Logic Design II

CSci 2021: Machine Architecture and Organization

With Slides from: Stephen McCamant, Xi Zhang and Hai Zhou

Combinational Logic Design

- Given: description of circuit behavior
  - Word problem, or truth table
- Goal: efficient circuit implementation
  - Usually most important: fewest gates and wires
  - Secondarily: reduce number of levels (propagation delay)
- Kinds of techniques
  - Up to 6 inputs: pencil and paper approaches
  - Large but structured: split into repeated pieces
  - Large and unstructured: computer algorithm

DNF / SOP

- An input or its negation is called a *literal*
  - E.g.: a, !b
- An AND of literals is a *product term* or *cube*
  - E.g.: (a & c), (a & !b), (!a & !b & !c), c
- An OR of product terms is a *sum of products* (SOP), or in *disjunctive normal form* (DNF)
  - E.g.: (a & b) | (a & c)
- (Dual: *product of sums* (POS), or *conjunctive normal form* (CNF))

Inefficiency of Straight DNF

- Consider another example:

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

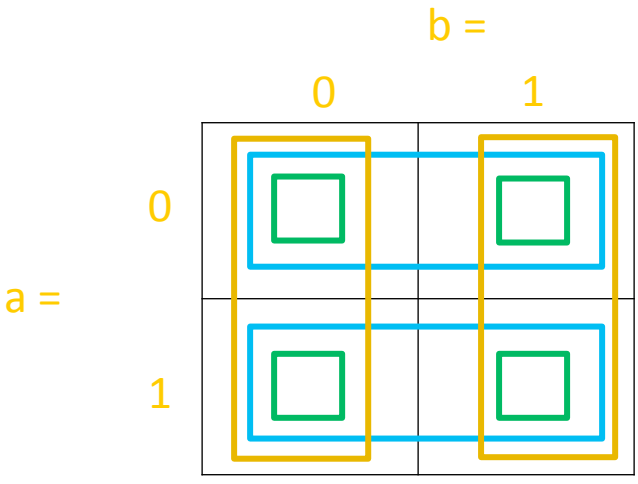
Result: (!a & b) | (a & b)

- By algebra, can simplify back to "b"
  - Factor, (!a | a) = 1, 1 & b = b
- Can we recognize these patterns earlier?

Karnaugh Map Idea

- Write truth table entries in an array
- Product terms represented by certain rectangles
- Visually, find small number of rectangles to cover 1 bits
  - OK to cover more than once, combine with OR
  - Fewer rectangles = smaller circuit

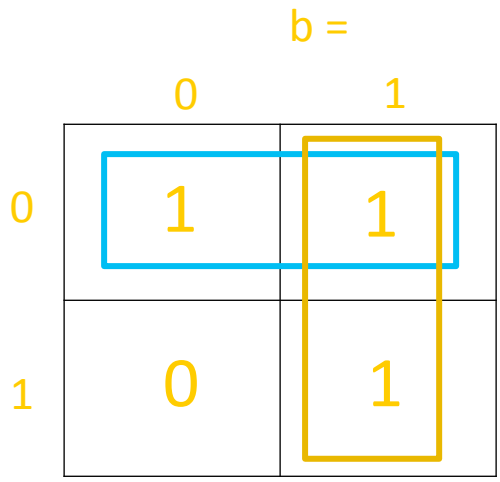
2-variable "Karnaugh Map"



2-variable "Karnaugh Map" example

Result:

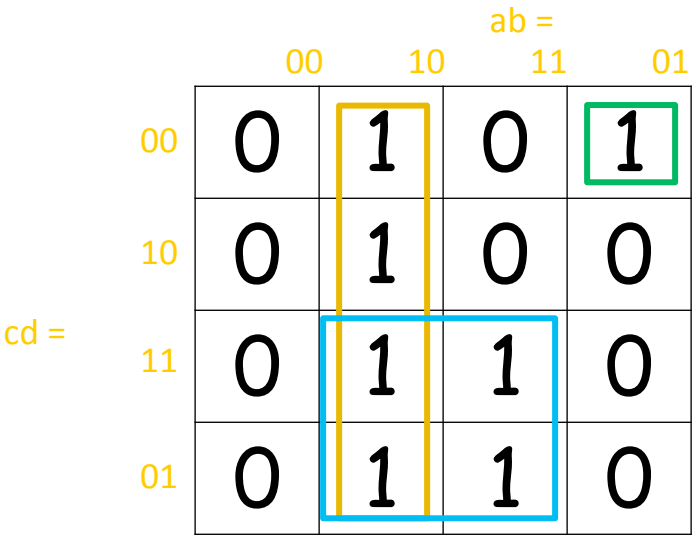
!a | b



Extending to 3 and 4 Variables

- Put two variables on a side
  - Weird order: 00 01 11 10
  - "Gray Code": change only one bit at a time
- Rectangles can enclose 1, 2, 4, or 8 entries
  - Bigger is better
- Rectangles can wrap around the edges
  - 00 is adjacent to 10

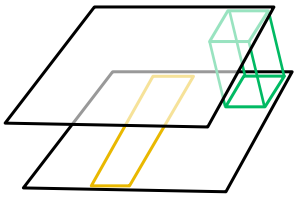
4-variable Karnaugh Map Example



Extending to 5 and 6 Variables

- 2D is no longer enough
  - No way to order 3 variables to capture 12 adjacencies
- Approach: stacking
  - Make 2 (for 5 inputs) or 4 (for 6 inputs) 4-input Karnaugh maps
  - Corresponding entries are "on top of" each other
  - Rectangles become 3D
  - Usually still drawn as 2D
  - With 6, more possibilities for wrapping too

5-variable Karnaugh Map Example



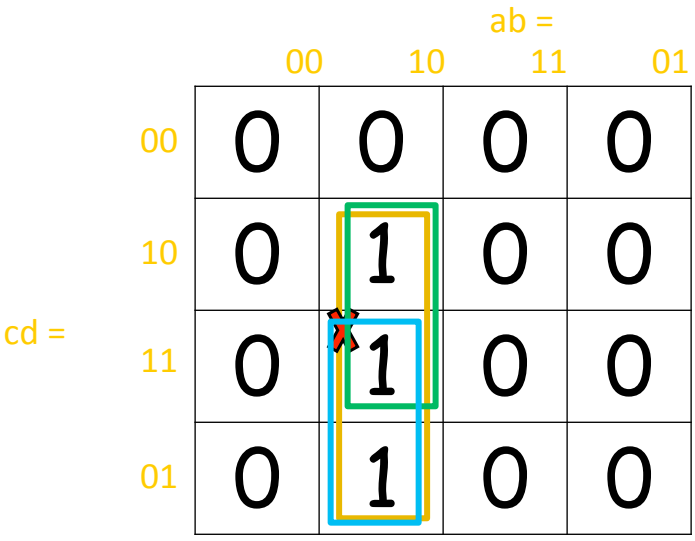
0	1	0	1
0	1	0	0
0	1	1	0
0	1	1	0

0	0	0	1
0	0	0	0
0	1	1	0
0	1	1	0

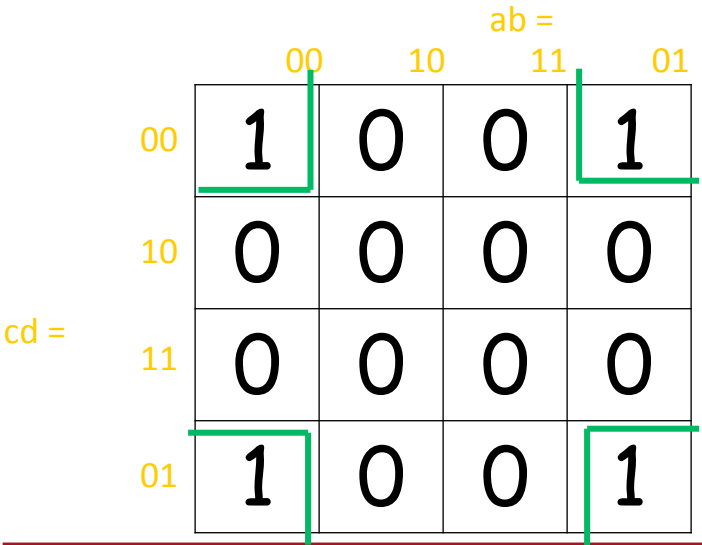
Karnaugh Map Tips: Overlap is Good

cd = \ ab =	00	10	11	01
00	0	1	0	0
10	0	1	0	0
11	0	1	1	0
01	0	1	1	0

Karnaugh Map Tips: No 3s



Karnaugh Map Tips: Wrap Around



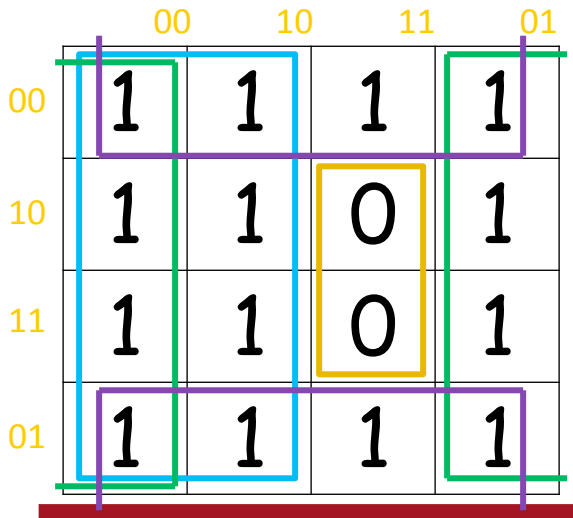
Don't Cares

- Some results don't matter
  - Domain of function is a subset of all n-bit strings
  - Unused bit patterns in encodings
  - Bits sometimes ignored by other circuits
- "Don't care" value could be 0 or 1
  - Usually denoted by X
- Don't-cares allow designs to be simpler
  - Choose the value that allows a simpler circuit
- In early CPUs, led to undocumented instructions
  - Example: x86 ASL vs. SHL
  - On modern CPUs, more error checking

Karnaugh Map Tips: Don't Cares

cd = \ ab =	00	10	11	01
00	x	0	x	1
10	0	x	x	x
11	x	x	x	x
01	1	x	x	x

Dual (POS) Karnaugh Maps



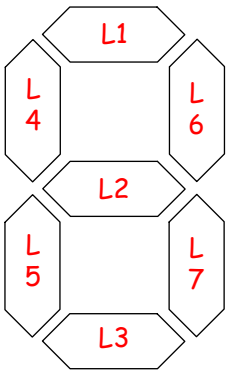
- Pretend 0s are 1s
  - And vice-versa
- Negate final result

!(a & b & c)

!a | !b | !c

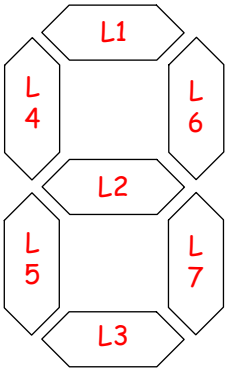
Case Study of a Simple Logic Design: Seven Segment Display

- Chip to drive digital display

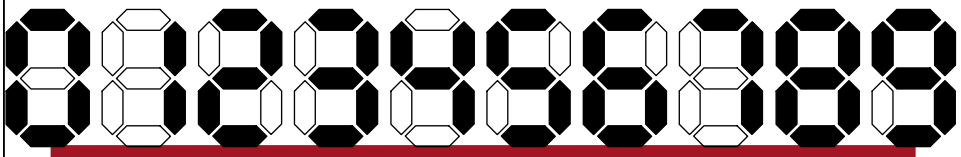


B3	B2	B1	B0	Val
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9

Case Study (cont.)



B3	B2	B1	B0	Val	L1	L2	L3	L4	L5	L6	L7
0	0	0	0	0	1	0	1	1	1	1	1
0	0	0	1	1	0	0	0	0	0	1	1
0	0	1	0	2	1	1	1	0	1	1	0
0	0	1	1	3	1	1	1	0	0	1	1
0	1	0	0	4	0	1	0	1	0	1	1
0	1	0	1	5	1	1	1	1	0	0	1
0	1	1	0	6	1	1	1	1	1	0	1
0	1	1	1	7	1	0	0	0	0	1	1
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	1	1	0	1	1



Case Study (cont.)

- Implement L4:

B3	B2	B1	B0	L4
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1

Boolean function for L4

B3B2 \ B1B0	00	10	11	01
00				
10				
11				
01				

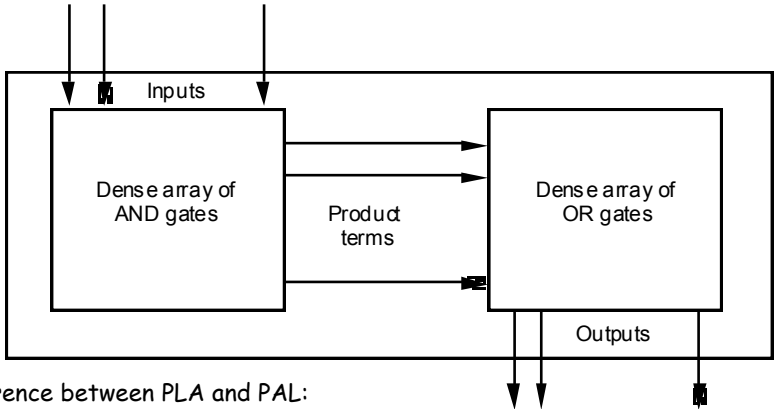
ASIC vs. Programmable Logic Device

- PLDs programmable logic devices
  - Simple PLD (SPLD)
    - Programmable logic array (PLA)
    - Programmable array logic (PAL)
  - Complex PLD (CPLD)
  - Field-programmable gate arrays (FPGA)
- ASICs
  - application specific integrated circuits

PALs and PLAs

Pre-fabricated building block of many AND/OR gates (or NOR, NAND)
"Personalized" by making or breaking connections among the gates

Programmable Array Block Diagram for Sum of Products Form

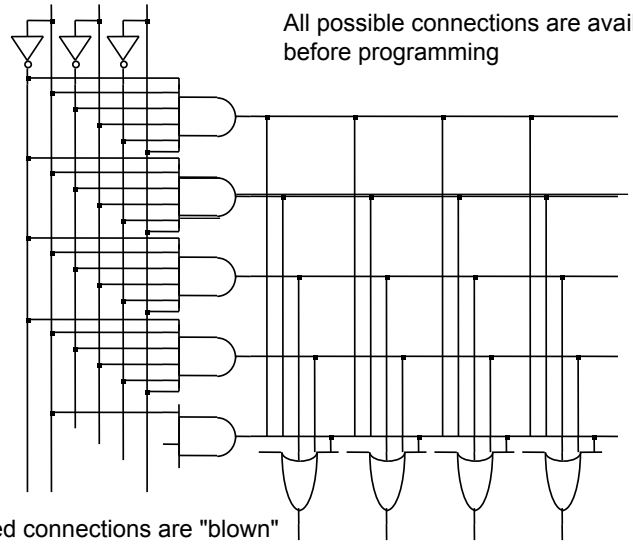


Difference between PLA and PAL:

PLA: both AND plane and OR plane are programmable.

PAL: Only AND plane is programmable, while OR plane is fixed.

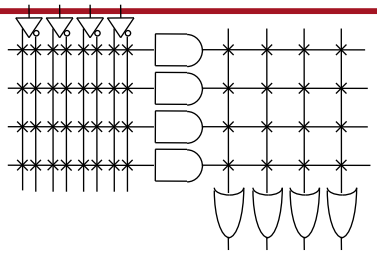
Example of PALs and PLAs



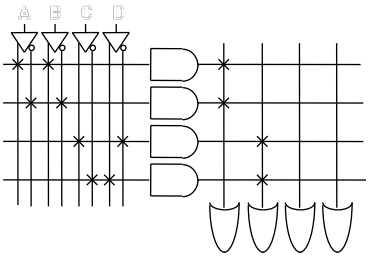
All possible connections are available before programming

Unwanted connections are "blown"

Alternative Representations

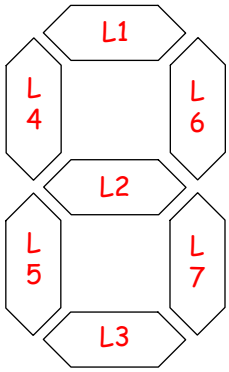


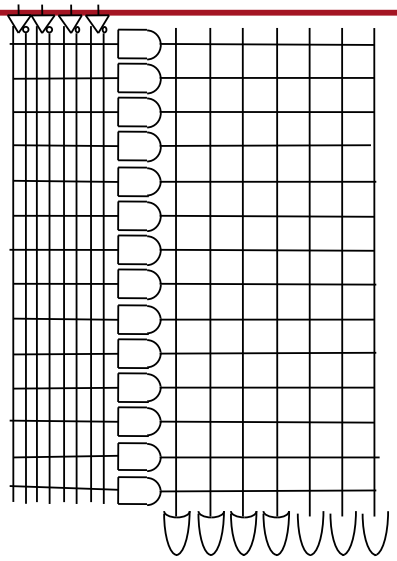
Short-hand notation so we don't have to draw all the wires!



Notation for implementing
F0 = A B + A' B'
F1 = C D' + C' D

Design Example





Automated Methods

- Karnaugh maps don't scale well beyond 6 inputs
- Good job for a computer!
- Quine-McCluskey algorithm
  - Tabular analog to Karnaugh maps
  - Optimal, but suffers from exponential blowup
- Heuristic methods like "`espresso`"
  - First, greedily achieve coverage
  - Then, opportunistically improve
  - No optimality guarantee, but good scalability
- Now a standard part of CAD systems
  - Like compilers for software