數位系統設計

Digital System Design

臺大電機系/電子所
吳安宇教授

2025/02/20

Course Website

- Course Website (NTU COOL)
  - https://cool.ntu.edu.tw/courses/45284
  - Course materials announcement
  - Homework submission
  - Fundamentals of Computer Architecture (CA)
  - Homework discussion, Problem settlement

Objective

- **Digital system design (DSD)** plays an important role in implementing digital functions in modern **system-on-chip (SOC)** design.

- In this course, we will focus on developing design skills for <u>undergraduate students</u> so that they can be familiar with state-of-the-art **digital <u>front-end</u> design skills and flow.**

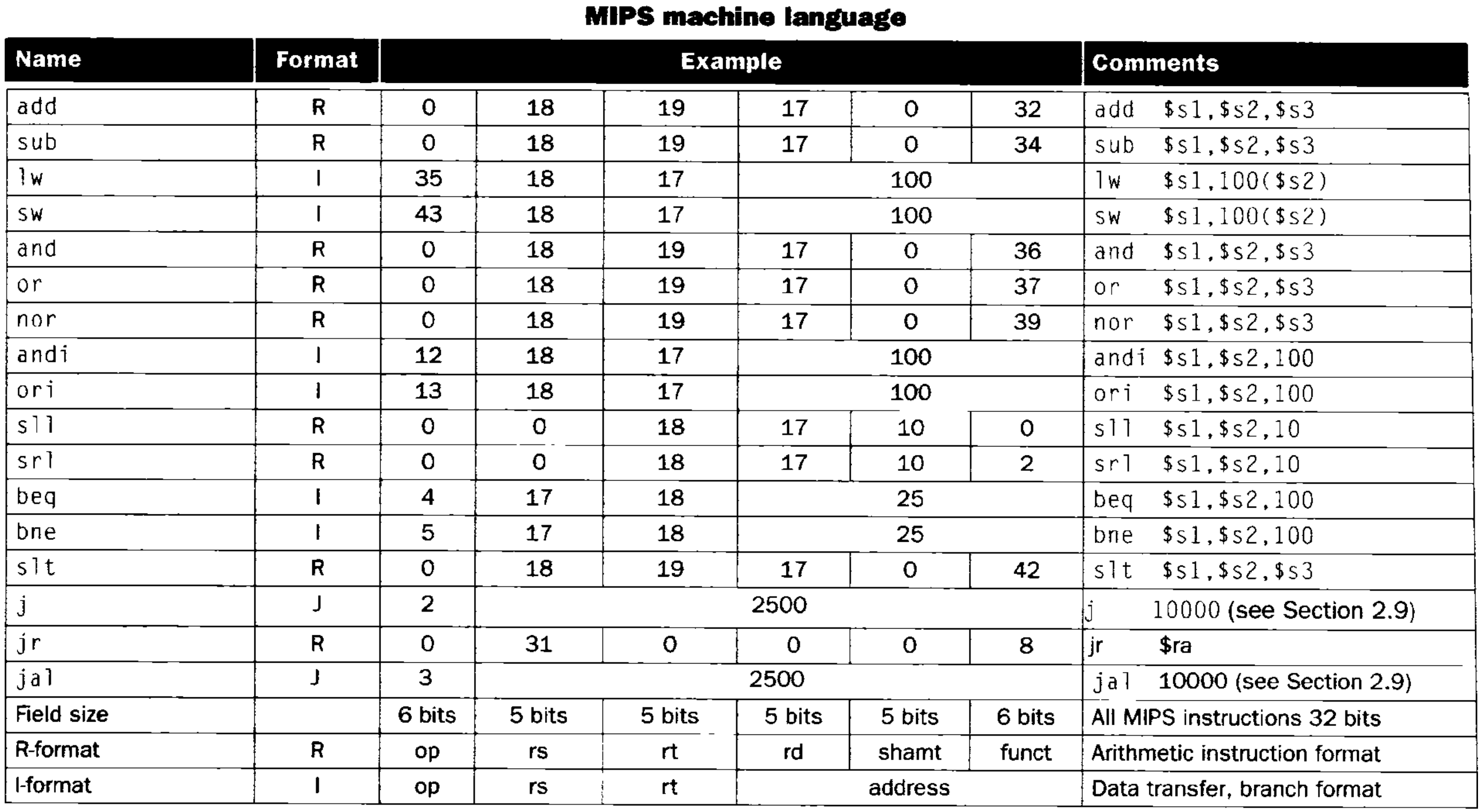
Course Content – Verilog HDL

- Firstly, we will introduce a **Hardware Description Language (HDL).** The chosen HDL is **Verilog**. We will formally cover following topics:
  - Verilog **syntax (basics)**
  - Verilog coding guideline (how to write **elegant** codes)
  - **Synthesizable** coding guideline (how to write **feasible/implementable** Verilog codes)
  - Modern **cell-based synthesis flow** (how to link with modern **EDA tools** and know ASIC design flow)
  - **Coding style** that follows Reuse Manual Methodology (RMM) – How to create **"reusable"** codes for IP (Intellectual property) reuse

Course Content – CPU Processor

- Secondly, we will ask students to design an **advanced RISC-V CPU.** It is based on the knowledge of "**MIPS**-based Computer Architecture (CA)." The assignment covers following topics:
  - Implement **Arithmetic Logic Unit (ALU)** and **Control Unit (CU)**
  - Implement **Single-Cycle CPU** capable of processing *assembly code* (machine language – binary programs for CPUs)
  - Implement **Cache** (a kind of memory that stores data so that future requests for that data can be served faster)
  - Implement a **Pipelined CPU design** with Forwarding Unit, Hazard Control, Branch Prediction, L1/L2 cache, etc.

MIPS Instruction & Machine Language



MIPS machine language

Name	Format	Example						Comments
add	R	0	18	19	17	0	32	add $s1,$s2,$s3
sub	R	0	18	19	17	0	34	sub $s1,$s2,$s3
lw	I	35	18	17	100			lw $s1,100($s2)
sw	I	43	18	17	100			sw $s1,100($s2)
and	R	0	18	19	17	0	36	and $s1,$s2,$s3
or	R	0	18	19	17	0	37	or $s1,$s2,$s3
nor	R	0	18	19	17	0	39	nor $s1,$s2,$s3
andi	I	12	18	17	100			andi $s1,$s2,100
ori	I	13	18	17	100			ori $s1,$s2,100
sll	R	0	0	18	17	10	0	sll $s1,$s2,10
srl	R	0	0	18	17	10	2	srl $s1,$s2,10
beq	I	4	17	18	25			beq $s1,$s2,100
bne	I	5	17	18	25			bne $s1,$s2,100
slt	R	0	18	19	17	0	42	slt $s1,$s2,$s3
j	J	2	2500					j 10000 (see Section 2.9)
jr	R	0	31	0	0	0	8	jr $ra
jal	J	3	2500					jal 10000 (see Section 2.9)
Field size		6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	All MIPS instructions 32 bits
R-format	R	op	rs	rt	rd	shamt	funct	Arithmetic instruction format
I-format	I	op	rs	rt	address			Data transfer, branch format

Multi-cycle Implementation of MIPS CPU (from CA Textbook)

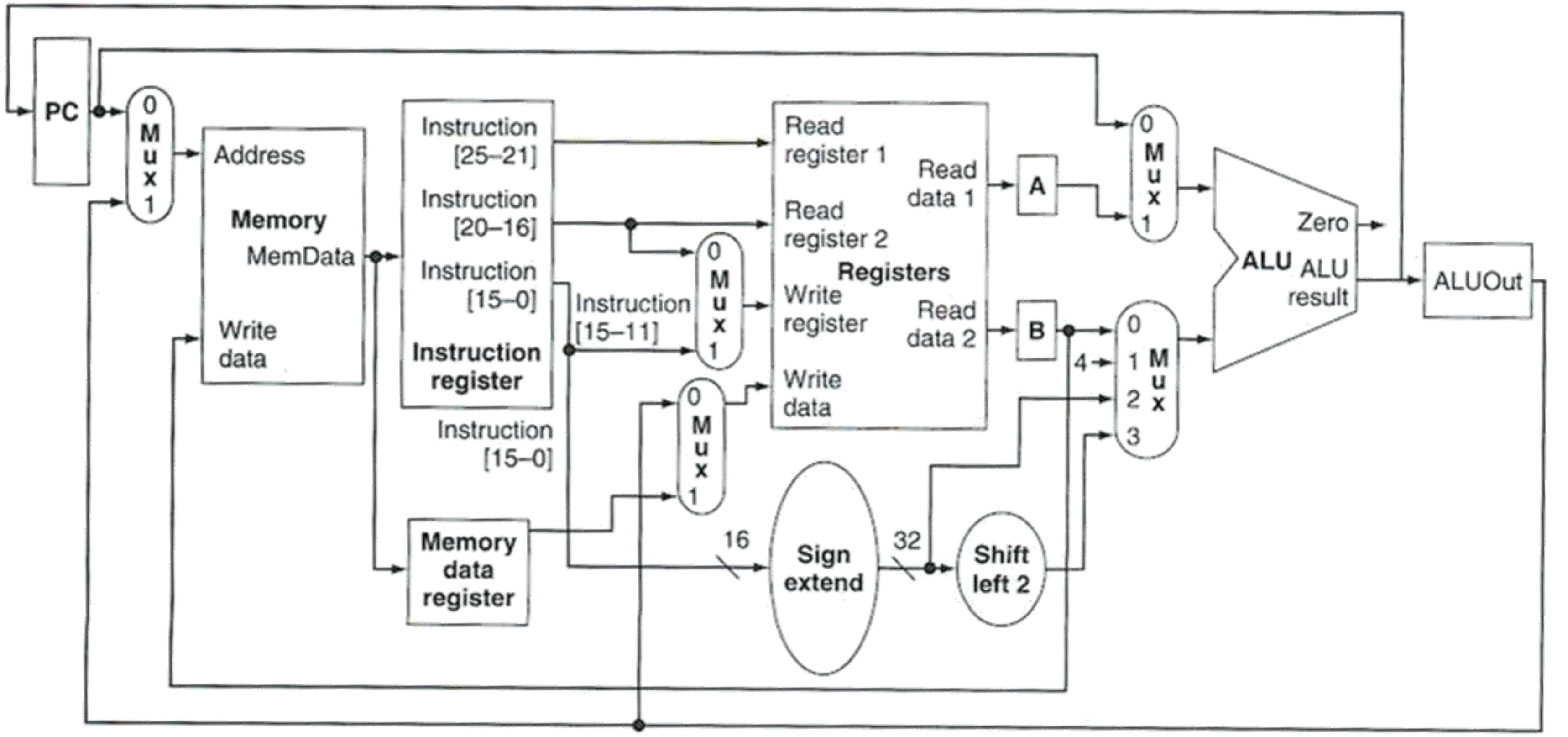
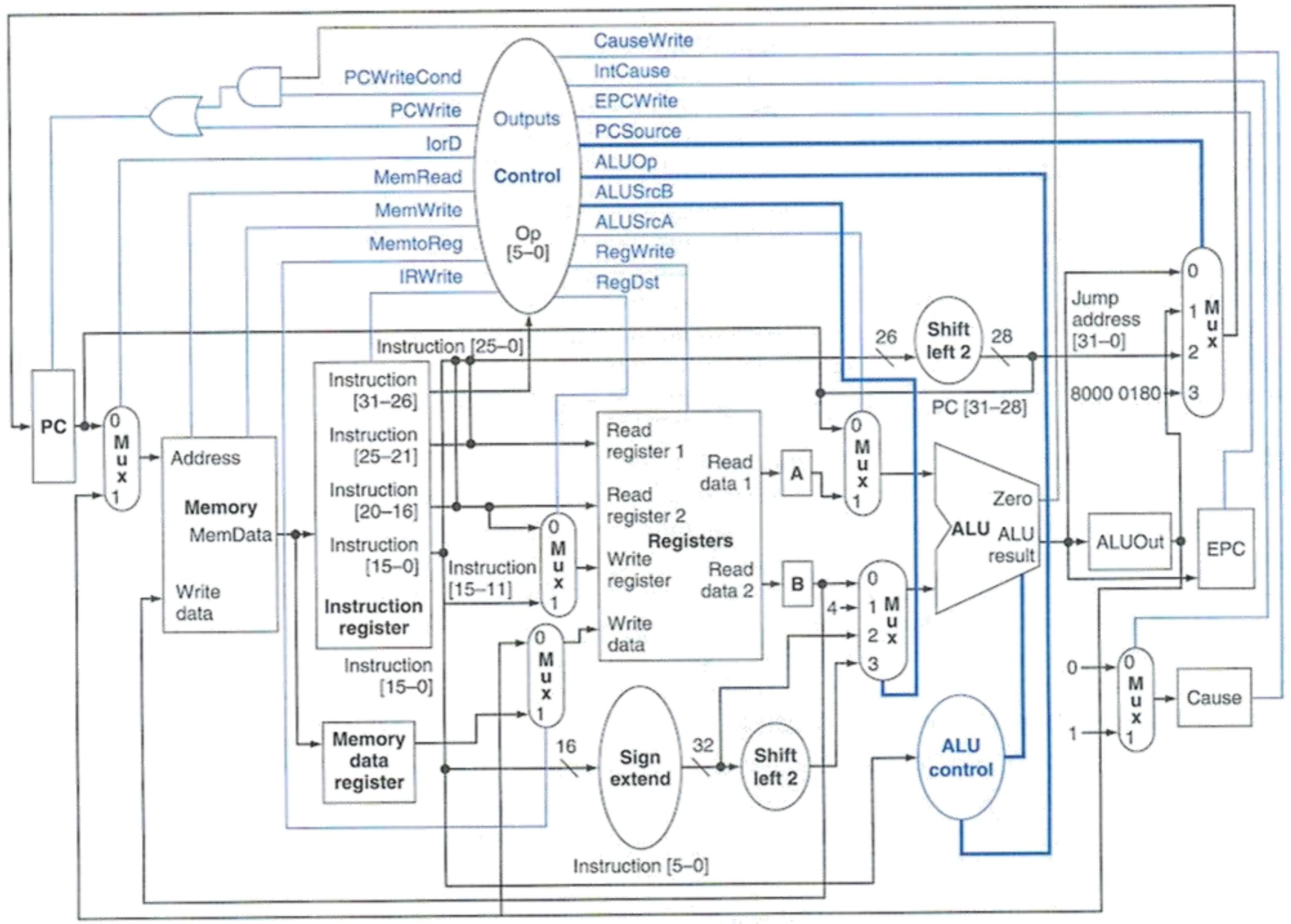


FIGURE 5.26 Multicycle datapath for MIPS handles the basic instructions. Although this datapath supports normal incrementing of the PC, a few more connections and a multiplexor will be needed for branches and jumps; we will add these shortly. The additions versus the single-clock datapath include several registers (IR, MDR, A, B, ALUOut), a multiplexor for the memory address, a multiplexor for the top ALU input, and expanding the multiplexor on the bottom ALU input into a four-way selector. These small additions allow us to remove two adders and a memory unit.

Multi-cycle Implementation with CU



What is RISC-V?

- **Design Concept: As Simple As Possible**
- RISC-V is an open source implementation of a *Reduced Instruction Set Computing (RISC)* based on an open *Instruction Set Architecture (ISA)*
  - Completely open ISA
  - Small base integer ISA + *Optional* standard extension
  - Both 32-bit and 64-bit address space variants for applications
  - Supports general purpose of software development (Linux)
- RISC-V was originated in 2010 by researchers in the Computer Science Division at UC Berkeley
  - Create a RISC-V foundation in 2015
  - In 2018, v2.2 of the user space ISA is fixed
- RISC-V is modularized
  - ARM: (de facto standard): Application, Real-Time, Embedded System are not compatible

Why Do We Need RISC-V?

- Commercial ISAs are proprietary (e.g., ARM)
  - Major concern for groups wishing to share actual RTL implementations
- market domains
  - ARM architecture is Commercial ISAs are only popular in certain not well supported in the **server space**
  - Intel x86 architecture is not well supported in the **mobile space**
- Popular commercial ISAs are complex
  - Nearly all the complexity is due to bad and outdated ISA design decisions (legacy instructions, e.g.,x86)
- Popular commercial ISAs were not designed for extensibility
  - added considerable instruction encoding complexity as their instruction sets have grown

AI
FREE
RISC-V
CPU

Course Contents - Exercise

- Design a circuit based on a given specification
- Duration: 2 weeks
- Grading rule:
  - RTL and Synthesis (70%)
    - Class **A**: pass *all* test patterns in RTL simulation and gate-level simulation
    - Class **B**: pass *all* test patterns only in RTL simulation
    - Class **C**: pass *partial* test patterns in RTL simulation
  - AT Ranking (25%)
    - AT ranking with other groups
  - Report (5%)

Course Contents - Final Project

- 3 people/team, RISC-V with Advanced Design

Baseline (focus on implementation)

- Implement Pipelined RISC-V by Verilog
  - HW2: Single Cycle RISC-V
  - Final Project: Pipelined RISC-V Processor
    - With Instruction cache and data cache

Extension (focus on discussion and analysis)

- Branch Prediction
- Compression (RVC instructions)
- Multiplication (RVM instructions)

Course Format

- Basic content
  - Basic materials of digital system design
- Supplement
  - Additional design materials
  - HW example from TA
- Lab
  - Demonstration of synthesis

Course Schedule (1/2)

Week	Date	Topic	Content	Homework Release	Homework Deadline
1	02/20	Intro.	Course Overview, Intro. to Digital System Design	HW0	
2	02/27	HDL 1	Fundamentals of Hardware Description Language		
3	03/06	HDL 2	Logic Design at Register Transfer Level	HW1	
4	03/13	HDL 3	Logic Design with Behavior Coding		
5	03/20	HDL 4	Testbench Writing, Verification Tool		
6	03/27	SYN 1	Synthesizable Coding	HW2	HW1
7	04/03	Spring break			
8	04/10	SYN 2	Synthesis Environment and Constraint		
9	04/17	Opt.	Complexity Management and Improvement	Exercise	HW2
10	04/24	**Midterm Exam**			

Course Schedule (2/2)

Week	Date	Topic	Content	Homework Release	Homework Deadline
11	05/01	CA	Memory Hierarchy and Pipelined Architecture	HW3	Exercise
12	05/08	CA	Advanced Topic: Branch Prediction and RVC ISA		
13	05/15	CA	Advanced Topic: M Extension and Applications	Final Project	HW3
14	05/22	TA Hour for Final Project			
15	05/29	Project Check Point			Final Check.
16	06/05	TA Hour for Final Project			
17	06/12	Final Project Presentation			Final Report

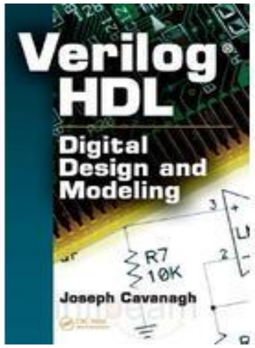
Verilog HDL Outlines (W2-W10)

- Overview of Hardware Describe Languages (HDL)
- Modeling and Verification with Verilog
- Logic Design with Behavior Models
- Introduction to synthesis with Verilog
  - Synthesizable coding for Combinational Circuits (ALU)
  - Synthesizable coding for Sequential Circuits (Control Unit)
- Finite State Machines (FSM) and Datapath Controllers
- Architecture and Algorithm mapping
- Coding Style (Reuse Manual Methodology)

CPU Outlines (W11-W17)

- Review of RISC-V CPU Architecture
  - HW: Single-cycle CPU design + ALU design
  - Control Flow Design
  - Memory hierarchy
- Extension of RISC-V CPU Architecture
  - Instruction Sets of RISCV-IMC
  - Design of branch predictor

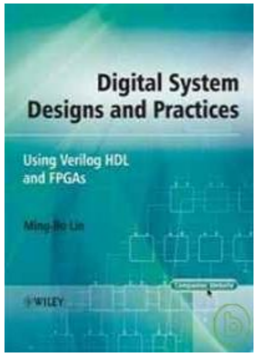
Textbooks



- (Main Verilog coding textbook)
  "**Verilog HDL: Digital design and modeling**,"
  Joseph Cavanagh, CRC Press, 2007.

COMPUTER ORGANIZATION AND DESIGN
THE HARDWARE/SOFTWARE INTERFACE
RISC-V EDITION
MK
DAVID A. PATTERSON
JOHN L. HENNESSY

- (Reference CPU textbook)
  "**Computer organization and design: The hardware/software interface**,"
  David A. Patterson and John L. Hennessy, 2014, 5th Edition and **RISC-V Edition**

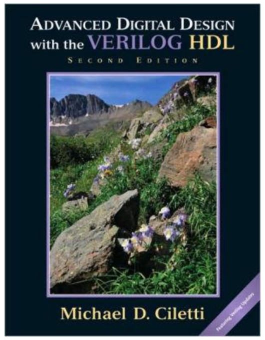

Digital System Designs and Practices
Using Verilog HDL and FPGAs
Ming-Bo Lin
WILEY

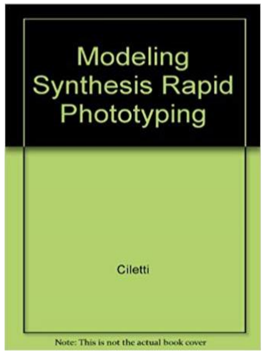
- (Reference Verilog coding textbook)
  "**Digital system designs and practices: Using Verilog HDL and FPGAs**,"
  Ming-Bo Lin, Wiley, 2008.

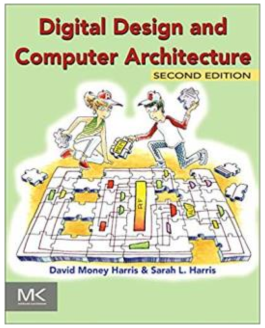
Textbooks

Other reference textbooks:

- *Advanced Verilog Coding:*
  - "**Advanced Digital Design with the Verilog™ HDL**", By Michael D. Ciletti, Published by Prentice Hall, 2003
  - "**Modeling, Synthesis, and Rapid Prototyping with the Verilog™ HDL**," By Michael D. Ciletti, Published by Prentice Hall, 1999
- *For Verilog design and microarchitecture:*
  - "**Digital Design and Computer Architecture, 2nd Edition**" by Harris & Harris , 2012




Modeling Synthesis Rapid Phototyping
Ciletti
Note: This is not the actual book cover


Digital Design and Computer Architecture
SECOND EDITION
David Money Harris & Sarah L. Harris
MK

Course Grading

- **A+ (ranking > 20% and final score > 90)**
  A   (ranking > 40% and final score > 85)
- Homework: 36%
  - HW1: 8%, HW2: 8%, HW3: 8%, Exercise: 12%
- Midterm Exam: 32% (closed book on Verilog)
- Final Project: 30%
  - Check Point (Baseline Design): 10%
  - Final Report and Submission: 20%
- Impression: 2% (Attendance & Attitude)

Suggested Background

- Programming Language : **Required**
- Logic Design : **Required**
- Computer Architecture (CA): **Required**
- VLSI Design and VLSI/EDA tools: **Optional**

Limitation

- Lab intensive and need close discussions.
- Limited to **60** students.
- 二類加簽
    - https://forms.gle/XecNarhLmo6SP62C9
    - 表單開放時間至2/23 23:59
    - **EE4 = EE3** > EE2 > EE1 > 外系
    - 研究生建議NTU COOL旁聽
    - Random drawing for remaining quota

Teaching Assistants Information

- **Lecture TA: Ben Wang (王景平)**
  - Room: EE2-232
  - Office hours: Wed. 14:20-15:10 (email in advance)
  - E-mail: benwang@access.ee.ntu.edu.tw
    Title should start with **[113-2 DSD]**
- **Homework TA:**
  - **謝言鼎**, alex@access.ee.ntu.edu.tw
  - HW1: 王泓予, pony@access.ee.ntu.edu.tw
  - HW2, HW3: 鄭至盛, sam@access.ee.ntu.edu.tw
  - Final: 江浩瑋, jackson@access.ee.ntu.edu.tw
    楊昀澔, howard@access.ee.ntu.edu.tw