EELE 414 – Introduction to VLSI Design

Module #1 – Introduction & Economy

- **Agenda**

  1. Course Logistics
  2. Course Content
  3. Design Implementation Options
  4. VLSI Economy

- **Announcements**

  1. Welcome
  2. All assignments are posted to the course website
  3. Please check your email regularly
  4. There is a digital circuit review at the end of these lecture notes
  5. Read Chapter 1

Course Overview

• **Office Hours:**	Check my website for most recent schedule. Also available by email appointment	
• **Requisites:**	Pre-requisite EE262, EE317	
• **Grading:**	Homework	- 40%
	Exam #1	- 20%
	Exam #2	- 20%
	Exam #3	- 20%

- Homework Assignments will be posted on Tuesdays

- Homework Assignments are due at the beginning of class on the following Thursday. (i.e., 1.5 weeks later)

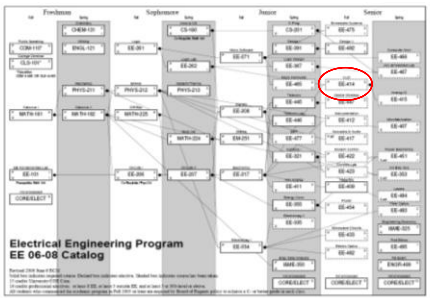
- Late homework will be accepted for one week after the due date with a penalty of 50% point reduction. No credit will be given for assignments over one week late.

- No make up exams will be given. Plan on being available on the exam dates.

- Check website frequently for the most up to date schedule.

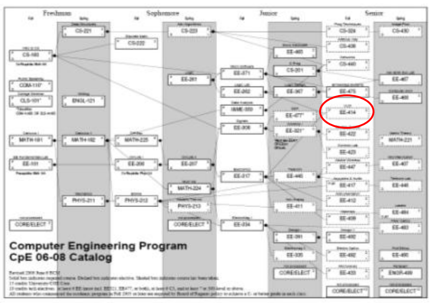
Course Overview

- **Where does this course fit into the <u>Electrical Engineering</u> curriculum?**



Course Overview

- **Where does this course fit into the <u>Computer Engineering</u> curriculum?**



Course Content

- **What is this course?**

  - An introductory course into VLSI Circuit Design

  - We will learn the transistor level implementation of the digital logic blocks used in VLSI designs

  - We will learn the design and analysis, simulation, layout, and fabrication of these circuits

  - We will learn to use modern CAD tools to design these circuits

- **What topics will be covered?**

  1) VLSI Implementation Options / Economy
  2) MOSFET Characteristics
  3) SPICE Modeling
  4) Circuit Layout
  5) CMOS Fabrication
  6) Static Behavior of a MOS Inverter
  7) Dynamic Behavior of a MOS Inverter
  8) CMOS Logic Gates
  9) Sequential Logic
  10) SRAM
  11) DRAM

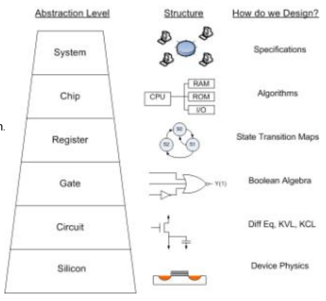
Course Content

- **At What level can we design?**

  - a level of abstraction allows the description of larger and more complex systems

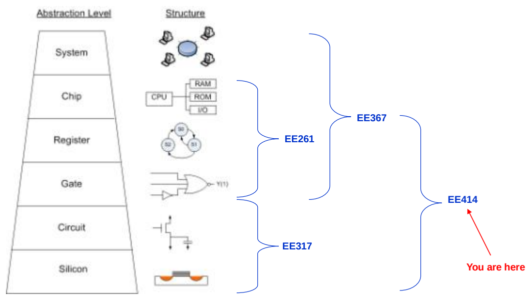
  - but we lose touch with the details of the implementation

  - as engineers, we want to have experience in all levels of abstraction.



Course Content

- **What areas of VLSI do my EE/CpE courses cover?**



Course Content

- **CAD / CAE Tools**

- Modern designs are too large to be designed using manual techniques

- We rely on a set of CAD / CAE tools in order to manage the complexity of the designs

- We won't use all of the tools for a VLSI design in this course, but it is good to understand the basic options and how they are used in a typical VLSI design

Analog Simulators
- SPICE, Cadence Spectre
- Simulation Program for the Integrated Circuit Environment
- Numerical Solver that performs KVL, KCL, and Branch Equations
- Performs Numerical Integration for transient simulations

Advantages
- Real analog view of signals

Disadvantages
- long run time limits number of nodes that can be simulated
- typically used for 10's of transistors
- full VLSI designs can't be simulated using SPICE

Course Content

Digital Simulators
- ModelSim, Xilinx, NC-Verilog
- only gives the state of the node (0,1,X,x,Z,z,…)
- this simplifies computation which reduces run time
- allows 1000's of nodes to be simulated simultaneously

Advantages
- many nodes can be simulated
- relevant for VLSI systems

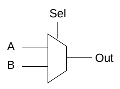
Disadvantages
- only see the state, not the analog nature

Analog vs. Digital Simulation

- We use Analog simulation to verify the operation of basic building blocks (inverters, NANDs)
- From the analog results, we define specs used in the digital simulation (t_{rise}, t_{fall}, t_{prop}, fanout)
- the digital simulations then incorporate the specs of the block as timing delays

Course Content

Design Entry
- Schematics

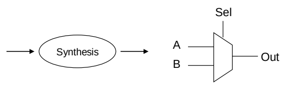


- Hardware Description Language (HDL)

```
if (Sel = 0)
   Out = A
else
   Out = B
```

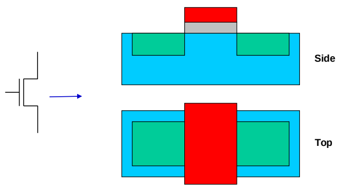
Synthesis
- Automatically generates the gate level netlist from an HDL

```
if (Sel = 0)
   Out = A
else
   Out = B
```



Course Content

Layout
- Physical implementation of the circuitry



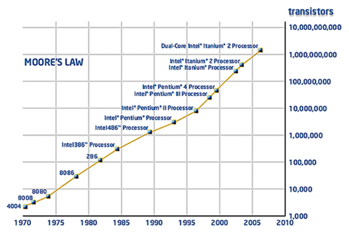
DRC / LVS
- Design Rule Checker: Were manufacturing capabilities violated
- Layout versus Schematic Check

Course Content

VLSI Design Sizes

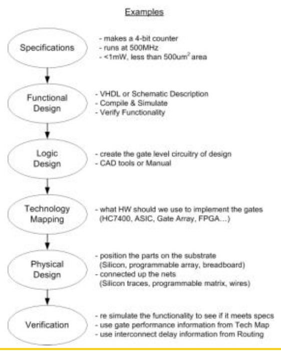
- more transistors are being integrated on chip

- this means we must rely more heavily on CAD/CAE



Course Content

VLSI Design Process


Examples
Specifications
- makes a 4-bit counter
- runs at 500MHz
- <1mW, less than 500um² area
Functional Design
- VHDL or Schematic Description
- Compile & Simulate
- Verify Functionality
Logic Design
- create the gate level circuitry of design
- CAD tools or Manual
Technology Mapping
- what HW should we use to implement the gates (HC7400, ASIC, Gate Array, FPGA...)
Physical Design
- position the parts on the substrate (Silicon, programmable array, breadboard)
- connected up the nets (Silicon traces, programmable matrix, wires)
Verification
- re simulate the functionality to see if it meets specs
- use gate performance information from Tech Map
- use interconnect delay information from Routing

Design Implementation Options

Implementations Options

- once a design is defined at the gate level, there are many options to implement the design

- each option is an engineering trade-off between

1) Cost	: up-front, engineering time, per-piece
2) Schedule	: how long does it take to get to market
3) Performance	: does the final design meet spec (features, speed, power, area)

- VLSI designs are especially difficult due to the rapid progression of fabrication technology

  - a new process is available every 18-36 months
  - each process is faster, small, and takes less power

- product life cycles can also be short

  - if development takes too long, a competitor can have a competing product in a better process
  - the product may be obsolete before enough are sold to cover the investment

Design Implementation Options

Custom ASIC

- "Application Specific Integrated Circuit"
- each gate is designed, laid out, and optimized by hand

Advantages
- Best circuit performance
- Best use of area on silicon

Disadvantages
- Long Design Cycle
- Full Custom Mask = More up-front $$$
- Takes skilled physical design engineers

Standard Cell

- all of the gates and basic building blocks are designed
- each block has a spec sheet, layout, symbol, HDL instance, and simulation deck
- the designer combines the pre-existing blocks to form the new ASIC
- still considered an ASIC

Advantages
- faster development
- still relatively fast in performance

Disadvantages
- not as much optimization in performance, area, power as a custom ASIC

Design Implementation Options

Gate Array

aka

"structured ASIC"

- transistors are already created but not connected
- the designer provides the interconnect design to implement the given functionality

Advantages
- faster development time

Disadvantages
- less performance

FPGA

- Field Programmable Gate Array
- an array of programmable logic blocks are designed and packaged
- the designer creates a programming file to implement the given functionality
- user downloads the file and is running in hardware without any fab

Advantages
- fastest development

Disadvantages
- lowest performance

Design Implementation Options

Trade-offs

	Up-Front Cost	Development Time	Per-Piece Cost
Custom ASIC	most ↑	most ↑	least ↓
STD Cell ASIC			
Gate Array			
FPGA	least	least	most

VLSI Economy

Economy

- there are two general types of economy in VLSI

  1) you are going to use the IC design in your own product you sell
  2) you design and sell the individual IC

- both require a financial analysis that includes:

  1) upfront cost
  2) engineering development cost
  3) per piece cost
  4) predicted monthly volume
  5) predicted life span of product

- items 1,2, and 3 are deterministic

- items 4 and 5 are marketing "best guesses" and probably the most difficult thing to predict in business

VLSI Economy

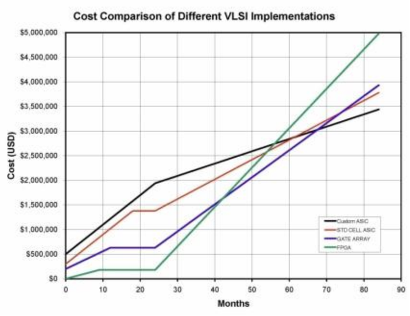
Economy Example of "Using your IC"

- your company is developing a product that will use a VLSI design. You are to choose which implementation option is the best between a Custom ASIC, STD Cell ASIC, Gate Array, or FPGA
- in this case, all 4 of the implementation options meet the electrical specifications so it a simple matter of economy in selecting the best option
- the product that this IC goes into will ship in 24 months regardless of the development time of your IC (assuming it is done of course!)
- your marketing department thinks that you will sell 5,000 product per month and that the product life cycle is 5 years (starting after the 24 months of development)
- you are given the following:

	NRE	Engineering	Per-Piece Cost
Custom ASIC	\$500k	4 engineers, 24 months, \$15k/engr/mo	\$5.00
STD Cell ASIC	\$300k	4 engineers, 18 months, \$15k/engr/mo	\$8.00
Gate Array	\$200k	3 engineers, 12 months, \$12k/engr/mo	\$11.00
FPGA	\$0k	2 engineers, 9 months, \$10k/engr/mo	\$16.00

VLSI Economy

Economy Example

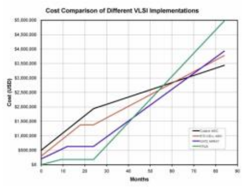


VLSI Economy

Economy Considerations

Quantitative considerations for this example….

	Answer
- Which options takes the most development?	ASIC
- If the product runs 5 years, which option is the most cost effective?	ASIC
- What if the product only sells for 1 year?	FPGA
- What if the product only sells for 2 years?	Gate Array

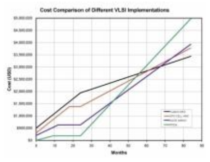


VLSI Economy

Economy Considerations

Qualitative considerations for this example….

	Idea
- If performance mattered, how could you justify doing a Custom ASIC?	You could charge more for the product
- What if you are a small company and can't afford the NRE?	You might be forced to use an FPGA
- What is the danger of doing an FPGA?	It might not take long for a competitor to develop the same thing if they see you are successful selling your product



VLSI Economy

Economy of Selling your IC

- if you design and sell your IC, you need a similar analysis.
- But now you need to consider

1) Profit Margin - for each IC sold, how much of that is gross profit versus the cost of simply making the IC

2) Break Even Point - you invest upfront NRE and engineering time. The gross profit from the first x IC's that you sell go toward paying off that investment. The # of IC's and the time it takes to reach that point is called "Break Even"

- If the product doesn't sell enough, you may not make enough to pay for the initial development.

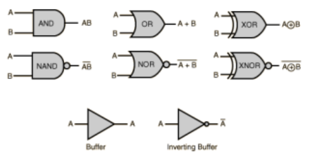
3) Opportunity Cost - if you only have 5 skilled IC designers, you want to make sure they are working on the most profitable product. If they are working on product A then they can't be working on product B. If B is more profitable, you are not using your resources effectively.

Digital Review

Combinational Logic

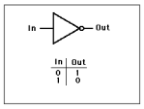
Combinational Logic Gates :

- Output depends on the logic value of the inputs
- no storage



Digital Review

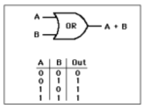
NOT



In	Out
0	1
1	0

out = in' = $\overline{\text{in}}$ f(in) = in' = $\overline{\text{in}}$

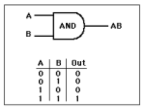
OR



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

out = a+b f(a,b) = a+b

AND

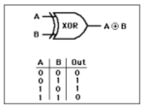


A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

out = a·b f(a,b) = a·b

Digital Review

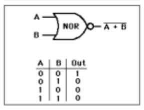
XOR



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

out = a⊕b f(a,b) = a⊕b

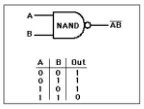
NOR



A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

out = $\overline{a+b}$ f(a,b) = $\overline{a+b}$

NAND

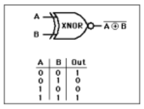


A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

out = $\overline{a \cdot b}$ f(a,b) = $\overline{a \cdot b}$

Digital Review

XNOR



A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1

out = $\overline{a \oplus b}$ f(a,b) = $\overline{a \oplus b}$

Also remember about XOR Gates:

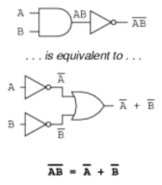
f(a,b) = a⊕b = (a'b + b'a)

Also remember the priority of logic operations (without parenthesis) is:
NOT, AND, OR

Digital Review

DeMorgan's Theorems

- Inverting the output of any gate results in the same function as the opposite gate (AND/OR) with inverted inputs


A
B
AB
$\overline{AB}$
. . . is equivalent to . . .
A
B
$\overline{A}$
$\overline{B}$
$\overline{A} + \overline{B}$
$\overline{AB} = \overline{A} + \overline{B}$

Digital Review

DeMorgan's Theorems

- Graphically : breaking the bar changes the logic function (AND-OR) under the break

out = $\overline{a+b}$

1) Break bar

out = $\overline{a}\overline{+}\overline{b}$

2) Change + to · under break

out = $\overline{a} \cdot \overline{b}$

Digital Review

Boolean Expressions Using SOP

- Logic functions can be described using a Sum of Products techniques
- Sum of Products (SOP) is the summation of all *minterms* resulting in the truth table
- A minterm is the expression for an input configuration which yields a TRUE output
- A minterm expression is the AND'ing of the input "1" signal configuration

Truth Table

a b	out	
0 0	0	
0 1	1	*minterm* m_1 = a'·b
1 0	1	*minterm* m_2 = a·b'
1 1	0	

SOP Expression : f(a,b) = a'·b + a·b'

Note : un-minimized Boolean expression

Digital Review

Boolean Expressions Using POS

- Logic functions can be described using a Product of Sums techniques
- Product of Sums (POS) is the multiplication of all *maxterms* resulting in the truth table
- A maxterm is the expression for an input configuration which yields a FALSE output
- A maxterm expression is the OR'ing of the input "0" signal configuration

Truth Table

a b	out
0 0	0
0 1	1
1 0	1
1 1	0

maxterm $m_0 = a+b$ (input configuration of 0's)

maxterm $m_3 = a'+b'$ (input configuration of 0's)

POS Expression : $f(a,b) = (a+b) \cdot (a'+b')$

Digital Review

Boolean Expressions Using SOP & POS

- SOP and POS functions are equivalent

SOP Expression : $f(a,b) = a' \cdot b + a \cdot b'$

is equal to

POS Expression : $f(a,b) = (a+b) \cdot (a'+b')$

Digital Review

Karnaugh Maps

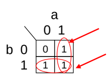
- K-maps provide a graphical method to find SOP/POS expressions
- K-maps also provide a graphical method to perform logic minimization

K-map SOP Process

1) Circle minterms to create SOP
2) Circle in Horizontal & Vertical manner
3) Circle in groups with powers of 2 (1,2,4,8,…)

Truth Table

a b	out
0 0	0
0 1	1
1 0	1
1 1	1



No dependency on b, minterm = a

No dependency on a, minterm = b

SOP expression : $f(a,b) = a + b$

Digital Review

Karnaugh Maps

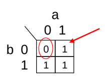
- K-maps provide a graphical method to find SOP/POS expressions
- K-maps also provide a graphical method to perform logic minimization

K-map POS Process

1) Circle maxterms to create POS
2) Circle in Horizontal & Vertical manner
3) Circle in groups with powers of 2 (1,2,4,8,…)

Truth Table

a b	out
0 0	0
0 1	1
1 0	1
1 1	1



Dependency on a' and b', maxterm = a+b

POS expression : $f(a,b) = a + b$

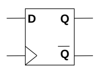
Digital Review

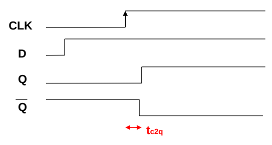
Sequential Logic

- Concept of "Storage Element"
- With Storage, logic functions can depend on current & past values of inputs
- Sequential State Machines can be created

D-Flip-Flop

- on timing event (i.e., edge of clock input), D input goes to Q output





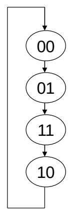
Digital Review

State Machines

- Moore :	Outputs depend on present state
- Mealy :	Outputs depend on present state and current inputs

Digital Review

State Machine Example : Design a 2-bit Gray Code Counter

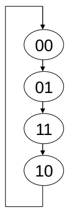


1) Number of States?	: 4
2) Number of bits to encode states?	: $2^n=4$, n=2
3) Moore or Mealy?	: Moore

For this counter, we can make the outputs be the state codes

Digital Review

State Machine Example : Design a 2-bit Gray Code Counter



STATE

Current A_{cur}	Current B_{cur}	Next A_{nxt}	Next B_{nxt}
0	0	0	1
0	1	1	1
1	1	1	0
1	0	0	0

A_{nxt} Logic

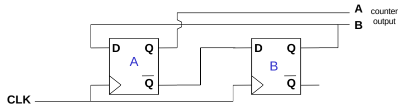
A_{cur} \ B_{cur}	0	1
0	0	1
1	0	1

$A_{nxt} = B_{cur}$

B_{nxt} Logic

A_{cur} \ B_{cur}	0	1
0	1	1
1	0	0

$B_{nxt} = A_{cur}'$


A
B
counter output
D
Q
A
$\overline{Q}$
D
Q
B
$\overline{Q}$
CLK