMS30026
MVB 1.15 PC
Wks:1 - 7, 11, 12
Design Verification -
Room Booking -
Remote Desktop
COMS30026
Wks:1 - 7, 11, 12
Design Verification
KIE
Seminar
Wednesday
Thursday
Friday
COMS30026
Wks:1 - 7, 11, 12
Design Verification
- Peer Learning
Teaching Other
Printed 2nd October 2020
Page 1 of 2

Design Verification Assessment

- Assessment
  - COMS30026 will be assessed in the January exam period (exam only)
  - COMS30025 will be assessed by coursework (only)

Literature and Study Resources

- **Writing Testbenches: Functional Verification of HDL Models** by Janick Bergeron. Second Edition, Kluwer, 2003.
- **Comprehensive Functional Verification** by Bruce Wile, John Goss and Wolfgang Roesner. Elsevier, 2005.
- verificationacademy.com
- In addition:
  - Lecture slides and on-line tutorials on unit web page
  - On-line documentation of ModelSim/Questa Simulator and SpecMan Elite
  - Watch the unit web page for further supplementary literature.

[**Credits:** Parts of the lecture notes contain material from the book “Comprehensive Functional Verification” by Bruce Wile etal, the book ”Writing Testbenches: Functional Verification of HDL Models” by Janick Bergeron, the book ”The Verilog Hardware Description Language” by Donald Thomas and from lecture slides developed at IBM (by Avi Ziv and Jaron Wolfstal), the University of Pittsburgh, Penn State University, North Carolina State University and Ohio State University. The HDL for the assignments has been developed at IBM.]

What is this unit about?

Aim: To familiarise you with the state of the art in Design Verification, and to give you the **technical background** plus some of the **practical skills** expected from a professional Design Verification Engineer.

- Pre-/Co-requisites: programming experience

On successful completion of this unit, you will be able to:

- understand the complexities and limits of verification;
- carry out functional verification and determine its effectiveness;
- set appropriate verification goals, select suitable verification methods and assess the associated risks;
- compile a verification plan that fits into the flow of a design project.

Unit Outline

Lecture Topics

- Introduction: What is Verification? What is a Testbench?
- Verification Flow and Tools including basic Verilog HDL coding
- Verification cycle, methodology and plan including coverage
- Simulation-based Verification: Stimulus Generation and Checking
- Assertion-based Verification (ABV)
- Advanced Testbench Design Methodology with SpecMan Elite
- (Functional Formal Verification and Property Checking)

Practical work

- Exercise 1: Evita Verilog interactive tutorial (do @ home asap)
- Exercise 2: Introduction to the ModelSim/Questa Simulator
- Practical 1, weeks 2-5: Verification of calculator design with ModelSim
- Exercise 3: How to collect Code Coverage with ModelSim/Questa
- Exercise 4: Introduction to SpecMan Elite
- Practical 2, weeks 6-7 & 11-12: Advanced testbench design with SpecMan Elite and formal verification with JasperGold

What is Design Verification?

What is Design Verification?

"Design Verification is the process used to gain confidence in the correctness of a design w.r.t. the requirements and specification."

Types of verification:

- Functional verification
- Timing verification
- ...
- What about performance?

Verification vs Validation

- Verification:
  - Confirms that a system has a given input / output behaviour, sometimes called the **transfer function** of a system.
- Validation:
  - Confirms that the system's transfer function results in the intended system behaviour when the system is employed in its target environment, e.g. as a component of an embedded system.
- Validation is sometimes used when verification is meant.

Annex A
(informative)

The Role of Testing in Verification and Validation

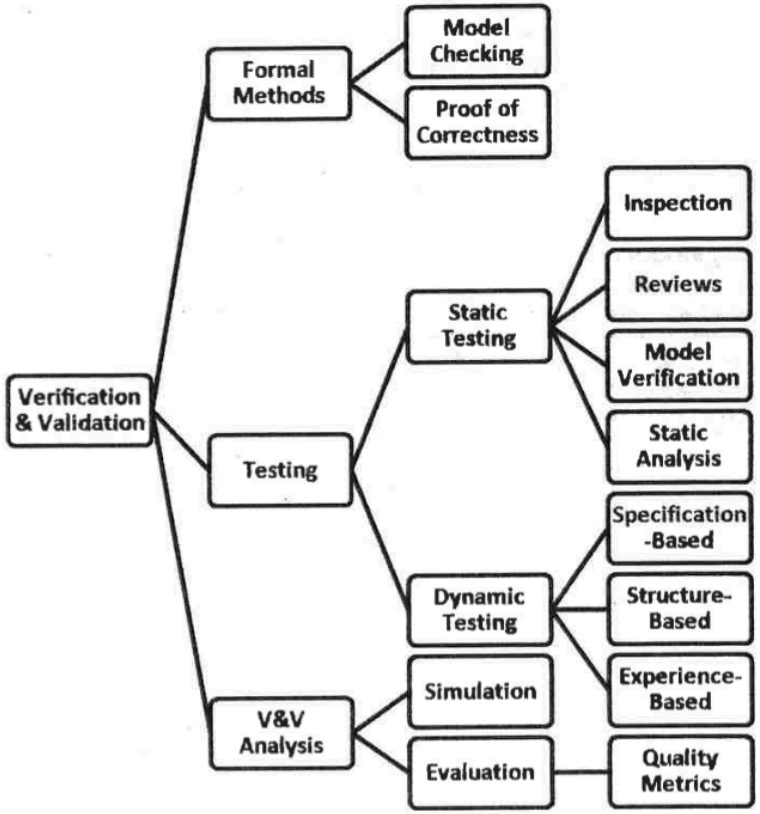


Figure 11 — Hierarchy of Verification and Validation activities

Figure 11 defines the complete nature of verification and validation (V&V) activities. V&V can be done on system, hardware, and software products. These activities and planning are defined and refined in IEEE 1012 and ISO/IEC 12207. Much of V&V is accomplished by testing. The ISO/IEC 29119 standard addresses the Dynamic and Static software testing (directly or via reference), thus covering parts of this verification and validation model. ISO/IEC 29119 is not intended to address all the elements of the V&V model, but it is important for a tester to understand where they fit within this model.

DRAFT INTERNATIONAL STANDARD ISO/IEC DIS 29119-1

ISO/IEC JTC 1 | Secretariat: **ANSI**

Voting begins on **2012-10-09** | Voting terminates on **2013-01-09**

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

Software and systems engineering — Software testing —

Part 1:
Concepts and definitions

Ingénierie du logiciel et des systèmes — Essais du logiciel —
Partie 1: Concepts et définitions

ICS 35.080

To expedite distribution, this document is circulated as received from the committee secretariat. ISO Central Secretariat work of editing and text composition will be undertaken at publication stage.

Pour accélérer la distribution, le présent document est distribué tel qu'il est parvenu du secrétariat du comité. Le travail de rédaction et de composition de texte sera effectué au Secrétariat central de l'ISO au stade de publication.

THIS DOCUMENT IS A DRAFT CIRCULATED FOR COMMENT AND APPROVAL. IT IS THEREFORE SUBJECT TO CHANGE AND MAY NOT BE REFERRED TO AS AN INTERNATIONAL STANDARD UNTIL PUBLISHED AS SUCH.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

RECIPIENTS OF THIS DRAFT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.

© International Organization for Standardization, 2012
International Electrotechnical Commission, 2012

Next

- Recordings of lectures on
  - Introduction to design verification and
  - Verification Hierarchy

  will be uploaded to BB within the coming two days. Recordings will then be uploaded on a weekly basis.
- **Tasks for you:**
  - **Attend the lab session** with Xuan tomorrow to set up remote access to the EDA tools
  - **Review "*The limits of correctness".*** *SIGCAS Comput. Soc. 14,15, 1,2,3,4 (Jan 1 1985), 18–26.* DOI: https://doi.org/10.1145/379486.379512
    - *Identify the main lines of argument*
    - *Why does the author question the notion of "correctness"?*
    - *What are the two or three key take-away messages for you?*

Paper review

Brian Cantwell Smith. 1985. The limits of correctness.
SIGCAS Comput. Soc. 14,15, 1,2,3,4 (Jan 1 1985), 18–26.
DOI: https://doi.org/10.1145/379486.379512

THE LIMITS OF CORRECTNESS†

Brian Cantwell Smith*

1. Introduction

On October 5, 1960, the American Ballistic Missile Early-Warning System station at Thule, Greenland, indicated a large contingent of Soviet missiles headed towards the United States.[1] Fortunately, common sense prevailed at the informal threat-assessment conference that was immediately convened: international tensions weren't particularly high at the time. The system had only recently been installed. Kruschev was in New York, and all in all a massive Soviet attack seemed very unlikely. As a result no devastating counter-attack was launched. What was the problem? The moon had risen, and was reflecting radar signals back to earth. Needless to say, this lunar reflection hadn't been predicted by the system's designers.

Over the last ten years, the Defense Department has spent many millions of dollars on a new computer technology called "program verification" - a branch of computer science whose business, in its own terms, is to "prove programs correct". Pro-

What, we do well to ask, does this new technology mean? How good are we at it? For example, if the 1960 warning system had been proven correct (which it was not), could we have avoided the problem with the moon? If it were possible to prove that the programs being written to control automatic launch-on-warning systems were correct, would that mean there could not be a catastrophic accident? In systems now being proposed computers will make launching decisions in a matter of seconds, with no time for any human intervention (let alone for musings about Kruschev's being in New York). Do the techniques of program verification hold enough promise so that, if these new systems could all be proven correct, we could all sleep more easily at night? These are the questions I want to look at today. And my answer, to give away the punch-line , is no. For fundamental reasons - reasons that anyone can understand - there are inherent limitations to what can be proven about computers and computer programs. Although program verification is an important new technology, useful, like so many