Design Verification
COMS30026

(COMS30066 with coursework)

WEEKLY LIVE SESSION – W3

Kerstin Eder

Trustworthy Systems Lab

University of BRISTOL

Department of COMPUTER SCIENCE

Topics W1

- ✓ Introduction to DV
- ✓ Verification Hierarchy
- ✓ Fundamentals of Simulation-based Verification
  - Driving & Checking

- **Lab W1:**
  - get remote access to the EDA software
  - teach yourself Verilog ☺

Topics W2

- ✓Verification Tools
- ✓Hardware Design Languages
- ✓Verification Cycle, Methodology & Plan

- **Lab W2:**
  - Introduction to ModelSim/Questa
    - installed on linux lab machines
  - Work through mux testbench from Exercise 2

    https://uobdv.github.io/Design-Verification/

Task(s) – Week 2

- **Review the foretellix blog** *https://blog.foretellix.com/*

HOME ABOUT WHY THIS BLOG TERMINOLOGY TOPICS ALL POSTS

GPT-3 and verification

July 20, 2020

Summary: This post talks about GPT-3, a new Machine Learning (ML) system currently making waves in the ML community. It explains why GPT-3 is a big deal, and then considers the verification implications of such systems. One way to look at GPT-3 (and the even-bigger GPT-4, GPT-5 etc. which are sure to follow) is as ...

More

Search ...

Archives

Select Month

RV2020 Keynote by Lane Desborough on *The Physical Side of Cyber-Physical Systems*

https://youtu.be/QRknQBMK9LA?t=457

1. It is possible and valuable to cross domains
2. The future is already here, it just hasn't been evenly distributed yet. – William Gibson
3. Bad things can happen during mode transitions, when the state of the system is changing
4. Never forget that the physical side of cyber-physical systems involves energy
5. Manage variation using hierarchical, temporal decoupling
6. In cyber-physical systems, most any parameter can be a critical parameter … so manage them all carefully … if you don't manage change, change will manage you
7. "Open" Process Automation Systems are nearly impossible to comprehend
8. Interoperability is not a panacea
9. So simple there are obviously no errors, or so complex there are no obvious errors
10. Hardware, software, wetware, and the cyber-physical systems comprised thereof are different and should be treated as such
11. Systems: the bigger they are, the less frequent – but harder - they fall
12. Take on only as much complexity as you can manage – lives may be at stake
13. When our commercial reach exceeds our technical grasp, we must look for new approaches
14. Humans and computers are good at different things; improper task allocation creates problems
15. Automation changes the nature of use error: from acts of commission to acts of omission
16. The methods, tools, and processes to compose small, medium, and large systems are different
17. Emergent properties will emerge
18. The challenges with complexity have been known for a long time
19. That which you do not have does not cause problems
20. Complexity is easy to add hard to remove
21. Complexity adds cost, risk, and delay (and technical debt, and late cycle surprises)
22. All models are wrong, but some are useful – George Box
23. We build models to efficiently characterize what the system will do
24. Modeling is a means not an end; different kinds of modeling serve different purposes
25. One must be prepared to use as many characterizations methods as necessary

Topics W3

- ✓ Stimuli Generation Part I: Foundations
- ✓ Stimuli Generation Part II: Test Automation
- Checking

- **Lab W3:**
  - Introduction to **Practical 1** (Weeks 2-5)
    - Specification review
    - Debug calculator design using Modelsim/Questa
    - Example testbench on Blackboard

DESIGN VERIFICATION Practical 1:
Debugging a Calculator Design
(Weeks 2-5)

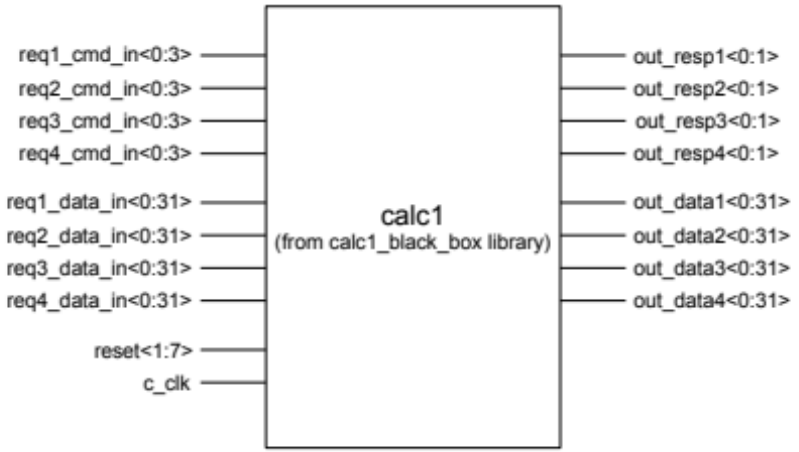
This practical requires you to find bugs in a calculator design. The calculator specification is given below. The design HDL code is not available for this practical. You are expected to use a black box verification approach.

For simulation with ModelSim, the calculator design has been provided in the pre-compiled library `calc1_black_box`. The library is available from BlackBoard as a ZIP file. BlackBoard also provides you with instructions on how to set up your workspace with this library. The top-level module of the calculator design is called `calc1`. Please use the example testbench file provided on BlackBoard to check that you have set everything up correctly.

Calculator Specification

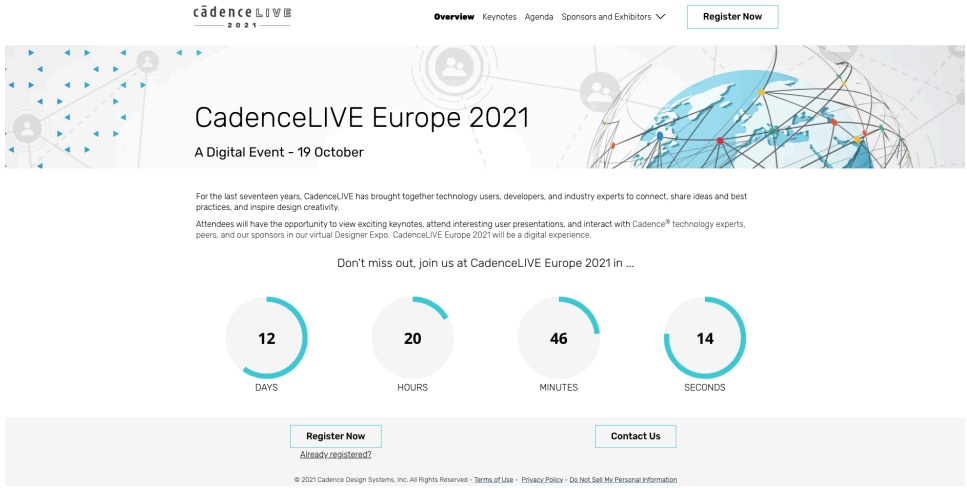
Input/Output Specification

The calculator has four commands: add, subtract, shift left and shift right. It can handle (but not process) four requests in parallel. All four requestors use separate input signals. All requestors have equal priority. Figure 1 shows the input output ports of the calculator.



Opportunities – Week 3

- ***CDN Live** 2021 is on 19 Oct 2021*
- *Registration is open at:*
  https://events.cadence.com/event/1220226a-f2b7-474c-b2a1-f08fdab24766/summary



- 2020 Wilson Research Group Verification Survey Results
  - Recording by Harry Foster on Blackboard

Challenge

Photo from
https://en.wikipedia.org/wiki/Edsger_W._Dijkstra

- Find out about the work of Edsger W. Dijkstra.
- In particular, find what he said about testing and its limitations.

Challenge

Photo from
https://en.wikipedia.org/wiki/Edsger_W._Dijkstra

- Find out about the work of Edsger W. Dijkstra.
- In particular, find what he said about testing and its limitations.

"Program testing can be used to show the presence of bugs, but never to show their absence!"

— Edsger W. Dijkstra

Four necessary and sufficient conditions for deadlock

System Deadlocks

E. G. COFFMAN, JR.

Pennsylvania State University, University Park, Pennsylvania

M. J. ELPHICK

University of Newcastle upon Tyne, Newcastle upon Tyne, England

A. SHOSHANI

System Development Corporation, Santa Monica, California

E. G. Coffman, M. Elphick, and A. Shoshani. 1971. System Deadlocks. ACM Comput. Surv. 3, 2 (June 1971), 67–78.

DOI: https://doi.org/10.1145/356586.356588

A problem of increasing importance in the design of large multiprogramming systems is the, so-called, deadlock or deadly-embrace problem. In this article we survey the work that has been done on the treatment of deadlocks from both the theoretical and practical points of view.

Key words and phrases: deadlocks, deadly embraces, system deadlocks, multiprogramming, interlock problems

CR categories: 4.10, 4.32

1 INTRODUCTION

One of the objectives of recent developments in operating systems—incorporating multiprogramming, multiprocessing, etc.—has been to improve the utilization of system resources (and hence reduce the cost to users) by distributing them among many concurrently executing *tasks*. In any operating system of this type the problem of deadlock must be considered. Requests by separate tasks for resources may possibly be granted in such a sequence that a group of

But if we assume that T_2 is in the middle of some file on disk, then, because of hardware or software limitations, a preemption of the disk from T_2 can be effectively tantamount to a sacrifice in the processing already accomplished by T_2. Under these circumstances, if T_1 and T_2 have no (temporary) alternative to waiting, they will be deadlocked and never able to proceed.

As shown by this example, deadlocks, or "deadly embraces" as E. W. Dijkstra has called them, can arise even though no single task requires more than the total resources

This offers you an opportunity to learn material beyond what has been taught in this unit.

Next

- Recordings of lectures for **Week 3:**
  - ✓ Stimuli Generation Part I: Foundations
  - ✓ Stimuli Generation Part II: Test Automation
  - ✓ Checking
  - uobdv.github.io/Design-Verification/ shows a **weekly schedule of topics** to watch BEFORE the next online session, ideally
  - Recordings are available from Blackboard unit page
- Tasks for you this week:
  - **Attend the lab session** with Xuan on Friday to get help with using ModelSim/Questa so you can do **Practical 1**
  - Submit issues found during your **Specification review** to Blackboard Discussion Forum

Questions


novel solutions
secure
reliable
full stack
graceful degredation
V&V
publications
verification
integrated
machine learning
by design
hardware
demonstrable
integrated
reliable
functional
confidence
formal
credibility
privacy
safe
research
robustness
function
design
security
energy
proof
hardware
robust
software
trustworthy
reliable
V&V
security
systems
V&V
safe
integrity
holistic
robotics
security
dependability
privacy
embedded
proof
safety
AI
correctness
implementation
transparent
simulation
V&V
intelligent testing
trusted
software dependable
expert
analysis
secure
verification
validation
Demonstrably trustworthy systems for reliable, secure computing.
https://www.bristol.ac.uk/tsl