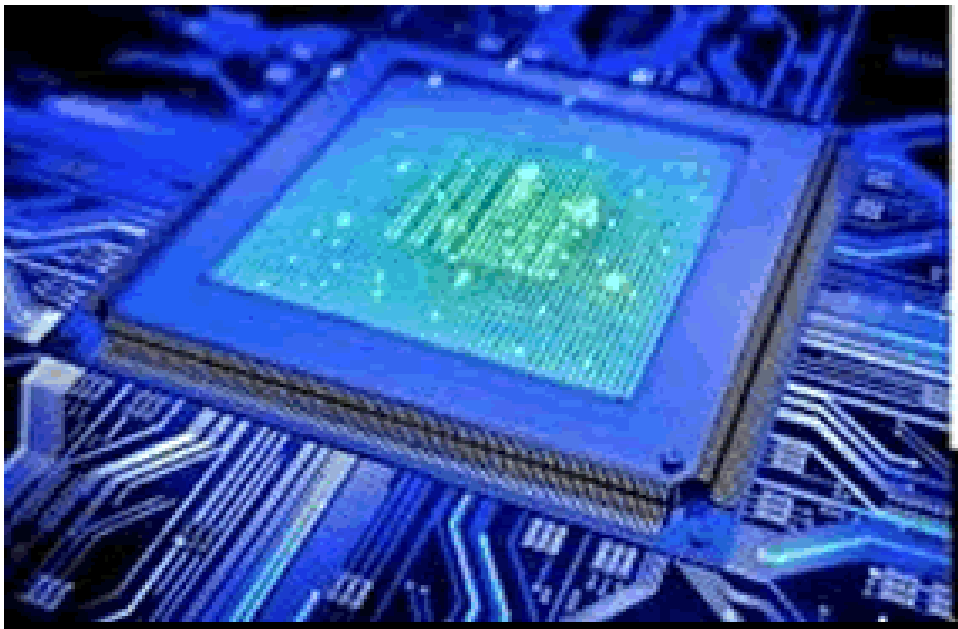
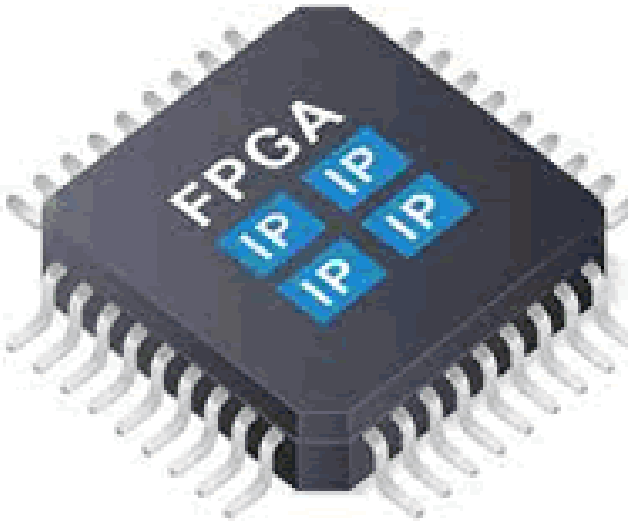


python™

Hardware Verification in Python

A Comprehensive Review

Mark Chen



HARDWARE VERIFICATION IN PYTHON

A Comprehensive Review

First Edition

Mark Chen

Angelia Technologies

www.ipcoredesign.net

Table of Contents

PREFACE

CHAPTER 00 FOREWORD

CHAPTER 01 WHY PYTHON?

- PYTHON IS PROBABLY THE LANGUAGE IN ACADEMIA
- PYTHON PROS AS PROGRAMMING LANGUAGE
  - MULTIPLE PROGRAMMING PARADIGMS
  - DYNAMIC TYPE SYSTEM
  - AUTOMATIC MEMORY MANAGEMENT
  - READABILITY AND MAINTAINABILITY
  - FASTER TO BUILD AND RUN
  - BROAD PYTHON ECOSYSTEM OF MODULES
  - PYTHON'S BROADER TALENT BASE
  - PYTHON'S WIDER RANGE OF APPLICATIONS WITH EASE
  - MANY INDUSTRY ACTORS SUPPORT AND USE PYTHON
- PYTHON CONS AS PROGRAMMING LANGUAGE
- VERIFICATION RELEVANT & MAIN PURPOSE
  - PURPOSE OF PYTHON VERIFICATION
  - PYTHON TO DEVELOP BRINGUP TESTS
  - USING PYTHON TO WRITE TESTS IN PYTEST
  - PROXY-DRIVEN TESTBENCHES WRITTEN IN PYTHON
  - EASIER TO CREATE TESTBENCHES FOR COMPLEX APPLICATIONS SUCH AS MACHINE LEARNING
  - PYTHON TO BOOST UVM FLOWS
  - CLIENT-SERVER STRUCTURE WITH CLIENT WRITTEN IN PYTHON
  - STUPID VERIFICATION TASKS
  - PYTHON FOR DESIGNERS WHO HATE THE VERIFICATION TEAM
  - LOAD TESTS DYNAMICALLY
  - CUT DOWN ON THE NUMBER OF MEMORY LEAKS
- HARDWARE DESIGN COMPANIES USING PYTHON FOR VERIFICATION

CHAPTER 02 HARDWARE VERIFICATION (TESTBENCHES) IN PURE PYTHON

- THE CODE MAC (MULTIPLY AND ACCUMULATE)
- THE VERILOG TESTBENCH
- THE GOLDEN MODEL IN PYTHON
- THE AUTOMATED TESTBENCH IN PYTHON
- THE ENTIRE PYTHON TEST SCRIPT WITH THE GOLDEN MODEL
- TEST RESULTS
- LINKS

CHAPTER 03 COCOTB: CO-ROUTINE AND CO-SIMULATION OF TESTBENCH IN PYTHON

INTRODUCTION
WHY COCOTB?
SOME COCOTB SPECIFIC KEYWORDS
FANCY PYTHON FEATURES
THE FRAMEWORK
HOW DOES COCOTB WORK?
HOW IS COCOTB DIFFERENT?
COCOTB IN PYTHON VS. UVM IN SYSTEMVERILOG
    PROPONENTS:
    OPPONENTS:
SIMULATOR INDEPENDENT
SIMULATORS TO WORK WITH COCOTB
LANGUAGE INDEPENDENT
CODE YOUR TESTS IN PYTHON
NO ADDITIONAL RTL CODE NEEDED
MANIPULATE SIGNALS INSIDE YOUR MODULE FROM PYTHON
INTERFACE BETWEEN SIMULATOR AND PYTHON WITH COROUTINES AND CO-SIMULATION
POST-SYNTHESIS SIMULATIONS
TRIGGERS
YIELDING MULTIPLE TRIGGERS
COROUTINES
FORKING COROUTINES
JOINING FORKED COROUTINES
COMMUNICATING WITH COROUTINES
COROUTINES AND CLASSES
MODIFYING THE HIERARCHY
INSTALLING COCOTB
THE SIMPLEST EXAMPLE OF COCOTB VERIFICATION
    THE VERILOG CODE
    THE COCOTB TEST BENCH
    THE CONTROL FILE (THE MAKEFILE):
    RUNNING:
LINKS AND LITERATURE

CHAPTER 04 COCOTB DERIVATIVE – PYUVM: PYTHON IMPLEMENTATION OF UVM USING COCOTB

INTRODUCTION
INSTALLATION
RUNNING FROM THE REPOSITORY
RUNNING THE SIMULATION
    THE PYUVM TESTBENCH FOR THE TINYALU EXAMPLE
IMPORTING PYUVM
THE ALUTEST CLASSES
THE ALUENV CLASS
THE SCOREBOARD
THE MONITOR
COVERAGE
DRIVER
THE ALU SEQUENCE
ALU SEQUENCE ITEM

TRANSACTION-LEVEL MODELING (TLM) 1.0 IN PYUVM
BLOCKING OPERATIONS
Blocking put
Blocking get
NON-BLOCKING OPERATIONS
Non-blocking put
Non-blocking get
CONNECTING THE PRODUCER AND CONSUMER
THE CONFIGURATION DATABASE IN PYUVM
WHAT IS THE CONFIGDB()?
BASIC SETTING AND GETTING
UVM HIERARCHICAL CONTROL
PATH OVERRIDES
DEBUGGING THE CONFIGDB()
THE UVM FACTORY
INSTANTIATING OBJECTS USING THE FACTORY
OVERRIDING CLASSES
CLEARING OVERRIDES FROM THE FACTORY
PRINTING THE STATE OF THE FACTORY
CREATING A STRING FROM THE FACTORY
LOGGING IN PYUVM
LOGGING LEVELS
CHANGING THE LOGGING LEVEL
LOGGING MESSAGES IN PYUVM
Examining the log message
CONTROLLING OUTPUT
Writing to a file
Disabling Logging
ODDS AND ENDS
PYTHON AND SYSTEMVERILOG
PYUVM CHANGES TO RUN_TEST()
Passing classes to run_test()
pyuvm does not need uvm_test
Keeping singletons
ENABLING DEEP UVM LOG MESSAGES
Changing the default logging level
pyuvm Does Not Need uvm_subscriber
LINKS

CHAPTER 05 COCOTB DERIVATIVE - UVM PYTHON: UVM PORT TO PYTHON

WHY BOTHER?
INSTALLATION
RUNNING THE EXAMPLES
THE CODE
HDL SIMULATORS
RELATED PROJECTS
LINKS

CHAPTER 06 COCOTB DERIVATIVE – VARIFOG: OPEN SOURCE PYTHON AND COCOTB BASED HARDWARE VERIFICATION TOOL

INTRODUCTION
TOOL LIMITATION
PROPOSED FLOW
VERIFOG TOOL GUI
CONFIGURATION
RUNNING THE TOOL
    STEPS TO BE FOLLOWED IN THE TOOL:
TESTING EXAMPLE_ADDER.V
    PARSE IT.
    SET THE NUMBER OF TEST VECTORS AS 5.
    VERIFY:
    REGRESSION SUMMARY.
LITERATURE

CHAPTER 07 COCOTB DERIVATIVE – VERLPY WITH REINFORCEMENT LEARNING

REINFORCEMENT LEARNING (RL)
    REINFORCEMENT LEARNING (RL), WHAT IS IT?
    HOW TO FORMULATE A BASIC REINFORCEMENT LEARNING PROBLEM?
VERLPY
INSTALLATION
IDENTIFYING VERIFICATION GOALS AND DEFINING THE MDP (MARKOV DECISION PROCESS)
INHERITING COCOTBENV
INSTANTIATING THE VERIFICATION ENVIRONMENT OBJECT
ADDING COROUTINES TO TRACK EVENTS
CONFIGURATION FILE
FILLING IN THE VERIFICATION LOGIC
MULTI-STEP RL
MAKE FILE
LINKS

CHAPTER 08 COCOTB EXTENSION - COCOTB-COVERAGE

INTRODUCTION
INSTALLATION
CODE EXAMPLE
FIFO EXAMPLES
LINKS

CHAPTER 09 COCOTB EXTENSION - COCOTB-TEST

INTRODUCTION
USAGE
ARGUMENTS FOR SIMULATOR.RUN:
ENVIRONMENTAL VARIABLES
PYTEST ARGUMENTS
TIPS AND TRICKS
RUNNING (SOME) TESTS AND EXAMPLES FROM COCOTB

CHAPTER 10 COCOTB EXTENSION - USB TEST SUITE

INTRODUCTION
SETUP
PREREQUISITES
STEPS
USAGE
LINKS

CHAPTER 11 MYHDL - PYTHON BASED HARDWARE DESCRIPTION AND VERIFICATION LANGUAGE

OVERVIEW
    MODELING
    SIMULATION AND VERIFICATION
    CONVERSION TO VERILOG AND VHDL
INTRODUCTION TO MYHDL
    A BASIC MYHDL SIMULATION
    SIGNALS AND CONCURRENCY
    PARAMETERS, PORTS AND HIERARCHY
    TERMINOLOGY REVIEW
    SOME REMARKS ON MYHDL AND PYTHON
SUMMARY AND PERSPECTIVE
INSTALLATION - CO-SIMULATION
    INSTALLATION
        Installation using pip
        Installation using distutils
    INSTALLATION FOR CO-SIMULATION
        Setup ModelSim
    HOW CAN I RUN CO-SIMULATION ON WINDOWS?
SIMULATION AND VERIFICATION
    D FLIP-FLOP
        Specification
        Description
        Simulation
CO-SIMULATION WITH VERILOG
    INTRODUCTION
    THE HDL SIDE
    THE MYHDL SIDE
    RESTRICTIONS
        Only passive HDL can be co-simulated
        Race Sensitivity Issues
    IMPLEMENTATION NOTES
        Icarus Verilog
        Cver
        Other Verilog Simulators
        Interrupted System Calls
        What about VHDL?
UNIT TESTING
    DEFINING THE REQUIREMENTS
    WRITING THE TEST FIRST
    TEST-DRIVEN IMPLEMENTATION
    ADDITIONAL REQUIREMENTS

CONVERSION TO VERILOG AND VHDL
INTRODUCTION
SOLUTION DESCRIPTION
FEATURES
THE CONVERTIBLE SUBSET
Introduction
Coding style
Supported types
Supported statements
Supported built-in functions
Docstrings
CONVERSION OF LISTS OF SIGNALS
CONVERSION OF INTERFACES
ASSIGNMENT ISSUES
Name assignment in Python
Signal assignment
intbv objects
EXCLUDING CODE FROM CONVERSION
USER-DEFINED CODE
TEMPLATE TRANSFORMATION
CONVERSION OUTPUT VERIFICATION BY CO-SIMULATION
CONVERSION OF TEST BENCHES
METHODOLOGY NOTES
Simulate first
Handling hierarchy
KNOWN ISSUES
AUTOMATIC CONVERSION TO VERILOG OR VHDL
LINKS & LITERATURE

CHAPTER 12 METAPROGRAMMING – FAULT: A PYTHON EMBEDDED DOMAIN-SPECIFIC LANGUAGE FOR METAPROGRAMMING PORTABLE HARDWARE VERIFICATION COMPONENTS

METAPROGRAMMING - WIKI
INTRODUCTION
OVERVIEW
DESIGN
FRONTEND: TESTER API
ACTIONS IR
BACKEND TARGETS
EVALUATION
RELATED WORK
SUPPORTED SIMULATORS
INSTALLATION
USING PIP
PYTHON 3.7.2
FROM SOURCE
COREIR
TESTER ABSTRACTION
TESTER ACTIONS
POKE
EVAL
EXPECT

STEP
EXECUTING TESTS
EXAMPLE
EXERCISE 1
EXTENDING THE TESTER CLASS
EXERCISE 2
PYTEST PARAMETRIZATION
EXERCISE 3
ASSUME/GUARANTEE
CONSTRAINED RANDOM
FORMAL VERIFICATION
EXERCISE 4
MORE EXAMPLES
HOW DO I GENERATE WAVEFORMS WITH FAULT?
LINKS & LITERATURE

CHAPTER 13 HARDWARE DESIGN LANGUAGE BASED ON PYTHON PYRTL

HARDWARE PROTOTYPING IN PYTHON - DEFINITION
INTRODUCTION
PYRTL IS
FEATURES INCLUDE:
PACKAGE CONTENTS
PYRTL WORKFLOW
AUTOMATIC INSTALLATION
CONFIGURATION FILE FOR THE SPHINX DOCUMENTATION BUILDER
DESIGN, SIMULATE, AND INSPECT IN 15 LINES
DEVELOPING AND VERIFYING WITH SCIKIT-LEARN
LINKS & LITERATURE
CIRCUITPYTHON
MICROPYTHON: AN INTRO TO PROGRAMMING HARDWARE IN PYTHON

CHAPTER 14 PYMT - PYTHON-BASED HARDWARE GENERATION, SIMULATION AND VERIFICATION FRAMEWORK

PYMTL BASICS
PYMTL: A UNIFIED FRAMEWORK ENABLING MODELING TOWARDS LAYOUT
PYMTL: A UNIFIED FRAMEWORK FOR VERTICALLY INTEGRATED COMPUTER ARCHITECTURE RESEARCH
PYMTL FOR COMPUTER ARCHITECTURE TEST CHIPS
EIGHT FEATURES THAT MAKE PYMTL PRODUCTIVE
PYMTL3 WORKFLOW
PYMTL FRAMEWORK
PYMTL3 EMBEDDED DSL
WHAT IS PYMTL FOR AND (CURRENTLY) NOT FOR?
MULTI-LEVEL MODELING WITH PYMTL
PYMTLV3 HIGH-LEVEL MODELING
PYMTLV3 LOW-LEVEL MODELING
HIGHLY PARAMETRIZED STATIC ELABORATION
PYMTL PASSES
PYMTL/SYSTEMVERILOG INTEGRATION
PROPERTY-BASED RANDOM TESTING
FAST PURE-PYTHON SIMULATION

INSTALLATION
INSTALL THE PYMTL3 FRAMEWORK
INSTALL THE VERILATOR SIMULATOR
BITS ARITHMETICS
FULL ADDER EXAMPLE
REGISTER INCREMENTOR EXAMPLE
LINKS & LITERATURE

CHAPTER 15 PYHVL - A VERIFICATION TOOL

WHY PYTHON?
WHAT IS PYHVL?
DEVELOPMENT ENVIRONMENT
INSTALLATION
CONFIGURING PYTHON
LINKING WITH YOUR SIMULATOR
TESTING PYHVL
CONFIGURING PYHVL
SIMULATOR SPECIFIC CONCERNS
PYHVL AND GENERATORS
A BRIEF DESCRIPTION OF PYTHON GENERATOR FUNCTIONS
USING A GENERATOR IN A VERILOG SIMULATION
USING PYHVL
BUILDING THE TOP-LEVEL
For each output port
For each input port
For each inout port
For internal nodes
CONNECTING TO VERILOG NETS
TASKS AND TYPICAL PATTERNS
COMPOSING TASKS TO FORM TRANSACTIONS
USING CLASSES TO GROUP TASKS
EXAMPLES
OTHER EXAMPLES
CONSIDERATIONS IN IMPLEMENTING $PYHVL
A UVM LAYER FOR PYHVL
SIMPLIFIED VPI ITERATORS USING PYHVL GENERATORS
ABOUT GORDON MCGREGOR
LINKS & LITERATURE

CHAPTER 16 FPGA HARDWARE SIMULATION FRAMEWORK - FPGA_HW_SIM_FWK

HARDWARE SIMULATION PYTHON - DEFINITION
ARCHITECTURE OVERVIEW
DETAILED DESIGN
CODE
APPLICATION GUI
FEATURES
LIMITATIONS
USE CASES
PROJECT SETUP

GENERATION OF EXECUTABLE FILE
SUMMARY
LINKS

CHAPTER 17 HARDWARE SIMULATION ENVIRONMENT INTEGRATING PYTHON AND VHDL - PYHDL

INTRODUCTION
FEATURES
REQUIREMENTS
INSTALL PYVHDL
ZAMIACAD
RUN THE PLASMA DEMO
    SETUP A NEW PROJECT
    IMPORT THE ARCHIVED PROJECT
    CONFIGURE A SIMULATION
    RUN THE SIMULATION
WRITE A PYTHON TESTBENCH
LINKS

CHAPTER 18 SPICE-LIKE ELECTRONIC CIRCUIT SIMULATOR WRITTEN IN PYTHON - AHKAB

INTRODUCTION
SUPPORTED SIMULATIONS
DOWNLOAD AND INSTALL
    REQUIREMENTS
    INSTALL WITH DISTUTILS
RUN STANDALONE
DOCUMENTATION
SIMULATING FROM PYTHON
    A FIRST OP EXAMPLE
SIMULATING FROM THE COMMAND LINE
LINKS

CHAPTER 19 CO-SIMULATION OF HDL USING PYTHON AND MATLAB - COSIMTCP

INTRODUCTION
CO-SIMULATION ARCHITECTURE
    PROTOCOL
    SERVER
    CLIENT
DESIGN FLOW
EXAMPLE PROJECT SHOWING THE BASIC IDEA OF THE COSIMTCP
    SERVER SIDE
        ModelSim
        Vivado
    CLIENT SIDE
        Python

LINKS & LITERATURE

CHAPTER 20 CPU SIMULATOR WRITTEN IN PYTHON - PYCPUSIMULATOR

WHAT ARE THE MAIN FEATURES ?
INSTALLATION
    DEPENDENCIES
    INSTALLATION FROM PYPI REPOSITORY
    INSTALLATION FROM SOURCE
AVR DATASHEET
    ATMEGA640/1280/1281/2560/2561 REGISTER SUMMARY
    HOW TO EXTRACT DATA FROM ATMEL DATASHEETS
    AVR REGISTERS
        SREG – AVR Status Register
        General Purpose Register File
        RAMPZ – Extended Z-pointer Register for ELPM/SPM
        EIND – Extended Indirect Register
    AVR MEMORIES
    REGISTER SUMMARY
    INSTRUCTIONS
MICRO CODE LANGUAGE
INSTRUCTION SET YAML FORMAT
LINKS & LITERATURE

CHAPTER 21 PYTHON LIBRARY FOR INTERFACING TO VARIOUS SIMULATORS - PYOPUS

INTRODUCTION
WHAT PACKAGES DOES PYOPUS DEPEND ON
SUPPORTED SIMULATORS
SUPPORTED OPERATING SYSTEMS
BUILDING PYOPUS FROM SOURCES
    REQUIREMENTS
    UNPACKING PYOPUS SOURCES
    BUILDING PYOPUS FOR LINUX (WHEEL, DEMOS AND DOCUMENTATION, AND SOURCE)
    BUILDING FOR WINDOWS (WHEEL ONLY)
    INSTALL PYOPUS FOR DEVELOPMENT (IN SOURCE FOLDERS)
PYOPUS DESIGN AUTOMATION GUI
    STARTING THE GUI
    THE GUI WINDOW
    SETTING UP A PROJECT
        Netlist fragments, models, and the project file
        Defining variables
        Simulator setup
        Analyses
        Parameters
        Setting up the operating parameters.
        Setting up the statistical parameters.
    A SIMPLE CIRCUIT EVALUATION TASK
        Creating a new evaluation task
        Task settings
        Starting the task

Viewing the log file
Using the log file for debugging
Viewing the Results and Postprocessing Saved Waveforms
Opening and browsing the results
EVALUATING PERFORMANCE MEASURES ON SAVED WAVEFORMS
Visualization of saved waveforms
POSTPROCESSING SETUP FILE
LINKS

CHAPTER 22 PYTHON INTERFACE TO THE NGSPICE AND XYCE CIRCUIT SIMULATORS - PYSPICE

WHAT IS PYSPICE ?
WHAT ARE THE MAIN FEATURES ?
INSTALL ON WINDOWS
INSTALL A MORE RECENT VERSION FROM GITHUB USING PIP
INSTALLATION FROM SOURCE
HOW TO RUN THESE EXAMPLES
NETLIST MANIPULATIONS
FAST FOURIER TRANSFORM
LINKS

CHAPTER 23 CONSTRAINTS AND COVERAGE - PYVSC PACKAGE

INTRODUCTION
WHAT IS PYVSC?
FUNCTIONAL VERIFICATION AND CONSTRAINED RANDOM STIMULUS
KEY REQUIREMENTS
PYVSC BASICS
PYVSC FEATURES
DEBUG
INSTALLING PYVSC
PYVSC CONSTRAINTS
CONSTRAINT BLOCKS
CONSTRAINT EXPRESSIONS
CONSTRAINT STATEMENTS
CUSTOMIZING DECLARED CONSTRAINTS
PYVSC FUNCTIONAL COVERAGE
COVERGROUPS
COVERPOINTS, BINS, AND CROSSES
SAMPLING COVERAGE DATA
REPORTING
ENVIRONMENT INTEGRATION
RANDOM SEED MANAGEMENT
SAVING COVERAGE DATA
LINKS & LITERATURE

CHAPTER 24 CONSTRAINTS AND COVERAGE, YOSYSHQ/MCY - MUTATION COVER WITH YOSYS

INTRODUCTION TO MUTATION COVERAGE WITH YOSYS (MCY)
DEPENDENCIES
INSTALLATION
INSTALLING TABBY CAD SUITE OR OSS CAD SUITE
    INSTALLING FROM SOURCE
METHODOLOGY
    EQUIVALENCE CHECK
    FUNDAMENTAL PRINCIPLE
COMMAND REFERENCE
EXAMPLE
CONFIGURATION FILE FORMAT
FORMAL EQUIVALENCE TEST
TAGGING LOGIC
    LOGIC
    REPORT
    TESTS
WRITING A TEST SCRIPT
EXPORTING THE MUTATED SOURCE
RUNNING THE TESTBENCH
REPORTING THE RESULT
RUNNING MCY
RUNNING THE TESTS
TESTBENCH SCRIPT
WRITING THE EQUIVALENCE CHECK
    CREATING THE MITER CIRCUIT
    SETTING UP THE TEST SCRIPT FOR THE EQUIVALENCE CHECK
        Mutation export
    RUNNING THE EQUIVALENCE CHECK
    REPORTING THE RESULT
MUTATION EXPORT OPTIONS
THE CREATE_MUTATED.SH SCRIPT
THE TASK MUTATION LIST INPUT.TXT
WRITING A CUSTOM MUTATION EXPORT SCRIPT
THE YOSYS MUTATE COMMAND
MUTATION GENERATION
APPLYING A MUTATION
LINKS & LITERATURE

CHAPTER 25 SYMBOLIC MODEL CHECKING - COSA (COREIR SYMBOLIC ANALYZER)

SYMBOLIC MODEL CHECKING FOR HARDWARE VERIFICATION, WHAT IT IS?
INTRODUCTION
OVERVIEW
BACKGROUND
    MODEL CHECKING
    SYMBOLIC TRANSITION SYSTEM
    LINEAR TEMPORAL LOGIC (LTL)
INPUT FORMATS
    VERILOG
    SYSTEMVERILOG
    VERILOG/SYSTEMVERILOG WITH YOSYS
    COREIR

BTOR2
SYMBOLIC TRANSITION SYSTEM (STS)
EXPLICIT STATE TRANSITION SYSTEM (ETS)
INITIAL STATE CONSTRAINTS (INIT)
PROPERTIES
INVARIANT
LINEAR TEMPORAL LOGIC
SYNTACTIC SUGAR
GENERATORS
VERIFICATION DEFINITION
ENVIRONMENTAL ASSUMPTIONS
SIMULATION
SAFETY AND LTL VERIFICATION
Formula Syntax
EQUIVALENCE CHECKING
PARAMETRIC MODEL CHECKING
PROBLEM FILES
USEFUL HINTS
EXAMPLES
RESULTS ANALYSIS
COUNTEREXAMPLE TRACES
Finite traces
Infinite traces
GOOD PRACTICE
ENCODINGS
PERFORMANCE OPTIMIZATIONS
LEMMAS
STRATEGY
ASSUME IF TRUE
CONE OF INFLUENCE
CIRCUIT OPTIMIZING
FILES CACHING
DEBUGGING
LINKS AND LITERATURE

CHAPTER 26 FRONT-END DRIVER PROGRAM YOSYSHQ/SYMBIYOSYS

INTRODUCTION
GETTING STARTED
FIRST IN, FIRST OUT (FIFO) BUFFER
VERIFICATION PROPERTIES
SYMBIYOSYS
EXERCISE
CONCURRENT ASSERTIONS
LINKS

CHAPTER 27 AMIQ OFC - OPEN-SOURCE FRAMEWORK FOR CO-EMULATION USING PYNQ

HARDWARE EMULATION IN PYTHON - DEFINITION
WHAT IS CO-EMULATION?
WHAT IS PYNQ?

WHAT IS OFC?
BASIC CONCEPTS
    LAYER 1: HOST – VERIFICATION ENVIRONMENT
    LAYER 2: PYNQ – PROCESSING SYSTEM
    LAYER 3: PYNQ – PROGRAMMABLE LOGIC
OFC SV-PYTHON CONNECTION
    1. OFC DRIVER
    2. OFC SERVER CONNECTOR
    3. OFC PYTHON SERVER.
    4. TESTBENCH
OFC PYTHON-FPGA INTERACTION
    1. CONFIGURING THE PL SIDE OF THE PYNQ BOARD.
    2. MA TO THE PL SIDE.
    3. L SIDE THROUGH DMA.
FPGA
INTEGRATION
    STEP 1: REPLACING THE ORIGINAL DRIVERS WITH THE OFC DRIVER
    STEP 2: CREATING A SPECIFIC OFC MONITOR
    STEP 3: COMPUTING RESPONSE WITHIN THE PYTHON SERVER
    STEP 4: CREATING HDL COMPONENTS
CONCLUSIONS
DVCON U.S. 2021 EXPERIENCE
RESOURCES
LINKS & LITERATURE

CHAPTER 28 EMULATE INTEGRATED CIRCUITS IN PYTHON - ICEMU

ICEMU - EMULATE INTEGRATED CIRCUITS
SEE IT IN ACTION
WHERE'S THE PYTHON IMPLEMENTATION?
REQUIREMENTS
HOW TO USE
EXAMPLES
DOCUMENTATION
LICENSE

PREFACE

I write this book because I cannot find a similar one that can help me grasp a full understanding of how a traditional programming language like Python can be used as a primary verification language, as I previously surmised that verification must be always done with a special language called Hardware Verification Language (HVL), such as SystemVerilog, the e Language, OpenVera and a bunch of others. During my design of IP (Intellectual Property) core, I have met with a number of test scripts, many of them are written in Python and other languages such as Matlab, and even Java. In other words, we don't necessarily have to learn the overly sophisticated verification language of SystemVerilog in order to verify a hardware design, especially for smaller designs.

On other hand, I cannot find a similar book describing in full about the use of Python language and its rich sets of libraries for hardware verification. There you can find various sets of libs here and there sporadically about Python modules used for verification. In my design career, I need to compile a list of all these modules so that I can use them for my present as well as future design and verification projects. And this book is largely the result of my collection work of the designs.

How to Use This Book

This book is provided in PDF format in digital form, no paper work will be produced. We are in digital age, yeah? To save money for publishing, to facilitate the distribution, to save our earth and also to save money for our readers, I refuse to publish anything in paper forms, and I also refuse to let the expensive publishing houses and online platforms to do the same. Traditional printing is gone, forever. At time of self-publishing, we are all our own publishing houses, aren't we?

MC

Nov. 29, 2022