

Digital Logic Appendix A

Boolean Algebra
Gates
Combinatorial Circuits
Sequential Circuits

6.9.2002

Copyright Teemu Kerola 2002

1

Boolean Algebra

- George Boole
 - ideas 1854
- Claude Shannon,
 - apply to circuit design, 1938 (piirisuunnittelu)
- Describe digital circuitry function
 - programming language?
- Optimise given circuitry
 - use algebra (Boolean algebra) to manipulate (Boolean) expressions into simpler expressions

6.9.2002

Copyright Teemu Kerola 2002

2

Boolean Algebra

- Variables: A, B, C
- Values: TRUE (1), FALSE (0)
- Basic logical operations:
 - binary: AND (\bullet), OR ($+$) $A \bullet B = AB$ $B + C$
 - unary: NOT ($\bar{\quad}$) \bar{A} (ja, tai, ei)
- Composite operations, equations
 - precedence: NOT, AND, OR
 - parenthesis $D = A + \bar{B} \bullet C = A + ((\bar{B})C)$

6.9.2002

Copyright Teemu Kerola 2002

3

Boolean Algebra

- Other operations
 - NAND $A \text{ NAND } B = \text{NOT}(A \text{ AND } B) = \overline{AB}$
 - NOR $A \text{ NOR } B = \text{NOT}(A \text{ OR } B) = \overline{A + B}$
- Truth tables
 - What is the result of the operation?

Table A.1

AND	0	1
0	0	0
1	0	1

P	Q	P AND Q
0	0	0
0	1	0
1	0	0
1	1	1

6.9.2002

Copyright Teemu Kerola 2002

4

Postulates, Identities in Boolean Algebra

- How can I manipulate expressions?
 - Simple set of rules?
- Basic identities Table A.2
 - commutative laws (vaihdantalait)
 - distributive laws (osittelulait)
 - identity elements (identiteetit)
 - inverse elements (vasta-alkiot)
 - associative laws (liitântälait)
 - DeMorgan's theorem (DeMorganin laki)

6.9.2002

Copyright Teemu Kerola 2002

5

Gates

(portit)


- Fundamental building blocks
 - easy to build <http://tech-www.informatik.uni-hamburg.de/applets/cmos/cmosdemo.html>
 - implement basic Boolean algebra operations
- Combine to build more complex circuits
 - memory, adder, multiplier (yhteenlaskupiiri, kertolaskupiiri)
- 1-3 inputs, 1 output  AND

Fig. A.1
- Gate delay (viive)
 - change inputs, new output available after gate delay 1 ns? 10 ns?

6.9.2002

Copyright Teemu Kerola 2002

6

Functionally Complete Set

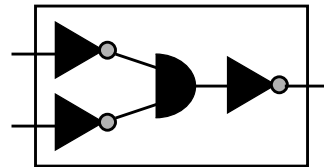
- Can build all basic gates (“and”, “or”, “not”) from a smaller set of gates

- With “and”, “or”, “not” (trivial!)

- With “and”, “not”

- “or”?

$$A + B = \overline{\overline{A} \cdot \overline{B}}$$



OR

- With “or”, “not”

- With “nand” alone

Fig A.2

- With “nor”

Fig A.3

6.9.2002

Copyright Teemu Kerola 2002

7

Combinational Circuits ⁽³⁾

(yhdistelmäpiirit)

- Interconnected set of gates
 - change input, wait for gate delays, new output
- Output is Boolean function of input signals
 - m (binary, Boolean) inputs
 - n (binary, Boolean) outputs
- Described in three ways
 - describe function with Boolean equations (one for each output)

$$F = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\overline{C}$$

- describe function with truth table

Table A.3

- describe implementation with graphical symbols for gates and wires

Fig A.4

6.9.2002

Copyright Teemu Kerola 2002

8

Simplify Presentation (and Implementation) ⁽³⁾

- Boolean equations

- Sum of products form (SOP)

$$F = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\overline{C}$$

Fig A.4

- Product of sums form (POS)

$$F = (A + B + C) \cdot (A + B + \overline{C}) \cdot (\overline{A} + B + C) \cdot (\overline{A} + B + \overline{C}) \cdot (\overline{A} + \overline{B} + \overline{C})$$

Fig A.5

Boolean algebra

Which presentation is better?

Fewer gates? Smaller area on chip?

Smaller circuit delay? Faster?

6.9.2002

Copyright Teemu Kerola 2002

9

Algebraic Simplification

- Circuits become too large to handle?
- Use basic identities to simplify Boolean expressions

$$F = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\overline{C}$$

$$= \overline{A}B + B\overline{C} = B(\overline{A} + \overline{C})$$

Fig A.5

Fig A.6

- May be difficult to do
- How to do it automatically?
- Build a program to do it “best”?

$$f = \overline{a}b\overline{c}d + \overline{a}bcd + a\overline{b}\overline{c}d + ab\overline{c}d$$

$$+ abcd + abc\overline{d} + a\overline{b}cd + a\overline{b}c\overline{d}$$

6.9.2002

Copyright Teemu Kerola 2002

10

Karnaugh Map Squares

- Each square represents complete input value combination

- canonical form: each term has each variable once
- adjacent squares differ only in one input value (wrap around)

CD:	00	01	11	10
<u>AB</u>				
00	0000	0001	0011	0010
01	0100	0101	0111	0110
11	1100	1101	1111	1110
10	1000	1001	1011	1010

order!!

Square for input value combination

$$\overline{A}BCD = 1001$$

6.9.2002

Copyright Teemu Kerola 2002

11

Karnaugh Maps

- Represent Boolean function (I.e., circuit) truth table in another way

Fig A.7

- each square is **one product** in sums-of-products (SOP) presentation
- value is one (1) iff corresponding input values give value 1, o/w value is “empty”
- value is 1 if function value for those input values is one

		CD			
		00	01	11	10
AB	00			1	
	01				
	11	1			
	10		1		

$$F = \overline{A}BCD + A\overline{B}CD + A\overline{B}C\overline{D}$$

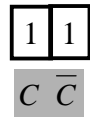
6.9.2002

Copyright Teemu Kerola 2002

12

Karnaugh Map Simplification

- Starting point:
 - Adjacent squares differ only in one input variable value



Adjacent squares have value 1

Input values differ only in one variable

Value of that variable is irrelevant
(when all other input variables are fixed to corresponding values for those squares)

Can ignore that variable for those expressions

6.9.2002

Copyright Teemu Kerola 2002

13

Using Karnaugh Maps to Minimize Boolean Functions ⁽⁶⁾

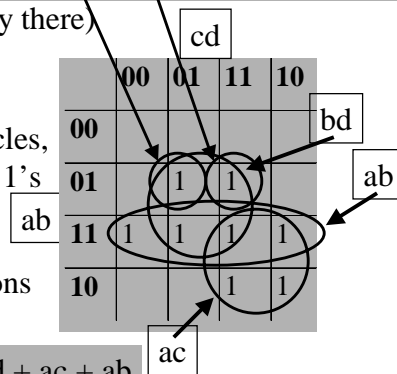
Original function

$$f = \overline{a}b\overline{c}d + \overline{a}bc\overline{d} + a\overline{b}c\overline{d} + a\overline{b}cd + a\overline{b}c\overline{d} + a\overline{b}cd + a\overline{b}c\overline{d} + a\overline{b}cd$$

Canonical form (now already there)

Karnaugh Map

Find smallest number of circles, each with largest number of 1's



Select parameter combinations corresponding to the circles

Get reduced function $f = bd + ac + ab$

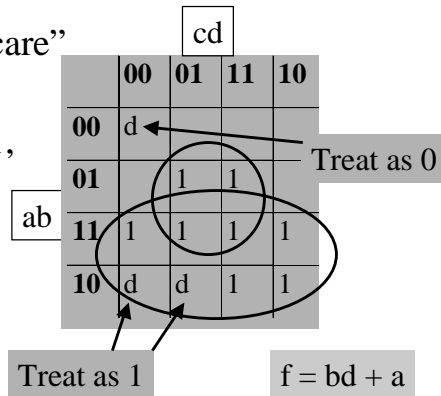
6.9.2002

Copyright Teemu Kerola 2002

14

Impossible Input Variable Combinations

- What if some input combinations can never occur?
 - Mark them "don't care" or "d"
 - treat them as 0 or 1, whichever is best
 - more room to optimize

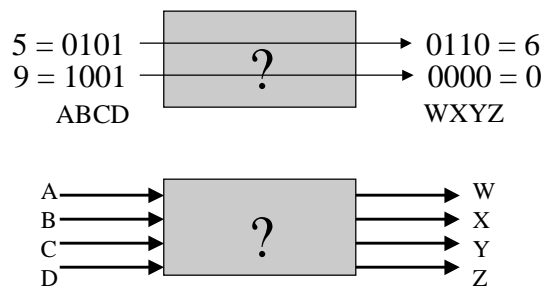


6.9.2002

Copyright Teemu Kerola 2002

15

Example: Circuit to add 1 (mod 10) to 4-bit BCD decimal number ⁽⁴⁾



Truth table?

Table A.4

Karnaugh maps for W, X, Y and Z?

Fig. A.10

6.9.2002

Copyright Teemu Kerola 2002

16

Quine-McKluskey Method

- Another method to minimise Boolean expressions
- Why?
 - Karnaugh maps become complex with 6 input variables
- Quine-McKluskey method
 - tabular method
 - automatically suitable for programming
 - details skipped

6.9.2002

Copyright Teemu Kerola 2002

17

Basic Combinational Circuits

- Building blocks for more complex circuits
 - Multiplexer
 - Encoders/decoder
 - Read-Only-Memory
 - Adder

6.9.2002

Copyright Teemu Kerola 2002

18

Multiplexers ⁽²⁾

- Select one of many possible inputs to output
 - black box Fig A.12
 - simple truth table Tbl A.7
 - implementation Fig A.13
- Each input/output “line” can be many parallel lines
 - select one of three 16 bit values Fig A.14
 - $C_{0..15}$, $IR_{0..15}$, $ALU_{0..15}$
 - simple extension to one line selection
 - lots of wires, plenty of gates ...

6.9.2002

Copyright Teemu Kerola 2002

19

Encoders/Decoders

- Only one of many Encoder input or Decoder output lines can be 1
- Encode the line number as output
 - hopefully less pins (wires) needed this way
 - optimise for space, not time
 - Example:
 - encode 8 input wires with 3 output pins
 - route 3 wires around the board Fig A.15
 - decode 3 wires back to 8 wires at target



6.9.2002

Copyright Teemu Kerola 2002

20

Read-Only-Memory (ROM) ⁽⁴⁾

- Given input values, get output value
 - Like multiplexer, but with fixed data
- Consider input as address, output as contents of memory location
- Truth tables for a ROM Table A.8
 - 64 bit ROM
 - 16 words, each 4 bits wide

Mem (7) = 4

Mem (11) = 14

- Implementation with decoder & or gates

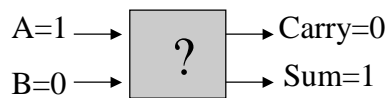
6.9.2002

Copyright Teemu Kerola 2002

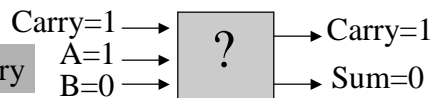
21

Adders ⁽⁴⁾

1-bit adder



1-bit adder with carry



Implementation

Table A.9, Fig A.22

Build 4-bit adder from 4 1-bit adders

Fig A.21

6.9.2002

Copyright Teemu Kerola 2002

22

6.9.2002

Copyright Teemu Kerola 2002

23

Sequential Circuit

(sarjalliset piirit)

- Circuit has (modifiable) internal state
- Output of circuit depends (also) on internal state
 - not only from current inputs
 - $\text{output} = f(\text{input}, \text{state})$
 - $\text{new state} = f(\text{input}, \text{state})$
- Processor control, registers, memory

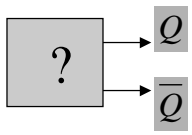
6.9.2002

Copyright Teemu Kerola 2002

24

Flip-Flop (kiikku)

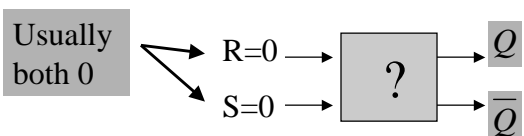
- 2 states for Q (0 or 1, true or false)
- 2 outputs Q and \bar{Q}
 - complement values
 - both always available on different pins
- Need to be able to change the state (Q)




6.9.2002
Copyright Teemu Kerola 2002
25

S-R Flip-Flop or S-R Latch ⁽⁴⁾ (laituri)

Usually both 0

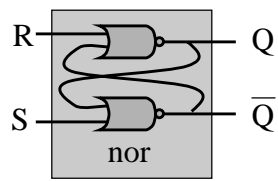


“Write 1” = “set S=1 for a short time” = “set”



“Write 0” = “set R=1 for a short time” = “reset”

nor (0, 0) = 1
 nor (0, 1) = 0
 nor (1, 0) = 0
 nor (1, 1) = 0

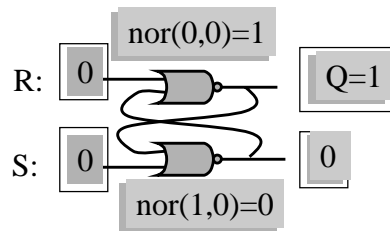


6.9.2002
Copyright Teemu Kerola 2002
26

S-R Latch Stable States (3)

- 1 bit memory (value = value of Q)
- bi-stable, when R=S=0
 - Q=0?
 - Q=1?

nor (0, 0) = 1
 nor (0, 1) = 0
 nor (1, 0) = 0
 nor (1, 1) = 0



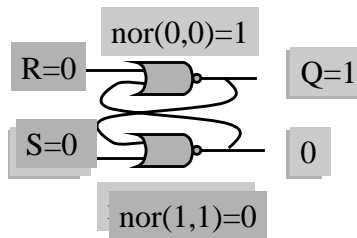
6.9.2002

Copyright Teemu Kerola 2002

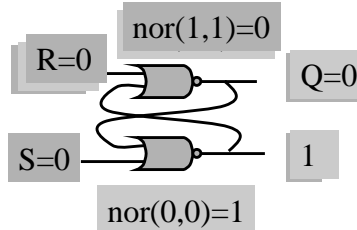
27

S-R Latch Set (1) and Reset (0) (n)

Write 1: S=0 → 1 → 0



Write 0: R=0 → 1 → 0



nor (0, 0) = 1
 nor (0, 1) = 0
 nor (1, 0) = 0
 nor (1, 1) = 0

6.9.2002

Copyright Teemu Kerola 2002

28

Clocked Flip-Flops

- State change can happen only when clock is 1
 - more control on state changes
- Clocked S-R Flip-Flop Fig. A.26
- D Flip-Flop Fig. A.27
 - only one input D
 - D = 1 and CLOCK → write 1
 - D = 0 and CLOCK → write 0
- J-K Flip-Flop Fig. A.28
 - Toggle Q when J=K=1 Fig. A.29

6.9.2002

Copyright Teemu Kerola 2002

29

Registers (2)

- Parallel registers Fig. A.30
 - read/write
 - CPU user registers
 - additional internal registers
- Shift Registers Fig. A.31
 - shifts data 1 bit to the right
 - serial to parallel?
 - ALU operation?
 - rotate?

6.9.2002

Copyright Teemu Kerola 2002

30

Counters ⁽⁴⁾

- Add 1 to stored counter value
- Counter
 - parallel register plus increment circuits
- Ripple counter
 - asynchronous
 - increment least significant bit, and handle “carry” bit as far as needed
- Synchronous counter
 - modify all counter flip-flops simultaneously
 - faster, more complex, more expensive

6.9.2002

Copyright Teemu Kerola 2002

31

Summary

- Boolean Algebra → Gates → Circuits
 - can implement all with NANDs or NORs
 - simplify circuits: Karnaugh, Quine-McKluskey
- Components for CPU design
 - ROM, adder
 - multiplexer, encoder/decoder
 - flip-flop, register, shift register, counter

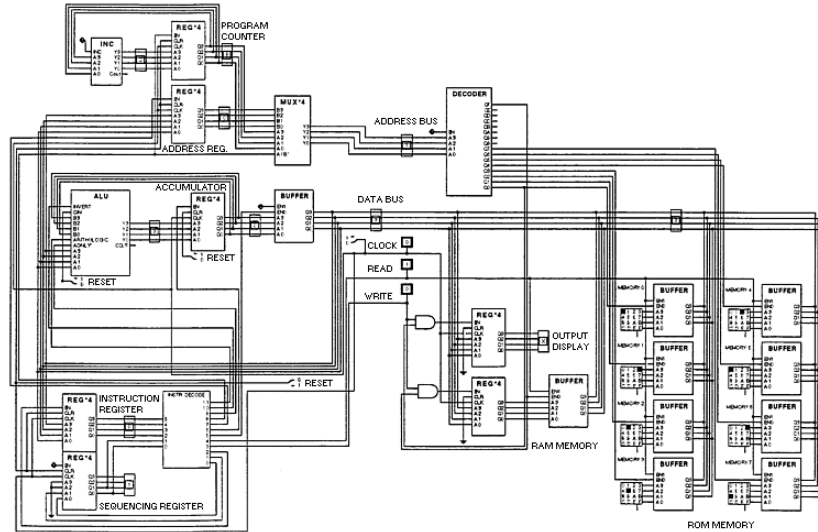
6.9.2002

Copyright Teemu Kerola 2002

32

-- End of Appendix A: Digital Logic --

Simple processor



http://www.gamezero.com/team-0/articles/math_magic/micro/stage4.html

6.9.2002

Copyright Teemu Kerola 2002

33