



Outline

Combinational Logic

1 Properties of Digital Signals

1.1 Compare Analog and Digital

1.2 Boolean Algebra

i Gates: NOT, AND, OR, XOR, NAND, NOR XNOR

ii Basic Laws: up through distributive laws

iii Proofs: algebraic, exhaustive.

iv Duality

v Simplification, Absorption, Swap $(X+a)(\bar{X}+b)$, Consensus rules

vi Uses of Duality: the dual of the common laws,

vii XNORs XORs and applications

viii Some 3 input gates: XOR, Select, Reject, Mux, Majority

ix Deriving Circuits from Truth Tables

x Parity Checks; Full Adders

1.3 Common Mistakes.

2 DeMorgan's laws.

i Two variable forms

As equations and as gates with inverted inputs

ii Why Real Gates are NAND/NOR, not AND/OR

iii Two symbols for NAND; two for NOR

iv AND-OR designs for thinking: NAND-NOR for real gates

Design with AND-OR; Implement with NAND-NOR

AND/OR to NAND/NOR Conversions

graphical form

v Generalized DeMorgan's Theorem

2.1 Common Errors

3 Karnaugh Maps

i Another form of the truth table

Labelling Maps

Circling Maps

ii Minimizing Algebra by Maps

Precautions when circling



Combinational Logic

3.1 Don't Cares

Where don't cares come from
- BCD digits

- i Don't cares on Karnaugh Maps

3.2 Common errors when using maps

4 Multiplying Out and Factoring

4.1 Sum-of-Products, Product-of-Sums; two Canonical forms

- i Multiplying Out

Use D1

Use D2 and Swap, then D1

Use a Karnaugh Map

- ii Factoring

Use D2

Take the dual \Rightarrow multiply out \Rightarrow take the dual back

Find \bar{F} using DeMorgan, then use a Karnaugh map

4.2 Common errors you are almost sure to make

5 More Complex K-Maps

5.1 Five-variable maps

Dual 4-variable maps

Variable-entered maps

Split-square maps

Two illustrations showing when variable entered maps are easier

5.2 Multiple-output maps

Example: braille decoder

Methods of circling multiple-output maps

1. Do half-maps first;
2. Circle lone "1"s; Circle recently orphaned, lone "1"s
3. Circle "1"s which have no same map neighbors

5.3 Appendix example: BCD to 7-segment decoder

5.4 Common Errors

6 Logic Implementation

- i How logic is really implemented:

74xxx chips, CPLDs, FPGAs, Standard-Cells

6.1 Logic Arrays

- i NMOS FET Arrays

AND arrays



Combinational Logic

- OR arrays
- AND-OR arrays
- ii PLA both AND and OR programmable
- PAL only AND plane programmable
- RAM only OR plane programmable
- 6.2 Commercial PALsTM
 - i Examples of multiple output logic using PALsTM
 - ii Minimizing the number of AND terms.
 - iii Using unneeded outputs to obtain more AND terms
- 6.3 Mux Logic
 - i Mux Binary-Tree (Binary Decision Diagram)
 - Expansion with 2-input MUXs
 - Expansion with 4-input MUXs
 - Removal of duplicate MUXs

Sequential Logic

- 1 Storage Devices
 - i Storing with feedback; Basic Latches
 - ii RS latches; Gated R-S latches
 - iii Transparent latches (D-latches)
 - 1.1 Edge triggered flip-flops
 - i D Flip Flops
 - ii Master-Slave D Flip Flops
 - iii Asynchronous Reset
 - iv Synchronous Reset
 - v Applications
 - Divide by Two
 - Shift Register
 - vi The *State*
 - vii Enabled D Flip Flops
 - 1.2 Review and common errors
- 2 Sequential Machines (machines with memory)
 - 2.1 States and Finite State machines
 - 2.2 Sequencing of States
 - i State Graphs
 - ii State Tables



- iii State Table to Karnaugh Map to Circuit
- iv Mealy and Moore Outputs
- v Constructing The Output Circuits From The State Table
- vi Properties of Mealy and Moore Outputs

2.3 Overview of Designing Finite State Machines

- i Construction of State Graphs
 - Timing waveforms
 - Examples:
 - Toggle Flip Flop
 - Detecting an Input Sequence, the 101 machine

2.4 Common Errors

3 Some Types of Machines and their State Graphs Checking for Dual Sequences

- i 1101 or 1011 Mealy Detection
 - Product Graph
- ii 010001 or 0101 Moore Detector
 - Product Graph
- iii One-Pulse-Per-Push Circuit
 - Mealy
 - Moore
- iv Coupled State Machines
- v Common errors

4 State Assignment

4.1 Assigning Bits to States

- How and why it is done
- Assigning on a Karnaugh map
- Reasons For Choosing A Particular Assignments
- Assignments that minimize logic

4.2 Guidelines

- a) Do something with unused states
- b) It doesn't matter where you start
- c) When you cuddle (on the K-map), it doesn't matter on which side

4.3 Heuristics

- 1. Parents of a child (for same input) should cuddle.
- 2) Sisters should cuddle
- 3. If rules 2) and 3) are inconclusive, states with the same output should cuddle.

4.4 Examples



Asynchronous Logic

4.5 Hazards

- i Glitches and hazards
Static-1, static-0 and dynamic hazards
- ii Hazards on Karnaugh maps
- iii Algebra and hazards
- iv Locating hazards algebraically
Using a mux tree like diagram (Binary Decision Diagram) to find all hazards
- v Implementing hazard-free circuits
Sum-of-Product and Product-of-Sum circuits have only one type of static hazard
- vi When are hazards important
Multiple-variable-change hazards
Clocked logic avoids most hazard problems.

5 Asynchronous Circuits

5.1 Storage in Feedback loops

- i Analysis of Asynchronous Circuits
- ii Asynchronous state-tables
- iii Races, and cycles
- iv Race-Free State Assignment
- v Hazards
- vi Overview of Where to Use, and Not Use, Anachronism Circuits