<u>ComSci M51A</u>
Lecture: Tuesdays, Thursdays 8:00 to 9:50
Discussion: Fridays 10:00 to 11:50

- **<u>Chapter 1-Introduction</u>**
  - **1.1-About Digital Systems**
    - Digital system-a system in which signals have a finite number of discrete values
    - Analog systems-signals have values from a continuous infinite set
    - Systems with instants labeled at digital values labeled at discrete instants in which signals change due to the time being discretized are known as synchronous
    - Those which changes may occur at any instant are called asynchronous
    - Why are digital systems important?
      - Digital systems are used in information processing
        - Digital representation is well suited both numerical and nonnumerical information processing
        - Information processing can use a general-purpose system (a computer) which is programmed for a particular processing test
        - The finite number in digital signals can be represented with binary signals
          - Signals all represent just open and closed
        - Digital signals are quite insensitive to variations of component parameter values
          - Especially true for binary signals
        - Numerical digital systems can be made more accurate by adding the number of digits used in the representation
        - The advances of microelectronics technology in recent years have made possible the fabrication extremely complex digital signals, which are small, fast and cheap
          - Complex digital systems are built as integrated circuits composed of a large number of very simple devices
        - It is possible to select among different implementations of systems that trade-off speed and amount of hardware
      - When are digital systems used?
        - Analog and digital signals
          - Signals in there real world are analog, so it is necessary for us to convert digital signals to analog signals
          - Process of converting from analog to digital is called quantization or digitization
        - Combinational and sequential systems
          - Digital systems have two classes: combinational systems and sequential systems
            - Combination systems-the output at time t depends only on the input
              - These systems have no memory, because the system does not depend on previous systems
            - Sequential systems-the output at time t depends on the input at time t and possibly also depends on the input prior to t
  - **1.2-Specification and Implementation, Analysis and Design**
    - The specification of a system refers to a description of its function and of other characteristics required for its use
    - Specification is related to what the system without reference to how it performs the operation
    - Specs should be complete and as simple as possible
    - Specifications of a systems must describe its function in a way that is adequate for two purposes:
      - To use the system as a component in more complex systems
      - To serve as the basis for the implementation of the system by a network of simpler components
    - On the other hand, an implementation of a system refers to how the system is constructed from simpler components
      - With digital networks, this consists of the interconnection of digital modules
        - This network can be defined at many levels, ranging from complexity of the primitive models
          - I.e. gates to complex processors
    - At the physical level, all digital systems are implemented by a complex interconnection of elementary elements such as transistors, resistors, and so on
    - It is necessary to define intermediate levels of modules of increasing complexity, whose descriptions

        - Modules are more than just conceptual entities used to simplify description of an implementation
            - Often built separately and then put together to finish the larger system
        - The distinction between separation of specification of a system and its implementation is very important in complex systems because
            - It shields the description required to use the system from irrelevant implementation details
            - It allows choosing an implementation from different alternatives without influencing the description required for using the system
        - Analysis of a system has the objective to determine its specification from implementation
        - Design process consists of obtaining an implementation that satisfies the spec of a system
            - Top-down decomposes itself into subsystems in which are decomposed into smaller systems
                - Has the disadvantage that no systematic procedure exists to assure the decomposition at a particular level optimizes the final implementation
            - Bottom-up-Connects available modules to form subsystems and these subsystems are connected to the subsystems until the required functional specification is fulfilled
                - Disadvantage is similar to the top-down case
        - Levels of an implementation
            - Module level consists of two registers and adder
            - Logical level consists of modules implemented with gates and flip-flops and signals are binary
            - Physical level are components that are realized in some technology
    - **1.3-Computer-aided design tools**
        - CAD is used to help design of digital systems be efficient, timely, and economical
        - Description of digital systems is performed in a hierarchical manner
            - Description provides a logic diagram of the system at different levels, showing the modules and their interconnections
            - This process is called a schematic capture because the tool used to capture the schematic description of the digital system
            - Process is supported by libraries of standard components
            - An alternative method is using a hardware-description language
        - Synthesis and optimization tools help in obtaining an implementation from a given description in improving some characteristics such as the number of modules and network delays
        - Simulation tools are used to verify the operation of a system
            - These can be used to detect errors in a design and to determine characteristics
- **<u>Chapter 2-Appendix A-Boolean Algebras</u>**
- Boolean algebras is an important class of algebras
- Switching algebra is an instance of boolean algebras
    - **A.1-Boolean Algebra**
        - Boolean algebra is a tuple {B, +, .}, where in
            - B is a set of elements
            - + and . Are binary operations applied over the elements of B
        - Which satisfies the following postulates
            - P1: Commutative
                - a+b=b+a
                - a . b=b . a
            - P2: Associative
                - a + (b . c) = (a +b) . (a + c)
                - a . (b + c) = (a . b) + (a . c)
            - P3: Identity
                - 0 + a = a + 0 = a
                - 1 . a = a.1 a
            - P4: Complement
                - a+a’=1
                - a . A’=0
    - **A.2-Switching Algebra**
        - Switching algebra is a system used to describe switching functions by means of switching expressions
        - Switching algebra contains B= {0,1} and operations AND and OR

        - Operations are used to evaluate switching expressions
    - Theorem 1
        - The switching algebra is a Boolean Algebra
    - P1: Commutativity of (+), (.)
        - Shown by inspection of the operation tables
            - Holds if operation is symmetric about the main diagonal
    - P2: Distributivity of (+), (.)
        - Shown by perfect induction, by considering all possible values of the elements a, b, c
    - P3: Existence of additive and multiplicative identity element
        - $0 + 1 = 1 + 0 = 1$
        - $0 \cdot 1 = 1 \cdot 0 = 0$
    - P4: Existence of the complement
        - 1 is the complement of 0 and 0 is the complement of 1
- **A.3-Important Theorems in Boolean Algebra**
    - Theorem 2: Principle of duality
        - Every algebraic identity deducible from the postulates of a Boolean algebra remains valid if
            - The operations + and . Are interchanged throughout
            - The identity elements 0 and 1 are also interchanged throughout
    - Theorem 3
        - Every element in B has a unique complement
    - Theorem 4
        - For any a in B p-p ,.
            - $1 + a = 1$
            - $0 \cdot a = 0$
    - Theorem 5
        - The complement of the element 1 is 0 and vice versa
    - Theorem 6 Idempotent Law
        - For every a in B
            - $a + a = a$
            - $a \cdot a = a$
    - Theorem 7 Involution Law
        - For every a in B,
            - $(a')' = a$
    - Theorem 8 Absorption Law
        - For every pair of element a, b in B
            - $a + a \cdot b = a$
            - $a \cdot (a + b) = a$
    - Theorem 9
        - For every pair of elements a, b in B
            - $a + a' = a + b$
            - $a(a' + b) = ab$
    - Theorem 10
        - In a Boolean algebra, each of the binary operators, (+) and (.) is associative. That is, for every a, b, c in B
            - $a + (b + c) = (a + b) + c$
            - $a(bc) = (ab)c$
    - Corollary 1
        - The order in applying the + and . operator among n elements does not matter
    - Theorem 11 DeMorgan's Laws
        - For every pair of elements a,b in B:
        - $(a + b)' = a'b'$
        - $(ab)' = a' + b'$
    - Theorem 12 Generalized DeMorgan's Laws
        - Let {a, b,…, c, d} be a set elements in a Boolean algebra. Then, the following identities hold:
            - $(a + b + \ldots + c + d)' = [(a + b + \ldots + c) + d]'$
                - $= (a + b + \ldots + c)'d'$

        - Other examples of Boolean Algebras
            - Algebra of sets
            - Algebra of Logic (Propositional Calculus)
- **<u>Chapter 3-Combinatorial Integrated Circuits: Characteristics and Capabilities</u>**
    - Logic gates-and, or, nor, etc
    - Circuit level-Physical representation of the gates
    - PMOS-Pull-up
    - NMOS-Pull-down
    - The not gate is formed by simply two gates inverting the input value to allow for output
    - AND and OR gates must be created by adding a NOT gate to invert the value of the circuit
    - Transmission gates are often used to represent XOR
    - We cannot get instantaneous change, so we will have propagation delay
    - Total load is the sum of all load factors
        - Assume load factor is 1
- **<u>Chapter 4-Description and Analysis of Gate Networks</u>**
    - Gate network-Interconnections of data
    - Connections all carried signals
    - Description of a gate network contains
        - Graphical representation (Logic diagram)
        - Tabular representation (a net list)
        - Representation based on a hardware description language (a set of language statements
    - Specification of a gate networks includes
        - Functional specification
        - The input load factors of the network inputs
        - The fanout factors of the network outputs
        - The propagation delays through the network
    - Universal sets
        - {AND, OR, NOT}
        - {AND, NOT}
        - {OR, NOT}
        - {NAND}
        - {NOR}
- **<u>Chapter 5-Design of Combinational Systems: Two-Level Gate Networks</u>**
    - Two level networks are networks with two layers of gates
    - Karnaugh maps are a method of determining a minimal expression via a visual representation
    - Implicant is a value 1 in the karnaugh map
    - Prime implicant is the larges 2^(n) value of grouping 1s
    - Essential Prime implicants contain at least one 1 which is not covered by any other primes
- **<u>Chapter 7-Specification of Sequential Systems</u>**
    - Sequential system
    - Synchronous and asynchronous
    - State diagrams
    - Initial case
    - States influence the output
    - Note input also influence output
    - Two types of state diagrams
- **<u>Chapter 8-Sequential Networks</u>**
    - Sequential Network is a collection of combinational and sequential networks
    - Canonical Implementation
        - Huffman-Moore implementation is based directly on the state description of a system
        - State register stores the state
        - Combinational network implements the transition and output functions
        - Synchronizing signal determines time instants at which the next state is loaded into the state register
        - Clock pulses
        - Initialize is the input to initialize the state

- Flip-Flop
  - JK
  - D
  - SR
  - T
- Only flips the output value
- We only really care about how everything initiates wrt the clock
- Clock will always run at any given time

- **Chapter 9-Standard Combinational Modules**
  - n-input binary decoder-combinational system that has n binary outputs
  - Coincident decoding and tree decoding
  - 2^(n) output binary-decoder
  - Additional I/O can be used with module enable E
  - 2^(n) input binary-multiplexer
  - 2^(n) output binary-demultiplexer
  - Multiplexer trees
  - Simple Shifter
  - p-shifter
  - Unidirectional shifter
- **Chapter 10**
  - Adders
    - Full
      - With carry in and carry out
    - Half
      - Without carry in and carry out
  - Carry ripple adder