ECE150 Overview

Jonathan Lam

January 14, 2020

Here's some of what I remember from digging up notes from the course, roughly in order. We were told about the Waverly textbook, *Digital Electronics for Logic Design*, 6th or 7th edition, but it was never referenced in class and I never consulted it. Note that this was over the summer, so we may have had more small projects than the students who took DLD over the school year – during the school year, they focused on the three larger projects: Braille, Traffic Light, and the final project.

1 Topics

1.1 Combinational Logic

1. zero vs. one, and connotations (from here on out, (almost always) "light on zero" to get out connotation that "one" always means "on")
2. basics of number systems (i.e., radix systems, binar, ternary, quaternary, octal, decimal, hex, etc.), binary-coded decimal (BCD), MSB/LSB, ordering with MSB first/least significant bit first (is this called endian-ness?), bit, bytes, nybbles
3. binary (i.e., logical or boolean) operators (AND, OR, NOT, NAND, NOR, XOR) (w/ truth tables, graphical symbol, symbol in a boolean expression)
4. CMOS gates, how to read a datasheet, learn which 4000-series chips to use, propagation delay, maximum ratings, etc.
5. basics of circuit analysis (i.e., Ohm's law and Kirchoff's laws) (enough to be able to the resistance given a voltage and desired current, etc.)
6. more tricks and properties of boolean expresssions (e.g., commutativity and associativity of both AND and OR, etc.), De Morgan's theorem, how to represent XOR in terms of OR, simplifying boolean expressions
7. Karnaugh maps (minterms, maxterms, XOR patterns/translation to boolean expressions), graycode, "don't cares"

8. Muxes, demuxes, decoders (and why use one over another); the logic necessary to chain/combine them, optimizing implementation to minimum number of muxes needed to map X inputs to an output, etc.

9. Pull-up/down resistors and purpose

1.2 Sequential Logic

1. what is a square wave? practical intro to how to use the 555 in all three modes (not really taught in class, we all kinda just looked up at home)

2. S/R latch design with NAND/NOR (quarter of a D-type F/F) *rightarrow* clocked S/R buffer (half of a D-type F/F) *rightarrow* full D-type M/S F/F

3. race conditions, state table

4. counters design: asynchronous/synchronous; sequence generators (like synchronous counters, but with arbitrary state tables/mappings)

5. J/K F/F design and purpose (e.g., examine how sequence generating logic is simpler), T-type F/Fs

6. shift registers and modes (shift left, shift right, parallel load, no change), implementation of bidirectional SRs

7. half-adder (review two's complement, fixed-width type overflows) design, full-adder design (half-adder with SRs)

8. debouncing (RC circuit, astable 555, S/R latch or F/F)

9. RAM vs. ROM, progression of ROM to EEPROM, RAM chip parameters, implementing a RAM chip (e.g., how to implement the bidirectional data lines without data loss/corruption)

10. only a very basic introduction to PLAs (definition and motivation). For our class, we never really got far in this discussion and nobody was allowed to use one on our final project. This material is covered in the Computer Architecture course – in my class, we learnt the CUPS language with the GAL22V10 chip).

2 Projects

minterms/maxterms Given a 4-to-1 mapping, write out the minterm and maxterm expressions. For us, we had to map the numbers 0-15 to: 0 if it was a prime between 2 and 13 inclusive, 1 for everything else, and 7,14 were don't cares. Basic logic gates only, 1 breadboard.

braille (major assignment 1) We were all given the same arbitrary 4-to-4 mapping (given a number from 0-9, map it to its 4-dot "braille" representation, where 10-15 are don't cares). Most of the time was spent trying to reduce it to the fewest number of gates possible – usually it can be reduced down to 11-15 gates. Documentation included K-map, boolean expression, boolean symbol representations of the final logic, as well as video (and/or live demo?). Basic logic gates only, maximum 1 chip of each logic gate type, 1-2 breadboards (XC for only one breadboard and handing in early)

counter/sequence generator Build an asynchronous 3-bit down-counter, and, given an arbitrary 8-item sequence, build it synchronously. 1-2 breadboards, D-type F/Fs and basic logic chips allowed.

adder Build a full-adder. I forget the requirements for this.

traffic light I just want to say that fulfilling all parts of this project in any of its forms was widely ruled to be impossible to finish with normal means and using only what was taught in class. However, it is good to at least get students to try and think out all of the implementations, and make some parts extra credit. (We had an all-or-nothing extra credit, so nobody got extra credit)

final project Had to choose teams and prepare proposals that had to be accepted by the prof.

3 Project requirements/expectations (in general, for larger projects)

1. boolean expressions and reductions, K-maps, logic diagrams, truth tables (especially for earlier projects)
2. schematics, block/functional diagrams, videos and/or live demo
3. timing diagrams, state tables (for sequential logic circuits)
4. written report (only for braille, traffic light, and final project) w/ diagrams, explanations of nonconventional tricks, acknowledgements