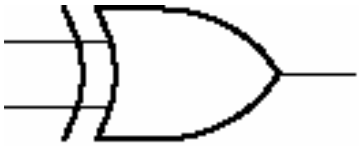
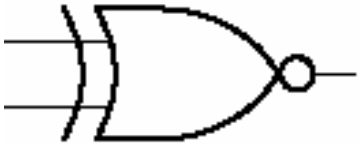
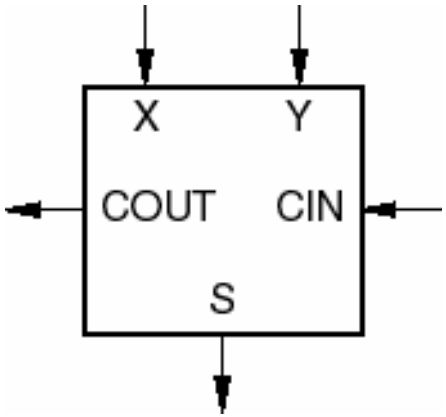


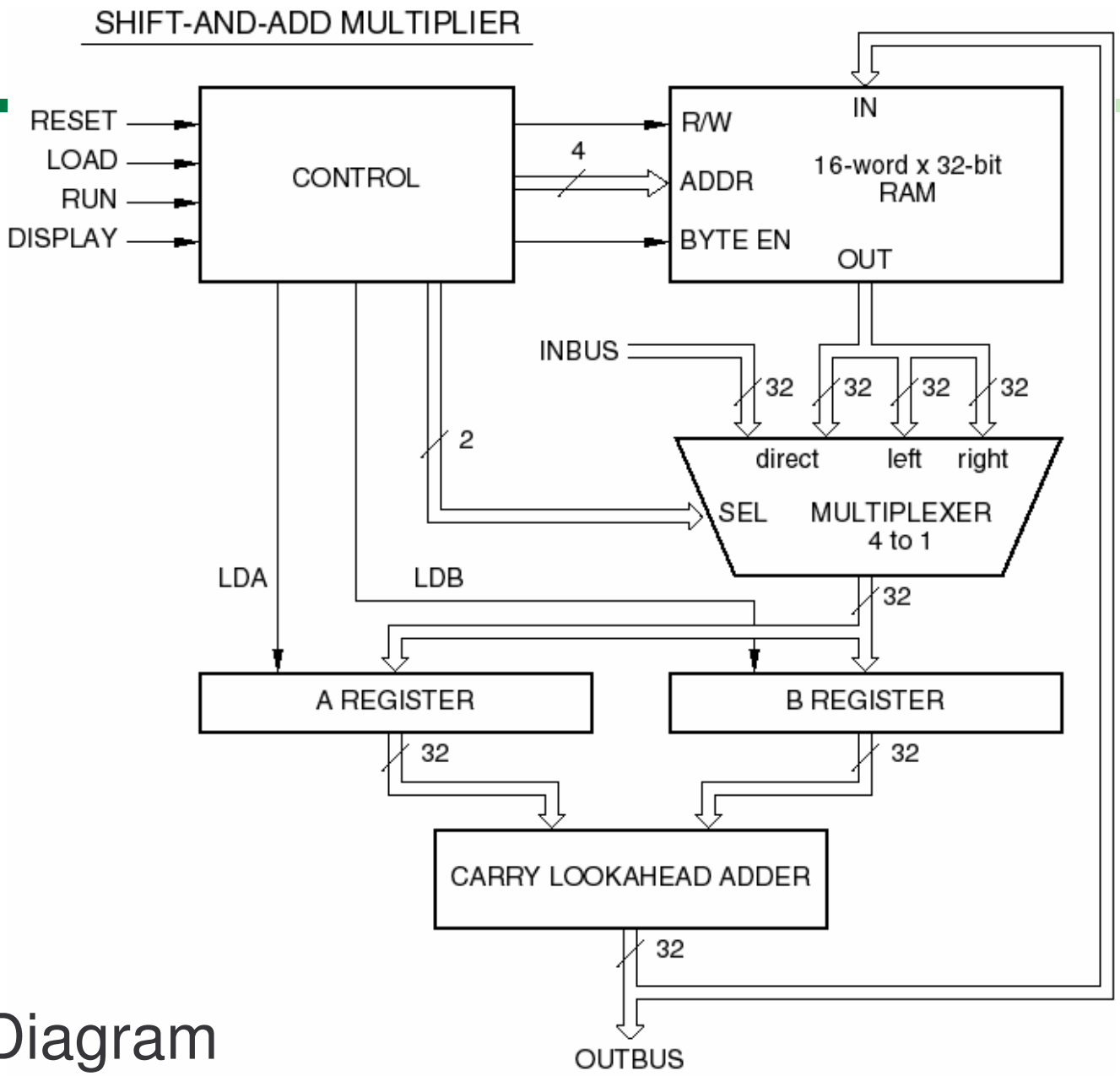


More Digital Design

ECGR2181

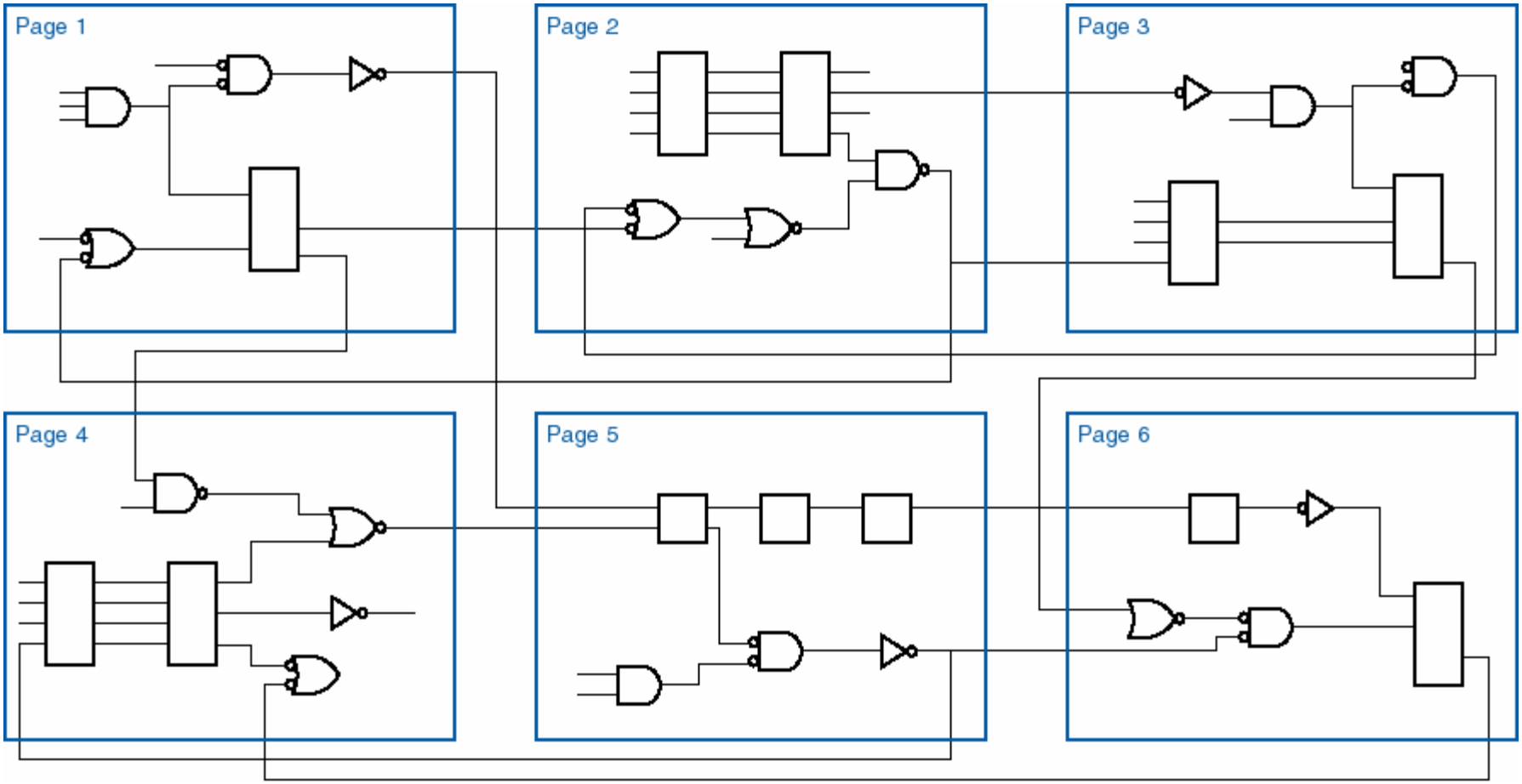


Reading: Chapter 5.1, 2, 3, 8, 10

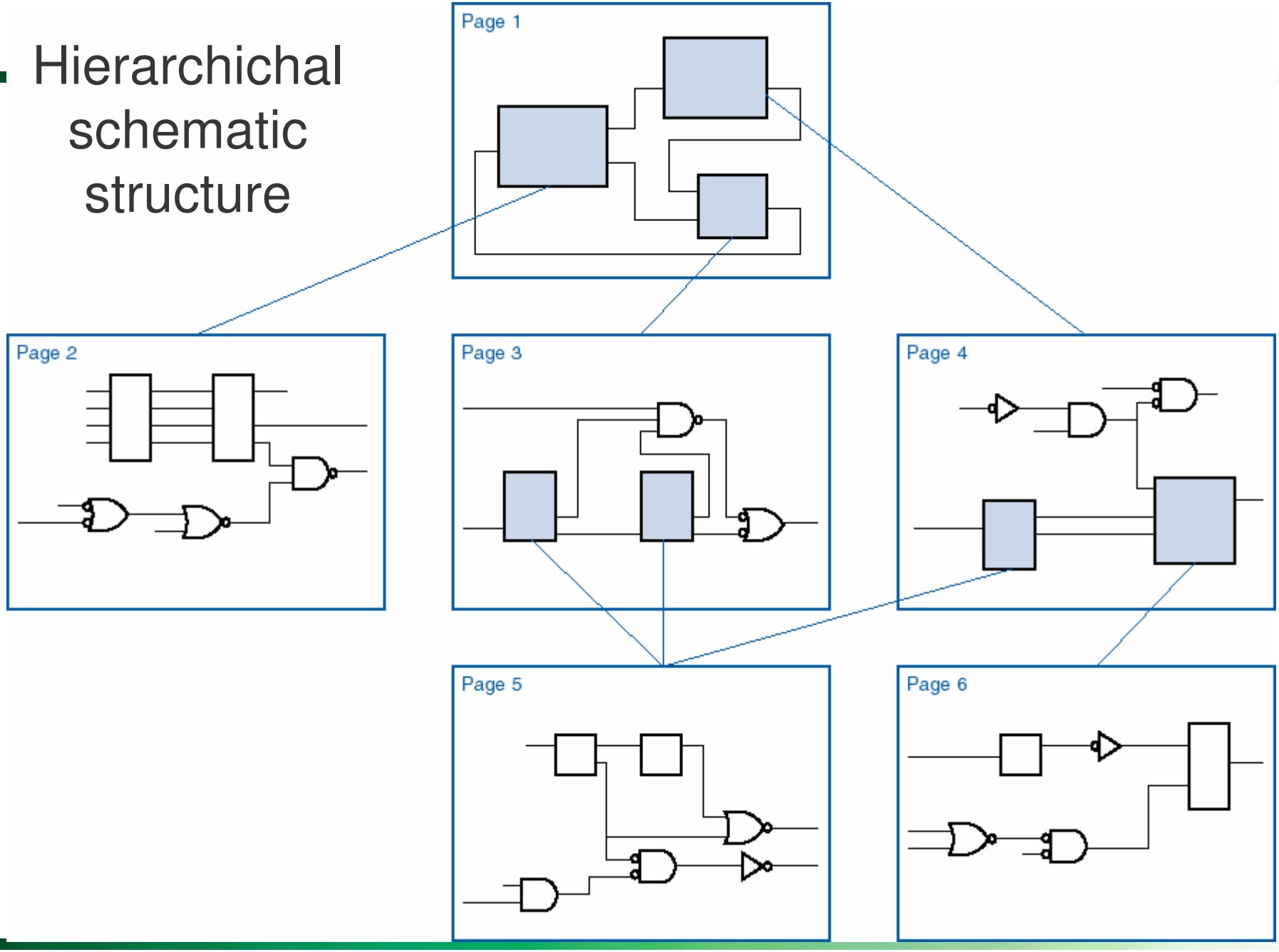


Block Diagram

Flat schematic structure



Hierarchichal schematic structure



Other Documentation

Timing diagrams

- Output from simulator
- Specialized timing-diagram drawing tools

Circuit descriptions

- Text (word processing)
- Can be as big as a book
- Typically incorporate other elements (block diagrams, timing diagrams, etc.)

Signal names and active levels

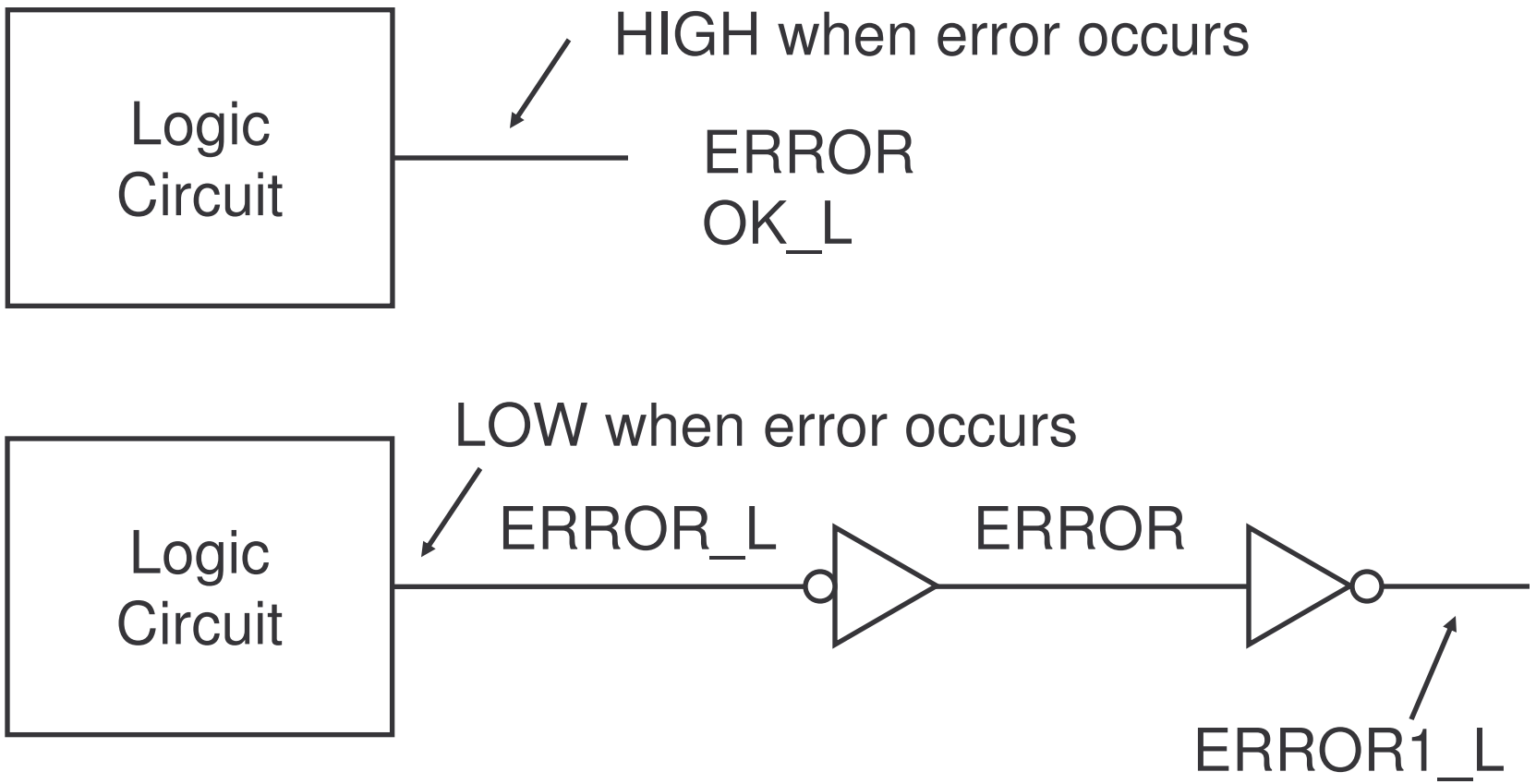
Signal names are chosen to be descriptive.

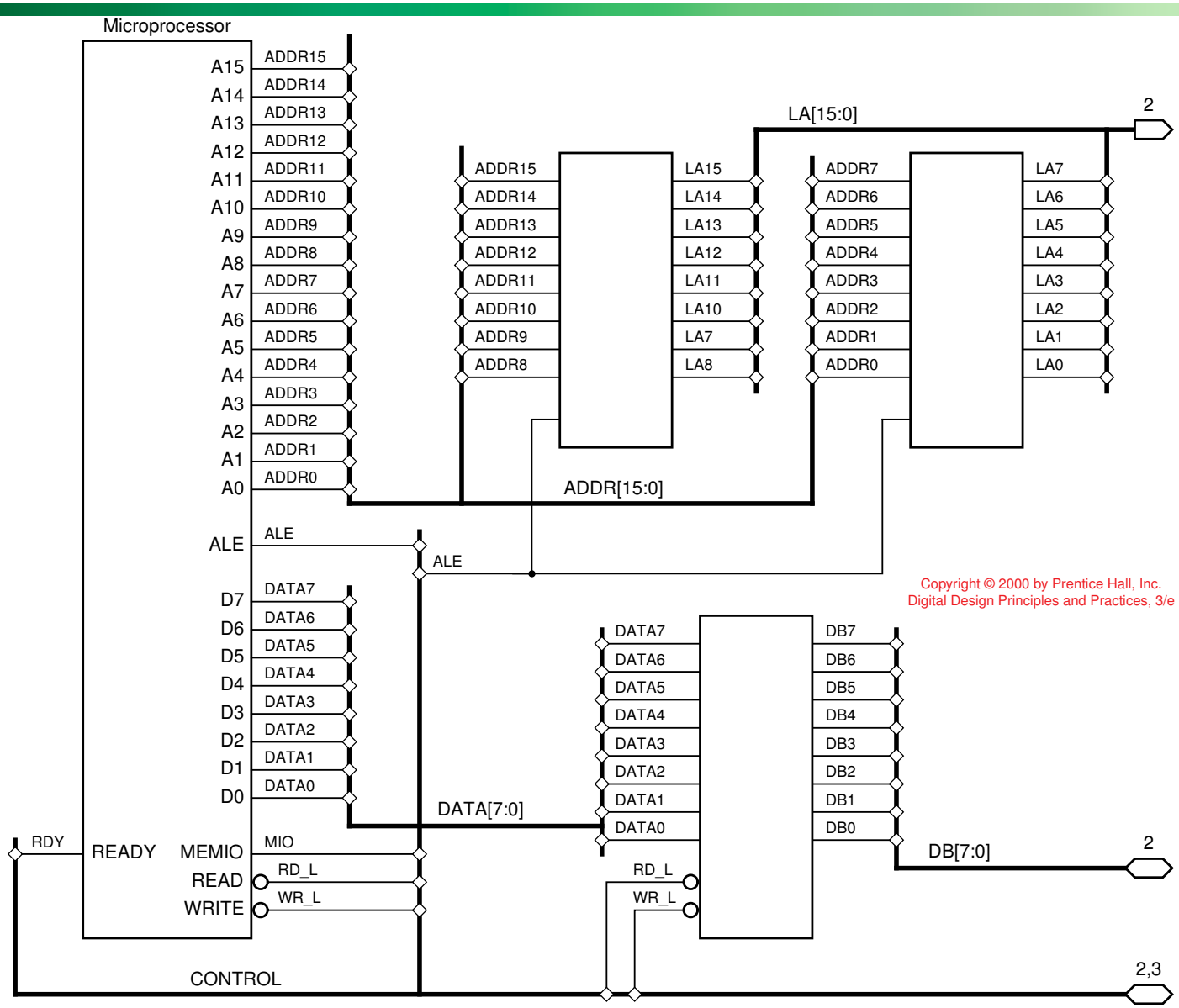
Active levels -- HIGH or LOW

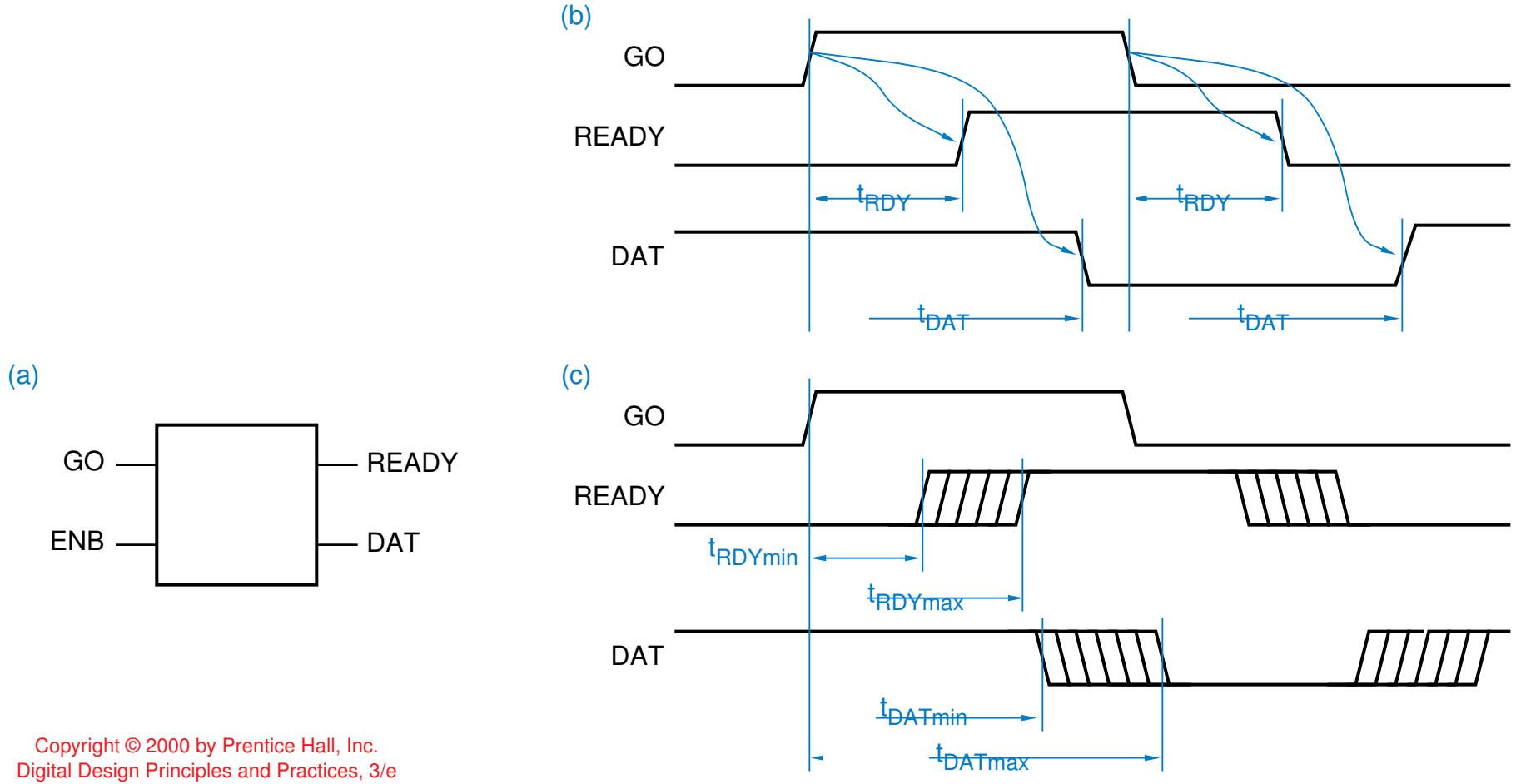
- named condition or action occurs in either the HIGH or the LOW state, according to the active-level designation in the name.

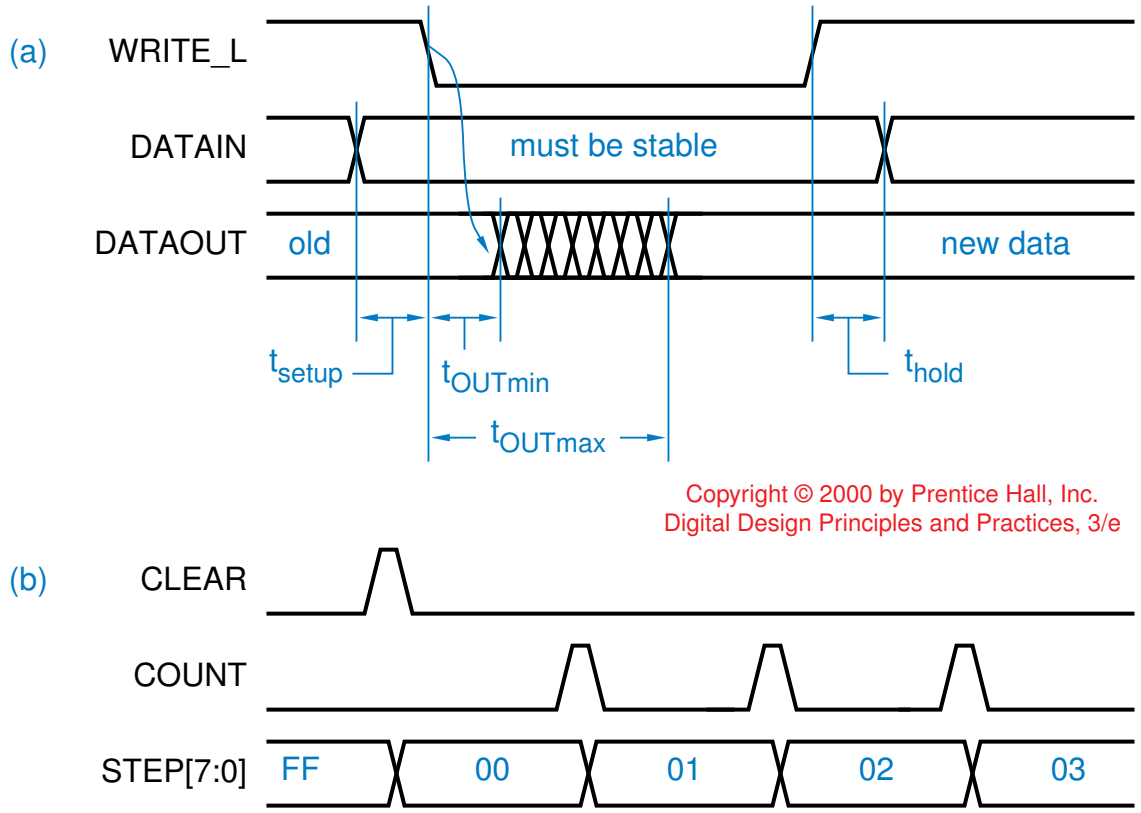
Active Low	*Active High*
READY–	READY+
ERROR.L	ERROR.H
ADDR15(L)	ADDR15(H)
RESET*	RESET
ENABLE~	ENABLE
~GO	GO
/RECEIVE	RECEIVE
TRANSMIT_L	TRANSMIT

Example


Logic
Circuit
HIGH when error occurs
ERROR
OK_L
Logic
Circuit
LOW when error occurs
ERROR_L
ERROR
ERROR1_L







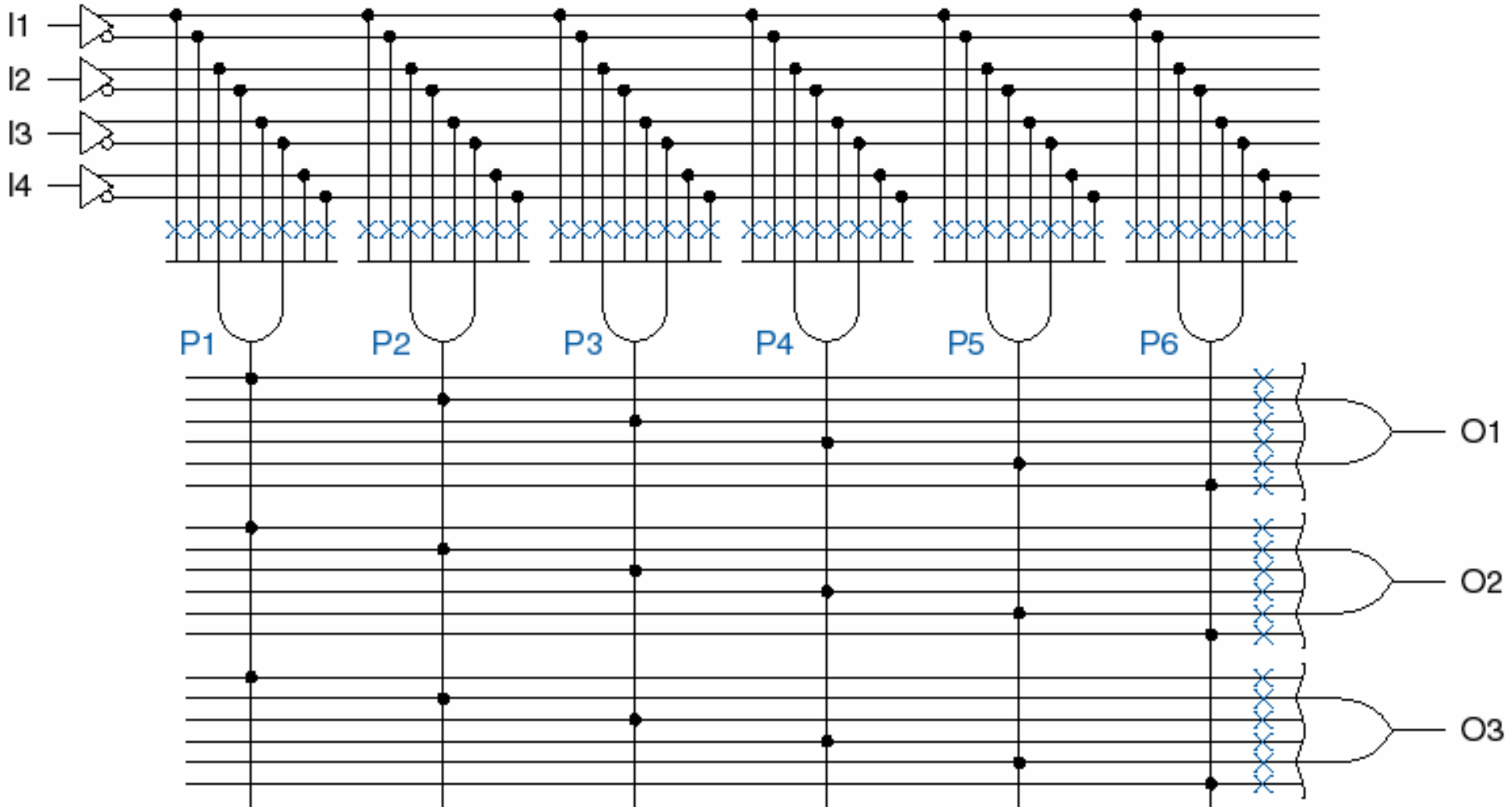
Programmable Logic Arrays (PLAs)

Any combinational logic function can be realized as a sum of products.

Idea: Build a large AND-OR array with lots of inputs and product terms, and programmable connections.

- *n* inputs
  - AND gates have 2*n* inputs -- true and complement of each variable.
- *m* outputs, driven by large OR gates
  - Each AND gate is programmably connected to each output's OR gate.
- *p* AND gates ($p << 2^n$)

Example: 4x3 PLA, 6 product terms



Programmable Array Logic (PALs)

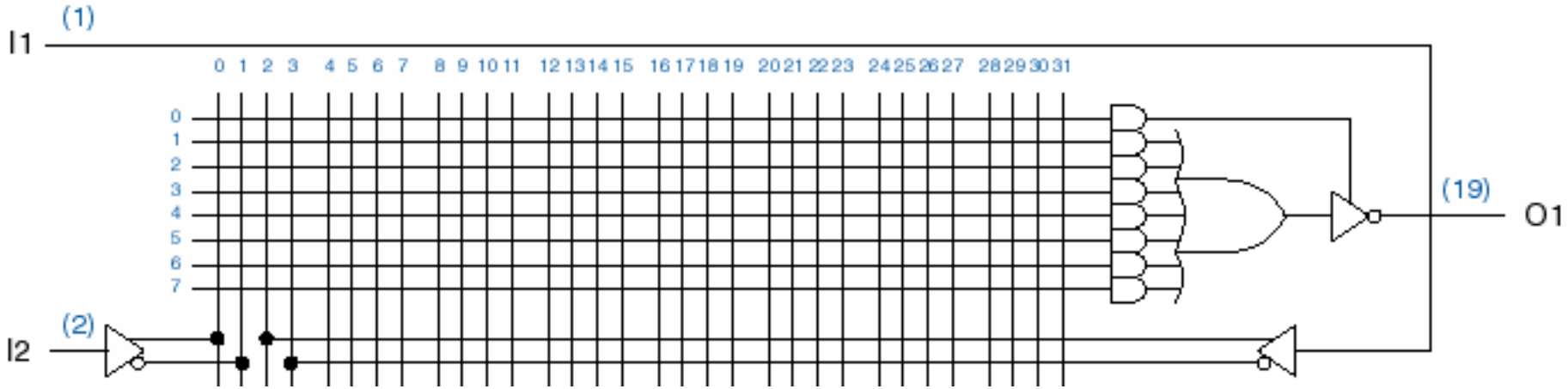
How beneficial is product sharing?

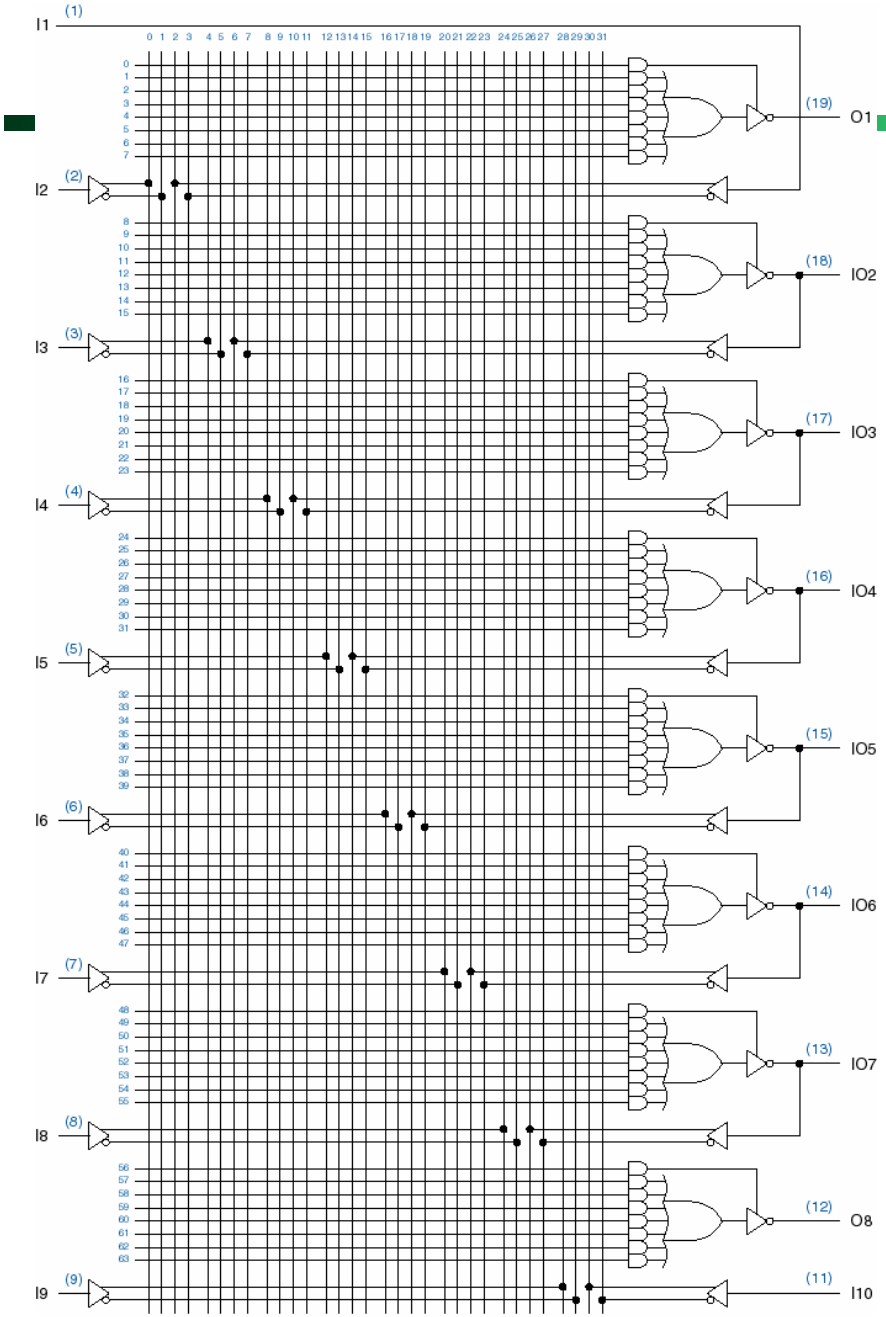
- Not enough to justify the extra AND array

PALs ==> *fixed* OR array

- Each AND gate is permanently connected to a certain OR gate.

Example: PAL16L8





- 10 primary inputs
- 8 outputs, with 7 ANDs per output
- 1 AND for 3-state enable
- 6 outputs available as inputs
  - more inputs, at expense of outputs
  - two-pass logic, helper terms
- Note inversion on outputs
  - output is complement of sum-of-products
  - newer PALs have selectable inversion

Designing with PALs

Compare number of inputs and outputs of the problem with available resources in the PAL.

Write equations for each output using HDL.

Compile the HDL program, determine whether minimized equations fit in the available AND terms.

If no fit, try modifying equations.

Documentation Standards

Block diagrams

- first step in hierarchical design

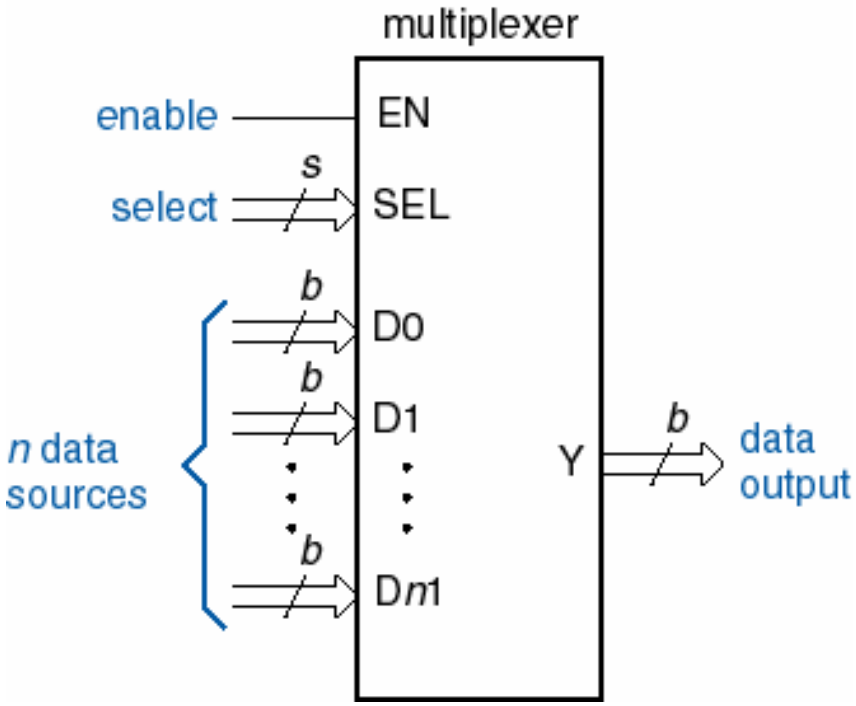
Schematic diagrams

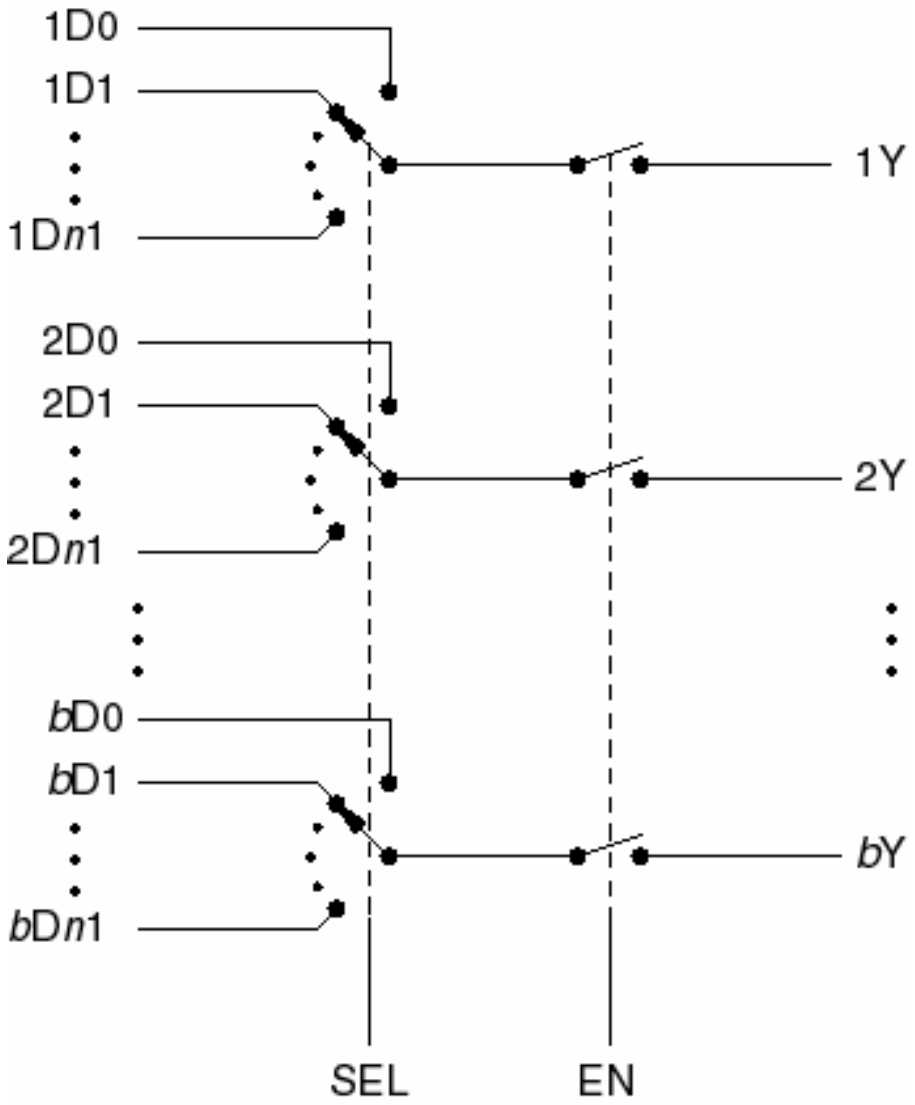
HDL programs (ABEL, Verilog, VHDL)

Timing diagrams

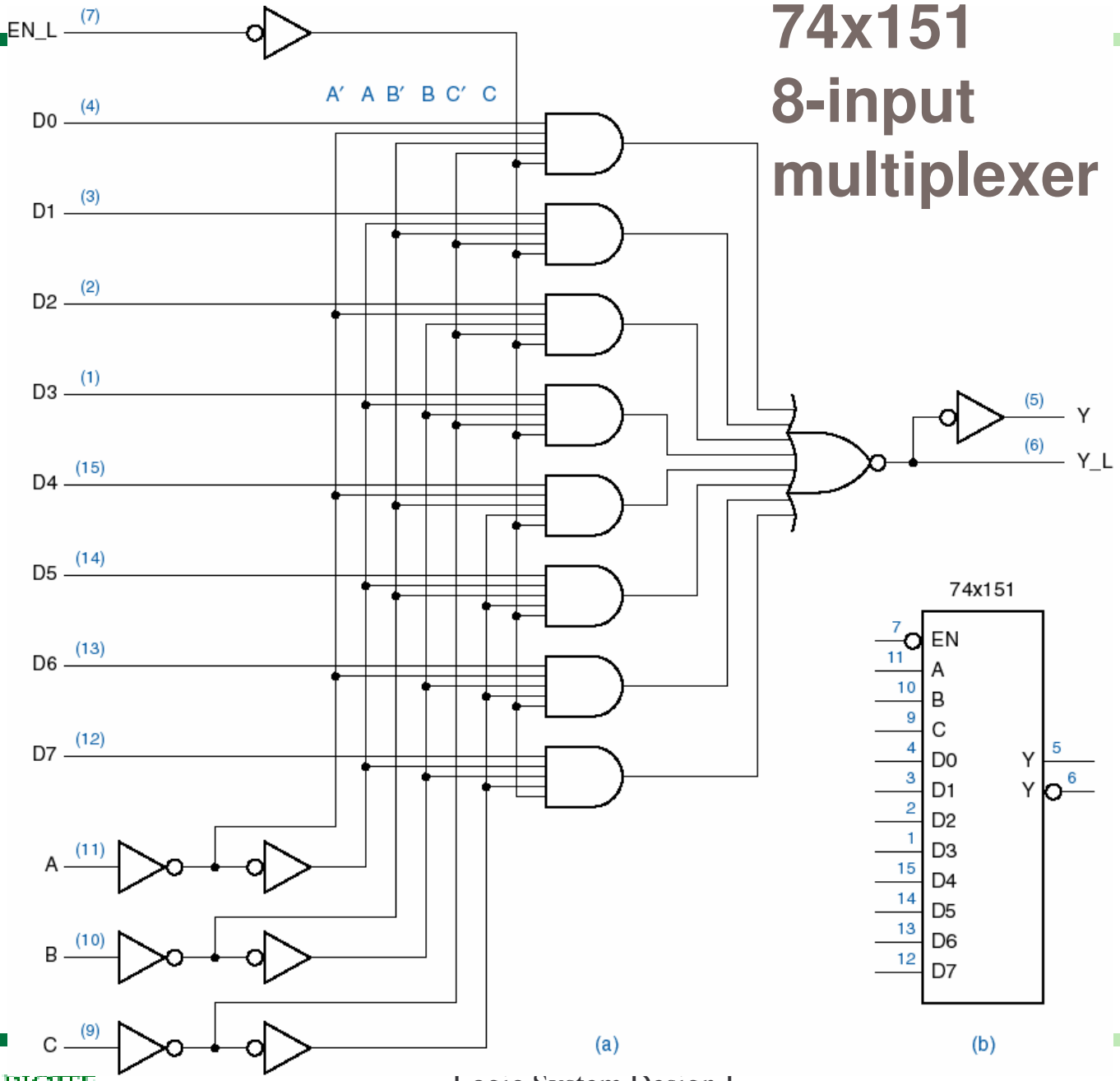
Circuit descriptions

Multiplexers





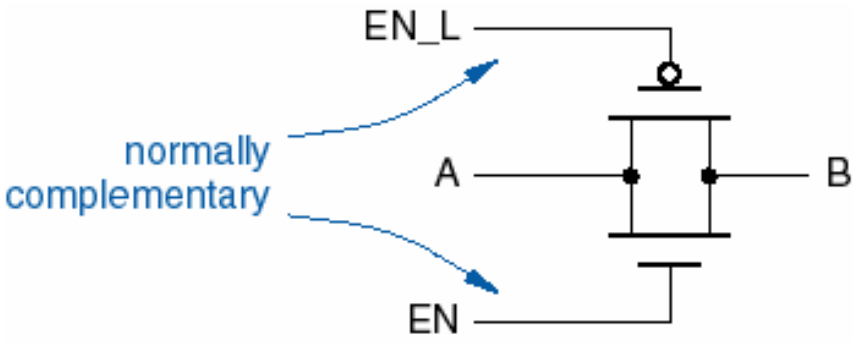
74x151 8-input multiplexer

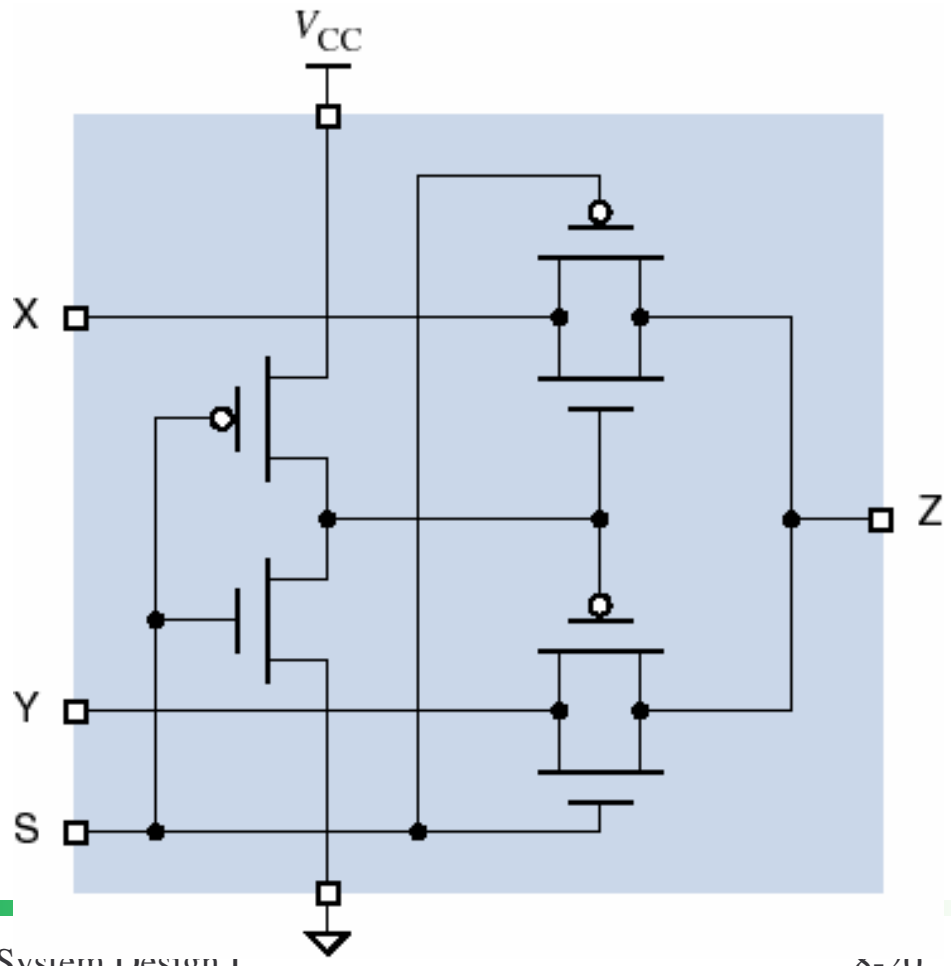


74x151 truth table

Inputs				Outputs	
EN_L	C	B	A	Y	Y_L
1	x	x	x	0	1
0	0	0	0	D0	D0′
0	0	0	1	D1	D1′
0	0	1	0	D2	D2′
0	0	1	1	D3	D3′
0	1	0	0	D4	D4′
0	1	0	1	D5	D5′
0	1	1	0	D6	D6′
0	1	1	1	D7	D7′

CMOS transmission gates





Other multiplexer varieties

2-input, 4-bit-wide
- 74x157

Inputs		Outputs			
G_L	S	1Y	2Y	3Y	4Y
1	x	0	0	0	0
0	0	1A	2A	3A	4A
0	1	1B	2B	3B	4B

4-input, 2-bit-wide
- 74x153

Barrel shifter design example

n data inputs, *n* data outputs

Control inputs specify number of positions to rotate or shift data inputs

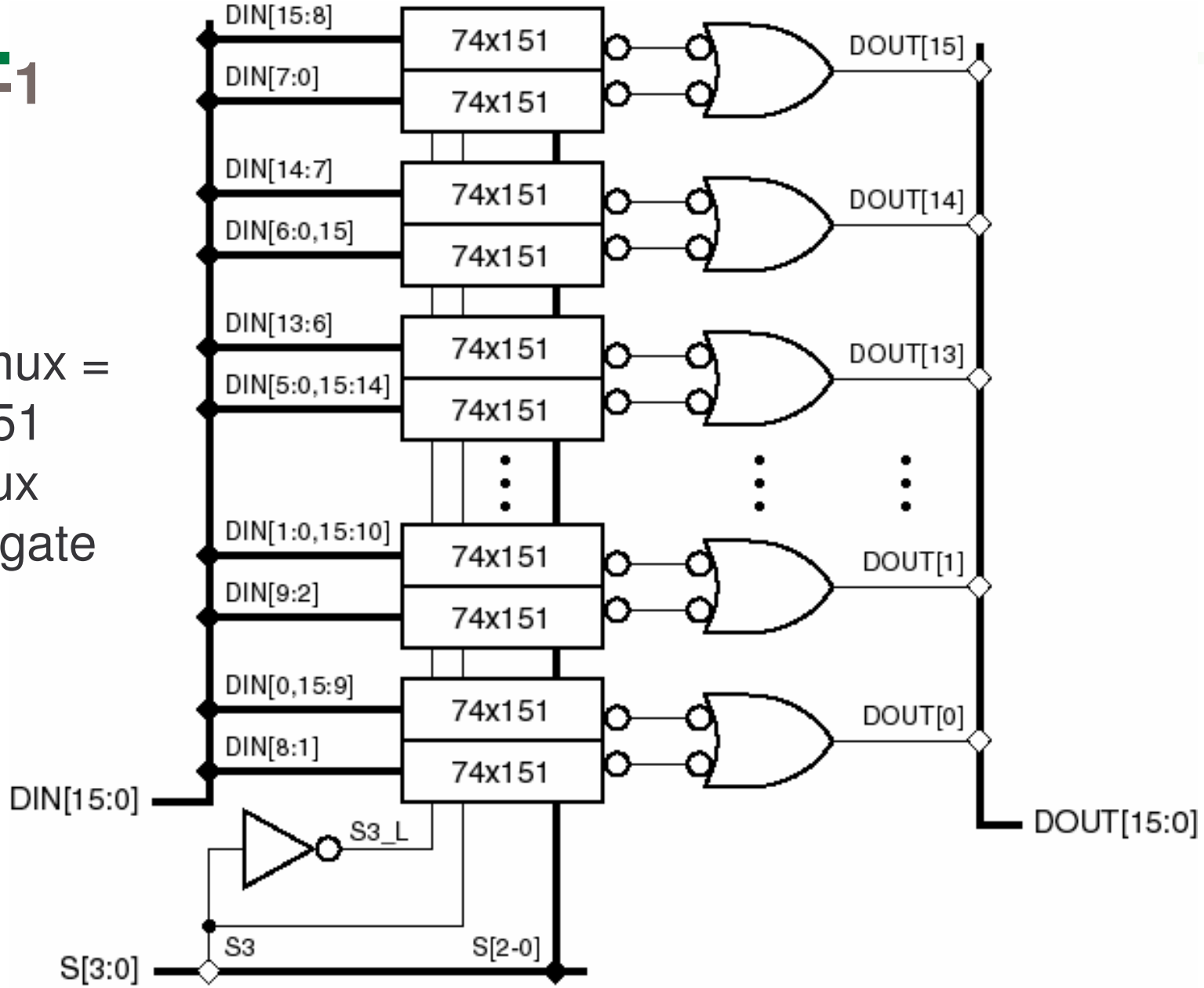
Example: n = 16

- DIN[15:0], DOUT[15:0], S[3:0] (shift amount)

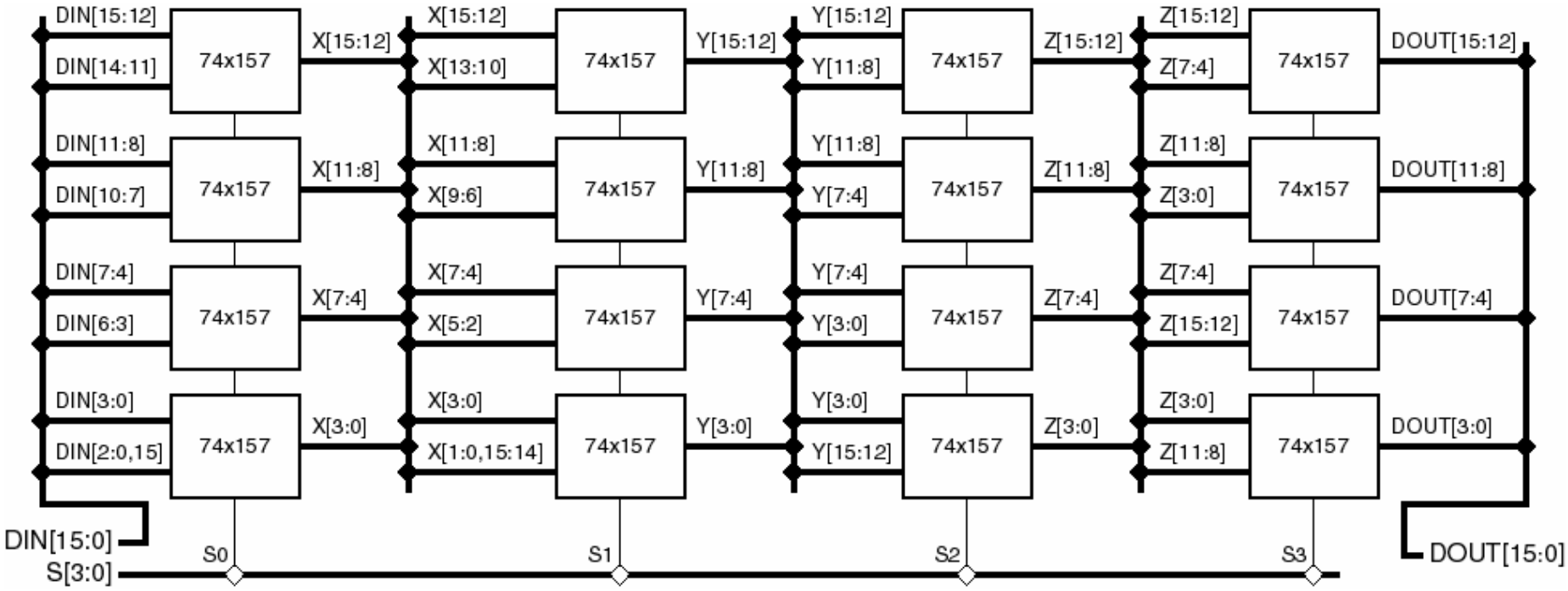
Many possible solutions, all based on multiplexers

16 16-to-1 muxes

16-to-1 mux =
2 x 74x151
8-to-1 mux
+ NAND gate



4 16-bit 2-to-1 muxes



16-bit 2-to-1 mux = 4 x 74x157 4-bit 2-to-1 mux

Properties of different approaches

Multiplexer Component	Data Loading	Data Delay	Control Loading	Total ICs
74x151	16	2	32	36
74x251	16	1	32	32
74x153	4	2	8	16
74x157	2	4	4	16

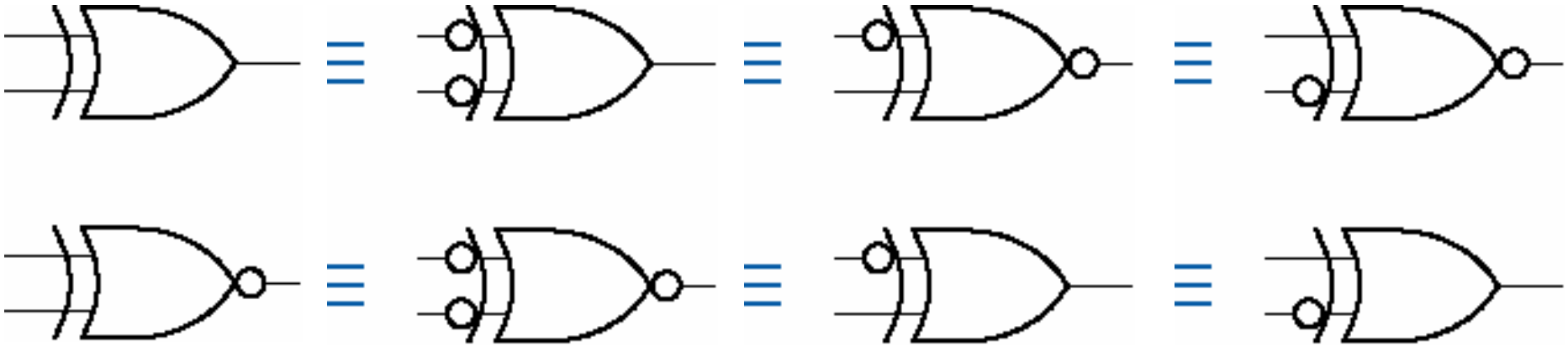
2-input XOR gates

Like an OR gate, but ***excludes*** the case where both inputs are 1.

X	*Y*	*X* ⊕ *Y* (XOR)	(*X* ⊕ *Y*)′ (XNOR)
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

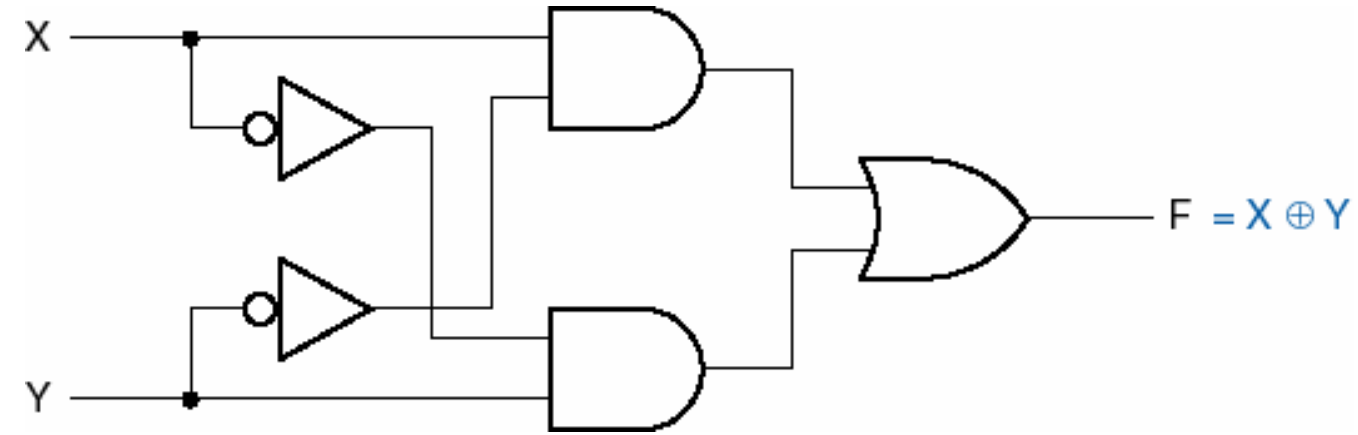
XNOR: complement of XOR

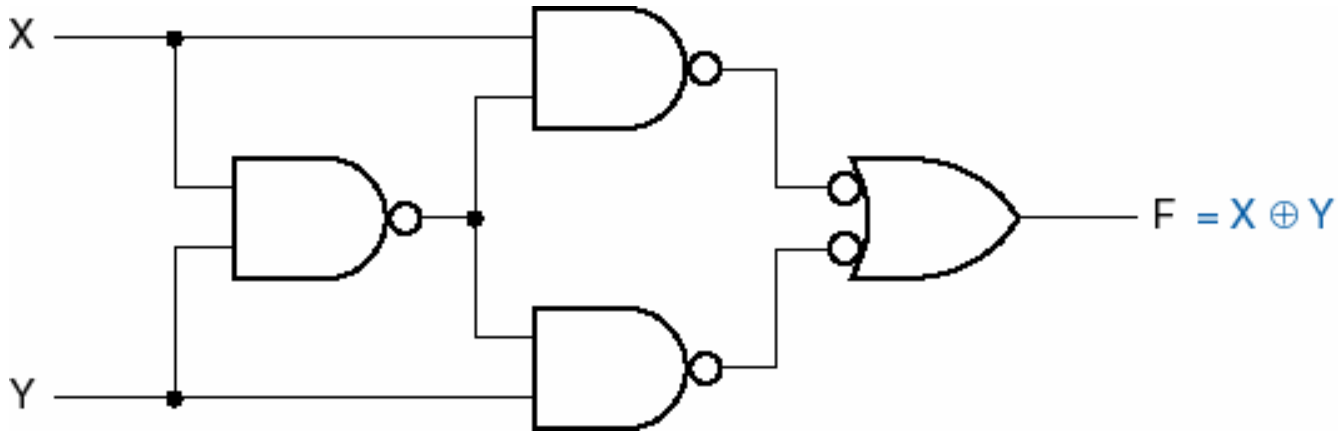
XOR and XNOR symbols



Gate-level XOR circuits

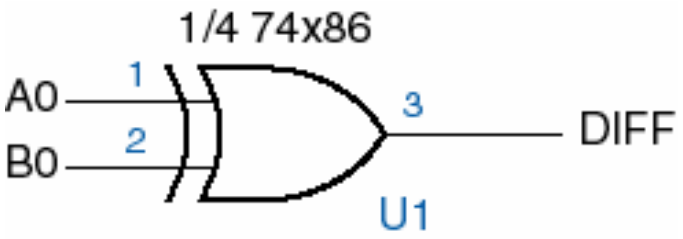
No direct realization with just a few transistors.



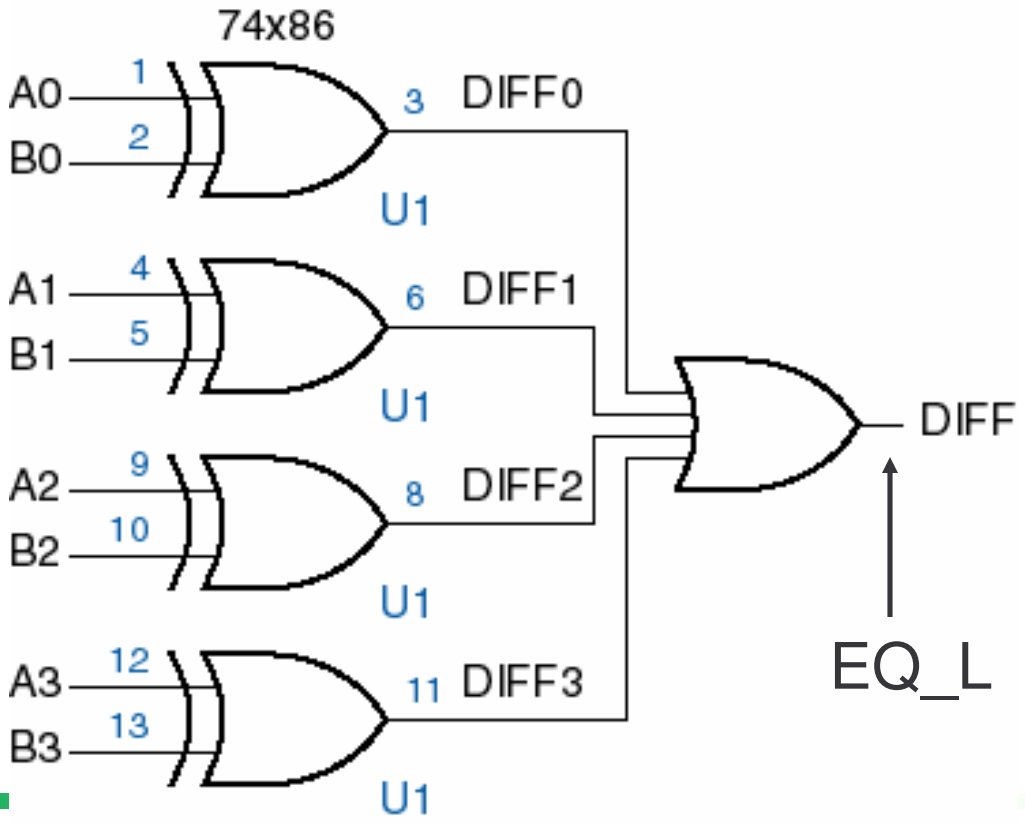


Equality Comparators

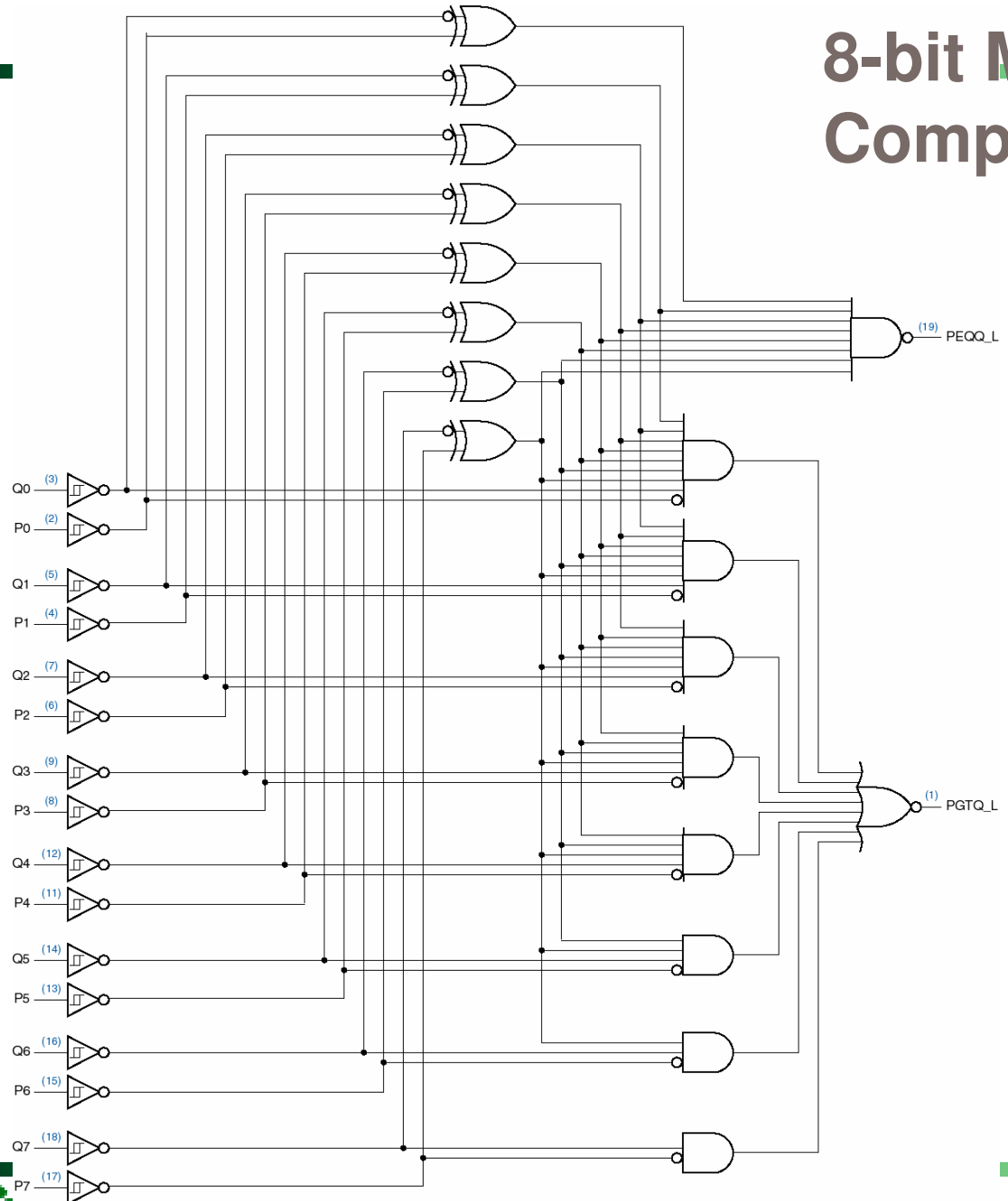
1-bit comparator

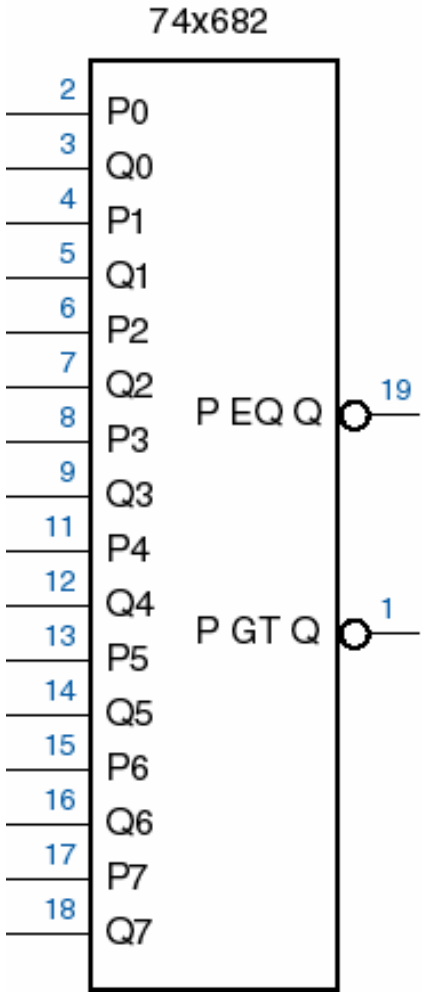


4-bit comparator

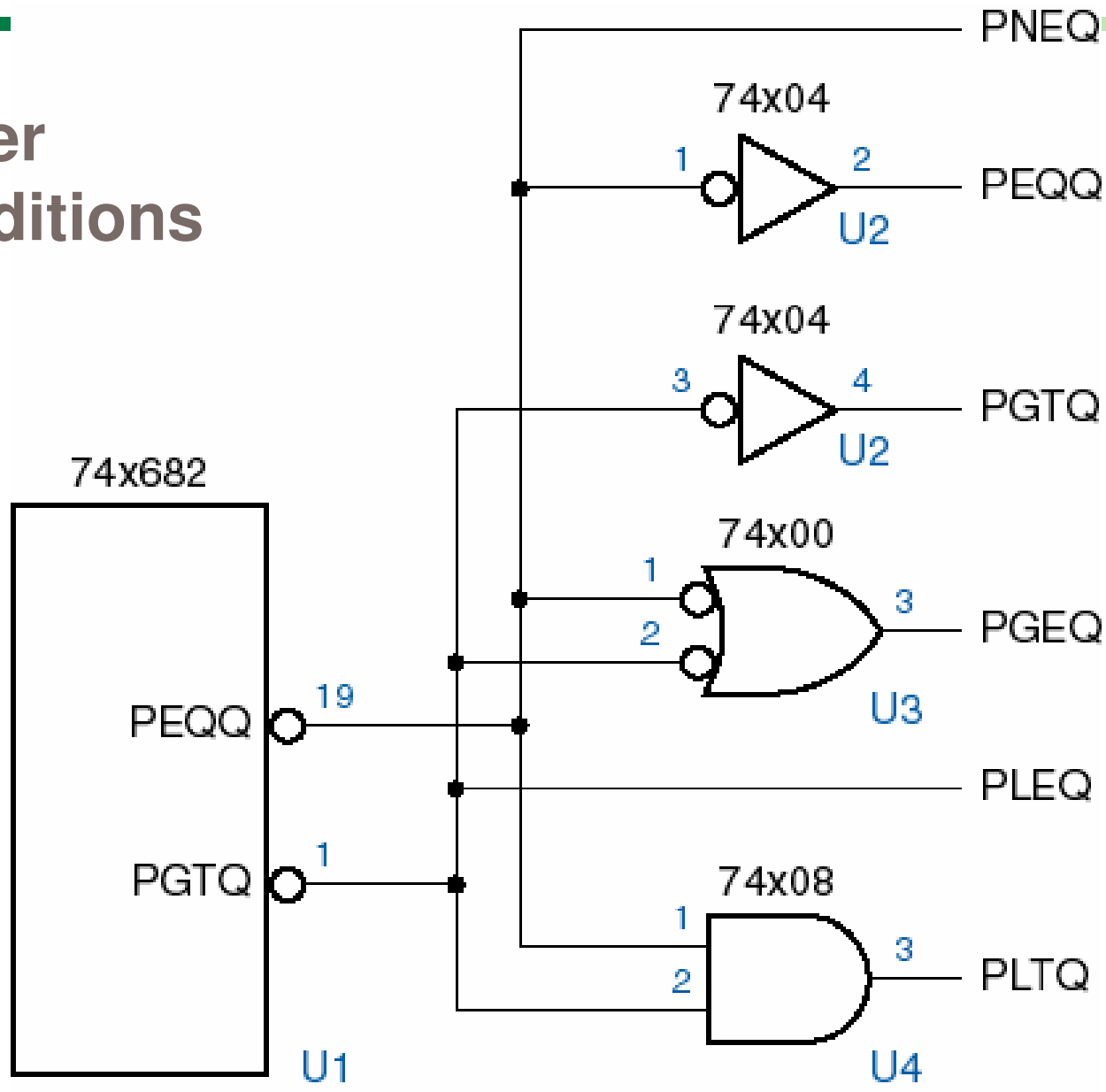


8-bit Magnitude Comparator





Other conditions



Adders

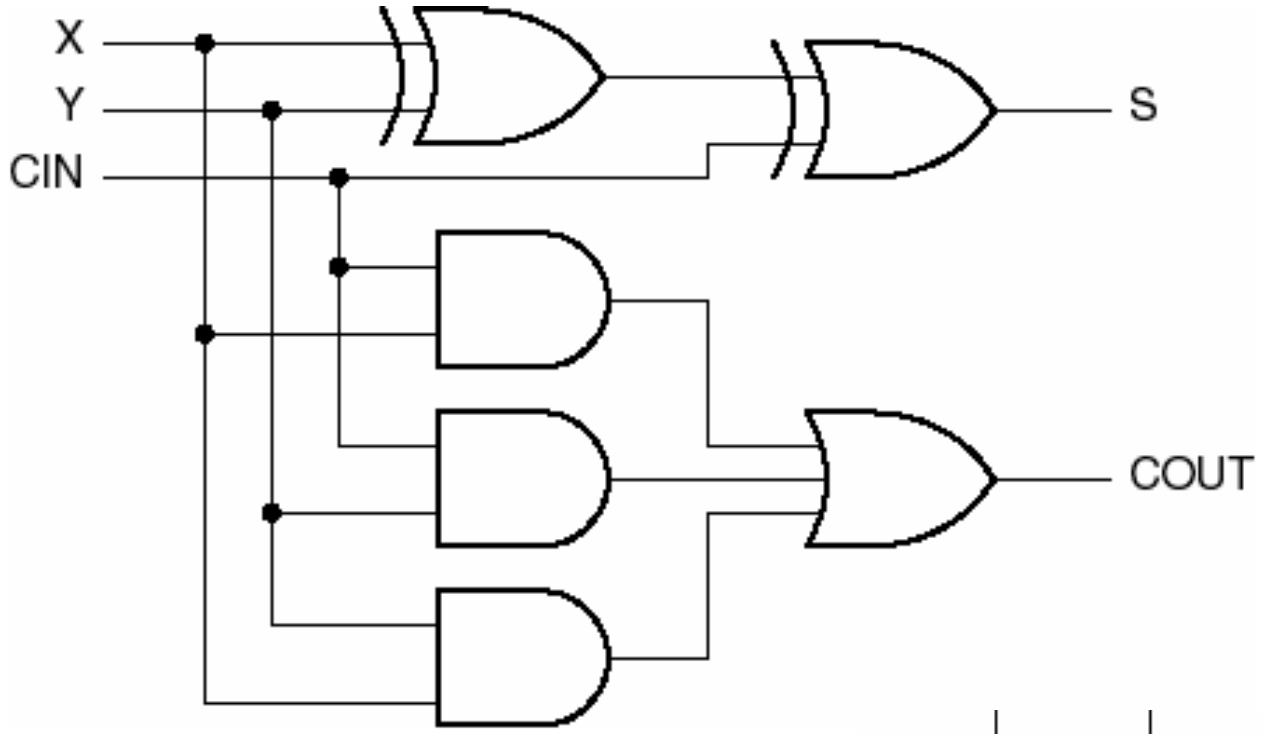
Basic building block is “full adder”

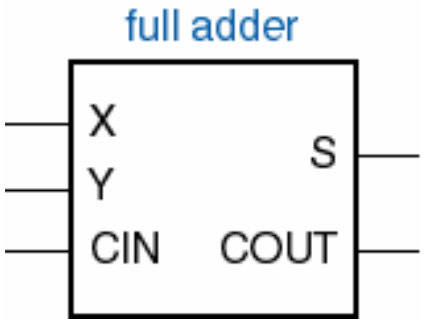
- 1-bit-wide adder, produces sum and carry outputs

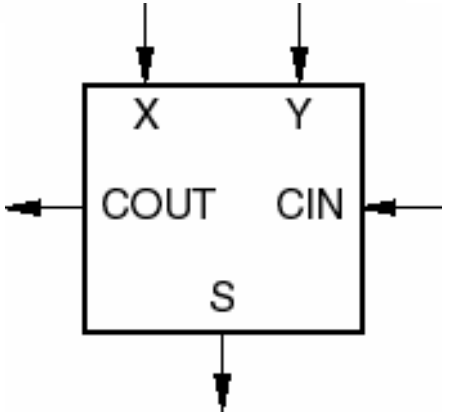
Truth table:

X	Y	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

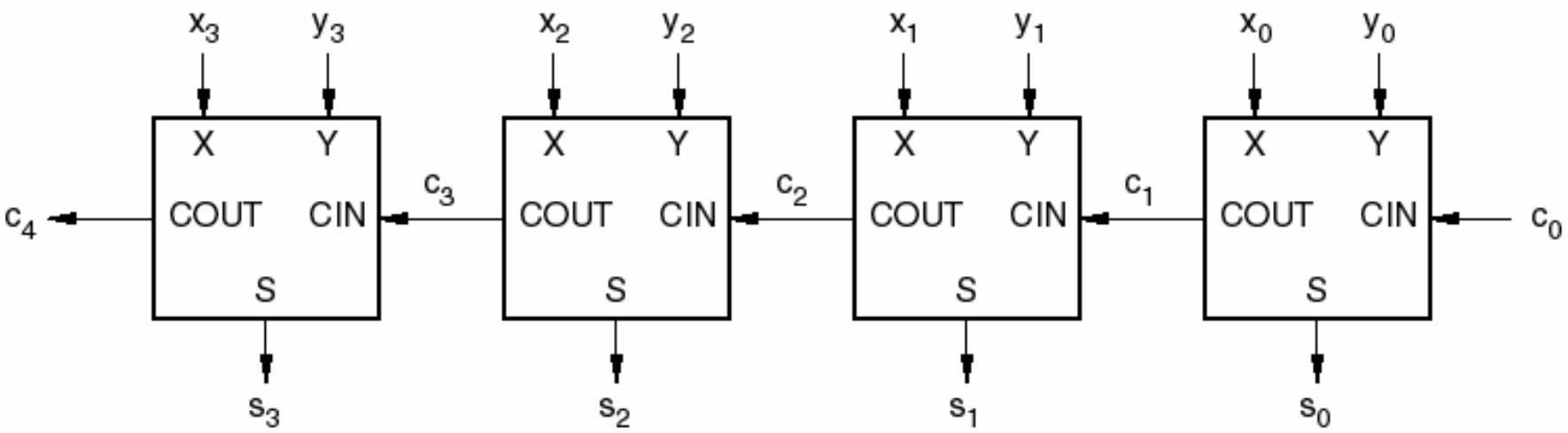
Full-adder circuit







Ripple adder

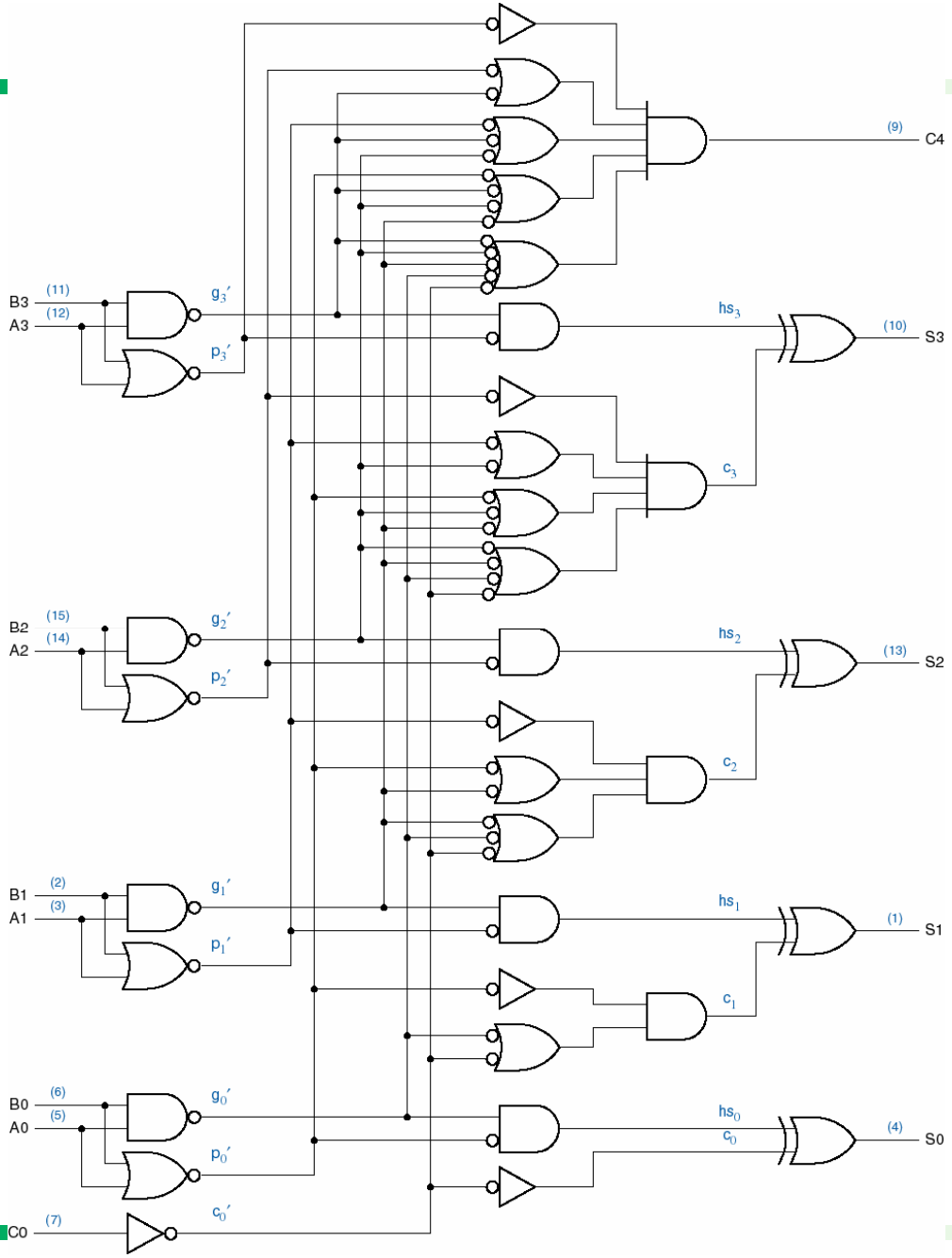


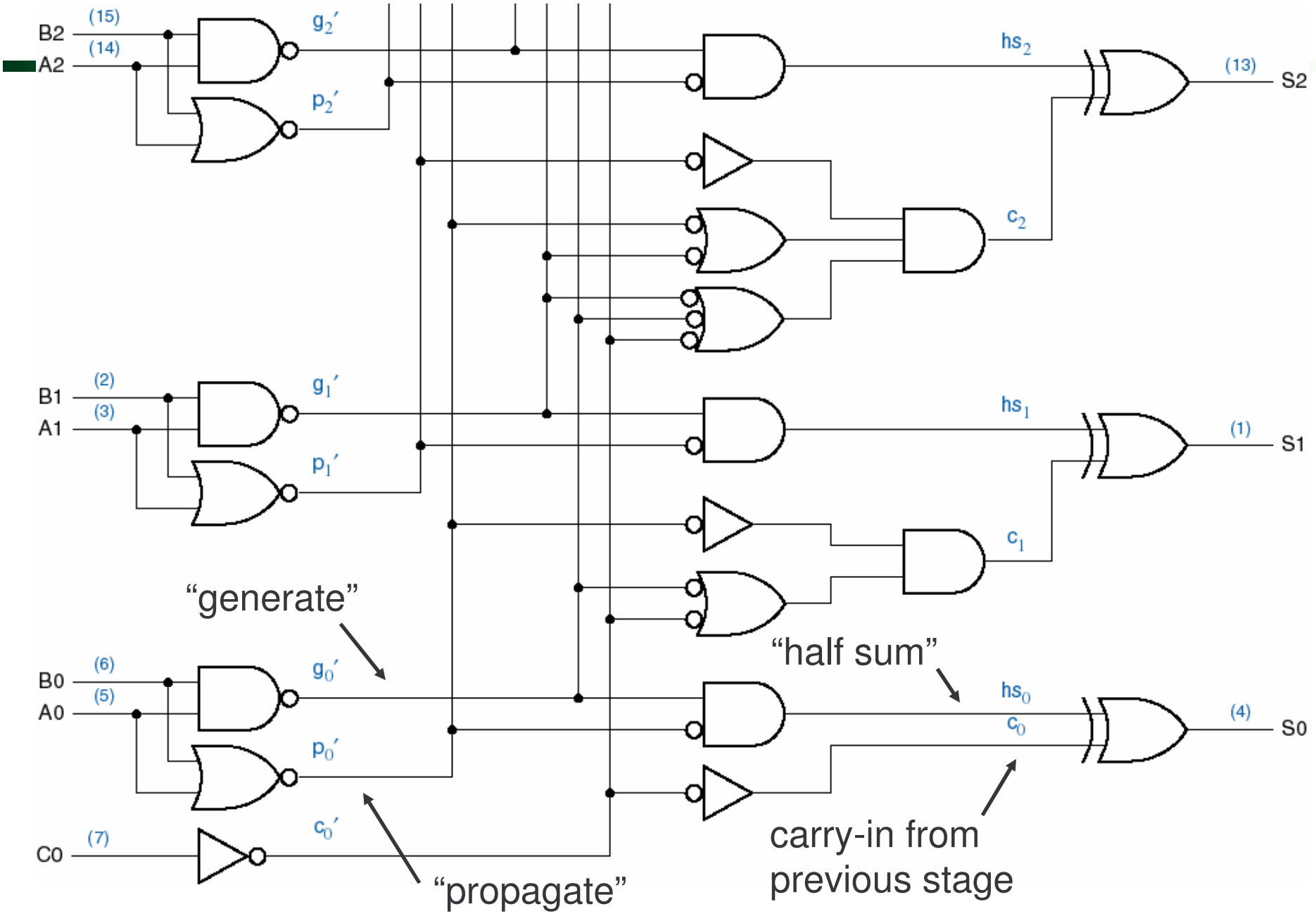
Faster adders eliminate or limit carry chain

- 2-level AND-OR logic ==> 2^n product terms
- 3 or 4 levels of logic, carry lookahead

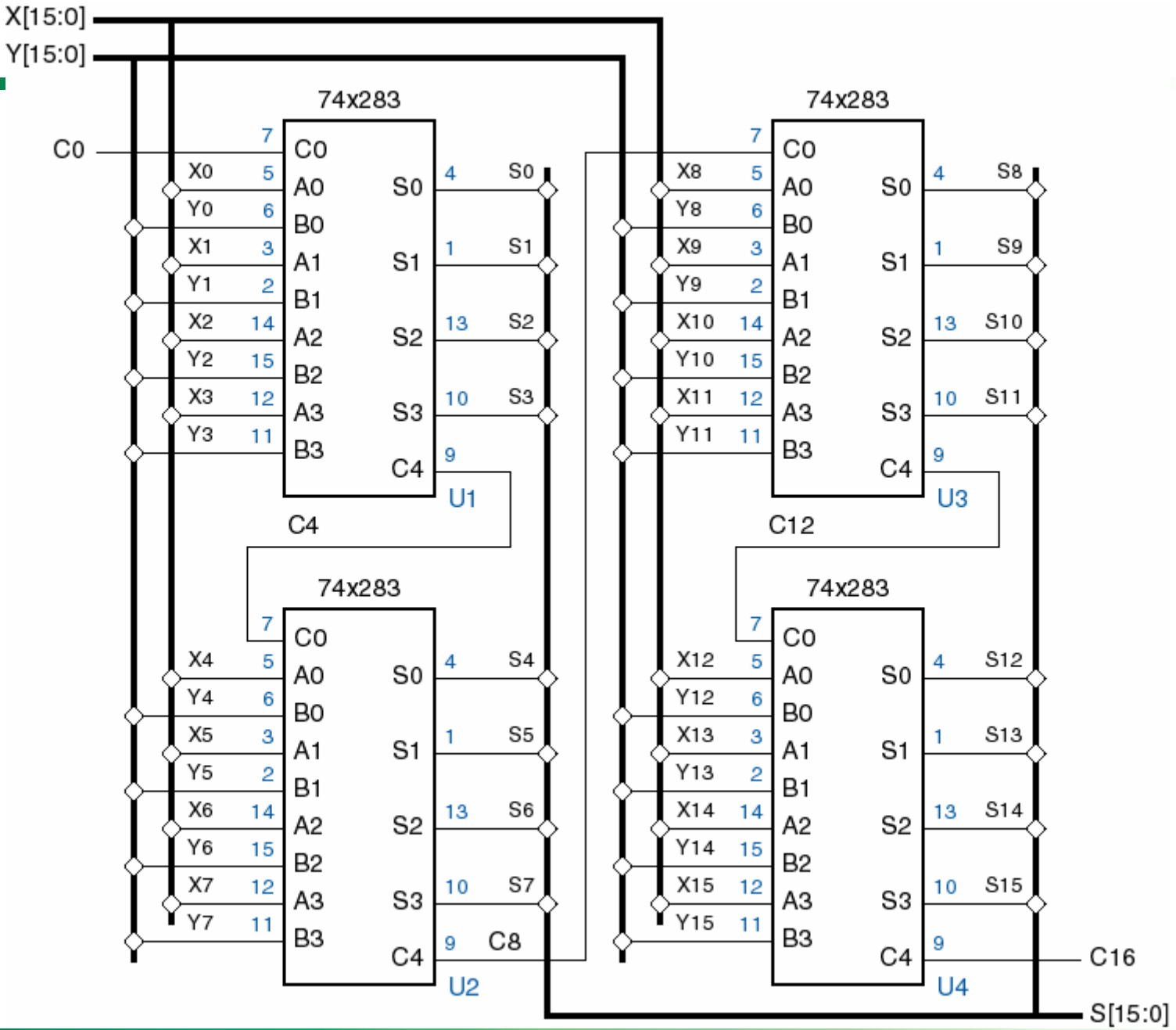
74x283
4-bit adder

Uses carry lookahead internally

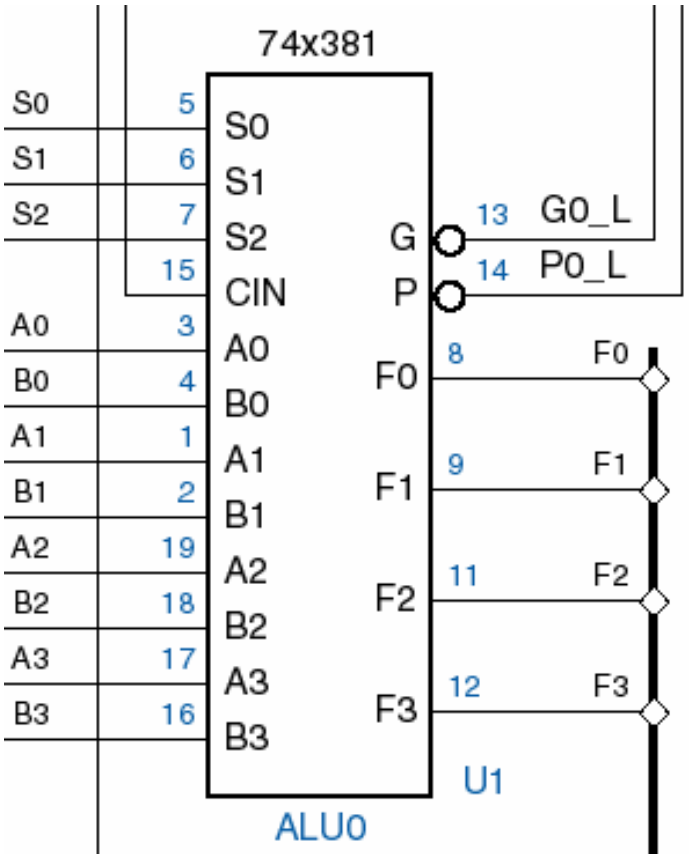


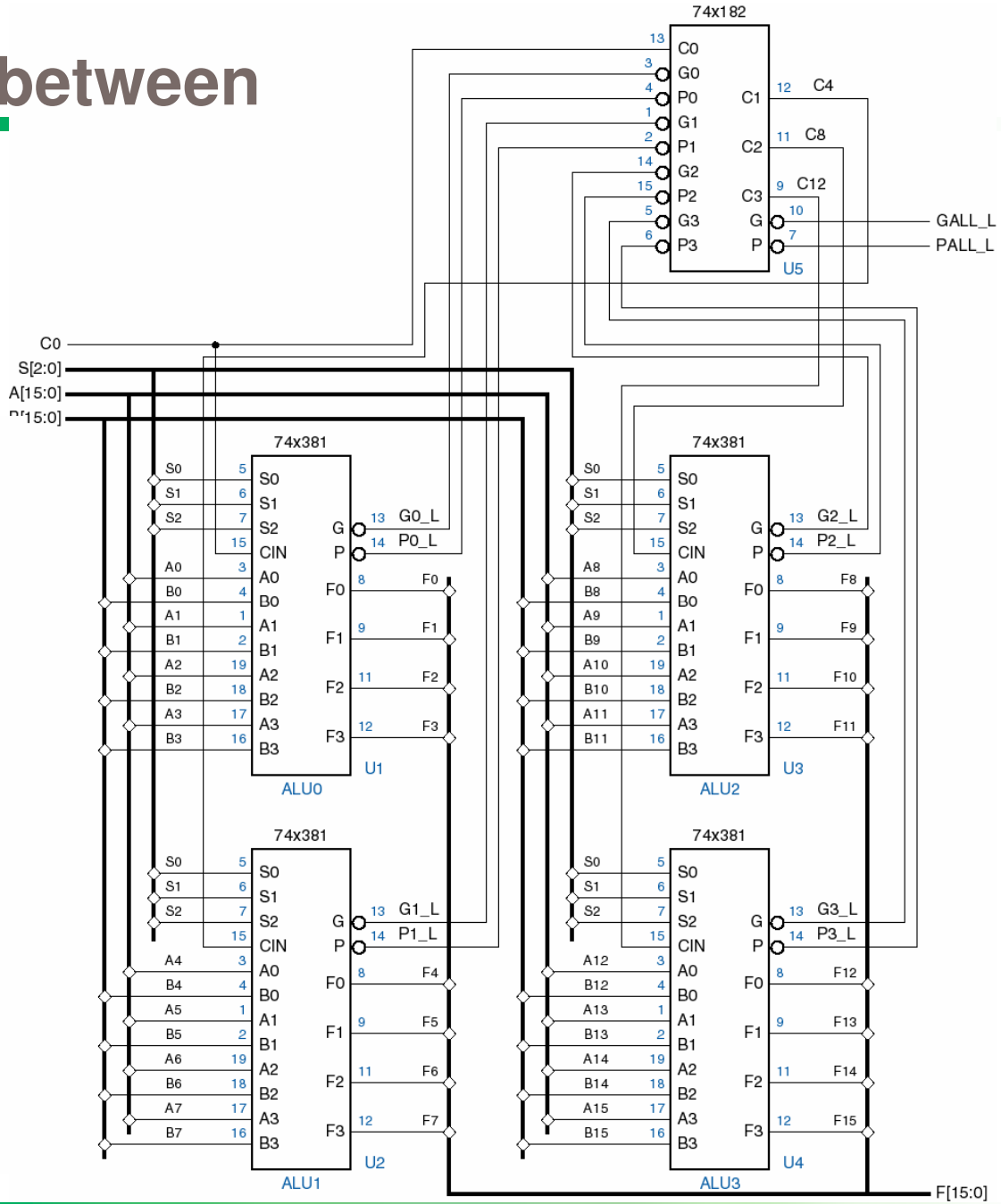

B2
(15)
A2
(14)
g_2'
p_2'
hs_2
(13)
S2
c_2
B1
(2)
A1
(3)
g_1'
p_1'
hs_1
(1)
S1
c_1
"generate"
"half sum"
B0
(6)
A0
(5)
g_0'
p_0'
hs_0
c_0
(4)
S0
C0
(7)
c_0'
"propagate"
carry-in from
previous stage

Ripple carry between groups



Lookahead carry between groups





Subtraction

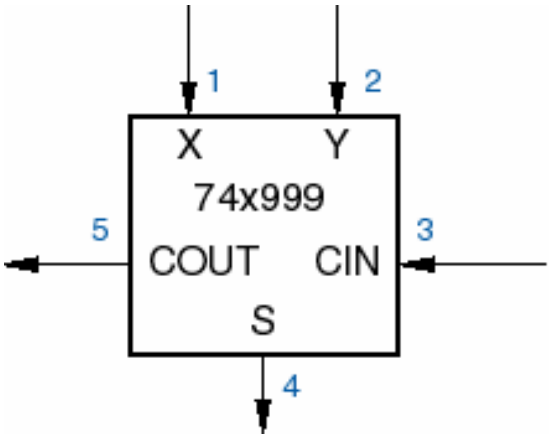
Subtraction is the same as addition of the two's complement.

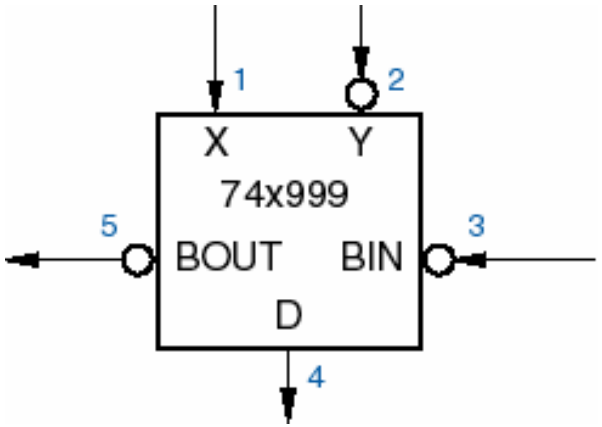
The two's complement is the bit-by-bit complement plus 1.

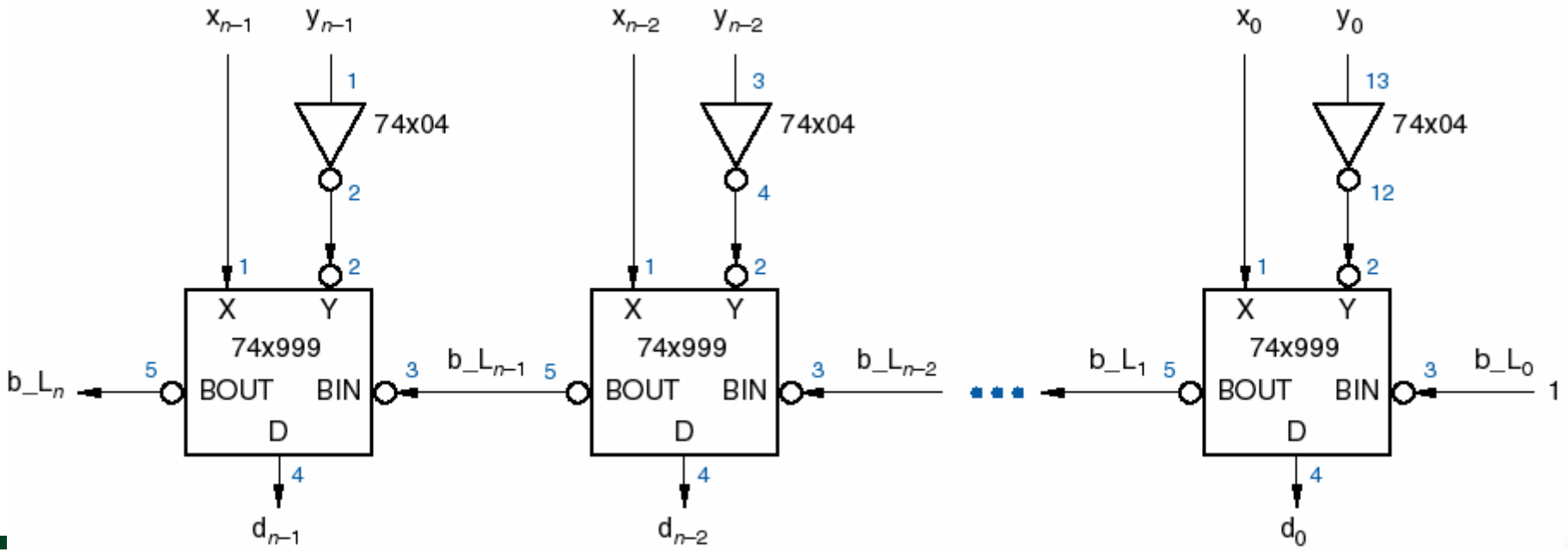
Therefore, $X - Y = X + \overline{Y} + 1$.

- Complement Y inputs to adder, set C_{in} to 1.
- For a borrow, set C_{in} to 0.

Full subtractor = full adder, almost

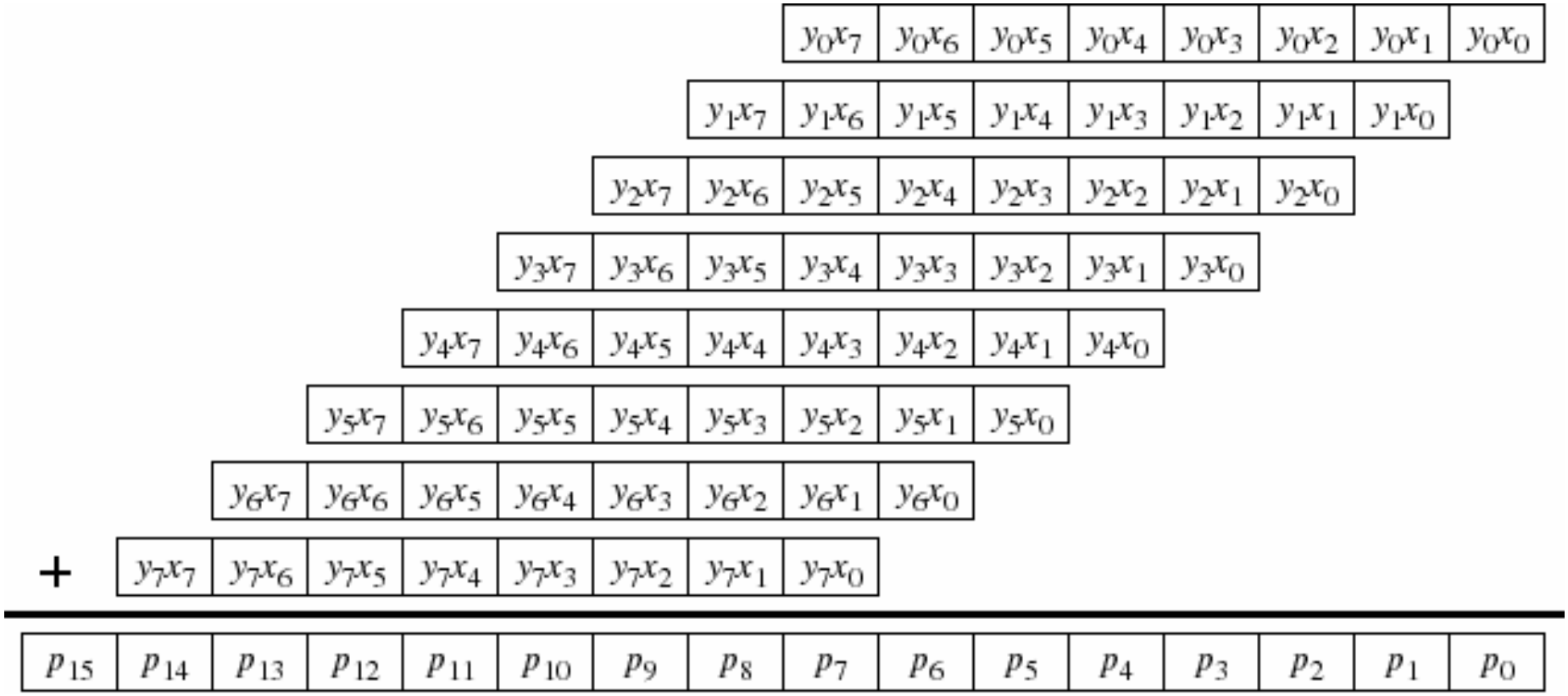




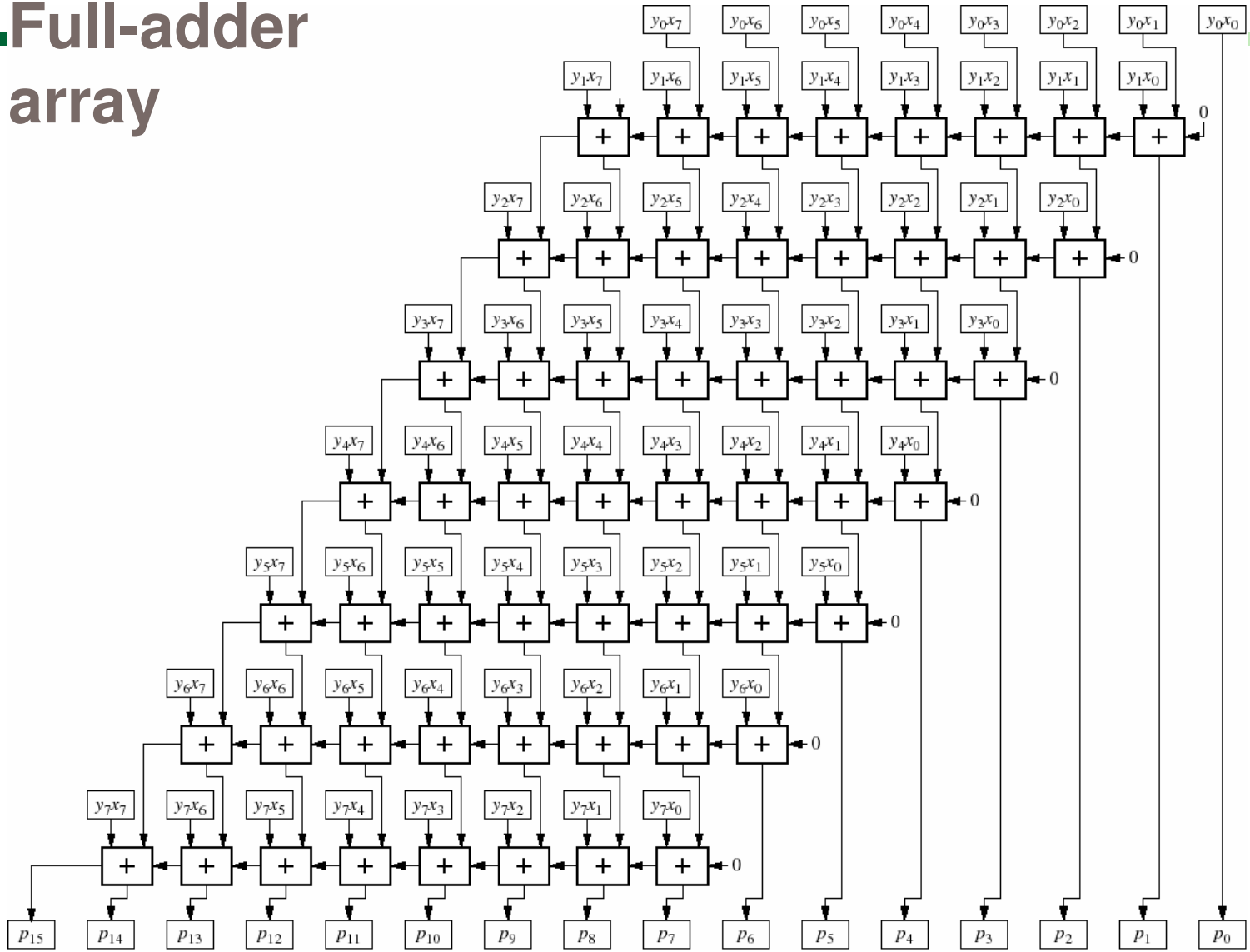


Multipliers

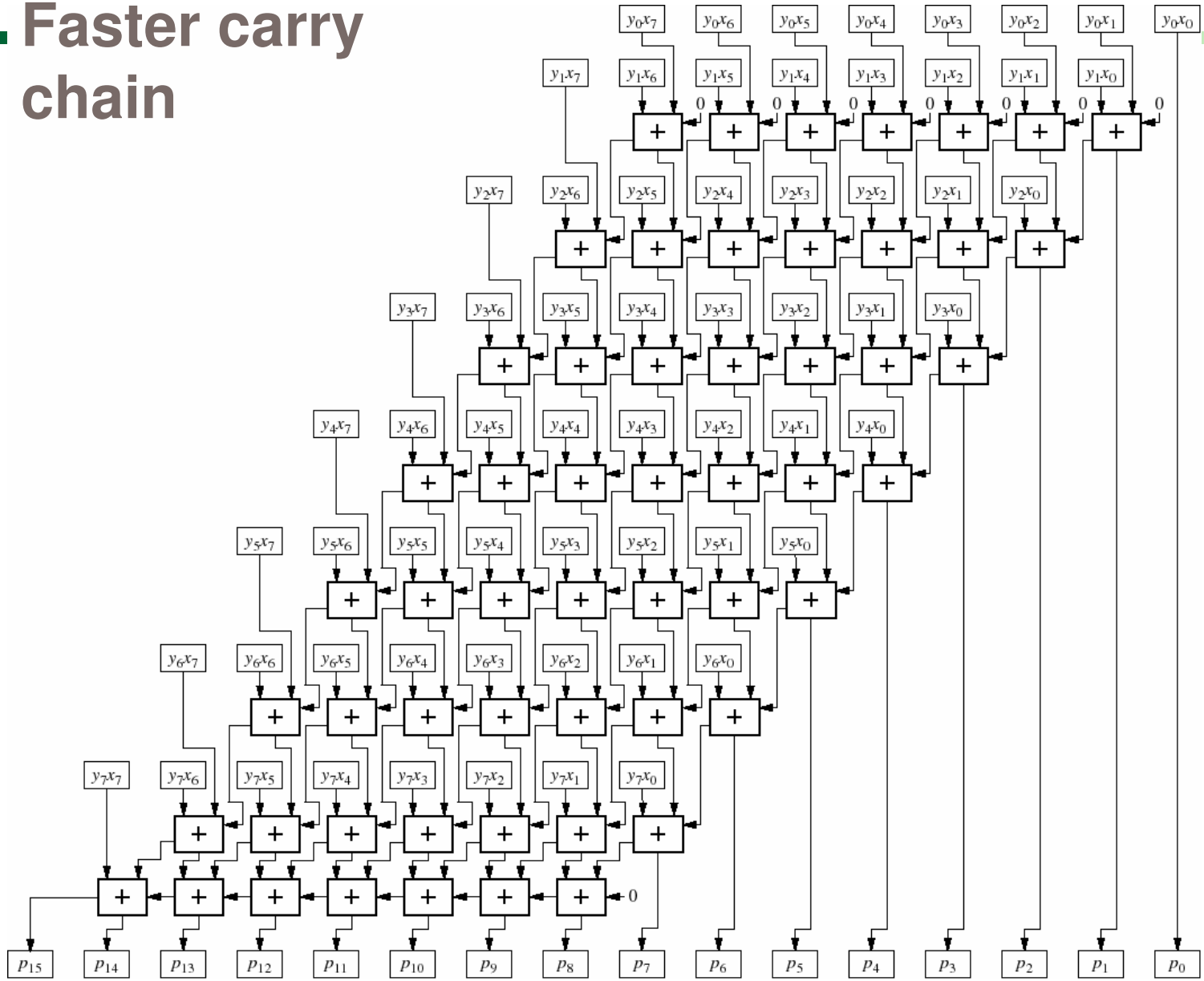
8x8 multiplier



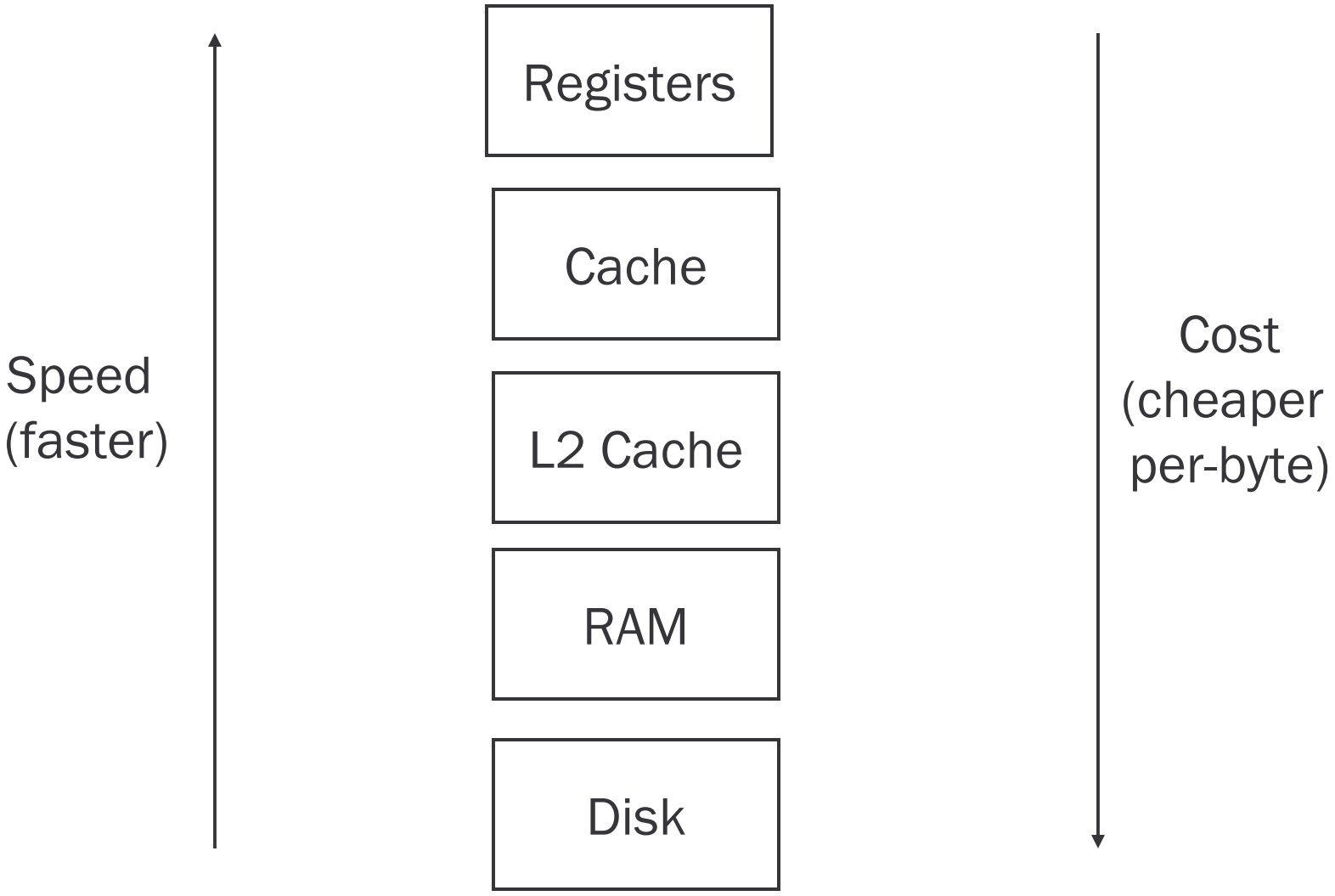
Full-adder array



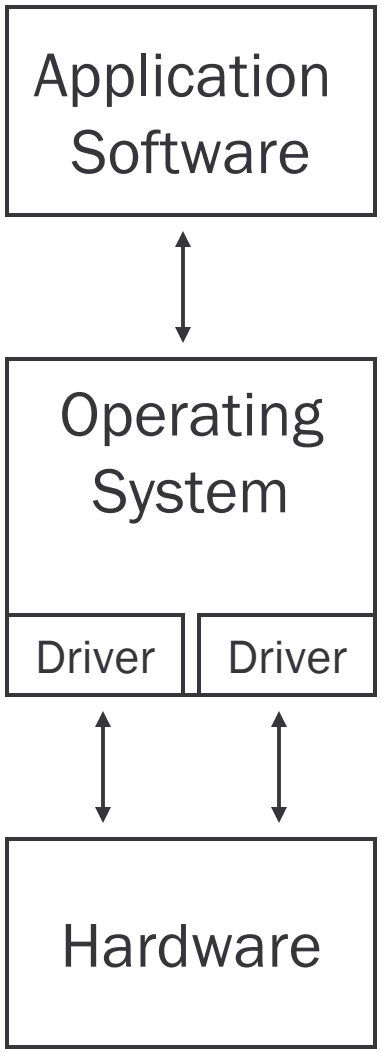
Faster carry chain



Memory Hierarchy



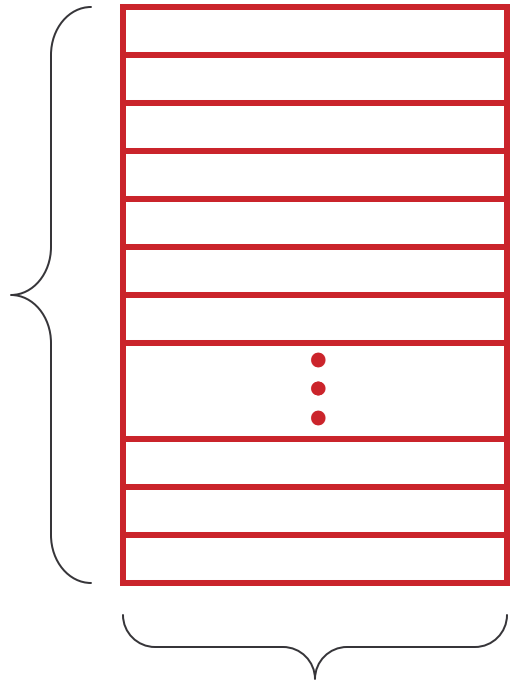
View of Computer System



Memory

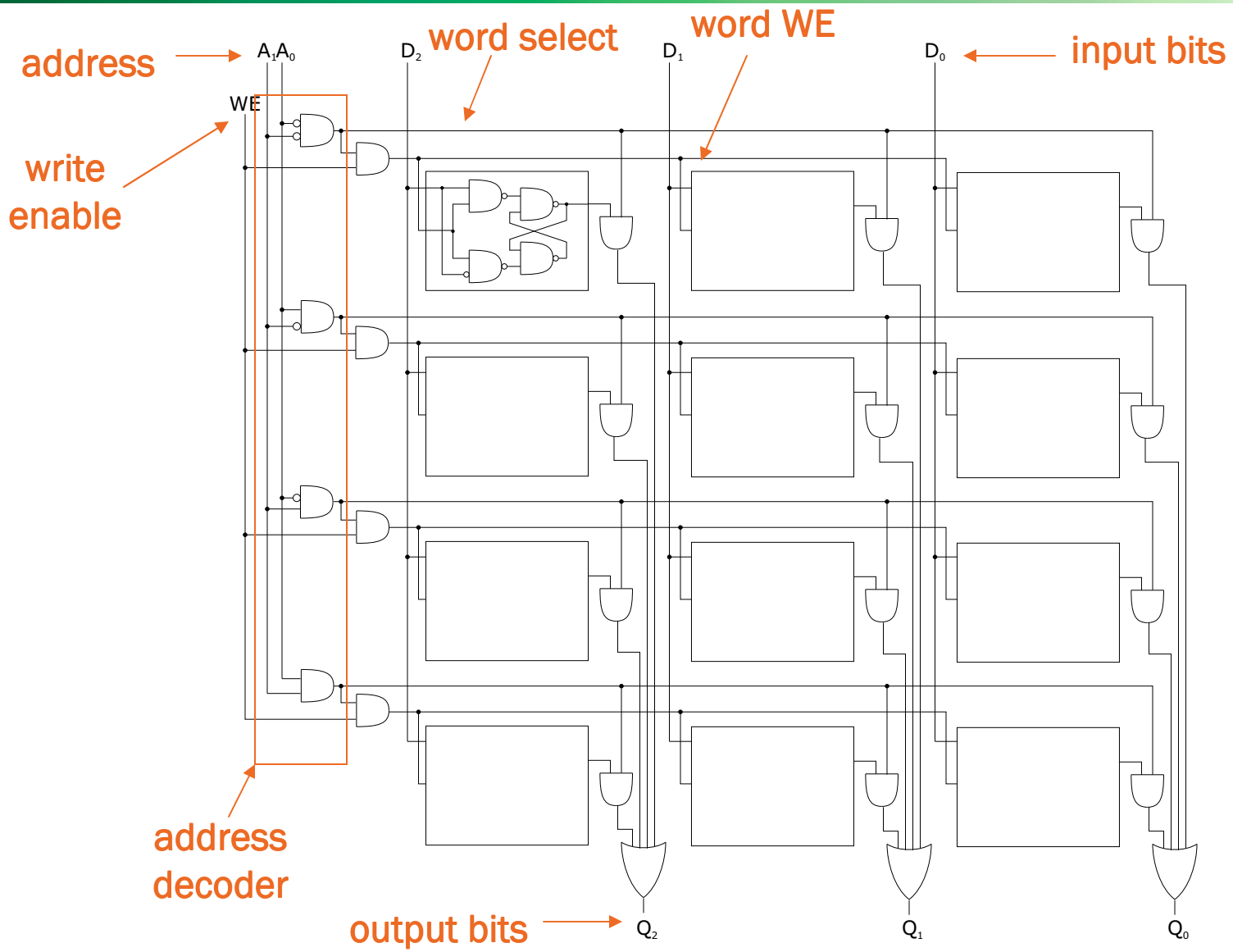
To build a memory -- a logical $k \times m$ array of stored bits.

Address Space:
number of locations
(usually a power of 2)



Addressability:
number of bits per location
(e.g., byte-addressable)

2^2 x 3 Memory


address
A1A0
D2
word select
D1
word WE
D0
input bits
WE
write
enable
address
decoder
output bits
Q2
Q1
Q0

More Memory Details

This is a not the way actual memory is implemented.

- fewer transistors, much more dense, relies on electrical properties

But the logical structure is very similar.

- address decoder
- word select line
- word write enable

Two basic kinds of memory (RAM = Random Access Memory)

Static RAM (SRAM)

- fast, maintains data without power refresh

Dynamic RAM (DRAM)

- slower but denser, bit storage must be periodically refreshed

Even More Memory Details

There are other types of “non-volatile” memory devices:

- ROM
- PROM
- EPROM
- EEPROM
- Flash

Can you think of other memory devices?

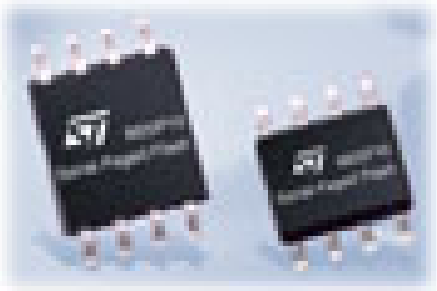
Electronics Packaging

- There are several packaging technologies available that an engineer can use to create electronic devices.
- Some are suitable for inexpensive toys but not miniature consumer products, and some are suitable for miniature consumer products but not inexpensive toys.
- These packages have metal leads that are the conductive wire that connect electricity from the outside world to the silicon inside the package.
- Leads between packages are connected with small copper traces on a printed circuit board (PCB), and the package leads are soldered to the PCB.

Examples of Electronics Packages

Dual In-line Package (DIP) Older technology, requires the metal leads to go through a hole in the printed circuit board.

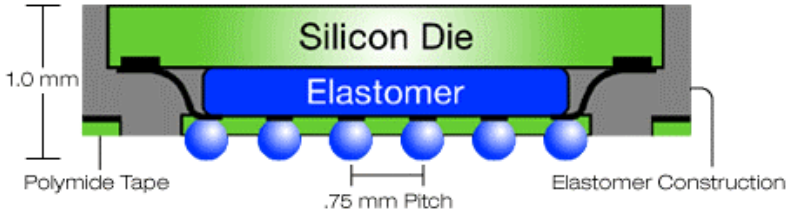
Dual Flat Pack (DFP) - A fairly recent technology, metal leads solder to the surface of the printed circuit board.



Examples of Electronics Packages

Quad Flat Pack (QFP) - like the Dual Flat Pack, except here are metal leads are on four sides.

Ball Grid Array (BGA) - The connections to the component are on the bottom of the chip, and have balls of solder on these connections.



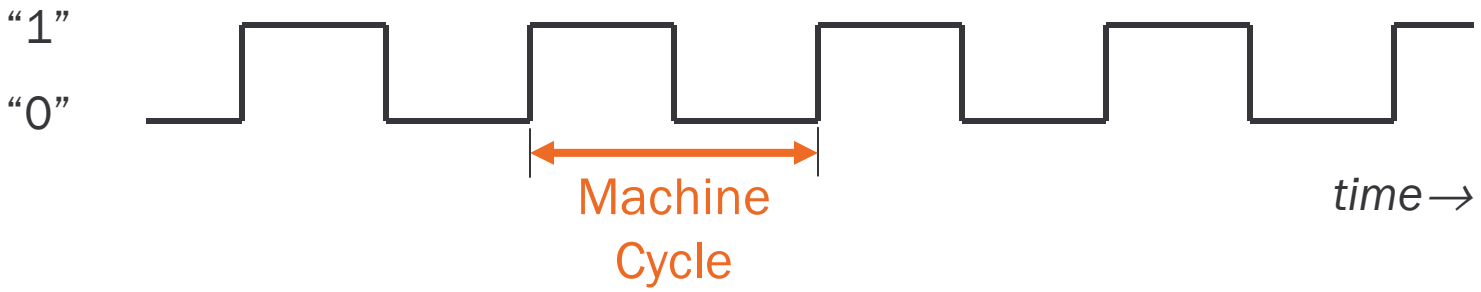
Driving Force: The Clock

The clock is a signal that keeps the control unit moving.

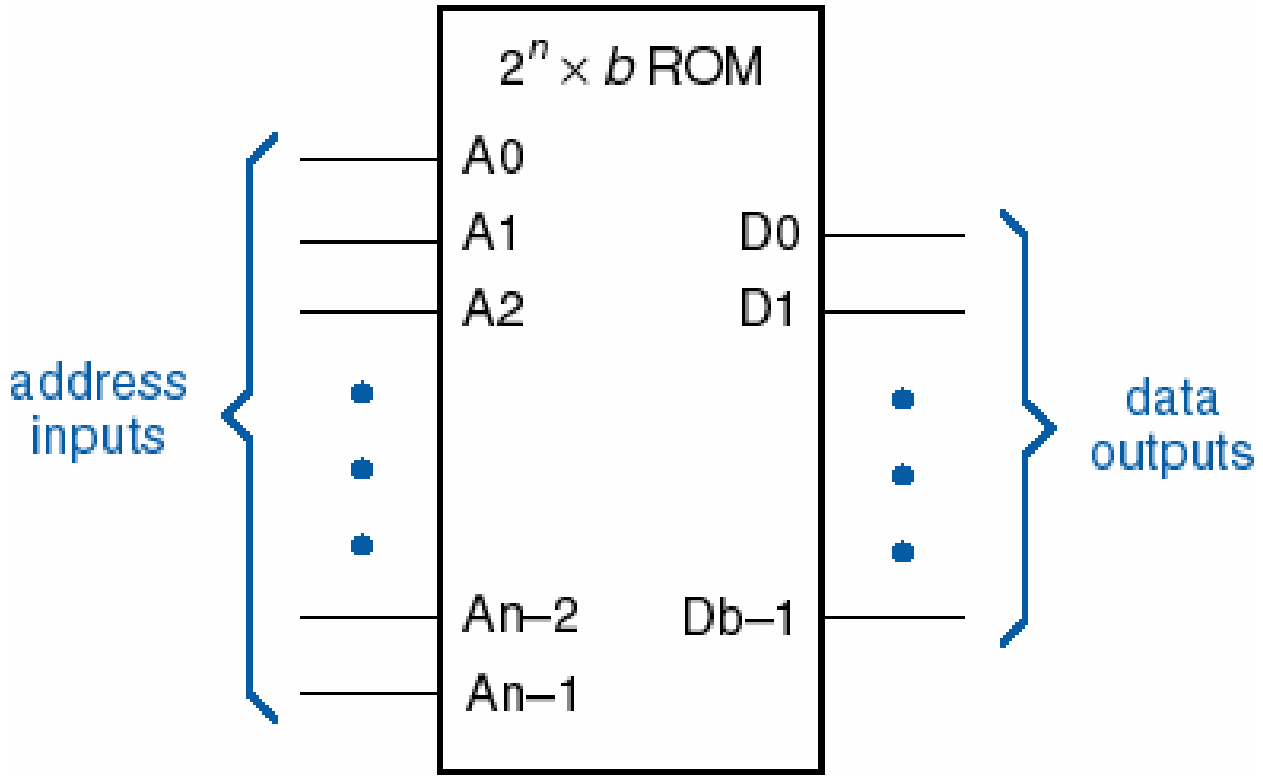
- At each clock "tick," control unit moves to the next machine cycle -- may be next instruction or next phase of current instruction.

Clock generator circuit:

- Based on crystal oscillator
- Generates regular sequence of "0" and "1" logic levels
- Clock cycle (or machine cycle) -- rising edge to rising edge



Read-Only Memories



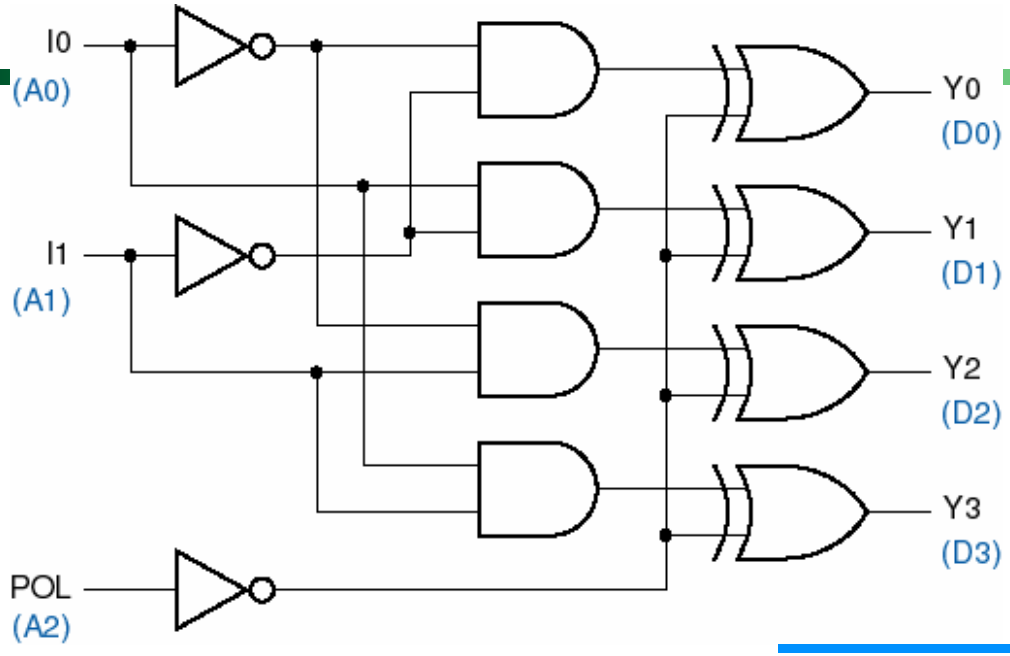
Why “ROM”?

Program storage

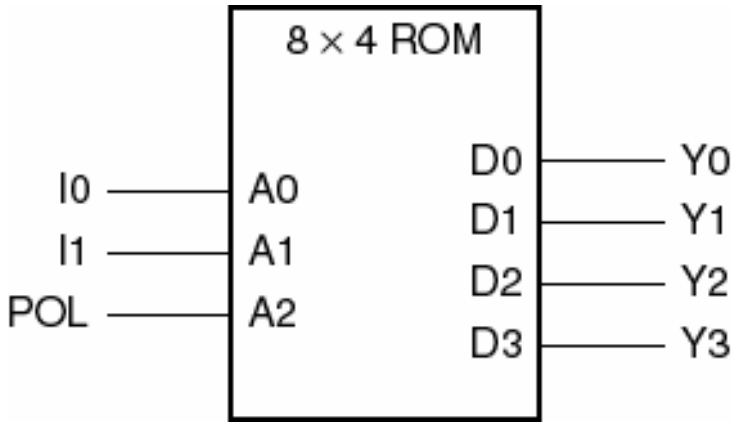
- Boot ROM for personal computers
- Complete application storage for embedded systems.

Actually, a ROM is a combinational circuit, basically a truth-table lookup.

- Can perform any combinational logic function
- Address inputs = function inputs
- Data outputs = function outputs

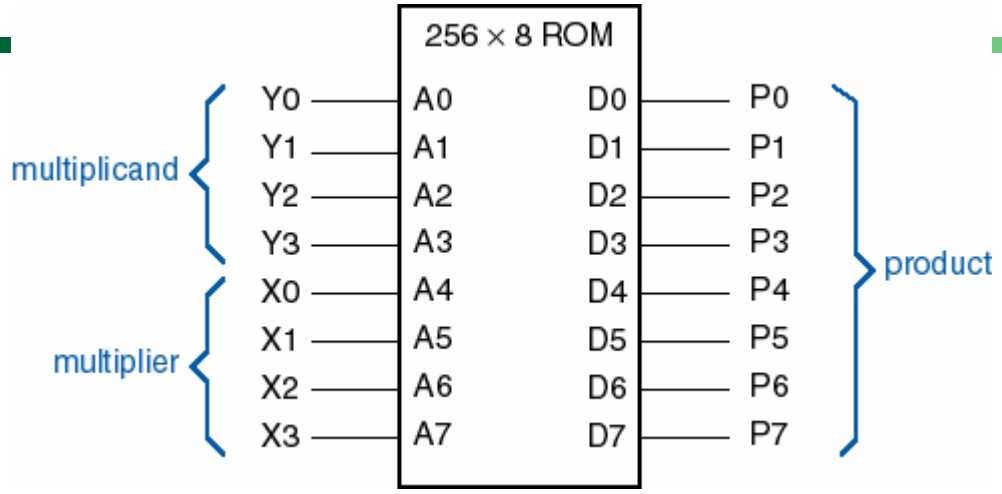


Logic-in-ROM example



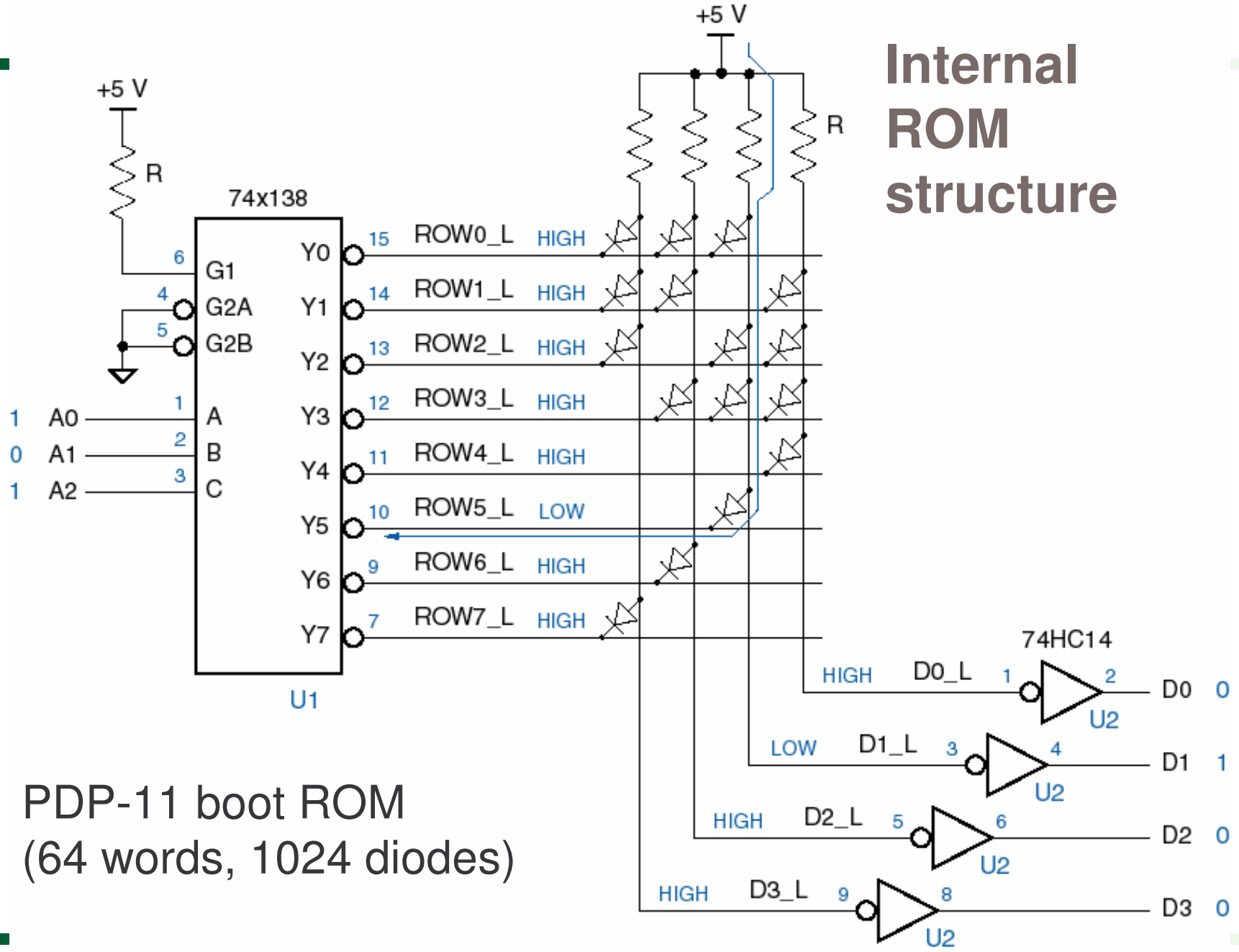
Inputs			*Outputs*			
A2	*A1*	*A0*	*D3*	*D2*	*D1*	*D0*
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

4x4 multiplier example



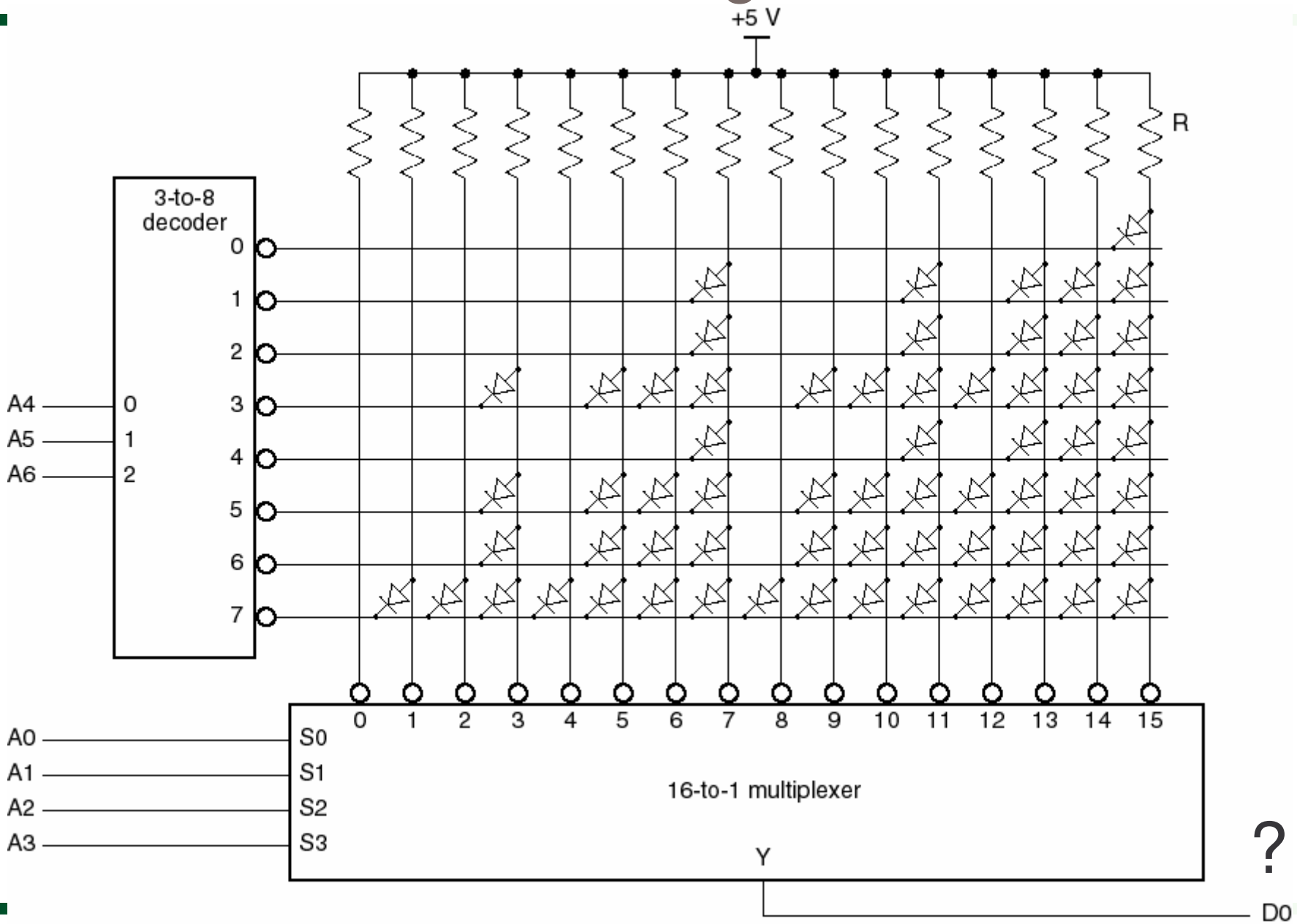

```
00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
20: 00 02 04 06 08 0A 0C 0E 10 12 14 16 18 1A 1C 1E
30: 00 03 06 09 0C 0F 12 15 18 1B 1E 21 24 27 2A 2D
40: 00 04 08 0C 10 14 18 1C 20 24 28 2C 30 34 38 3C
50: 00 05 0A 0F 14 19 1E 23 28 2D 32 37 3C 41 46 4B
60: 00 06 0C 12 18 1E 24 2A 30 36 3C 42 48 4E 54 5A
70: 00 07 0E 15 1C 23 2A 31 38 3F 46 4D 54 5B 62 69
80: 00 08 10 18 20 28 30 38 40 48 50 58 60 68 70 78
90: 00 09 12 1B 24 2D 36 3F 48 51 5A 63 6C 75 7E 87
A0: 00 0A 14 1E 28 32 3C 46 50 5A 64 6E 78 82 8C 96
B0: 00 0B 16 21 2C 37 42 4D 58 63 6E 79 84 8F 9A A5
C0: 00 0C 18 24 30 3C 48 54 60 6C 78 84 90 9C A8 B4
D0: 00 0D 1A 27 34 41 4E 5B 68 75 82 8F 9C A9 B6 C3
E0: 00 0E 1C 2A 38 46 54 62 70 7E 8C 9A A8 B6 C4 D2
F0: 00 0F 1E 2D 3C 4B 5A 69 78 87 96 A5 B4 C3 D2 E1
```

Internal ROM structure

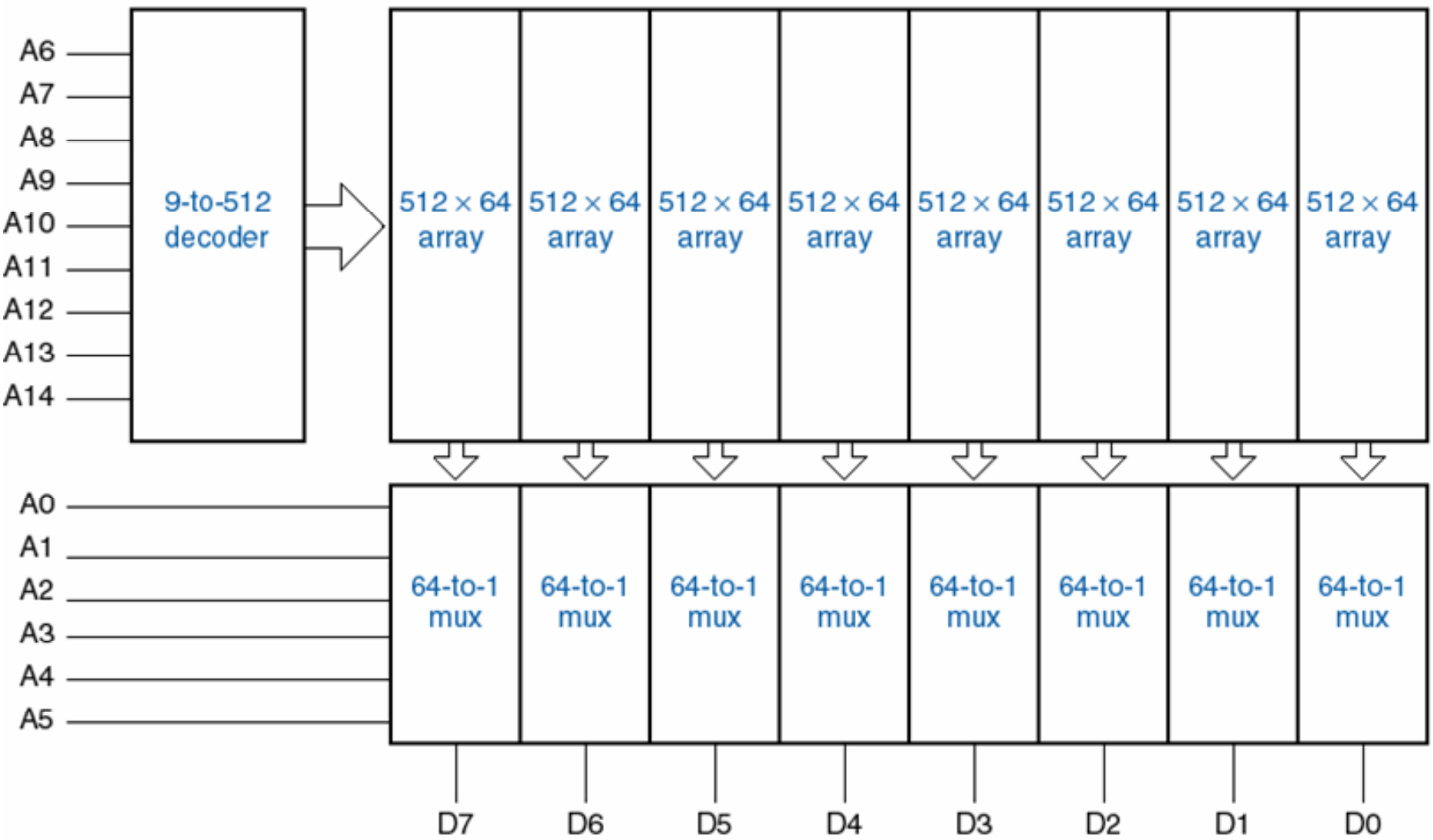


PDP-11 boot ROM
(64 words, 1024 diodes)

Two-dimensional decoding



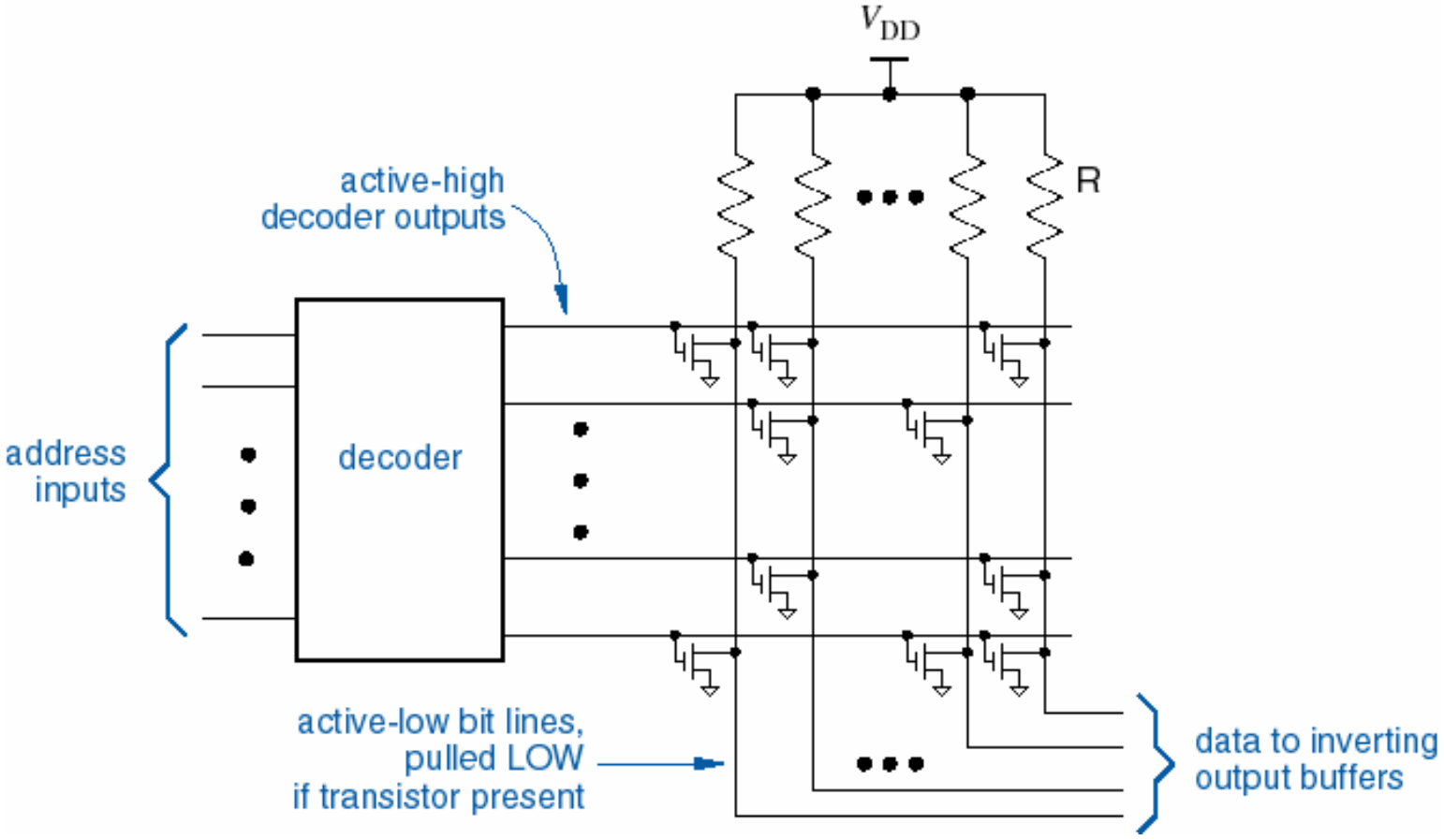
Larger example, 32Kx8 ROM



Today's ROMs

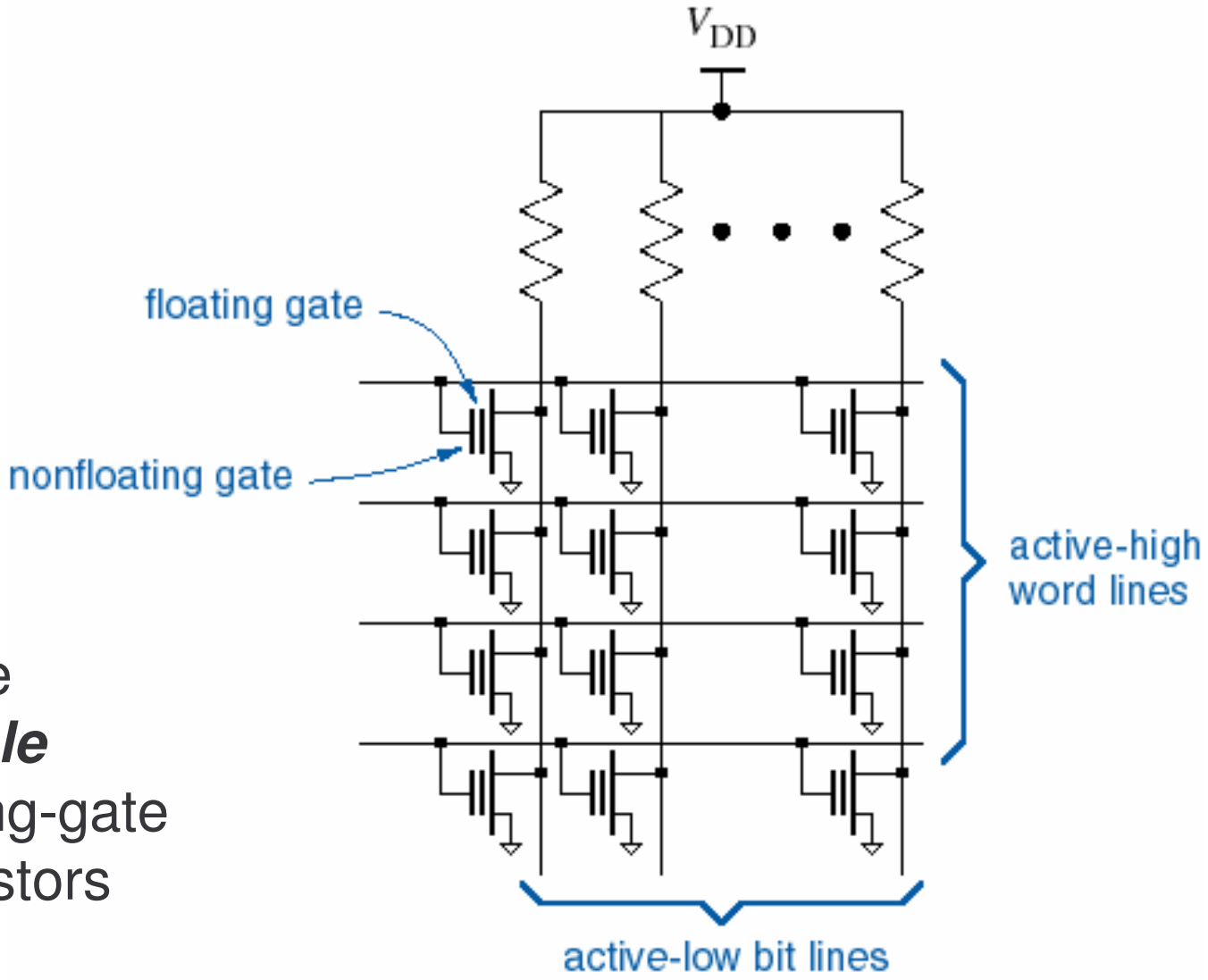
256K bytes, 1M byte, or larger

Use MOS transistors

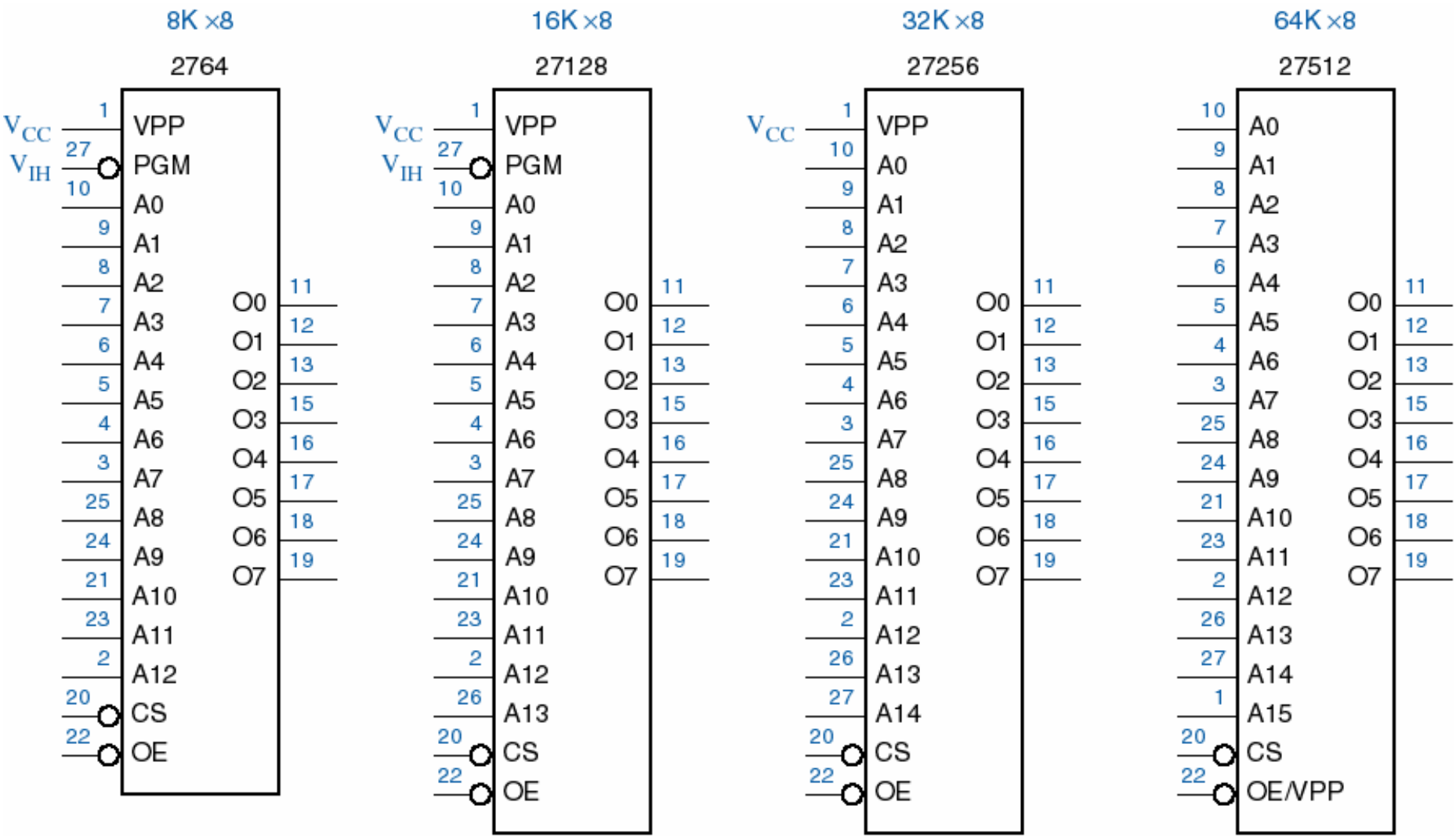


EEPROMs, Flash PROMs

Programmable
and ***erasable***
using floating-gate
MOS transistors



Typical commercial EEPROMs



EEPROM programming

Apply a higher voltage to force bit change

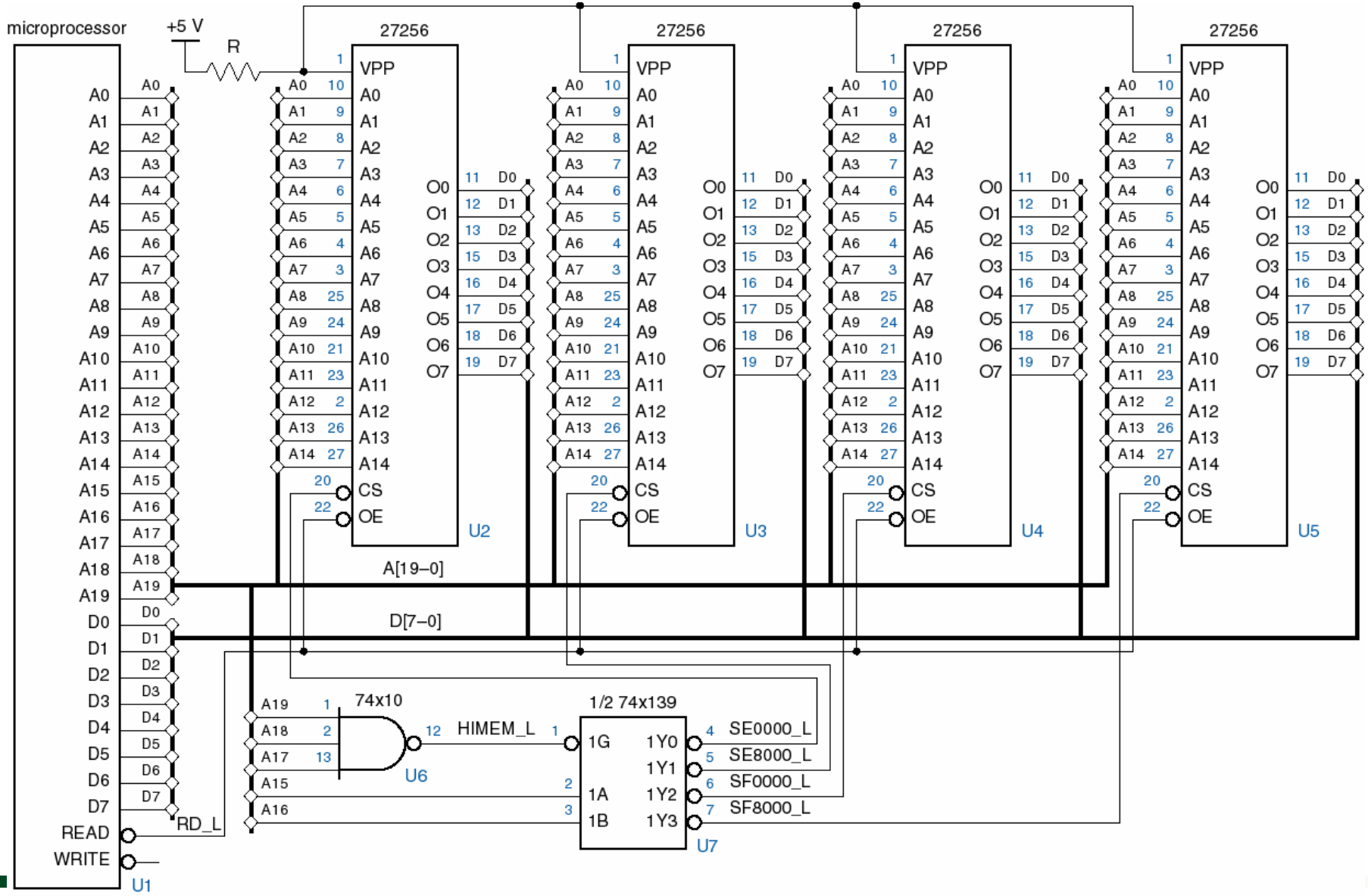
- E.g., VPP = 12 V
- On-chip high-voltage “charge pump” in newer chips

Erase bits

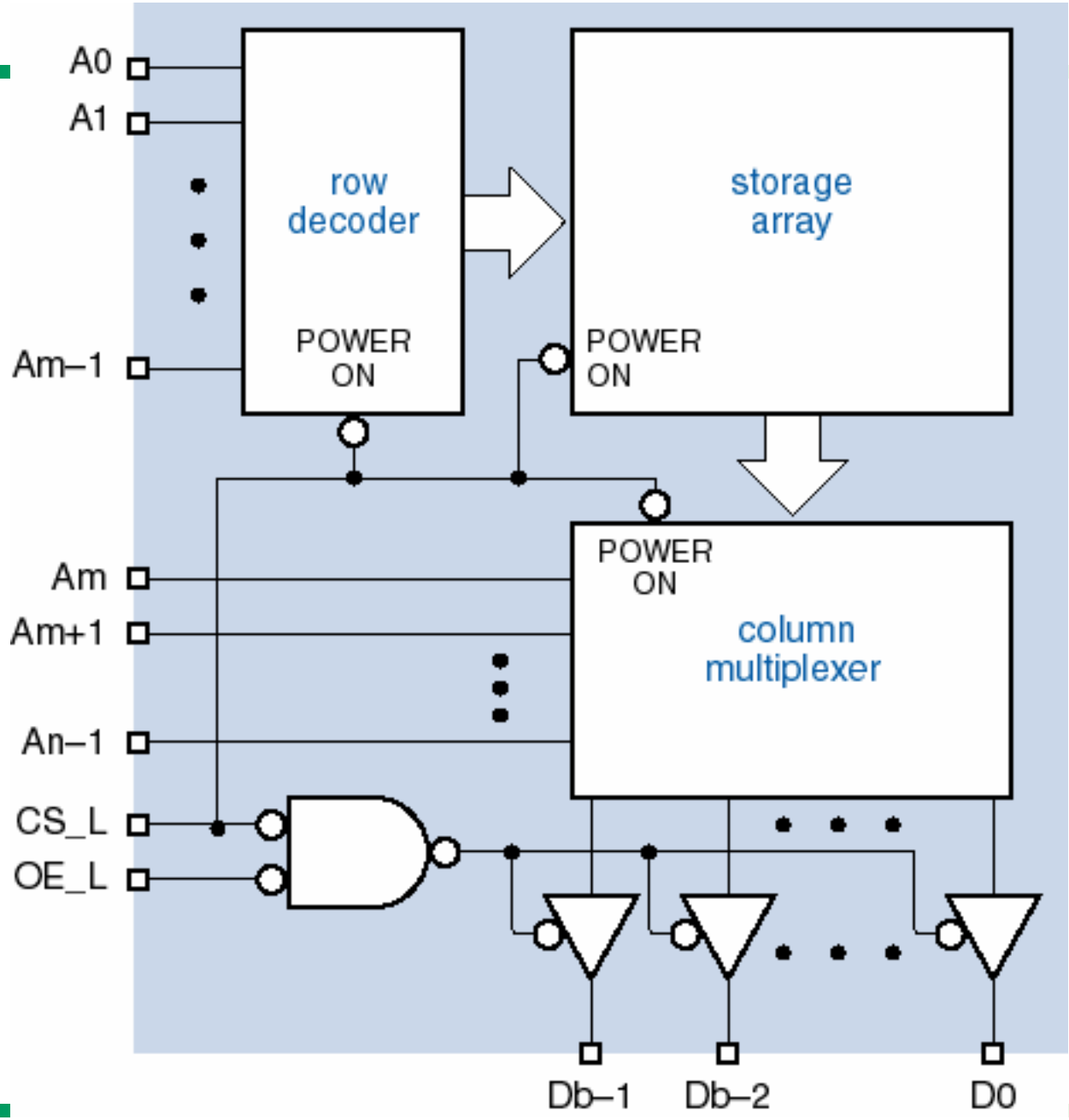
- Byte-byte
- Entire chip (“flash”)
- One block (typically 32K - 66K bytes) at a time

Programming and erasing are a lot slower than reading (milliseconds vs. 10’s of nanoseconds)

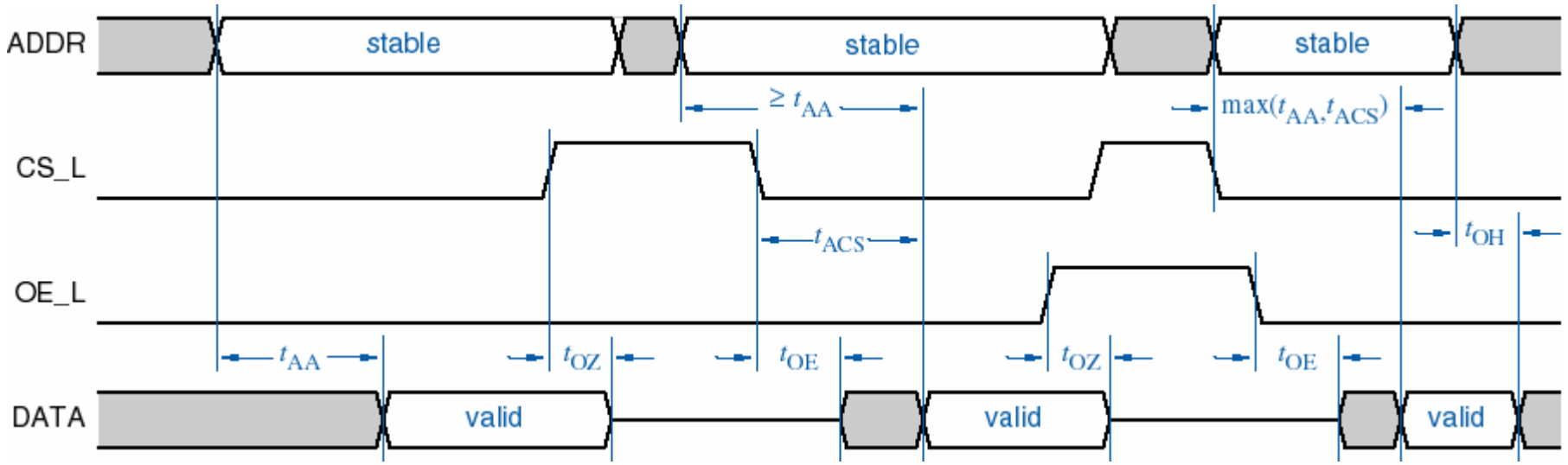
Microprocessor EPROM application



ROM control and I/O signals



ROM timing


ADDR
CS_L
OE_L
DATA
stable
stable
stable
≥ t_{AA}
max(t_{AA},t_{ACS})
t_{ACS}
t_{OH}
t_{AA}
t_{OZ}
t_{OE}
t_{OZ}
t_{OE}
valid
valid
valid