COURSE LEARNING OUTCOMES AND OBJECTIVES

A student who successfully fulfills the course requirements will have demonstrated:

1. **an ability to analyze and design CMOS logic gates**
    - 1-1. <u>convert</u> numbers from one base (radix) to another: 2, 10, 16
    - 1-2. <u>define</u> a binary variable
    - 1-3. <u>identify</u> the theorems and postulates of switching algebra
    - 1-4. <u>describe</u> the principle of duality
    - 1-5. <u>describe</u> how to form a complement function
    - 1-6. <u>prove</u> the equivalence of two Boolean expressions using perfect induction
    - 1-7. <u>describe</u> the function and utility of basic electronic components (resistors, capacitors, diodes, MOSFETs)
    - 1-8. <u>define</u> the switching threshold of a logic gate and identify the voltage ranges typically associated with a "logic high" and a "logic low"
    - 1-9. <u>define</u> assertion level and <u>describe</u> the difference between a positive logic convention and a negative logic convention
    - 1-10. <u>describe</u> the operation of basic logic gates (NOT, NAND, NOR) constructed using N- and P-channel MOSFETs and <u>draw</u> their circuit diagrams
    - 1-11. <u>define</u> "fighting" among gate outputs wired together and describe its consequence
    - 1-12. <u>define</u> logic gate fan-in and describe the basis for its practical limit
    - 1-13. <u>identify</u> key information contained in a logic device data sheet
    - 1-14. <u>calculate</u> the DC noise immunity margin of a logic circuit and describe the consequence of an insufficient margin
    - 1-15. <u>describe</u> the consequences of a "non-ideal" voltage applied to a logic gate input
    - 1-16. <u>describe</u> how unused ("spare") CMOS inputs should be terminated
    - 1-17. <u>describe</u> the relationship between logic gate output voltage swing and current sourcing/sinking capability
    - 1-18. <u>describe</u> the difference between "DC loads" and "CMOS loads"
    - 1-19. <u>calculate</u> V_{OL} and V_{OH} of a logic gate based on the "on" resistance of the active device and the amount of current sourced/sunk by the gate output
    - 1-20. <u>calculate</u> logic gate fan-out and identify a practical lower limit
    - 1-21. <u>calculate</u> the value of current limiting resistor needed for driving an LED
    - 1-22. <u>describe</u> the deleterious effects associated with loading a gate output beyond its rated specifications
    - 1-23. <u>define</u> propagation delay and list the factors that contribute to it
    - 1-24. <u>define</u> transition time and list the factors that contribute to it
    - 1-25. <u>estimate</u> the transition time of a CMOS gate output based on the "on" resistance of the active device and the capacitive load
    - 1-26. <u>describe</u> ways in which load capacitance can be minimized
    - 1-27. <u>identify</u> sources of dynamic power dissipation
    - 1-28. <u>plot</u> power dissipation of CMOS logic circuits as a function of operating frequency
    - 1-29. <u>plot</u> power dissipation of CMOS logic circuits as a function of power supply voltage
    - 1-30. <u>describe</u> the function and utility of decoupling capacitors
    - 1-31. <u>define</u> hysteresis and describe the operation of Schmitt-trigger inputs
    - 1-32. <u>describe</u> the operation and utility of a transmission gate
    - 1-33. <u>define</u> high-impedance state and describe the operation of a tri-state buffer
    - 1-34. <u>define</u> open drain as it applies to a CMOS logic gate output and calculate the value of pull-up resistor needed
    - 1-35. <u>describe</u> how to create "wired logic" functions using open drain logic gates
    - 1-36. <u>calculate</u> the value of pull-up resistor needed for an open drain logic gate

2. **an ability to analyze and design combinational logic circuits**

    2-1. identify minterms (product terms) and maxterms (sum terms)
    2-2. list the standard forms for expressing a logic function and give an example of each: sum-of-products (SoP), product-of-sums (PoS), ON set, OFF set
    2-3. analyze the functional behavior of a logic circuit by constructing a truth table that lists the relationship between input variable combinations and the output variable
    2-4. transform a logic circuit from one set of symbols to another through graphical application of DeMorgan's Law
    2-5. realize a combinational function directly using basic gates (NOT, AND, OR, NAND, NOR)
    2-6. draw a Karnaugh Map ("K-map") for a 2-, 3-, 4-, or 5-variable logic function
    2-7. list the assumptions underlying function minimization
    2-8. identify the prime implicants, essential prime implicants, and non-essential prime implicants of a function depicted on a K-map
    2-9. use a K-map to minimize a logic function (including those that are incompletely specified) and express it in either minimal SoP or PoS form
    2-10. use a K-map to convert a function from one standard form to another
    2-11. calculate and compare the cost (based on the total number of gate inputs plus the number of gate outputs) of minimal SoP and PoS realizations of a given function
    2-12. realize a function depicted on a K-map as a two-level NAND circuit, two-level NOR circuit, or as an open-drain NAND/wired-AND circuit
    2-13. define and identify static-0, static-1, and dynamic hazards
    2-14. describe how a static hazard can be eliminated by including consensus terms
    2-15. describe a circuit that takes advantage of the existence of hazards and analyze its behavior
    2-16. draw a timing chart that depicts the input-output relationship of a combinational circuit
    2-17. identify properties of XOR/XNOR functions
    2-18. simplify an otherwise non-minimizable function by expressing it in terms of XOR/XNOR operators
    2-19. describe the genesis of programmable logic devices
    2-20. list the differences between complex programmable logic devices (CPLDs) and field programmable gate arrays (FPGAs) and describe the basic organization of each
    2-21. list the basic features and capabilities of a hardware description language (HDL)
    2-22. list the structural components of a Verilog program
    2-23. identify operators and keywords used to create Verilog programs
    2-24. write equations using Verilog syntax
    2-25. define functional behavior of combinational circuits using Verilog constructs
    2-26. define the function of a decoder and describe how it can be use as a combinational logic building block
    2-27. illustrate how a decoder can be used to realize an arbitrary Boolean function
    2-28. define the function of an encoder and describe how it can be use as a combinational logic building block
    2-29. discuss why the inputs of an encoder typically need to be prioritized
    2-30. define the function of a multiplexer and describe how it can be use as a combinational logic building block
    2-31. illustrate how a multiplexer can be used to realize an arbitrary Boolean function

3. an ability to analyze and design sequential logic circuits

3-1. describe the difference between a combinational logic circuit and a sequential logic circuit
3-2. describe the difference between a feedback sequential circuit and a clocked synchronous state machine
3-3. define the state of a sequential circuit
3-4. define active high and active low as it pertains to clocking signals
3-5. define clock frequency and duty cycle
3-6. describe the operation of a bi-stable and analyze its behavior
3-7. define metastability and illustrate how the existence of a metastable equilibrium point can lead to a random next state
3-8. write present state – next state (PS-NS) equations that describes the behavior of a sequential circuit
3-9. draw a state transition diagram that depicts the behavior of a sequential circuit
3-10. construct a timing chart that depicts the behavior of a sequential circuit
3-11. draw a circuit for a set-reset (“S-R”) latch and analyze its behavior
3-12. discuss what is meant by “transparent” (or “data following”) in reference to the response of a latch
3-13. draw a circuit for an edge-triggered data (“D”) flip-flop and analyze its behavior
3-14. compare the response of a latch and a flip-flop to the same set of stimuli
3-15. define setup and hold time and determine their nominal values from a timing chart
3-16. determine the frequency and duty cycle of a clocking signal
3-17. identify latch and flip-flop propagation delay paths and determine their values from a timing chart
3-18. describe the operation of a toggle (“T”) flip-flop and analyze its behavior
3-19. derive a characteristic equation for any type of latch or flip-flop
3-20. identify the key elements of a clocked synchronous state machine: next state logic, state memory (flip-flops), and output logic
3-21. differentiate between Mealy and Moore model state machines, and draw a block diagram of each
3-22. analyze a clocked synchronous state machine realized as either a Mealy or Moore model
3-23. outline the steps required for state machine synthesis
3-24. derive an excitation table for any type of flip-flop
3-25. discuss reasons why formal state-minimization procedures are seldom used by experienced digital designers
3-26. describe how state machines can be specified in Verilog
3-27. draw a circuit for an oscillator and calculate its frequency of operation
3-28. draw a circuit for a bounce-free switch based on an S-R latch and analyze its behavior
3-29. design a clocked synchronous state machine and verify its operation
3-30. define minimum risk and minimum cost state machine design strategies, and discuss the tradeoffs between the two approaches
3-31. compare state assignment strategy and state machine model choice (Mealy vs. Moore) with respect to PLD resources (P-terms and macrocells) required for realization
3-32. compare and contrast the operation of binary and shift register counters
3-33. derive the next state equations for binary “up” and “down” counters
3-34. describe the feedback necessary to make ring and Johnson counters self-correcting
3-35. compare and contrast state decoding for binary and shift register counters

3-36. describe why "glitches" occur in some state decoding strategies and discuss how to eliminate them
3-37. identify states utilized by a sequence recognizer: accepting sequence, final, and trap
3-38. determine the embedded binary sequence detected by a sequence recognizer

4. an ability to analyze and design computer logic circuits

4-1. compare and contrast three different signed number notations: sign and magnitude, diminished radix, and radix
4-2. convert a number from one signed notation to another
4-3. describe how to perform sign extension of a number represented using any of the three notation schemes
4-4. perform radix addition and subtraction
4-5. describe the various conditions of interest following an arithmetic operation: overflow, carry/borrow, negative, zero
4-6. describe the operation of a half-adder and write equations for its sum (S) and carry (C) outputs
4-7. describe the operation of a full adder and write equations for its sum (S) and carry (C) outputs
4-8. design a "population counting" or "vote counting" circuit using an array of half-adders and/or full-adders
4-9. design an N-digit radix adder/subtractor circuit with condition codes
4-10. design a (signed or unsigned) magnitude comparator circuit that determines if A=B, A<B, or A>B
4-11. describe the operation of a carry look-ahead (CLA) adder circuit, and compare its performance to that of a ripple adder circuit
4-12. define the CLA propagate (P) and generate (G) functions, and show how they can be realized using a half-adder
4-13. write the equation for the carry out function of an arbitrary CLA bit position
4-14. draw a diagram depicting the overall organization of a CLA
4-15. determine the worst case propagation delay incurred by a practical (PLD-based) realization of a CLA
4-16. describe how a "group ripple" adder can be constructed using N-bit CLA blocks
4-17. describe the operation of an unsigned multiplier array constructed using full adders
4-18. determine the full adder arrangement and organization (rows/diagonals) needed to construct an NxM-bit unsigned multiplier array
4-19. determine the worst case propagation delay incurred by a practical (PLD-based) realization of an NxM-bit unsigned multiplier array
4-20. describe the operation of a binary coded decimal (BCD) "correction circuit"
4-21. design a BCD full adder circuit
4-22. design a BCD N-digit radix (base 10) adder/subtractor circuit
4-23. define computer architecture, programming model, and instruction set
4-24. describe the top-down specification, bottom-up implementation strategy as it pertains to the design of a computer
4-25. describe the characteristics of a "two address machine"
4-26. describe the contents of memory: program, operands, results of calculations
4-27. describe the format and fields of a basic machine instruction (opcode and address)
4-28. describe the purpose/function of each basic machine instruction (LDA, STA, ADD, SUB, AND, HLT)
4-29. define what is meant by "assembly-level" instruction mnemonics

4-30. draw a diagram of a simple computer, showing the arrangement and interconnection of each functional block
4-31. trace the execution of a computer program, identifying each step of an instruction's microsequence (fetch and execute cycles)
4-32. distinguish between synchronous and combinational system control signals
4-33. describe the operation of memory and the function of its control signals: MSL, MOE, and MWE
4-34. describe the operation of the program counter (PC) and the function of its control signals: ARS, PCC, and POA
4-35. describe the operation of the instruction register (IR) and the function of its control signals: IRL and IRA
4-36. describe the operation of the ALU and the function of its control signals: ALE, ALX, ALY, and AOE
4-37. describe the operation of the instruction decoder/microsequencer and derive the system control table
4-38. describe the basic hardware-imposed system timing constraints: only one device can drive a bus during a given machine cycle, and data cannot pass through more than one flip-flop (register) per cycle
4-39. discuss how the instruction register can be loaded with the contents of the memory location pointed to be the program counter *and* the program counter can be incremented on the same clock edge
4-40. modify a reference ALU design to perform different functions (e.g., shift and rotate)
4-41. describe how input/output instructions can be added to the base machine architecture
4-42. describe the operation of the I/O block and the function of its control signals: IOR and IOW
4-43. compare and contrast the operation of OUT instructions with and without a transparent latch as an integral part of the I/O block
4-44. compare and contrast "jump" and "branch" transfer-of-control instructions along with the architectural features needed to support them
4-45. distinguish conditional and unconditional branches
4-46. describe the basis for which a conditional branch is "taken" or "not taken"
4-47. describe the changes needed to the instruction decoder/microsequencer in order to dynamically change the number of instruction execute cycles based on the opcode
4-48. compare and contrast the machine's asynchronous reset ("START") with the synchronous state counter reset ("RST")
4-49. describe the operation of a stack mechanism (LIFO queue)
4-50. describe the operation of the stack pointer (SP) register and the function of its control signals: ARS, SPI, SPD, SPA
4-51. compare and contrast the two possible stack conventions: SP pointing to the top stack item vs. SP pointing to the top stack item
4-52. describe how stack manipulation instructions (PSH/POP) can be added to the base machine architecture
4-53. discuss the consequences of having an unbalanced set of PSH and POP instructions in a given program
4-54. discuss the reasons for using a stack as a subroutine linkage mechanism: arbitrary nesting of subroutine calls, passing parameters to subroutines, recursion, and reentrancy
4-55. describe how subroutine linkage instructions (JSR/RTS) can be added to the base machine architecture

4-56. analyze the effect of changing the stack convention utilized (SP points to top stack item vs. next available location) on instruction cycle counts

5. an ability to realize, test, and debug practical digital circuits

5-1. draw a logic circuit schematic using computer-aided design software (OrCAD)
5-2. construct a circuit consisting of discrete CMOS logic gates (NOT, NAND, NOR, XOR) and verify its operation
5-3. measure the output voltage swing (V_{OL}-V_{OH}) of a logic gate
5-4. measure the input voltage thresholds (V_{IL}-V_{IH}) of a logic gate
5-5. measure the input voltage thresholds (V_{IL}-V_{IH}) of a Schmitt trigger and compare them to the switching threshold of a standard CMOS gate
5-6. test the response of a logic gate to a “floating” input
5-7. measure the output current sourcing (I_{OH}) and sinking (I_{OL}) capability of a logic gate
5-8. measure the rise and fall propagation delays (t_{PLH} and t_{PHL}) of a logic gate
5-9. measure the rise and fall transition times (t_{TLH} and t_{THL}) of a logic gate
5-10. construct a clock generation circuit and measure its frequency of operation
5-11. verify the existence of a logic hazard in a combinational circuit and modify the circuit to eliminate it
5-12. create a hardware description language (Verilog) program that realizes a prescribed logic function (digital system) and test it on a programmable logic platform
5-13. diagnose and correct logic errors in a hardware description language (HDL) program