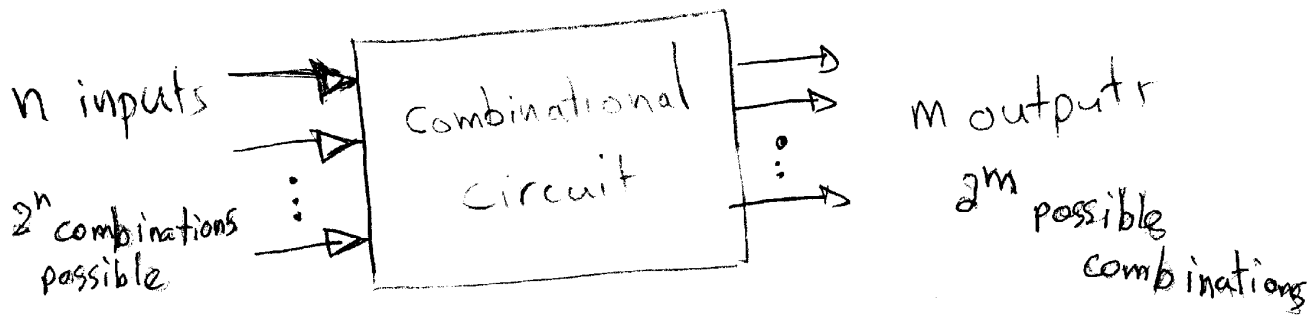
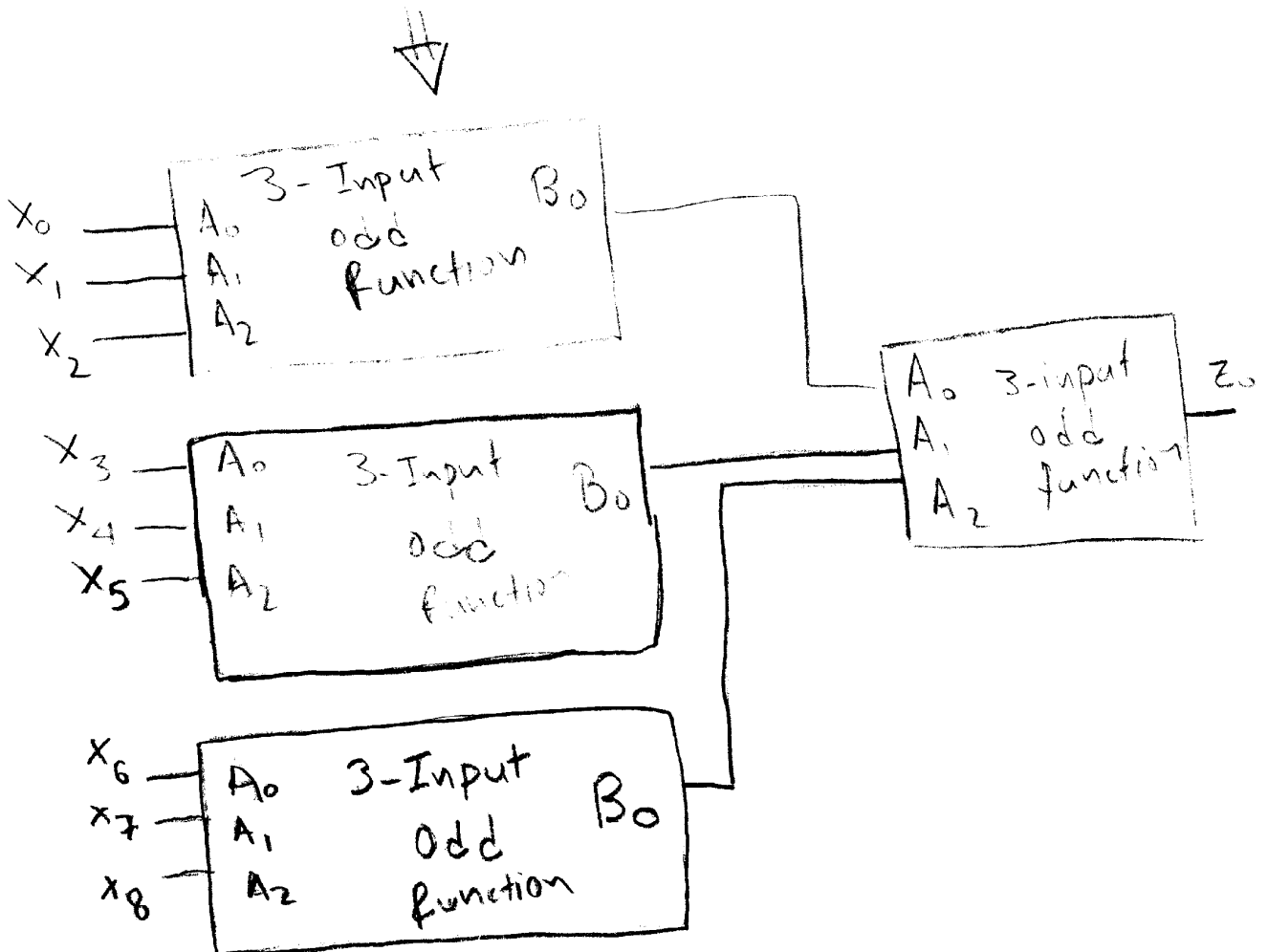
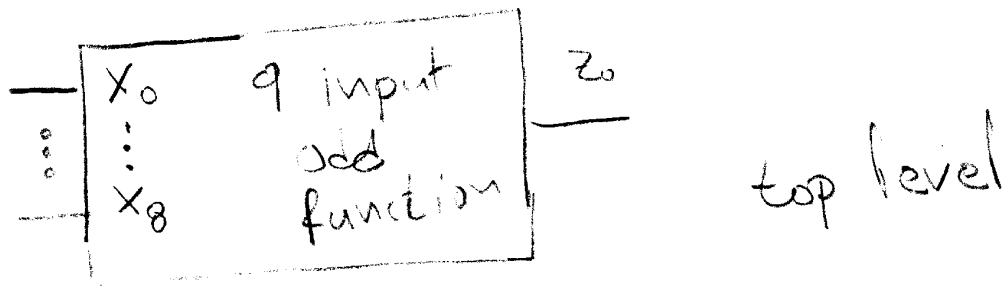
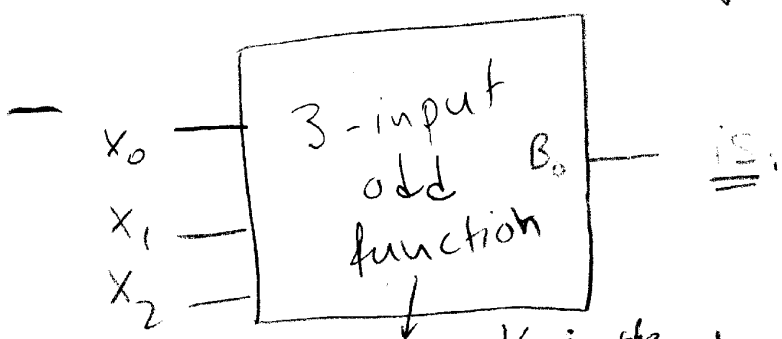


Chapter 3
Combinational Logic Design P. 91

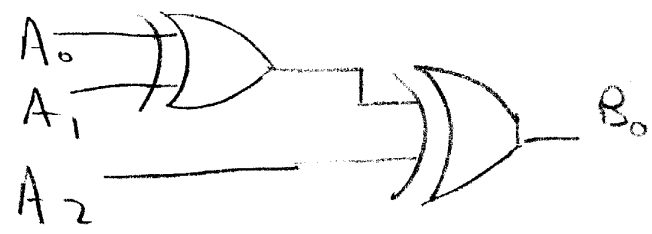


Design Hierarchy





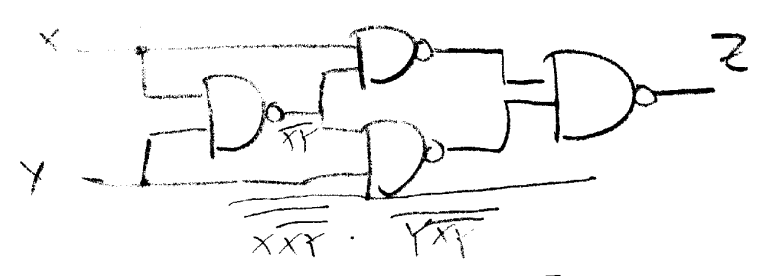
The block is reusable in the sense that it can be used in more than one place in the circuit design.



"Bottom" level.



is:



Top-Down vs. Bottom-Up

CAD for design & verification.

$$\begin{aligned}
 &= x\bar{y} + y\bar{x} \\
 &= x(\bar{x} + y) + y(\bar{x} + \bar{y}) \\
 &= \cancel{x\bar{x}} + x\bar{y} + y\bar{x} + \cancel{y\bar{y}} \\
 &= x\bar{y} + \bar{x}y
 \end{aligned}$$

Computer-aided design tools are implemented to design complex systems & integrated circuits.

Schematic Capture tools support the drawing of blocks & interconnections at all levels in the hierarchy.

At the level of primitives and functional blocks, libraries of graphics symbols are provided.

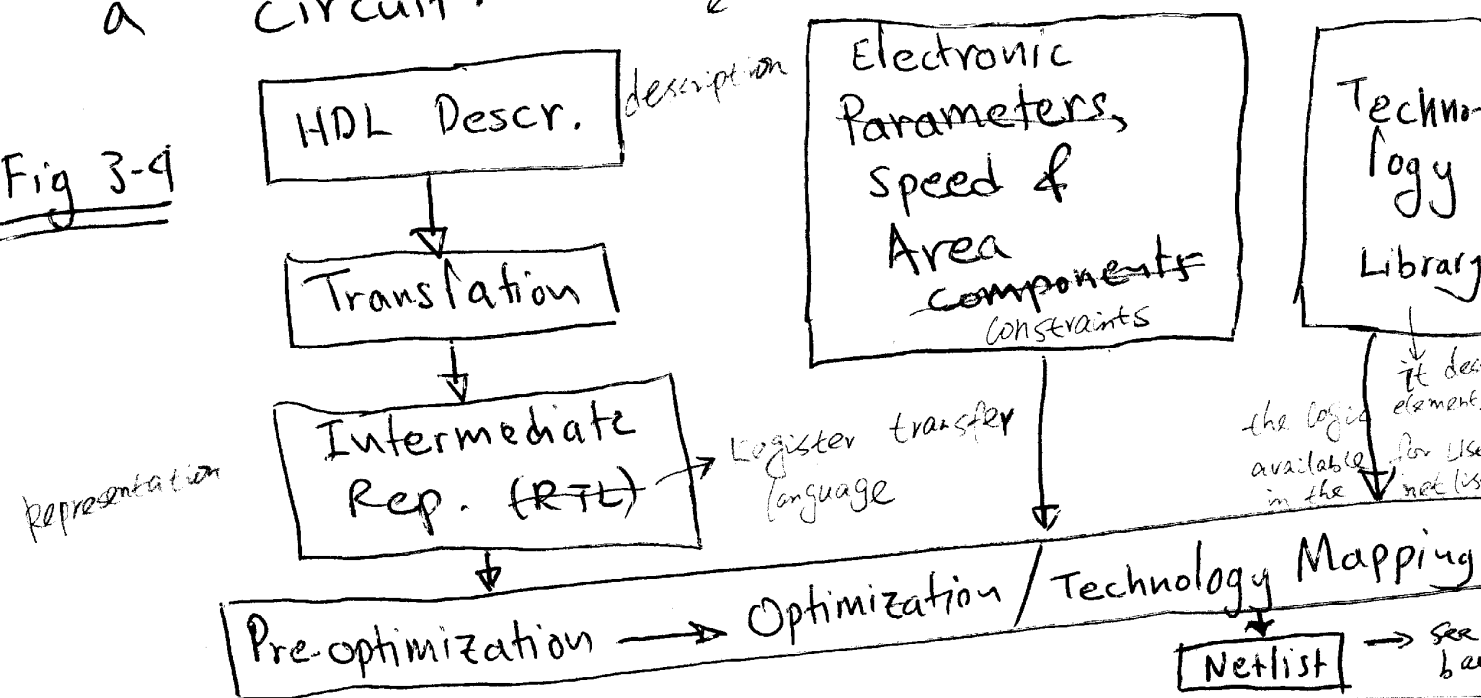
Hardware Description Languages

→ VHDL, Verilog

- ① structural description
 - used instead of schematics to describe the interconnection of components.
- ② → used in top-down design, very high-level description of the system.
- ③ → used to compute outputs for given inputs (but not at the same timing)

④ Logic synthesis
Hardware Description Language (HDL) is written in a register transfer language (RTL) and a logic synthesis tool with a library of components ^{use} converts this to a circuit. *High-level flow for Logic Synthesis Tool.*

Fig 3-9



The optimized netlist represents storage elements and combinational logic.

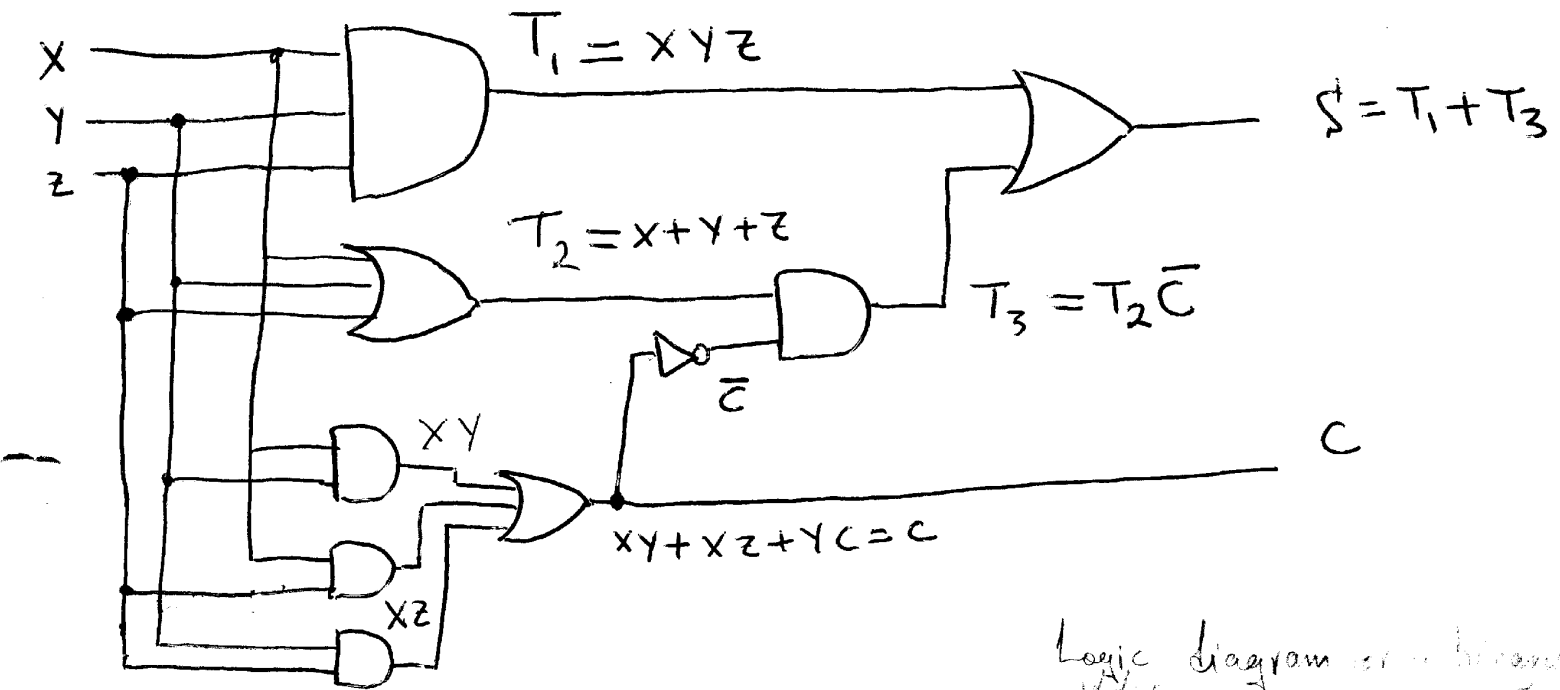
Subsequently, this netlist may be transformed by using physical design tools into an actual integrated circuit layout, which serves as the basis for integrated circuit manufacture.

Design optimization performance constraints:

Faster circuits \Rightarrow larger area \Rightarrow more expensive

lower circuits \Rightarrow smaller area \Rightarrow less expensive

3-3 Analysis. (p 102), Fig 3-6.



Logic diagram of a full adder

Derivation of output Boolean functions from a logic diagram:
Proceed from left to right indicating outputs in-terms of preceding inputs.

CKT Inputs: X, Y, Z

Outputs: C, S

Determine truth table. (p103)

Truth table for binary adder.

X	Y	Z	T ₁	T ₂	T ₃	C	S
0	0	0	0	0	0	0	0
0	0	1	0	1	1	0	1
0	1	0	1	0	0	0	1
0	1	1	1	1	0	1	0
1	0	0	1	0	1	1	0
1	0	1	1	1	0	1	1
1	1	0	0	1	1	1	0
1	1	1	0	0	1	1	1

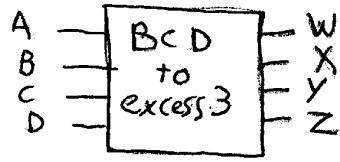
3-4 Design Procedure:

From the specifications of the circuit, determine the required number of inputs & outputs.

1. Determine inputs & outputs
2. Derive truth table.
3. Simplify Boolean functions
4. Draw logic diagram.
5. Verify correctness.

EX 3-2 P107

- BCD to Excess-3:



6/

① Input: BCD digits: $(0000)_2$ to $(1001)_2$

Output: Excess-3 = BCD-digit + 3.

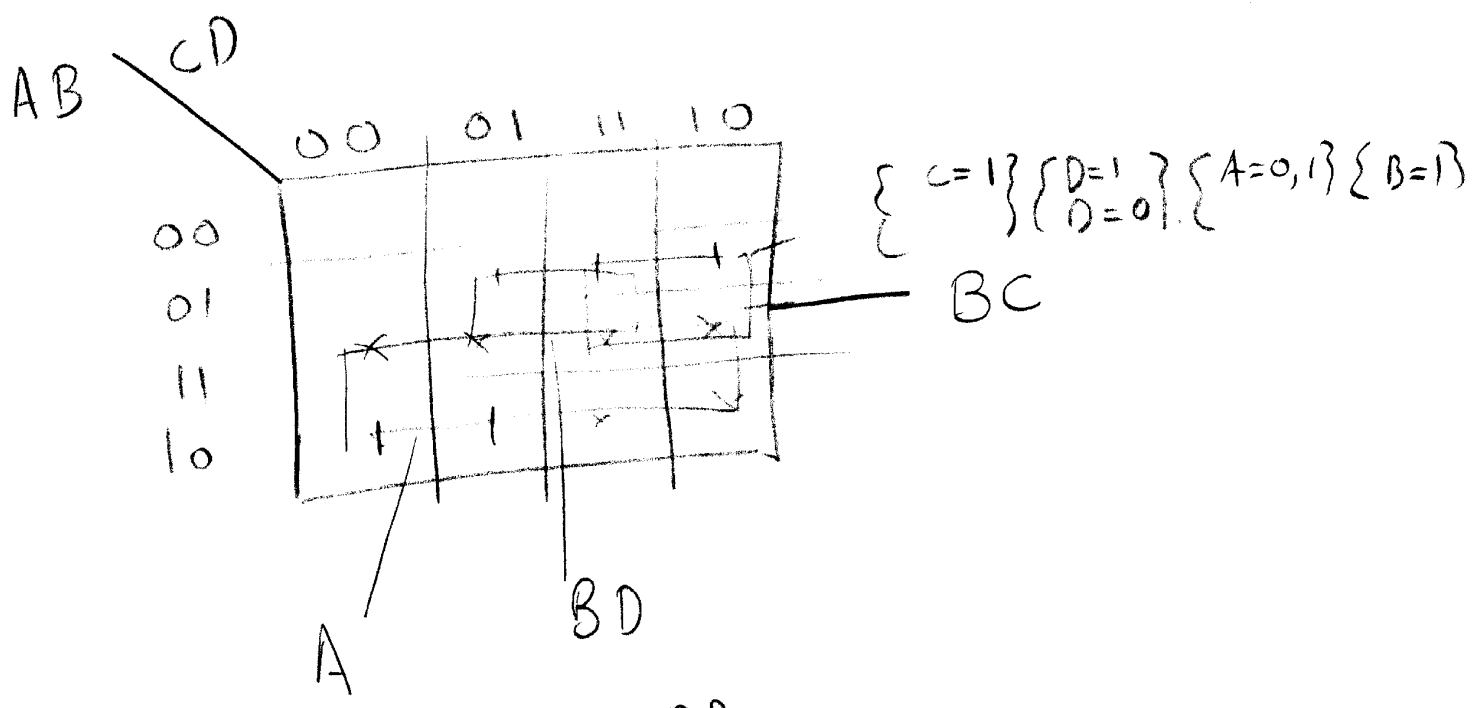
② Truth table

BCD				Excess-3			
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	0
1	0	1	1	1	1	1	1
1	1	0	0	1	1	1	1
1	1	0	1	1	1	1	1
1	1	1	0	1	1	1	1
1	1	1	1	1	1	1	1

Finish
for all inputs

③ Build k-maps for W, X, Y, Z

7



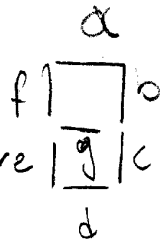
$$W = A + BC + BD$$

Similarly for X, Y, Z ...

④ show the circuit

Plot for W
Similar for the rest.

BCD-to-Seven-Segment Decoder



Digital readouts found in electronic calculators & digital watches use LEDs.

Each digit of the readout is formed from seven segments, each consisting of one LED that can be illuminated by digital signals.

① Inputs: BCD digit

Outputs: Seven segment decoder

② Truth table:

The truth table assumes that a logic 1 signal illuminates the segment, and a logic 0 signal turns the segment off.
Seven-segment Decoder.

See back.

BCD				a	b	c	d	e	f	g
A	B	C	D							
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
		⋮					⋮			
1	0	0	1							
1	0	1	0	0	0	0	⋮	0		
1	1	1	1	0	0	⋮	⋮	0		

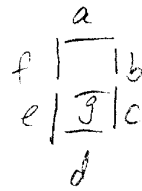
③ Use K-maps to simplify for a, b, c, ..., g.

expression

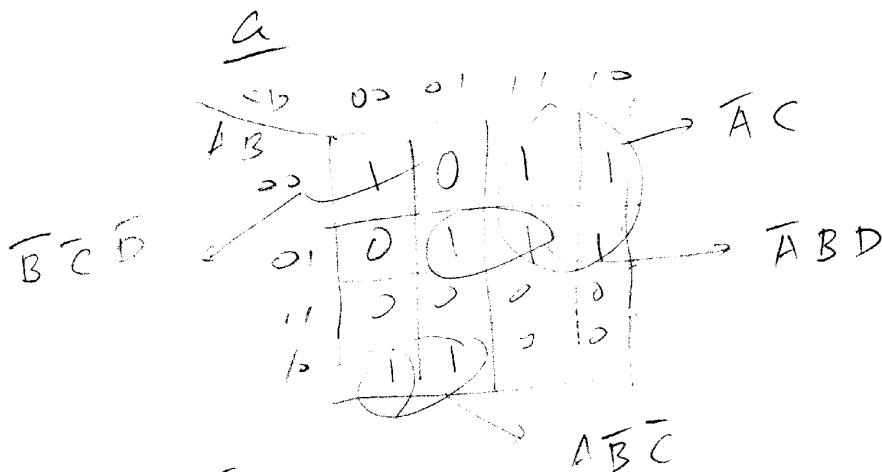
maybe to turn off all the segments

④ Indicate 1111

The binary combinations 1010 through 1111 have no meaning in BCD. In the previous example, we assigned these combinations to don't-care conditions. If we do the same here, the design will most likely produce some arbitrary and meaningless displays for the unused combinations \Rightarrow a better choice



BCD input				Seven-Segment							Decoder
A	B	C	D	a	b	c	d	e	f	g	
0	0	0	0	1	1	1	1	1	1	0	
0	0	0	1	0	1	1	0	0	0	0	
0	0	1	0	1	1	0	1	1	0	1	
0	0	1	1	1	1	1	1	0	0	1	
0	1	0	0	0	1	1	0	0	1	1	
0	1	0	1	1	0	1	1	0	1	1	
0	1	1	0	1	0	1	1	1	1	1	
0	1	1	1	1	1	1	0	0	0	0	
1	0	0	0	1	1	1	1	1	1	1	
1	0	0	1	0	0	0	0	0	0	0	
All other inputs				0	0	0	0	0	0	0	



$$a = \overline{A}C + \overline{A}BD + \overline{B}\overline{C}\overline{D} + A\overline{B}\overline{C}$$

Problem 3-12