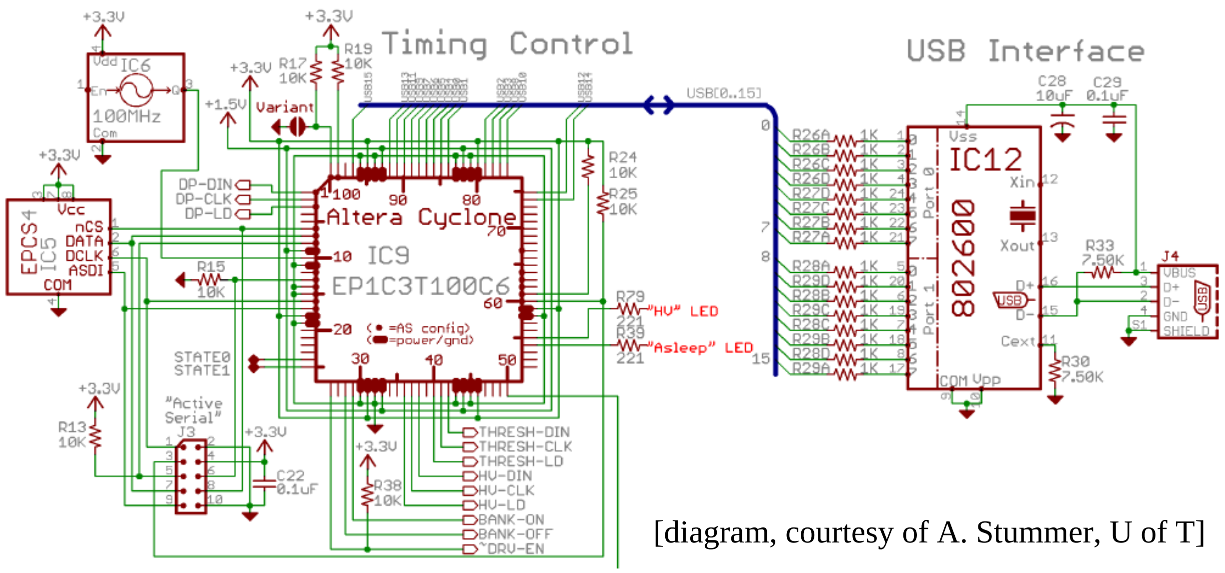
Physics 351: Electronics II
Introduction to Digital Circuits

Prerequisites: PHYS 252.

Introduction to digital electronics: Theory, design, and application of digital circuits … or how to understand and make circuits like these:



[diagram, courtesy of A. Stummer, U of T]

Small print: *If you don't have any experience with analog electronics you should talk to me after class.*

Instructors

Prof. Seth Aubin

Office: room 333 (3rd floor back hall), tel: 1-3545
Lab: room 15 (basement next to machine shop), tel: 1-3532
e-mail: saaubi@wm.edu
web: http://www.physics.wm.edu/~saubin/index.html

Austin Ziltz

Office: rooms 243, tel: 1-3570
e-mail: arziltz@wm.edu

Office hours:

Monday:	3-4 pm (Ziltz, room 243)
Tuesday:	4-5 pm (Aubin, room 333)

Course Objectives

Primary: Design and test both basic and advanced digital circuits for ***digital logic***, ***signal acquisition***, and ***digital signal processing***.

Secondary: Learn experimental research skills.

signal in
Digital Processing
signal out

Covered topics:

- Binary numbers, logic gates, and Karnaugh maps.
- Memory, flip-flops, and clocked latches.
- Clocks, timing, and one-shots.
- Counters, registers, and state machines.
- Analog-to-Digital Converters (ADC) and Digital-to-Analog Converters (DAC).
- Optical and magnetic isolation.
- **Field Programmable Gate Arrays (FPGA).**
- Verilog language FPGA programming.
- **Digital Signal Processing (DSP).**
- Microprocessors.

FPGAs for Physicists

Field Programmable Gate Array (FPGA) chips for physicists

- Contain 2,000-100,000 logic gates + memory.
- Rerogrammable via a computer (Quartus II v7-8).
- Stand alone circuitry (with flash memory).
- Parallel processing.
- ***Useful for complex circuits and Digital Signal Processing (DSP).***

Please download and install Quartus II WebEdition onto your laptop.

Please request a WebEdition license from Altera Inc.

(see course webpage for the necessary internet links)

Note: Quartus II is available on lab computers, in room 134 (Learning Center) of Swem library, and Morton 240 (24/7)

DSP design project (I)

A central component of the course is an FPGA-based digital signal processing (DSP) project. The general guidelines for the projects are:

- ➢ Teams of 2-3 students (depends on lab section distribution).
- ➢ Each team has a budget of $250 USD.
- ➢ All teams have the same project.
- ➢ This section of the course is a design and construction competition.

The purpose of the one month team project is to help you develop practical circuit design skills, as well as the following more general research skills:

- Complex device design.
- Project budgeting.
- Formal project proposal writing.
- Finding, selecting, and purchasing device components.
- Device construction.
- Troubleshooting and debugging.
- Oral and web presentations of the device.

DSP design project (II)

2 choices for the design project:

1. Phase coherent, dual output **DSP FUNCTION GENERATOR**

   → 2 analog outputs.

   → FPGA core.

   → External clock option.

   → Comments: more DSP, easier analog.

2. Digital **VOICE RECORDER** with playback

   → 1 analog input (i.e. microphone).

   → 1 analog output (i.e. speaker).

   → FPGA core.

   → Comments: more memory handling, more involved analog.

DSP design project (III)

- Easier project than last year, with more project time scheduled.
- The project will be based on an FPGA.
- The specific project requirements will be announced next week.
- The project is the most important part of the course.
- It will be graded according to the following weights:

Formal project proposal	10%
Device construction	15%
Device performance	20%
Oral presentation of device.	5%
Web presentation of device	5%
Project lab book and wiki	5%
Total	**60%**

Evaluation

Notebooks/Reports:	30%
Quizzes/Participation:	10%
DSP project:	60%
Total =	100%

Note: There is no final exam for the course

Due Dates

- **Lab books**

  In addition to lab notes, the lab books should include all design exercises.

  Lab books are due by 5pm 2 days after lab & will be returned by the next lab period:

  → Thursdays for the Tuesday section

  → Fridays for the Wednesday section

- **Reports**

  - Reports are due on monday the following week.

  - Instructor will indicate for which labs a report is due.

  - You can expect 1-2 reports for the semester.

  - Lab books should not be turned in when lab report is due.

  - Important measured numbers should include an estimated uncertainty.

Schedule (I)

Week 0: 8/25-27 **NO CLASS**

Week 1: 9/1-3 **Digital Logic**
Class & lab: Binary numbers, logic gates, Karnaugh maps.

Week 2: 9/8-10 **Introduction to FPGAs**
Class & lab: FPGAs, Verilog programming language, applications.

Week 3: 9/15-17 **Memory**
Class & lab: Memory, flip-flops.

Week 4: 9/22-24 **Timing**
Class & lab: Clocks, timing, one-shots, and counters.

Week 5: 9/29 -10/1 **Counters and Registers**
Class & lab: Counters II and registers.

Week 6: 10/6-8 **Analog-Digital Interfacing**
Class & lab: Analog-to-digital converters, digital-to-analog converters.
Project: Formal project proposals are due October 10, 2008.

Schedule (II)

Week 6.5: 10/15 Wednesday lab group jumps one week ahead
Class: No class; Lab: Project; project funds are released.

Week 6: 10/20-22 **State Machines and/or Microprocessors**
Class & lab: State machines and sequencing.

Week 8: 10/27-29 **Project week 1**
Class: Surface mount soldering review and machine shop training.
Project: Project construction begins.

Week 9: 11/3-5 **Project week 2**
Class: Digital signal processing.
Lab & Project: Project construction continues.

Week 10: 11/10-12 **Project week 3**
Class: Tri-state logic.
Lab & Project: Project construction continues.

Week 11: 11/17-19 **Project week 4**
Class: Ground loops and opto-isolation.
Lab & Project: Project construction and debugging. 2 sections equalize labs.

Schedule (III)

Week 12: 11/24-25 **Project week 5**
Class: Microprocessors on FPGAs *or* phase-lock loops.
Lab & Project: Project debugging and troubleshooting.

Week 13: 12/1-3 **Project week 6**
Class: Oral presentations and website launch.
Lab & Project: Device performance testing and review.

Introduction to Digital Logic

Digital Variables

A digital circuit has only 2 possible values HIGH (H or 1) and LOW (L or 0)

→ Does not need to be precision designed.

→ Not very sensitive to electronic noise.

Here are a few voltage-logic conventions:

Convention	Supply	LOW	HIGH	Speed
TTL	+ 5 V	< 0.7 V	> 2.0 V	~5 nS
LVTTL	+ 3.3 V	< 0.7 V	> 2.0 V	~5 nS
CMOS	+ 3-15 V	< 20% Supply	> 80% Supply	~10 nS
GaAs	undefined	undefined	undefined	~100 pS

Digital vs. Analog

Digital

- ➢ Easy to design (linear logic flow).

  → No feedback !

- ➢ Insensitive to electronic noise.
- ➢ Easy to design and make very complex circuits.
- ➢ Insensitive to specific components.
- ➢ Reliable isolation circuitry.
- ➢ Tends to consume a lot of power.
- ➢ Slower than analog equivalent.
- ➢ Very bad if a single bit is corrupted (std. error rate 1 part per 10^{10}).

  → Error correction is important.

Analog

- ➢ Harder to design and read a circuit, especially with feedback.
- ➢ Noise is critical.
- ➢ Complex circuits are hard to design.
- ➢ Sensitive to specific components and quality of assembly.
- ➢ Isolation circuitry reduces accuracy.
- ➢ Very fast.
- ➢ Can be low power.
- ➢ Some circuits must be analog.

Transistor-Transistor-Logic (TTL)

In this course, we will use almost exclusively the TTL family of logic chips.

Characteristics:

- Very reliable.
- Widely available.
- Silicon-based with bipolar transistors.
- Supply: + 5 V, High > 2 V, Low < 0.7 V
- 1 output can drive 10 inputs (fanout = 10).
- Never leave an input (or output) floating: it will tend to wander between H and L.

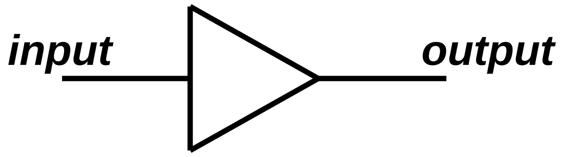
CAUTION: If any of your voltages are close to the range 0.7 – 2.0 V, then you should check your circuit and the components.

Boolean Operators

Identity

1 input → 1 output

0 input → 0 output

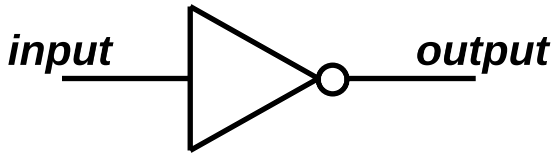


(also called a buffer)

Inverter

1 input → 0 output

0 input → 1 output



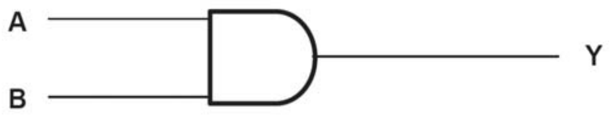
Note: Boolean (adj.) refers to something that is 2-valued
(named after G. Boole, 1815-1864).

2-input operators

AND

→ Outputs H only if both inputs are H.

→ Written as a product:
Y=AB

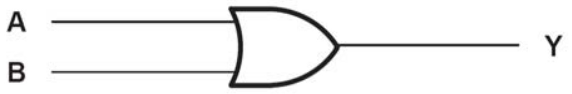


INPUTS		OUTPUT
A	B	AB = Y
L	L	L
L	H	L
H	L	L
H	H	H

OR

→ Outputs H only if either input is H.

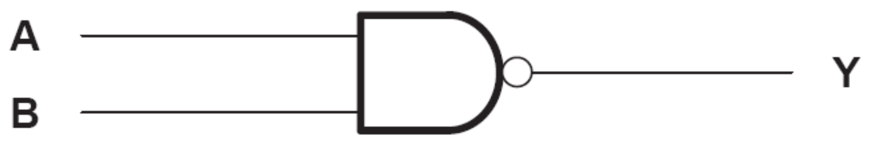
→ Written as a sum:
Y=A+B



INPUTS		OUTPUT
A	B	A+B = Y
L	L	L
L	H	H
H	L	H
H	H	H

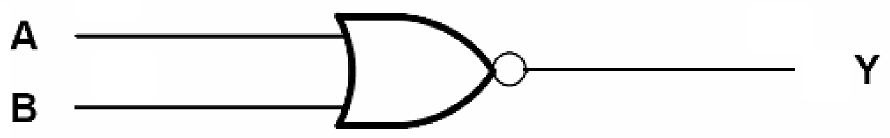
More operators

NAND



INPUTS		OUTPUT
A	B	$\overline{AB} = Y$
L	L	H
L	H	H
H	L	H
H	H	L

NOR

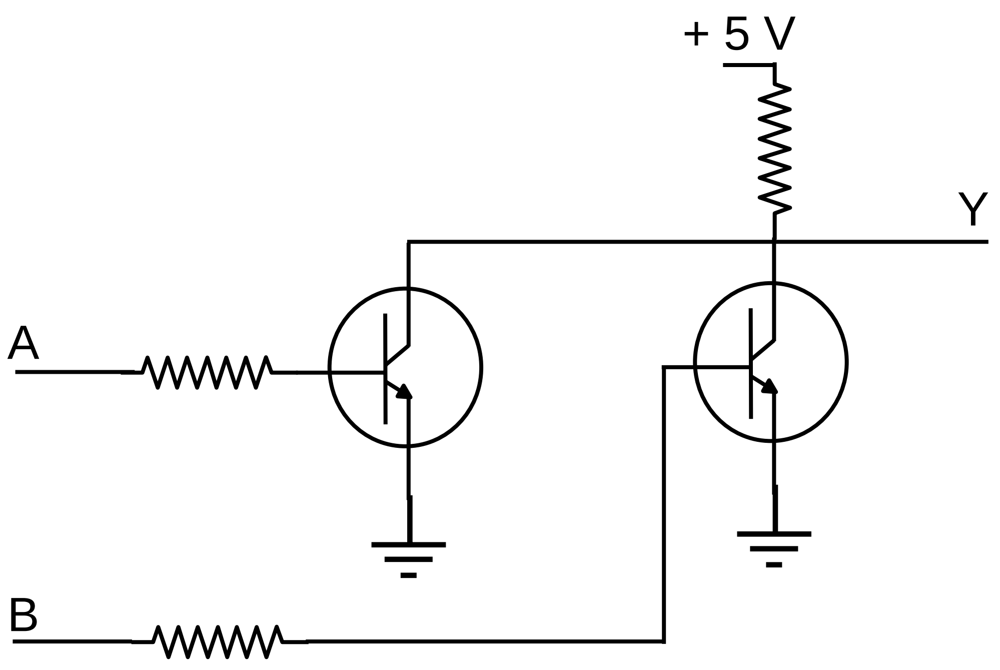


INPUTS		OUTPUT
A	B	$\overline{A+B} = Y$
L	L	H
X	H	L
H	X	L

X = don’t care (H or L)

A little bit of analog

Analog realization of a NOR gate



INPUTS		OUTPUT
A	B	$\overline{A+B} = Y$
L	L	H
X	H	L
H	X	L

Boolean logic identities

Associative

$ABC = (AB)C = A(BC)$	$A+B+C = (A+B)+C = A+(B+C)$

Commutative

$AB = BA$	$A+B = B+A$

Others

$AA = A$	$A1 = A$	$A0 = 0$
$A+A = A$	$A+1 = 1$	$A+0 = A$
$A + AB = A$	$A+BC = (A+B)(A+C)$	
$A + \overline{A} = 1$	$A\,\overline{A} = 0$	$A + \overline{A}\,B = A+B$

DeMorgan’s Theorem

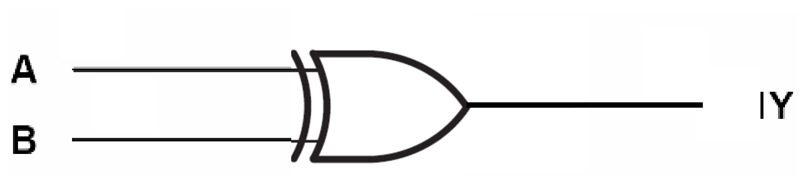
$\overline{A+B} = \overline{A}\,\overline{B}$	$\overline{AB} = \overline{A} + \overline{B}$

Exclusive OR

XOR

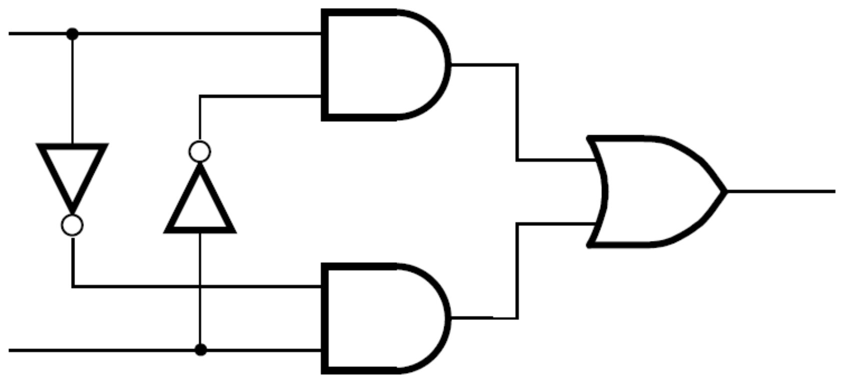
→ Outputs H if either input is H, but not both.

→ Written as a plus sign with a circle around it: Y=*A*⊕*B*



INPUTS		OUTPUT
A	B	A⊕B = Y
L	L	L
L	H	H
H	L	H
H	H	L

XOR realization



[diagram courtesy of Altera Inc.]

The NAND and NOR gates

DeMorgan’s theorem corollary:

Any logic gate or operation can be constructed exclusively of NAND gates (or NOR gates).

Note: a NAND gate with the inputs tied together is a NOT gate.

Hardware

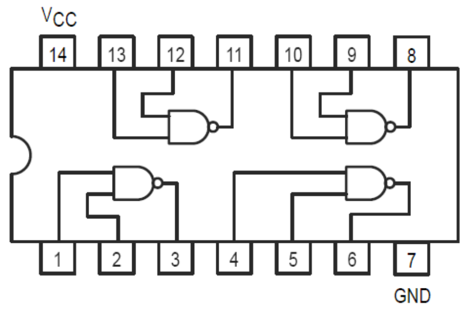
Name	Expression	Inputs	Part #
AND	AB	2 (also 3&4)	74xx08
NAND	$\overline{AB}$	2 (also 3&4)	74xx00
OR	$A+B$	2 (also 3&4)	74xx32
NOR	$\overline{A+B}$	2 (also 3&4)	74xx02
Invert	$\overline{A}$	1	74xx04
Buffer	A	1	74xx365
XOR	$A\oplus B$	2 (also 3&4)	74xx86 / 386
XNOR	$\overline{A\oplus B}$	2 (also 3&4)	74xx266

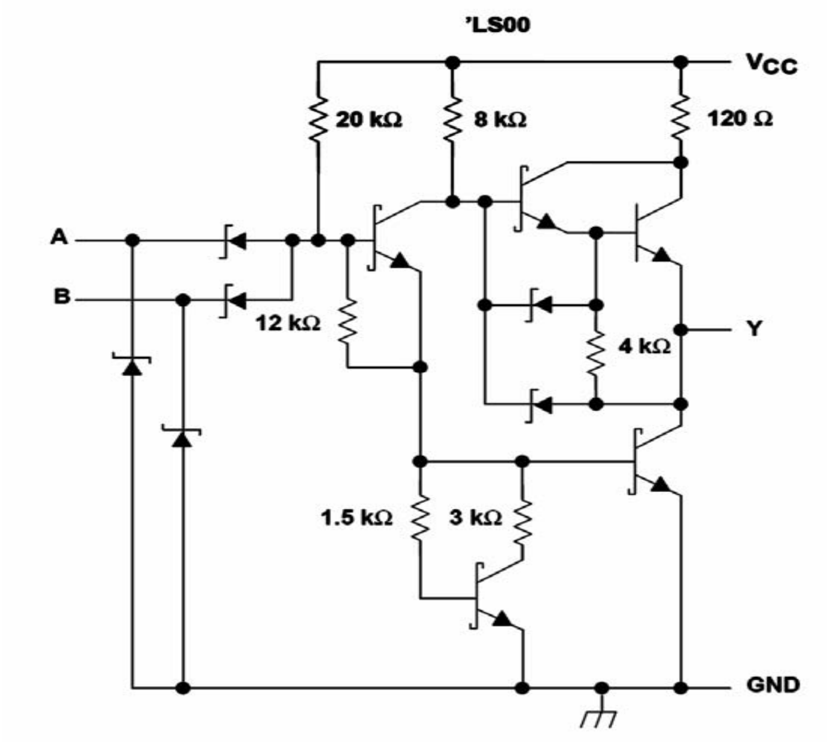
Note: We will use mostly Low Speed TTL (xx = LS).

Example: 74LS00

Quad NAND gate chip

- ➢ 4 gates per chip.
- ➢ Requires + 5 V of power at Vcc.
- ➢ Requires a ground connection at GND.
- ➢ Never float an input (i.e. it will wander between 0 and 1).
- ➢ Each gate consists of about 20 components.





Karnaugh Maps (I)

Logic table → Karnaugh Map → digital logic circuit

- Up to 4 inputs, 1 output.
- Always gives a solution, though not the most efficient one.

Example:

- 3 person vote.
- 2-person majority produces H output.

A	B	C	Q
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Karnaugh Maps (II)

- Arrange inputs on either one of the two table axes.
- Up to 2 inputs per axis.
- Order of inputs is important: only one input change per row or column.

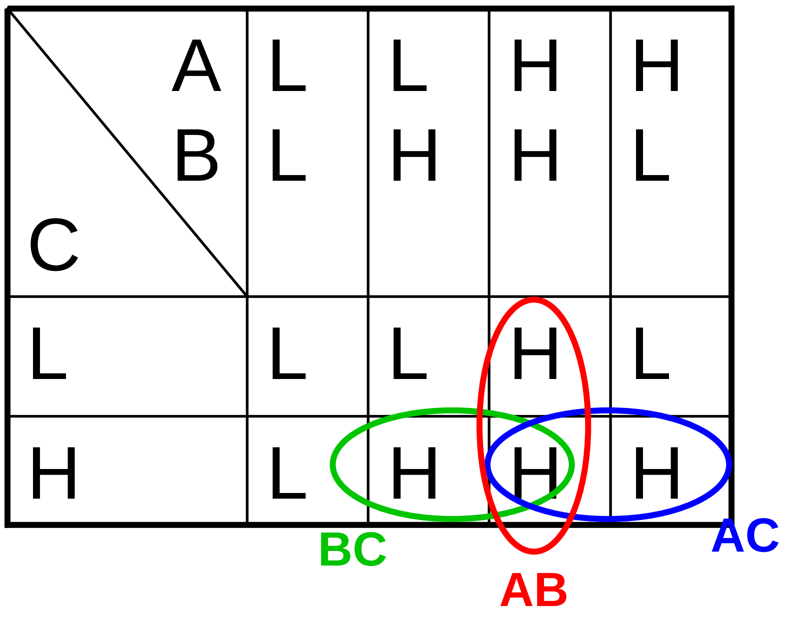
(note: column order is circular.)

A B C	L L	L H	H H	H L
L	L	L	H	L
H	L	H	H	H

Karnaugh Maps (II)

- ➢ Arrange inputs on either one of the two table axes.
- ➢ Up to 2 inputs per axis.
- ➢ Order of inputs is important: only one input change per row or column.
- ➢ Group together the adjacent “ones”: these correspond to AND gates.
- ➢ Alternatively, group adjacent “zeros”: these correspond to OR gates.
- ➢ Write down the corresponding AND gates: AB, BC, AC

Solution: AB + BC + AC

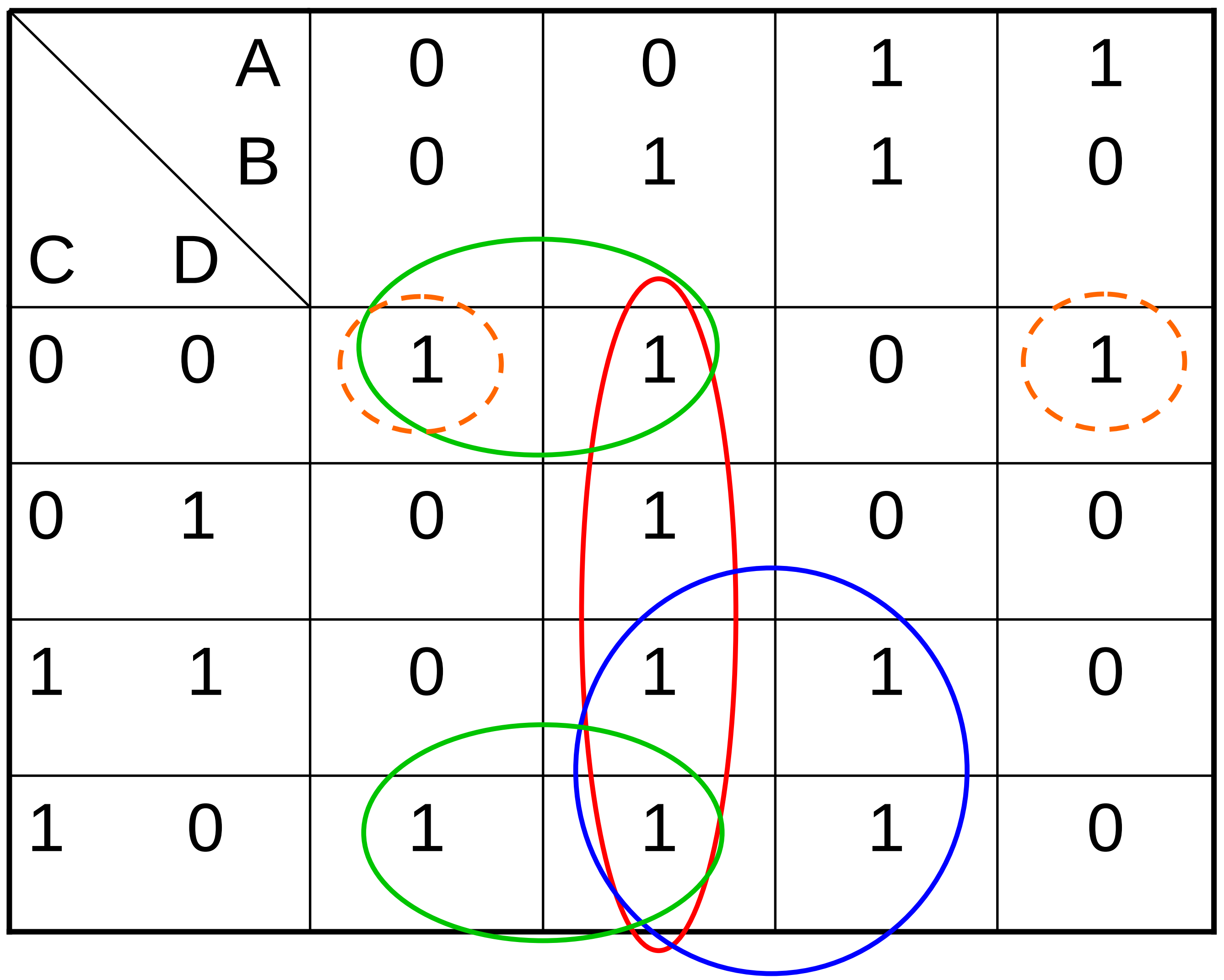


C \ A B	L L	L H	H H	H L
L	L	L	H	L
H	L	H	H	H

Karnaugh Maps (III)

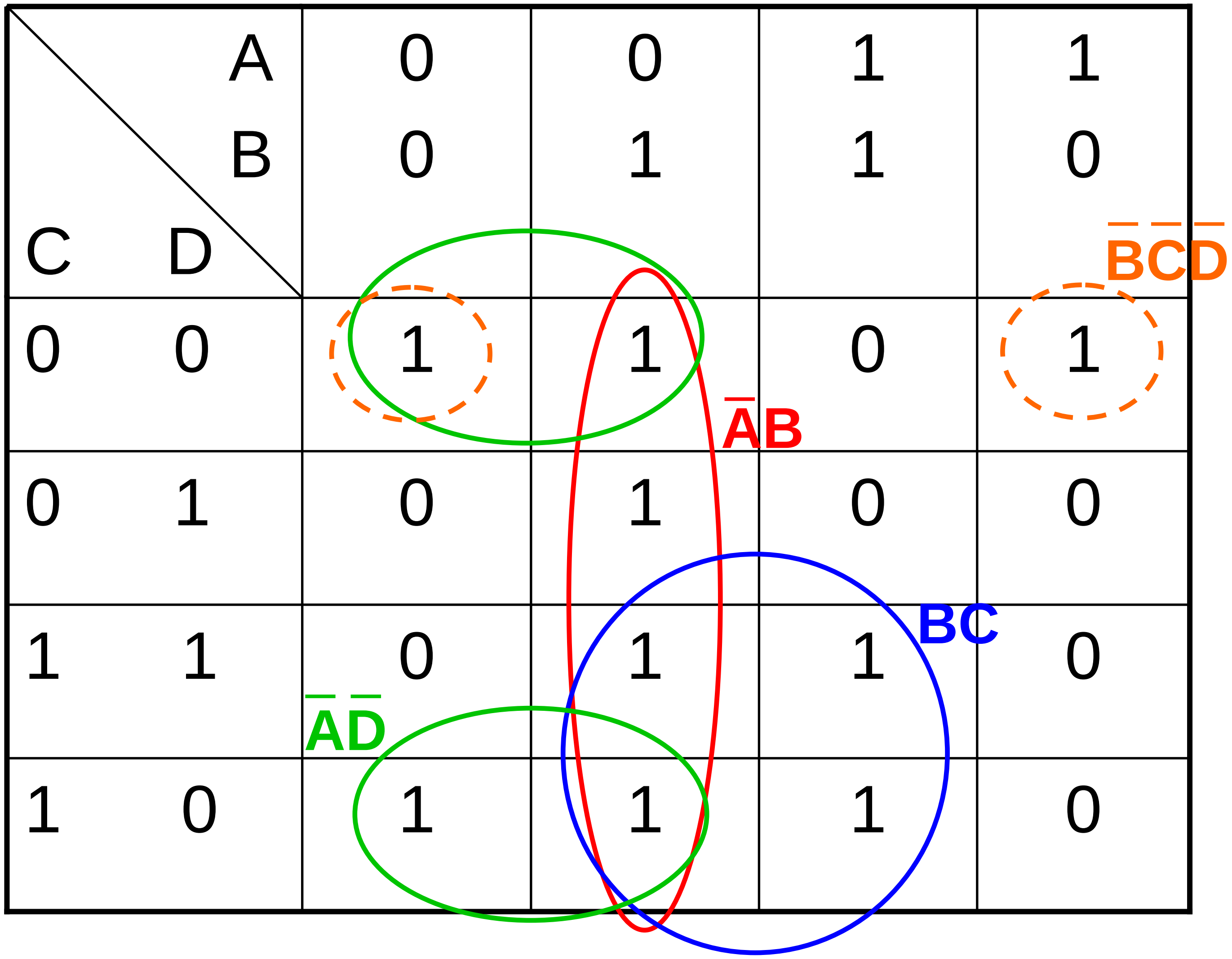
A B C D	0 0	0 1	1 1	1 0
0 0	1	1	0	1
0 1	0	1	0	0
1 1	0	1	1	0
1 0	1	1	1	0

Karnaugh Maps (III)



A B C D	0 0	0 1	1 1	1 0
0 0	1	1	0	1
0 1	0	1	0	0
1 1	0	1	1	0
1 0	1	1	1	0

Karnaugh Maps (III)



A B / C D	0 0	0 1	1 1	1 0
0 0	1	1	0	1
0 1	0	1	0	0
1 1	0	1	1	0
1 0	1	1	1	0

$\overline{B}\,\overline{C}\,\overline{D}$
$\overline{A}B$
BC
$\overline{A}\,\overline{D}$

Solution: $\overline{A}B + BC + \overline{A}\,\overline{D} + \overline{B}\,\overline{C}\,\overline{D}$

Binary Numbers

Base 10 (i.e. decimal numbers)

$73691 = 1\times10^0 + 9\times10^1 + 6\times10^2 + 3\times10^3 + 7\times10^4 = 73691_{10}$

We can represent any integer in a digital circuit if we use base-2 representation.

Base 2 (i.e. binary numbers)

$10011101 = 1\times2^0 + 0\times2^1 + 1\times2^2 + 1\times2^3 + 1\times2^4 + 0\times2^5 + 0\times2^6 + 1\times2^7 = 10011101_2$

$= 1 + 0 + 4 + 8 + 16 + 0 + 0 + 128 = 157_{10}$

Binary Numbers

Base 10 (i.e. decimal numbers)

$73691 = 1\times10^0 + 9\times10^1 + 6\times10^2 + 3\times10^3 + 7\times10^4 = 73691_{10}$

We can represent any integer in a digital circuit if we use base-2 representation.

Base 2 (i.e. binary numbers)

$1001\boxed{1}101 = 1\times2^0 + 0\times2^1 + 1\times2^2 + 1\times2^3 + 1\times2^4 + 0\times2^5 + 0\times2^6 + 1\times2^7 = 10011101_2$

1-bit

$= 1 + 0 + 4 + 8 + 16 + 0 + 0 + 128 = 157_{10}$

8-bits = 1 byte

Binary Numbers

Base 10 (i.e. decimal numbers)

$73691 = 1\times10^0 + 9\times10^1 + 6\times10^2 + 3\times10^3 + 7\times10^4 = 73691_{10}$

We can represent any integer in a digital circuit if we use base-2 representation.

Base 2 (i.e. binary numbers)

$$\begin{aligned}1001\underset{\text{1-bit}}{\boxed{1}}101 &= 1\times2^0 + 0\times2^1 + 1\times2^2 + 1\times2^3 + 1\times2^4 + 0\times2^5 + 0\times2^6 + 1\times2^7 = 10011101_2\\ &= 1 + 0 + 4 + 8 + 16 + 0 + 0 + 128 = 157_{10}\end{aligned}$$

8-bits = 1 byte

Base 16 (i.e. Hexadecimal numbers)

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Decimal → Binary

- To convert from decimal to binary
  - →Divides by 2 repeatedly & write the remainders

- To convert 13_{10} to binary

  13/2 = 6 remainder 1
  6/2 = 3 remainder 0
  3/2 = 1 remainder 1
  1/2 = 0 remainder 1

- The digits come out in right to left order
  - → $13_{10} = 1101_2$

Binary Addition

- Examples

  $0101_2 + 0010_2 = 0111_2$
  $0101_2 + 0001_2 = 0110_2$
  $0111_2 + 0001_2 = 1000_2$

- Differences between decimal & binary addition…
  - In binary we, on average, carry half the time.
  - There are only a limited number of possible operands & resultants (1s or 0s).
  - Makes digital implementation fairly simple.