McGill

ECSE 487

Computer Architecture Laboratory

Introduction

People

- Instructor: Prof. Warren Gross
- Office: McConnell 505
- Phone: (514) 398-2812
- Email: Warren.Gross@mcgill.ca

- Office Hours: Contact to arrange a time.

- TA:
  - Laurier Boulianne (Email: myCourses)

Webpage

- Check myCourses on a frequent basis !
  - email, announcements, handouts, links and pointers, schedule changes, etc…
- Discussion board on myCourses
  - Questions, discussion…

Calendar Description

- 2 Credits (0-3-3)
  - (Lectures - Labs and tutorials - outside work).

"Basic software tools used in the design, synthesis and analysis of computer and communication systems such as data-paths, switching circuits, and arithmetic and logic circuits. Behavioral and structural modeling of hardware designs in the IEEE standard hardware description language VHDL. Synthesis and implementation of hardware designs using Programmable Logic Devices."

What is 487 Really About ?

- Companion course to ECSE 425 Computer Organization and Architecture
- Hardware *design*
  - Modeling with hardware description languages, partitioning, verification…
- This is not a course in "VHDL"
- Software flow
  - Synthesis
- Presenting technical ideas effectively

What do you need to know?

- Prerequisite:
  - 455-206: Communication in Engineering.
- Co-requisite:
  - 304-425: Computer Organization and Architecture.
- Experience with VHDL.

What is Expected of You?

- Two assignments
  - A1: individual, A2: groups of up to 3
- Project
  - Abstract
  - Midpoint reports (x2)
    - » Written report
    - » Oral demonstration
  - End of term
    - » Final report
    - » Mini-conference presentations
- Project topics must obtain the approval of instructor before proceeding
- There is no guarantee that any particular project will be permitted

Grading Scheme

Assignment 1	10%
Assignment 2	25%
Final Project	65%

Lab

- Computer Architecture Laboratory
  - ENGTR 4120.
  - 24/7 access
- All course software is available in the lab

Tutorial and Lab Hours

- Software flow tutorial
  - ??
- Weekly TA lab hours
  - ??

Class Meetings

- Monday 2:30 – 3:30 PM, ENGTR 0070.
- There will NOT be a meeting every Monday.
- Attendance at all meetings is mandatory for all students.
- Course outline lists the schedule.
- Check myCourses for changes.

Schedule

- Jan. 5: Introduction and logistics.
- Jan 12: Assignment 1 discussion. Due Jan. 26
- Jan 19: Assignment 1 discussion (if necessary)
- Jan. 26: Assignment 2 discussion. Due Feb. 9
- Feb. 9: Project discussion.
- February 16: Project abstract due.
- March 2: Mid point review 1 report due.
- Midpoint Review 1 interviews: Week of March. 2-6. Schedule TBA.
- March 16: Mid point review 2 report due.
- Midpoint Review 2 interviews: Week of March 16-March 20. Schedule TBA.
- End of term Mini Conference: April 6, April 7, April 8, schedule TBA.
- Project material due on April 14.

Assignments

- All materials to be submitted electronically to WebCT Vista.
- Extensions only under ***exceptional*** circumstances.
  - You must obtain the instructor's permission at least one week in advance of the due date. Otherwise, no late submissions will be accepted. Medical notes will only be accepted within one week of the missed date.

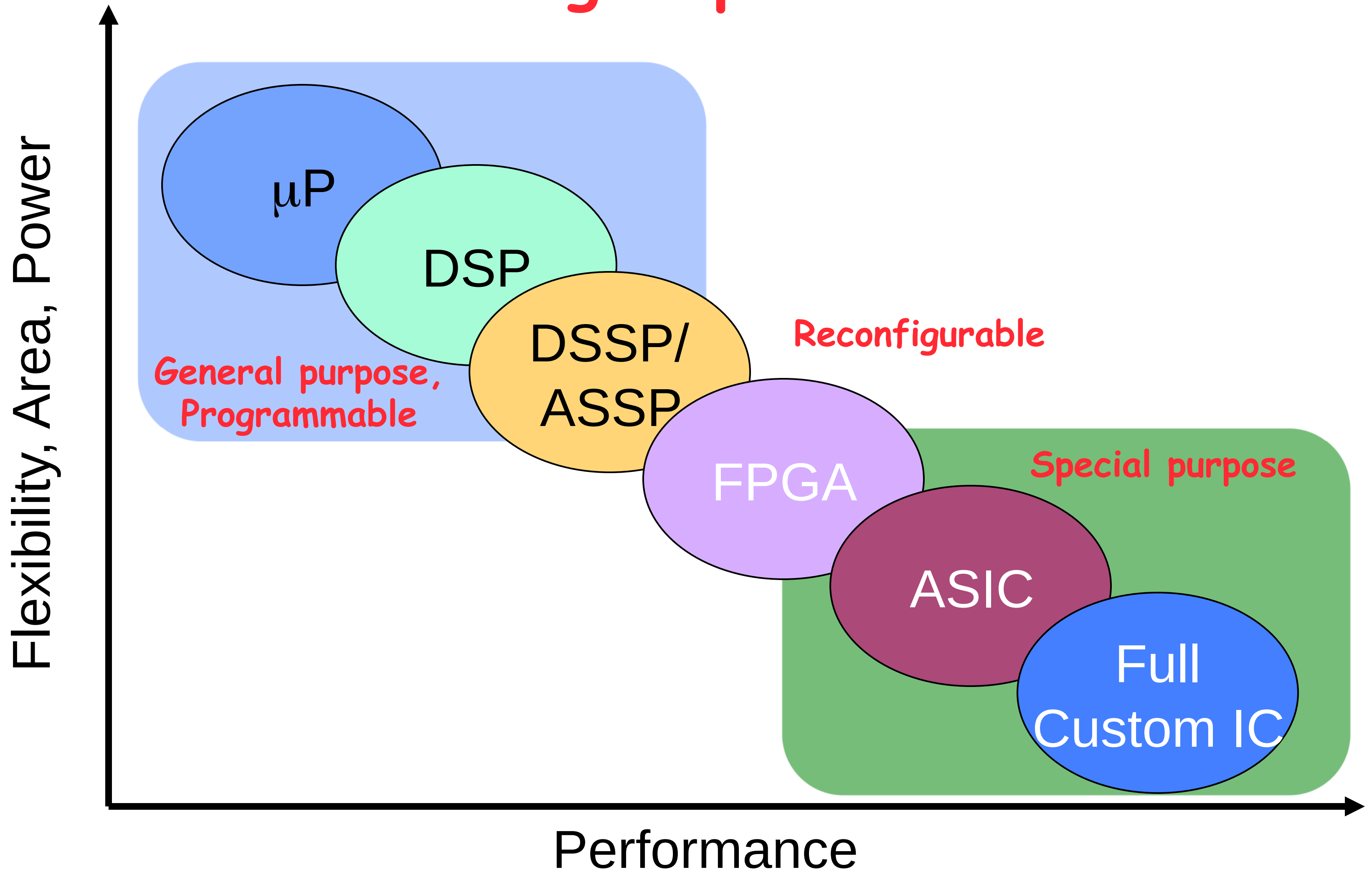
Academic Integrity

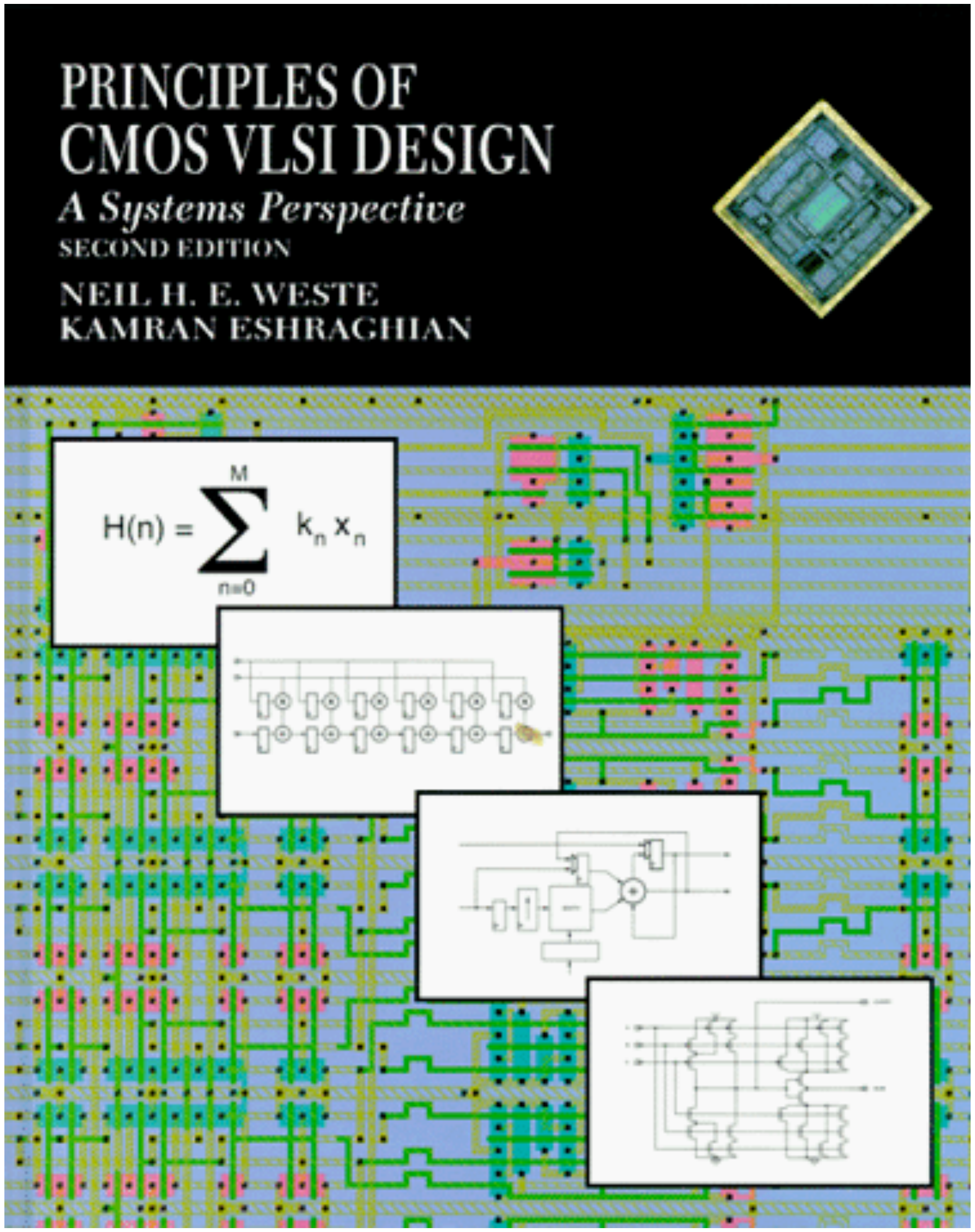
- All students are expected to be familiar with McGill's policies with respect to academic integrity:
- McGill University values academic integrity. Therefore all students must understand the meaning and consequences of cheating, plagiarism and other academic offences under the Code of Student Conduct and Disciplinary Procedures Procedures (see http://www.mcgill.ca/integrity for more information).
- Please see memo by Prof. Webb on myCourses

Reference Text

- P. J. Ashenden, The Designer's Guide to VHDL, Morgan Kauffman, 2nd edition.

Design Space





PRINCIPLES OF CMOS VLSI DESIGN

A Systems Perspective

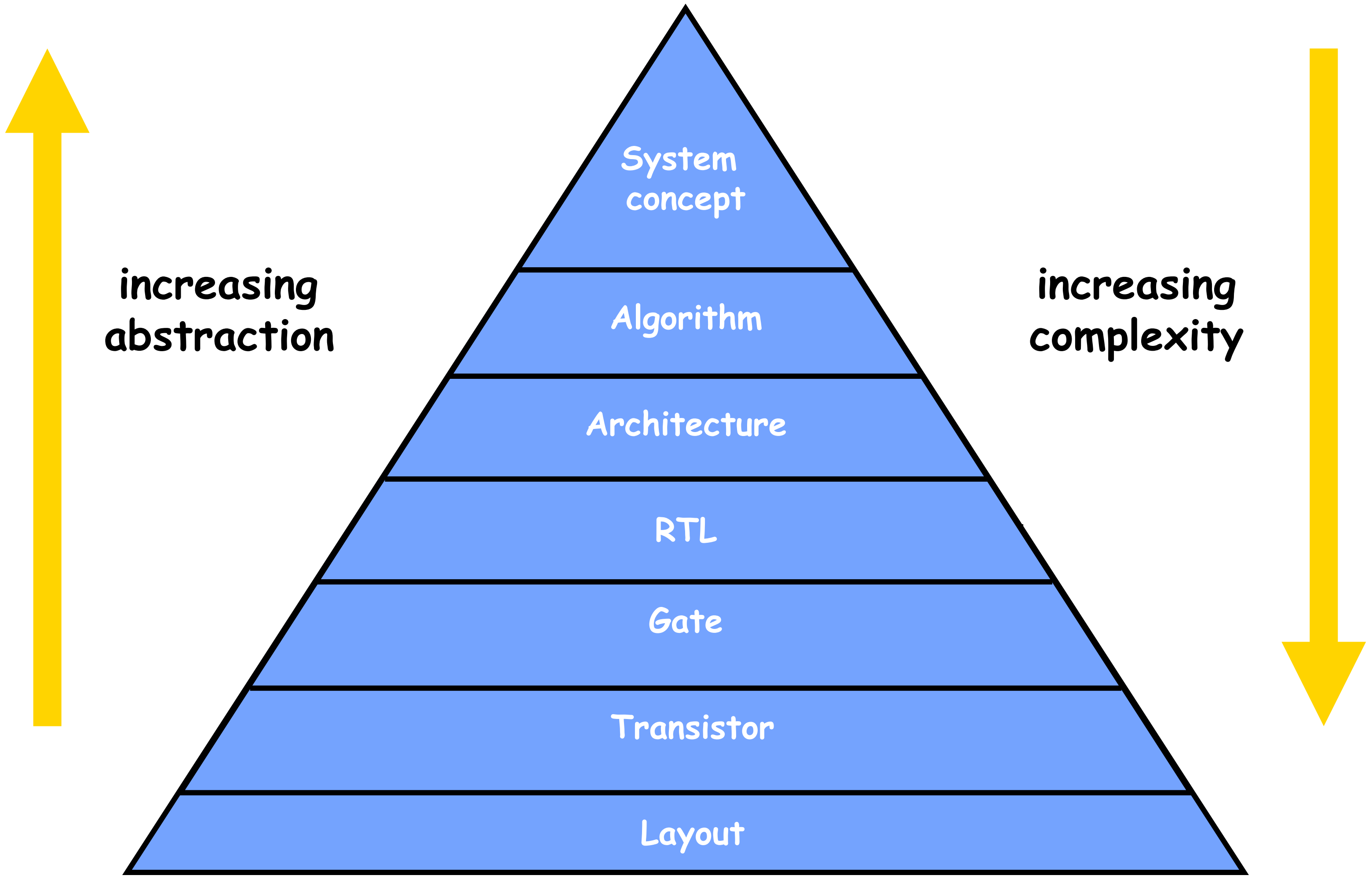
SECOND EDITION

NEIL H. E. WESTE

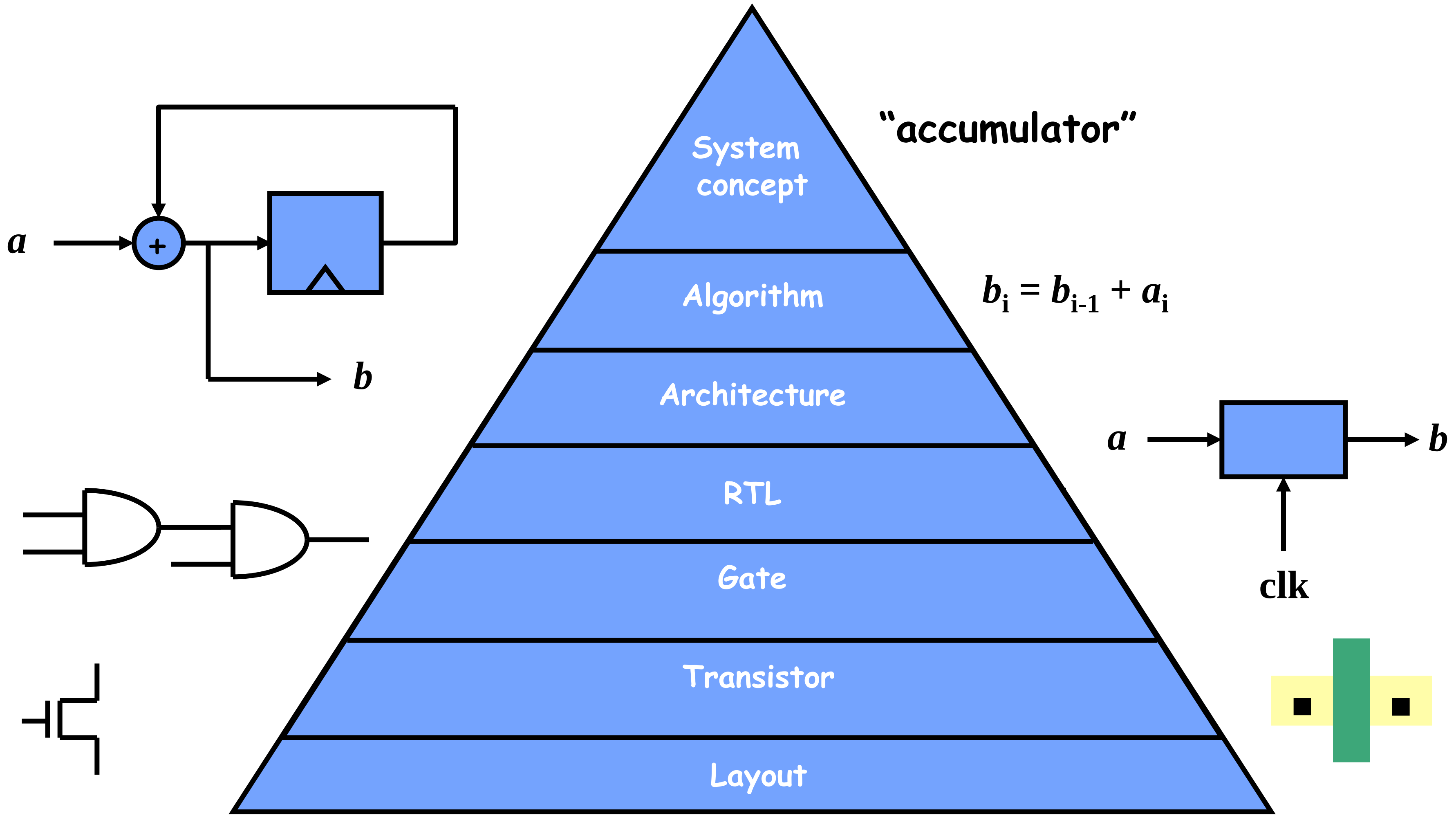
KAMRAN ESHRAGHIAN

$$H(n) = \sum_{n=0}^{M} k_n x_n$$

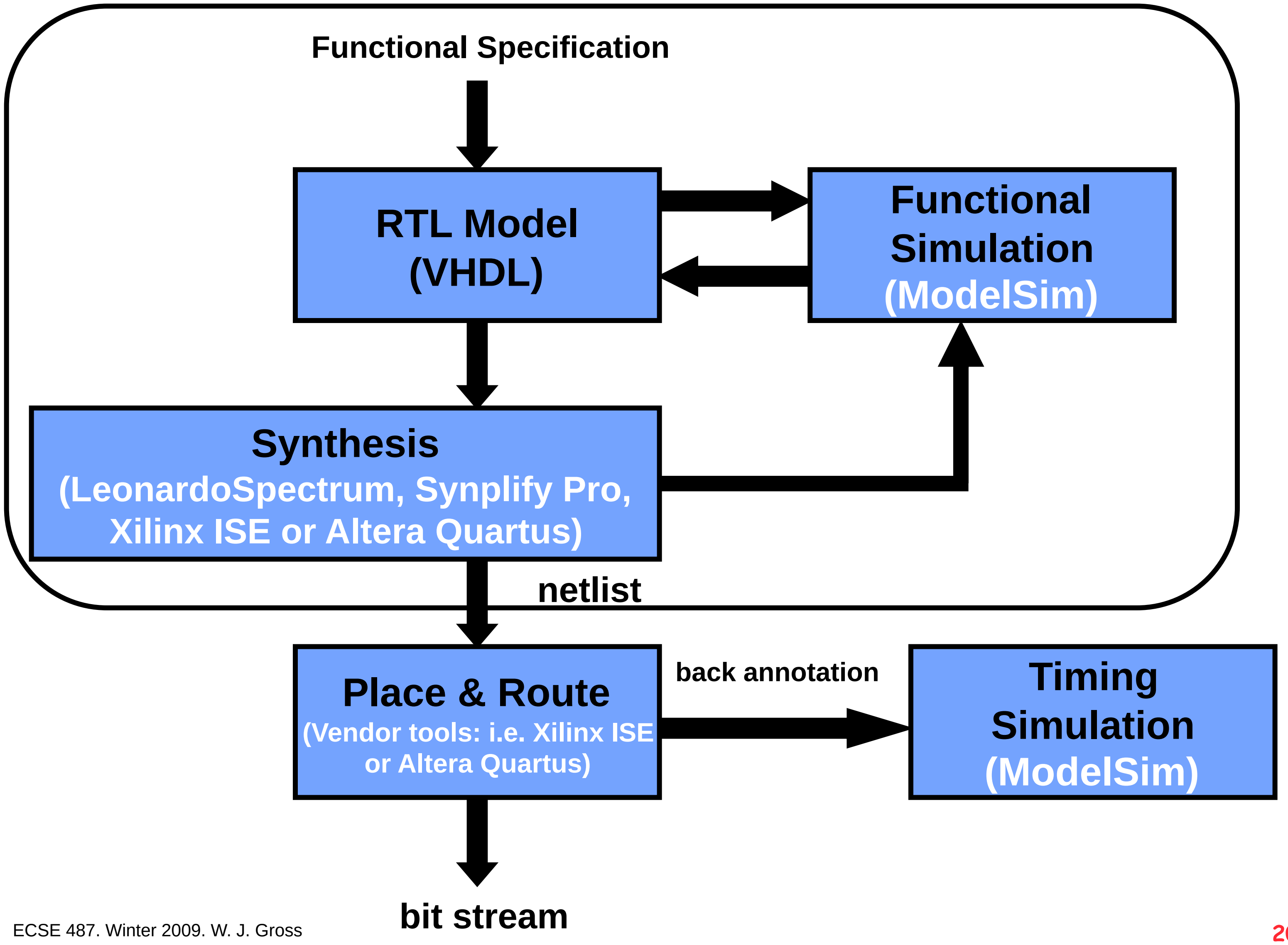
Behavioral Levels of Abstraction



Behavioral Levels of Abstraction

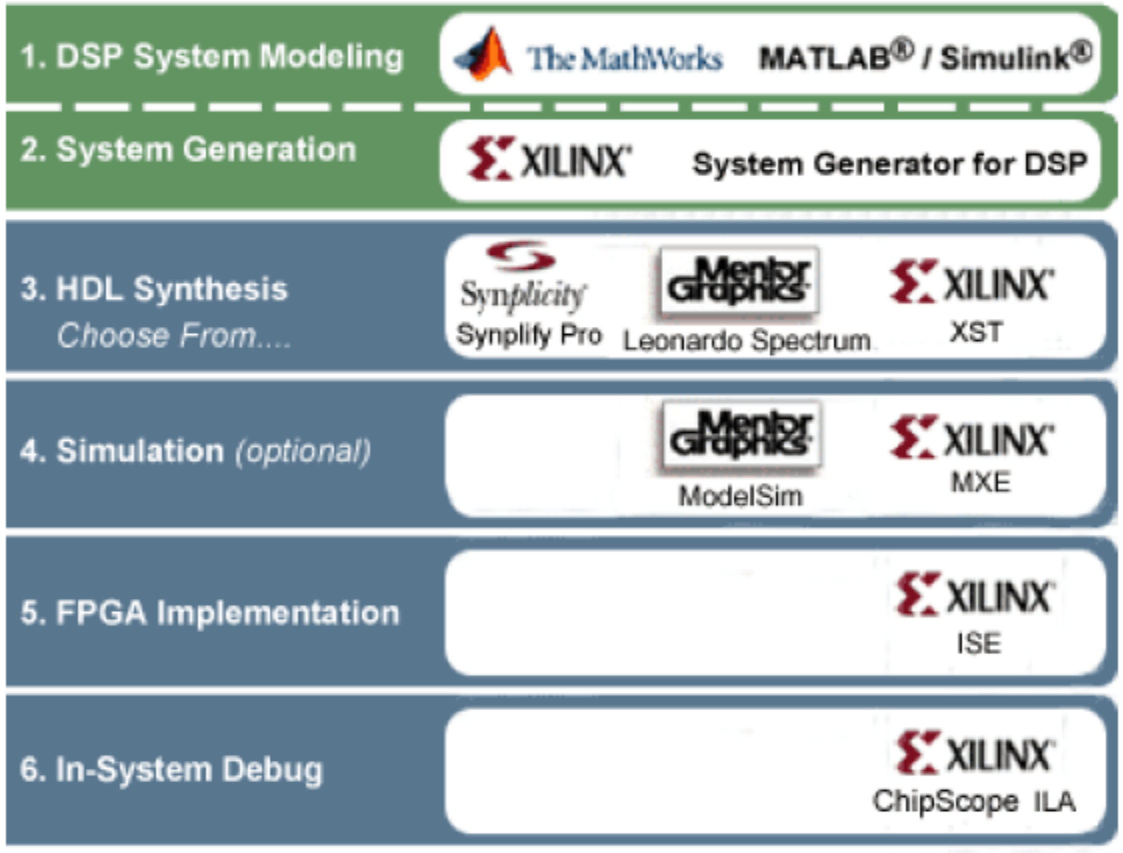


FPGA Software Flow



Functional Specification
RTL Model (VHDL)
Functional Simulation (ModelSim)
Synthesis (LeonardoSpectrum, Synplify Pro, Xilinx ISE or Altera Quartus)
netlist
Place & Route (Vendor tools: i.e. Xilinx ISE or Altera Quartus)
back annotation
Timing Simulation (ModelSim)
bit stream

XtremeDSP Design Flow

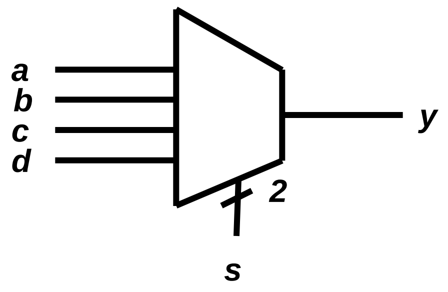


Hardware Description Languages

- Language used to model the intended operation of a piece of hardware (not a programming language)
- VHDL (VHSIC Hardware Description Language)
- Verilog
  - VHDL is slightly biased towards higher-level descriptions and Verilog slighlty biased towards lower-level descriptions (won't be a factor in 487)
  - Verilog is generally considered more concise and easier to learn and use
- We will use VHDL

Why VHDL is so Annoying

- Verbose
  - self documenting
- Multiple ways to model the same thing
- E.g. 4-1 multiplexor: 5 ways !



RTL Synthesis

- Complex designs need a way to separate the behavioral specification of the hardware from the actual implementation (e.g. gates)
- Register Transfer Level (RTL) synthesis is the process of translating an RTL model of hardware, written in a hardware description language at the register transfer level, into an optimized technology specific gate level implementation (D. J. Smith, 1996)

Synthesizable VHDL

- A subset of VHDL that can be used as the input to a synthesis program
- VHDL contains lots of programming language like constructs that can be used to make testbenches (file i/o, etc…) that can not result in synthesizable hardware

Synthesizable VHDL

- Some VHDL constructs are useful for modeling hardware, but not for synthesizing it
  - e.g.

```
        a <= '1' after 5 ns
```

is NOT synthesizable.

VHDL is NOT Software

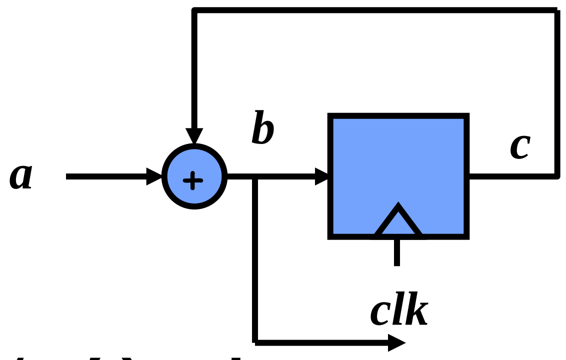
- If you insist on viewing it like software, it is most like a parallel programming model (communicating processes)

- "If you can't draw it, it won't work"
  - "If you don't simulate it, it won't work"
  - "If you don't test it, it won't work"

- ALWAYS draw an RTL level schematic (MUX's, registers, etc..)
  - (don't need to worry about the gate-level implementation, the synthesizer will take care of that)
  - EXTREMELY important in synchronous design
  - "I'm off by one clock cycle and I can't figure out why!"
  - Count flip-flops and latches !

Example: Accumulator

```
architecture RTL of accum is
begin
  b <= a + c;
  p1: process (b, clk)
  begin
     if (clk’event and clk = ‘1’) then
        c <= b;
     end if;
end architecture RTL;
```



The Commandments of Digital Design

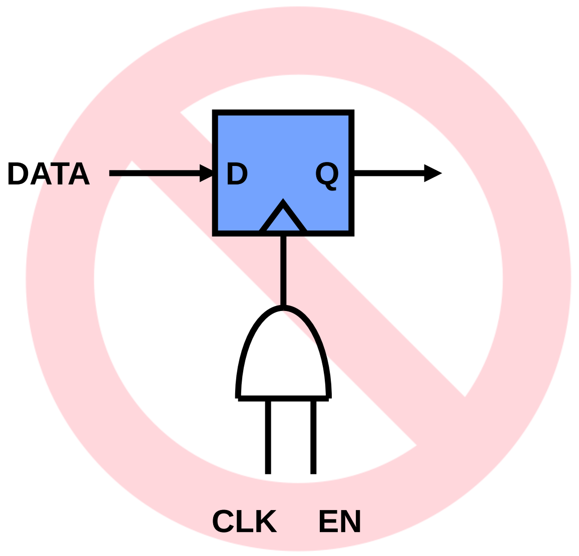
- Courtesy of Prof. John Knight (Carleton U.)
- We will look at the ones most important to us.

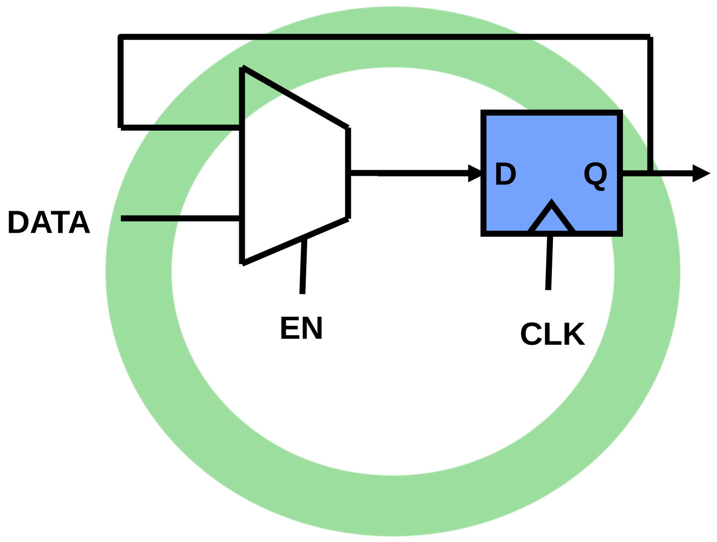
I. Only one clock

- Though shalt have only one Master clock and thou shalt not build false idolatrous clocks from astables

II. Do not "Gate" the Clock

- Though shalt not gate the clock, for that causeth false clock edges and skew.
  - Do not put any logic into the path of the clock – use enable inputs on flip-flops instead





III. Do not Use Asynchronous Circuits

- Thou shalt make all circuits synchronous unless thou canst convince he who pays thy salary, or assigns thy marks, that for reasons such as speed, pulse capture, or paper publishing, synchronous circuits cannot serve thy purpose.

V. Reset all Flip Flops

- Thou shalt have a master reset for all flip-flops so that the test engineer will love you, and your simulations will not remain undefined for time eternal.

VIII. Do Not Use Async. Reset

- Asynchronous reset was not conceived for tasks such as returning a count to zero. Verily I say thee, that six circuits created after that manner will clear properly and bring honour to thy name, but the seventh shall fail and carry thee down in shame and disgrace.

IX. Inputs must be Synchronized

- Raw asynchronous inputs are unclean, and must be cleansed by passing through a D flip flop, before they are allowed access to thy pure and chaste state variables.

X. Only Break the Commandments if you are ABSOLUTELY sure you know what you are doing

- Ye who completely understand the reasons for the commandments, then ye also know what liberties can be taken with them. Ye who would break them in ignorance, beware.

Next Time

- Refresher of basic VHDL concepts
- Assignment 1 Discussion
- Remember: Software flow tutorial.