

Lecture 2

EE 421 / CS 425

Digital System Design

Fall 2024

Shahid Masud

Topics in Lecture 2

- Background to Digital Systems
- Useful number systems, conversion
- Introducing RTL Design
- The Language and Abstraction of Digital Systems
- Review Boolean Algebra
- Review K-Maps, 5 and 6 variables ??
- Logic Minimization and Synthesis
- Introducing WinLogiLab tool

Review material, use reference books:

1. Digital Systems Principles and Applications, 12th edition, Widmer, Moss, Tocci
2. Digital Design, Mano and Ciletti

Useful Number Systems

- Initial electro-mechanical computers were based on analog and decimal numbers (base 10). The computers were difficult to scale, complex, and expensive.
- Binary number system (base 2) along with Boolean Algebra was found to be the most efficient (area \leftrightarrow speed \leftrightarrow power Δ) mechanism for computer implementation in microelectronics technology.
- Switching square waveforms in electronic circuits directly map to '1' and '0' in binary number system.
- In computer systems, Hex numbers (base 16) and 2's Complement numbers are used to represent information in a compact way.



Review of Useful Number Systems

- **Decimal to Binary**

Eg. Convert 37 (base 10) to a binary number (base 2)

$37 \div 2 = 18.5$ (whole number is 18, remainder is 0.5, so put a 1 here)	→	1 (LSB)
$18 \div 2 = 9$ (whole number is 9, remainder is 0 so put a zero here)	→	0
$9 \div 2 = 4.5$ (whole number is 4, remainder is 0.5, put a 1 here)	→	1
$4 \div 2 = 2$ (whole number is 2, remainder is 0, put a 0 here)	→	0
$2 \div 2 = 1$ (whole number is 1, remainder is 0, put a 0 here)	→	0
$1 \div 2 = 0.5$ (whole number is 0, remainder is 0.5, put a 1 here)	→	1 (MSB)

Write MSB to LSB → 1 0 0 1 0 1

Hence $(37)_{10} = (100101)_2$

Binary to Decimal conversion

- Convert Binary number $(1001101)_2$ to Decimal number system
- Each bit position has a proportional weight in the power of 2

1	0	0	1	1	0	1
2^6	2^5	2^4	2^3	2^2	2^1	2^0

MSB

LSB

Add up all the numbers with proportional weights

$$= 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$$= 64 + 8 + 4 + 1 = (77)_{10}$$

Signed Binary Numbers – 2's Complement

- 2's complement of $-(0110010)_2$ Keep one or more extra '0' bit in MSB

Step 1: Take 1's complement $\rightarrow (1001101)$

Step 2: Add '+1' to the 1's Complement

1 0 0 1 1 1 0 1

+ 1

1's Complement means invert all bits

(1 0 0 1 1 1 1 1 0)₂

negative numbers will always have MSB = 1

Hexadecimal	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Hexadecimal Number System (group of 4 bits)

Binary numbers up to 4 bits can be represented as **single Hex digit**
 7-Segment Display can show Hex Digits

Converting from Hex Number System

Hex to Decimal Conversion:

$$\begin{aligned}(356)_{16} &= 3 \times 16^2 + 5 \times 16^1 + 6 \times 16^0 \\ &= 768 + 80 + 6 \\ &= (854)_{10}\end{aligned}$$

Hex to Decimal Conversion:

$$\begin{aligned}(2AF)_{16} &= 2 \times 16^2 + 10 \times 16^1 + 15 \times 16^0 \\ &= 512 + 160 + 15 \\ &= (687)_{10}\end{aligned}$$

Hex to Binary Conversion:

Straightforward

$$\begin{aligned}(9F2)_{16} &= \quad 9 \quad \quad F \quad \quad 2 \\ &= (1001 \ 1111 \ 0010)_2\end{aligned}$$

Converting into Hex Number System

Binary to Hex Conversion:
Straightforward

$$(1110100110)_2 = \begin{array}{ccc} 0011 & 1010 & 0110 \\ = & 3 & A & 6 \end{array}$$

Answer = $(3A6)_{16}$

Decimal to Hex Conversion:

$$(378)_{10} =$$

$$378 \div 16 = 23 + \text{remainder of } 10_{10} = \mathbf{A}_{16}$$

$$23 \div 16 = 1 + \text{remainder of } \mathbf{7}$$

$$1 \div 16 = 0 + \text{remainder of } \mathbf{1}$$

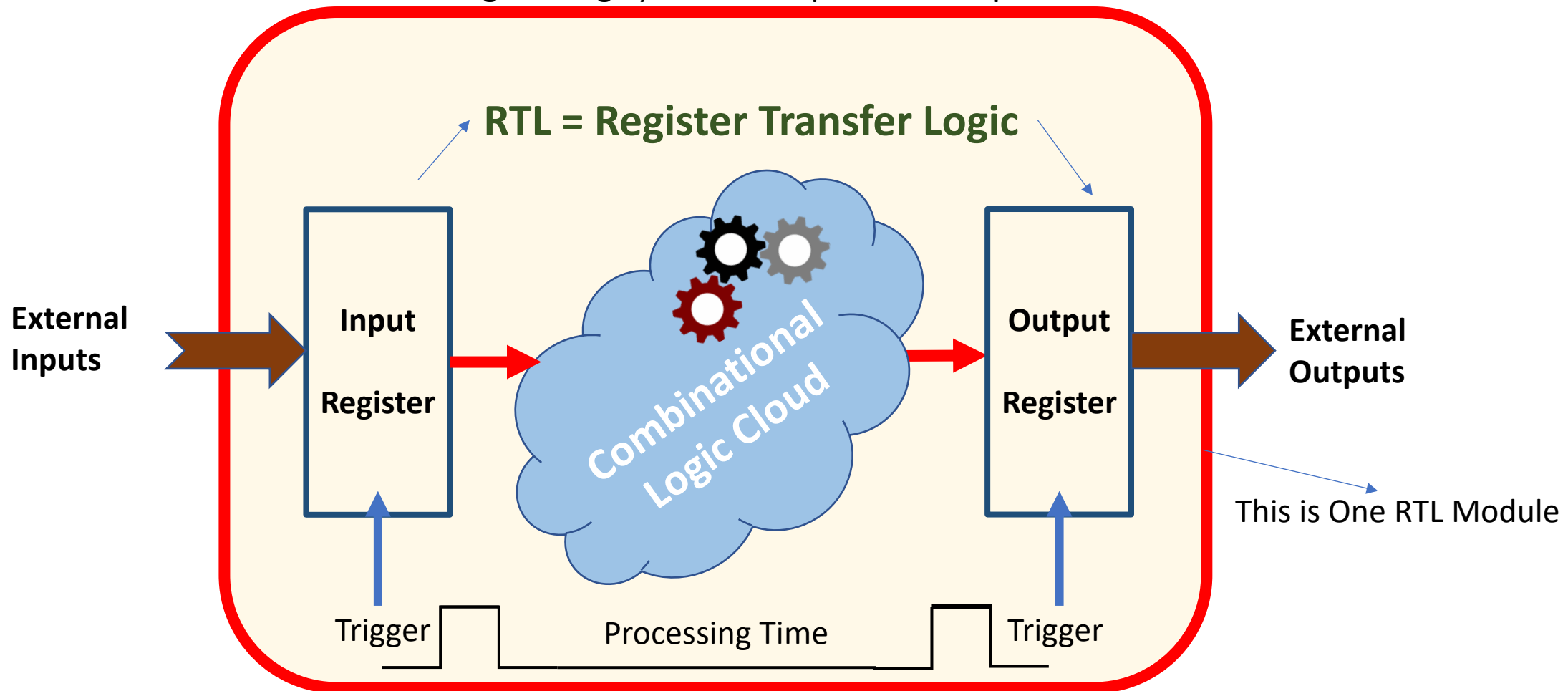
Answer = $(17A)_{16}$

In Binary form:

$$(17A)_{16} = (0001 \ 0111 \ 1010)_2$$

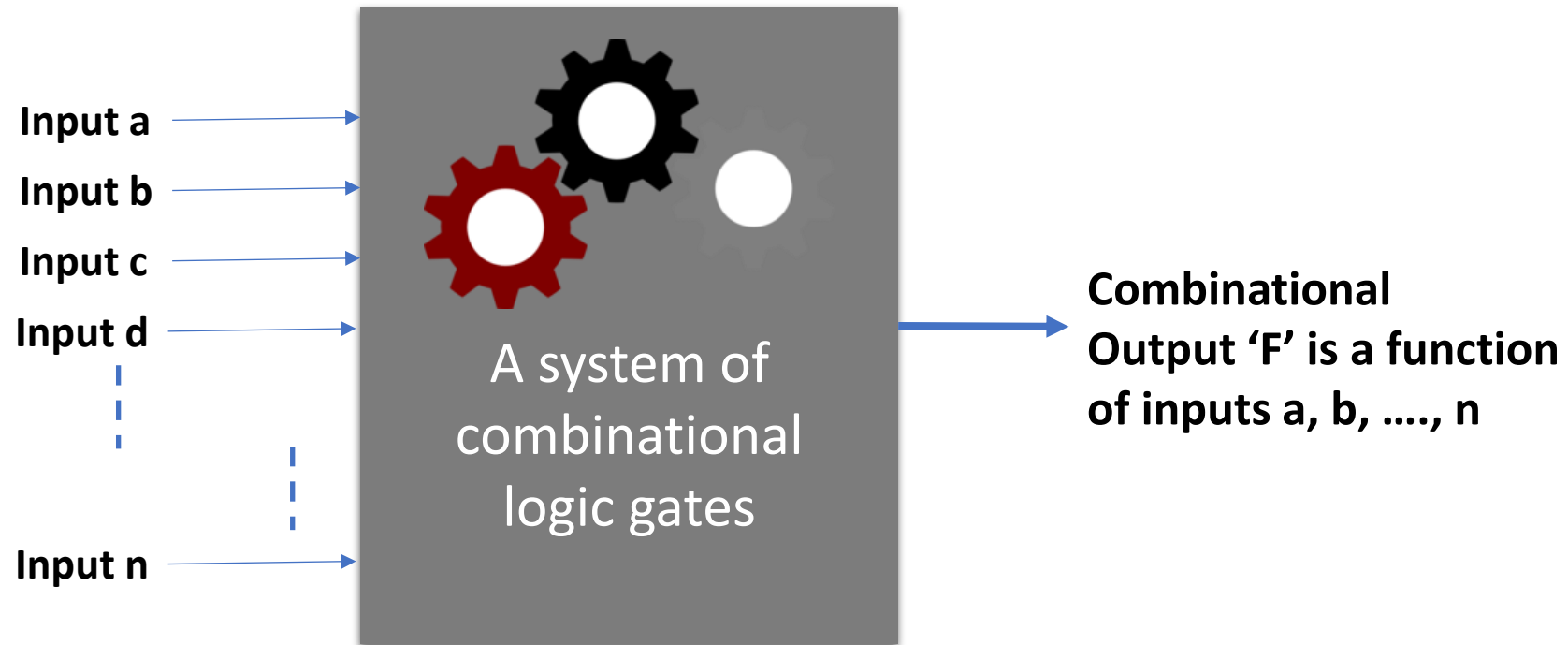
RTL View of Digital System Design

Engineering System has Inputs and Outputs



A Complex Digital System has many RTL Modules

Combinational Logic Circuits



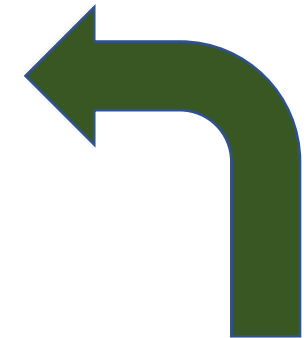
The Language of Digital Systems

- Minterms and Maxterms
- Truth Table
- Boolean Algebra
- K-Maps
- State Tables
- State Machines and Diagrams
- Schematic Diagram
- HDL Code

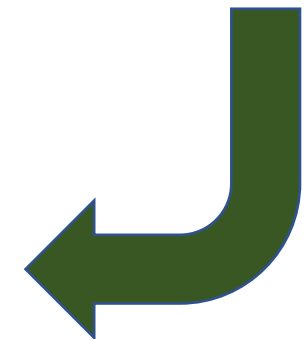
Combinational Logic

Sequential Logic


All types of complex
Logic Functions



linking inputs and
outputs
at different
Abstractions and
Hierarchy levels



Truth table, minterms, SOP




P	Q	R	S output	Expression	Minterms
0	0	0	1	$\bar{P}\bar{Q}\bar{R}$	m0
0	0	1	1	$\bar{P}\bar{Q}R$	m1
0	1	0	1	$\bar{P}Q\bar{R}$	m2
0	1	1	1	$\bar{P}QR$	m3
1	0	0	0	0	
1	0	1	0	0	
1	1	0	0	0	
1	1	1	1	PQR	m7

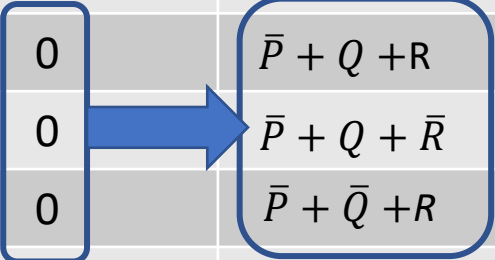
$$S = \bar{P}\bar{Q}\bar{R} + \bar{P}\bar{Q}R + \bar{P}Q\bar{R} + \bar{P}QR + PQR \quad \text{Sum of Products Expression}$$

$$S = \sum (m0, m1, m2, m3, m7) \quad \text{Minterms Expression}$$

Truth table, Maxterms, POS



P	Q	R	F output (POS)	Expression	Maxterms
0	0	0	1		
0	0	1	1		
0	1	0	1		
0	1	1	1		
1	0	0	0	$\bar{P} + Q + R$	M4
1	0	1	0	$\bar{P} + Q + \bar{R}$	M5
1	1	0	0	$\bar{P} + \bar{Q} + R$	M6
1	1	1	1		



$$F_{POS} = (\bar{P} + Q + R).(\bar{P} + Q + \bar{R}).(\bar{P} + \bar{Q} + R)$$

Product of Sum Expression

$$F_{POS} = \prod (M4, M5, M6)$$

Maxterms Expression

Output is **Zero** at F_{POS}

Boolean Algebra theorems

$$x + y = y + x$$

$$x \cdot y = y \cdot x$$

$$x + (y + z) = (x + y) + z = x + y + z$$

$$x(yz) = (xy)z = xyz$$

$$x(y + z) = xy + xz$$

$$(w + x)(y + z) = wy + xy + wz + xz$$

$$x + xy = x$$

$$x + \bar{x}y = x + y$$

$$\bar{x} + xy = \bar{x} + y$$

$$\overline{(x + y)} = \bar{x} \cdot \bar{y}$$

$$\overline{(x \cdot y)} = \bar{x} + \bar{y}$$

Prove through truth table

DeMorgan's Laws

Logic Minimization – using Boolean Algebra

From truth table example: $S = \bar{P}\bar{Q}\bar{R} + \bar{P}\bar{Q}R + \bar{P}Q\bar{R} + \bar{P}QR + PQR$

Identify and
Eliminate
Common terms

$$S = \bar{P}\bar{Q}(\bar{R} + R) + \bar{P}Q(\bar{R} + R) + PQR$$

$$S = \bar{P}\bar{Q} + \bar{P}Q + PQR$$

$$S = \bar{P}(\bar{Q} + Q) + PQR$$

$$S = \bar{P} + PQR$$

Use theorem

$$S = \bar{P} + QR$$

Final simplified expression

Logic Simplification – using Karnaugh K-Maps

2 Variable K-maps

A input	B input	X output
0	0	1
0	1	0
1	0	0
1	1	1

A, B	\bar{B}	B
\bar{A}	1	0
A	0	1

$$X = \bar{A}\bar{B} + AB$$

B, A	\bar{A}	A
\bar{B}	1	0
B	0	1

Three Variable K-Maps

A input	B input	C input	X output
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

AB,C	\bar{C}	C
$\bar{A}\bar{B}$	1	1
$\bar{A}B$	1	0
AB	1	0
$A\bar{B}$	0	0

A,BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
\bar{A}	1	1	0	1
A	0	0	0	1

$$X = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + AB\bar{C}$$

4 Variable K-Maps

A input	B input	C input	D input	X output
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

$$X = \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}D + AB\bar{C}D + ABCD$$

AB, CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	0	0
$\bar{A}B$	0	1	0	0
AB	0	1	1	0
$A\bar{B}$	0	0	0	0

Different orientation - 4 Variable K-Maps

A input	B input	C input	D input	X output
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

$$X = \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}D + AB\bar{C}D + ABCD$$

CD, AB	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	0	0
$\bar{C}D$	1	1	1	0
CD	0	0	1	0
$C\bar{D}$	0	0	0	0

K-Maps with Don't Care Literals

CD, AB	$\overline{A}\overline{B}$	$\overline{A}B$	AB	$A\overline{B}$
$\overline{C}\overline{D}$	0	0	0	0
$\overline{C}D$	1	1	1	0
CD	0	X	1	0
$C\overline{D}$	0	0	0	0

$A'C'D$

BD

Logic Minimization – using Karnaugh K-Maps

Procedure:

Find biggest groups of adjacent '1's

Edges can be folded

The literals that remain fixed in the group are written down

The literals that change, eg A to A' are removed

All 1's to be covered at least once

2 Variable K-maps

Example 1

A, B	\bar{B}	B
\bar{A}	1	0
A	1	0

A and A' are literals that change, so removed
B' remains common in the group
The minimized expression is **B'**

Example 2

B, A	\bar{A}	A
\bar{B}	0	0
B	1	1

A and A' are literals that change, hence dropped
B remains common in the group, retained
The minimized expression is **B**

Logic Minimization – 3 variable K-Maps

Example 3-1

AB,C	\bar{C}	C
$\bar{A}\bar{B}$	1	1
$\bar{A}B$	1	0
AB	1	0
$A\bar{B}$	0	0

The 1 at $A'BC'$ is covered in two groups, only one is sufficient
 So minimization will be on two non-redundant groups
 Minimized Expression is $A'B'$ (where C and C' are removed)
 And BC' (where A and A' are removed).
 Hence Answer is **$A'B' + BC'$** (in the form of Minterms)

A,BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
\bar{A}	1	1	0	1
A	0	0	0	1

Example 3-2

23

Three groups containing two 1's each are identified
 One group $A'B'C'$ and $A'BC'$ is redundant and can be Left out
 C and C' can be removed to give $A'B'$ as one expression
 And A and A' can be removed to give BC' as the other
 Final Minterms expression is **$A'B' + BC'$**

More Examples – 3 variable K-Maps

Example 3-3

AB,C	\bar{C}	C
$\bar{A}\bar{B}$	1	1
$\bar{A}B$	1	1
AB	0	0
$A\bar{B}$	0	0

Group of 4 1's
 C' and B' change and can be removed
 The answer is only A'

Example 3-4

AB,C	\bar{C}	C
$\bar{A}\bar{B}$	1	0
$\bar{A}B$	1	0
AB	1	0
$A\bar{B}$	1	0

Another group of 4 1's
 A and A' can be removed
 B and B' can be removed
 The answer is only C'

Example 3-5

AB,C	\bar{C}	C
$\bar{A}\bar{B}$	1	1
$\bar{A}B$	0	0
AB	0	0
$A\bar{B}$	1	1

Another group of 4 1's with folded edges
 A and A' change so can be removed
 C and C' change so can be removed
 The answer is only B'

Logic Minimization – 4 Variable K-maps

Example 4-1

AB, CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	0	0	0
$\bar{A}B$	0	1	1	0
AB	0	1	1	0
$A\bar{B}$	0	0	0	0

Group of 4 1's
 A' changes along vertical axis
 C' changes along horizontal axis
 Answer is remaining terms = **BD**

Example 4-2

AB, CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	0	0	1
$\bar{A}B$	0	0	0	0
AB	0	0	0	0
$A\bar{B}$	1	0	0	1

Group of 4 1's with folded edges
 C and C' change along horizontal axis
 A and A' change along vertical axis
 Answer is terms that do not change = **B'D'**

Logic Minimization – 4 Variable K-maps

Example 4-3

AB, CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	0	0	0
$\bar{A}B$	1	1	1	1
AB	1	1	1	1
$A\bar{B}$	0	0	0	0

Group of 8 1's
A and A' change in vertical axis
All variables change in horizontal axis
Answer = **B**

Example 4-4

AB, CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	0	0	1
$\bar{A}B$	1	0	0	1
AB	1	0	0	1
$A\bar{B}$	1	0	0	1

Group of 8 1's
Folded along horizontal edges for biggest group
All variables change along vertical axis
C and C' change along horizontal axis
Answer = **D'**

K-Maps for 5 variables ??

Starts to look complicated for manual working
 For any function $X=f\{A,B,C,D,E\}$
 Visualize two 4 input truth tables for $\{B,C,D,E\}$,
 First truth table is when $A=0$, second is when $A=1$

When $A=0$

K-map for $\{B,C,D,E\}$

A can be eliminated if is common for any
 Literal in both K-maps

When $A=1$

K-map for $\{B,C,D,E\}$

K-Maps for 6 or more variables

Generally computer programs are used to model multivariable K-Maps through multidimensional arrays

Conceptually, we can continue to extend 4-variable K-Maps

Eg for $X=f\{A,B,C,D,E,F\}$:

Consider there are four K-Maps of 4 variables $\{C,D,E,F\}$

There is one dedicated K-Map each for $AB=00$, $AB=01$, $AB=10$ and $AB=11$

After simplification in K-Maps, further simplification is done through Boolean Algebra

K-map for $\{C,D,E,F\}$

When $AB=00$

K-map for $\{C,D,E,F\}$

When $AB=01$

K-map for $\{C,D,E,F\}$
K-map for $\{C,D,E,F\}$

When $AB=10$

K-map for $\{C,D,E,F\}$

When $AB=11$

Higher level Logic Minimization

Quine McCluskey Tabular Method is a famous algorithm that is used in many computer programs

Logic Minimizer <https://www.logicminimizer.com/> (up to 8 variables, tabular method)

QMC Logic Minimizer <https://sourceforge.net/projects/qmclm/> (up to 16 variables, based on QM)

Logic Friday <https://logic-friday.software.informer.com/> (up to 16 variables, based on Espresso)

Mini Log https://softwiki.net/Logic_Minimizer.html (up to 40 variables, based on Espresso)

Espresso uses Graph theory and Cubes

<https://ptolemy.berkeley.edu/projects/embedded/pubs/downloads/espresso/index.htm>

An overview document by diligent <https://learn.digilentinc.com/Documents/412>

WinLogiLab program can minimize using both QM and Espresso algorithms


All this leads to:

LOGIC SYNTHESIS Synopsys, Cadence (discuss further in labs)

This is built into modern design automation tools for digital design and implementation on FPGA and ASIC

UC Berkeley is considered as the source of many simulation and synthesis tools, Eg SPICE, ESPRESSO and others

→ ↻ 🏠 ⚠ Not secure | www.hakasoft.com.au/winlogilab



HakaSoft Software

Navigation

- [Home](#)
- [About ...](#)
- [Licencing ...](#)
- [Donate ...](#)

ANDROID APPLICATIONS

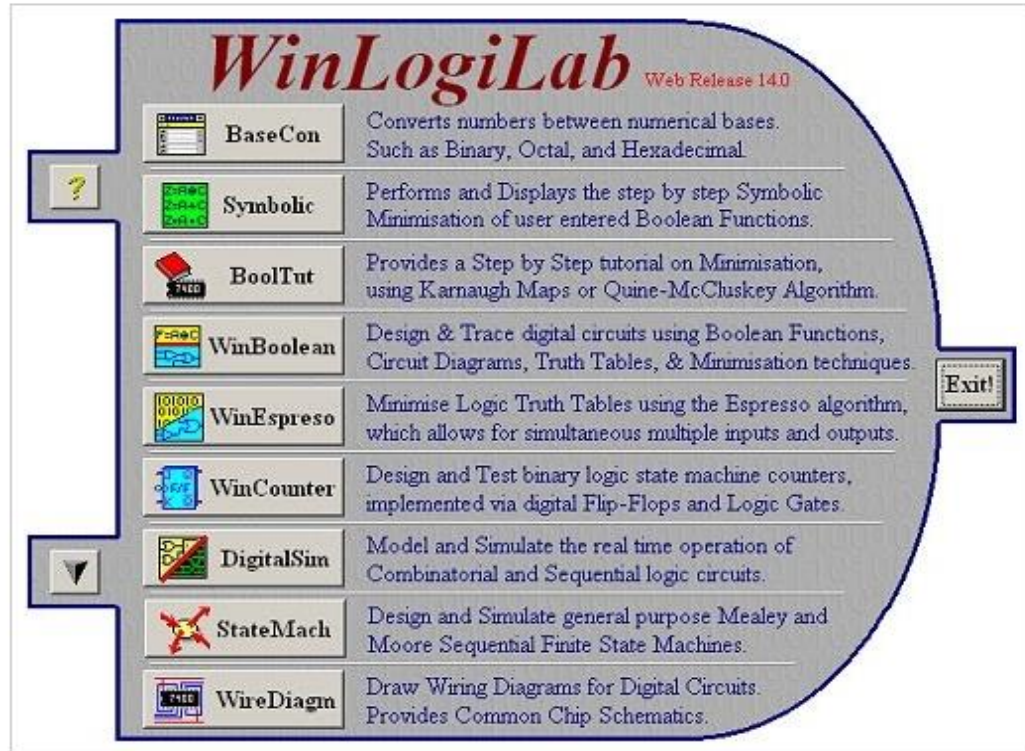
- [BaseCon](#)
- [Table_Minimiser](#)
- [TestKMap](#)
- [SpiceCircuit](#)

WINDOWS APPLICATIONS

- [AstralNav](#)
- [CircSchema](#)
- [GPS_Explorer](#)
- [RadDecay](#)
- [SpiceCircuit](#)
- [WinLogiLab](#)
- [WFM_Viewer](#)
- [WinHandy](#)

WinLogiLab

Computer-Based Teaching Software Suite for Digital Logic Design.



The screenshot shows the WinLogiLab software suite interface. It features a central panel with a list of tools, each with an icon and a description. The tools are: BaseCon (Converts numbers between numerical bases), Symbolic (Performs and Displays the step by step Symbolic Minimisation of user entered Boolean Functions), BoolTut (Provides a Step by Step tutorial on Minimisation, using Karnaugh Maps or Quine-McCluskey Algorithm), WinBoolean (Design & Trace digital circuits using Boolean Functions, Circuit Diagrams, Truth Tables, & Minimisation techniques), WinEspresso (Minimise Logic Truth Tables using the Espresso algorithm, which allows for simultaneous multiple inputs and outputs), WinCounter (Design and Test binary logic state machine counters, implemented via digital Flip-Flops and Logic Gates), DigitalSim (Model and Simulate the real time operation of Combinatorial and Sequential logic circuits), StateMach (Design and Simulate general purpose Mealey and Moore Sequential Finite State Machines), and WireDiagn (Draw Wiring Diagrams for Digital Circuits. Provides Common Chip Schematics). The interface also includes a 'Web Release 14.0' label, a 'Help' icon (question mark), a 'Run' icon (play button), and an 'Exit!' button.

Example Screen Shot