COMP 273
Introduction to Computer Systems

COMP 273

Digital Logic (Part 4)

Designing Gateway Networks

Prof. Joseph Vybihal

Announcements

- Note: these slides cross over two lectures.

Readings

* Web Resource:
  - http://www.play-hookey.com/digital/
  - http://en.wikipedia.org/wiki/Flip-flop_%28electronics%29

* Appendix C1 to C11 from our textbook

At Home

- Go through the Flip-Flop, Counter and ALU circuit diagrams to understand how they function
- Go on the web and look for the 6502 circuit CPU diagram and try to understand it. This is a classic simple CPU.
- Soul Of A New Machine

Try This Out At Home

- Create a circuit diagram for a candy dispenser machine. You type in a code number and one candy falls out.
  - Think about input of money
  - Returning change
  - Canceling the transaction

COMP 273
Introduction to Computer Systems

Lecture Outline

- Flip-flops & memory
- Designing circuit networks
- Combinatorial logic

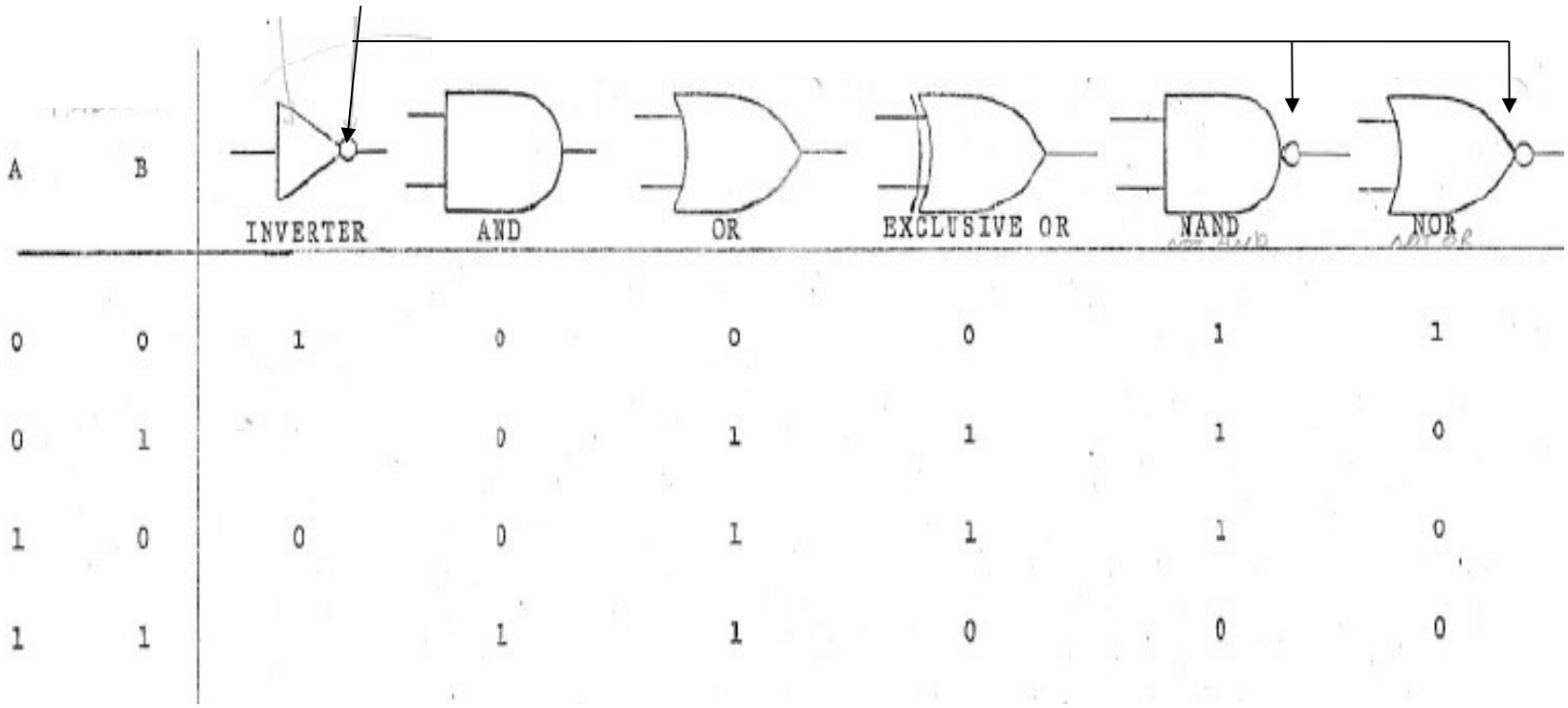
COMP 273
Introduction to Computer Systems

Part 1

The Bit, RAM, Counters, Clocks and Adders

Basic Logic Circuits

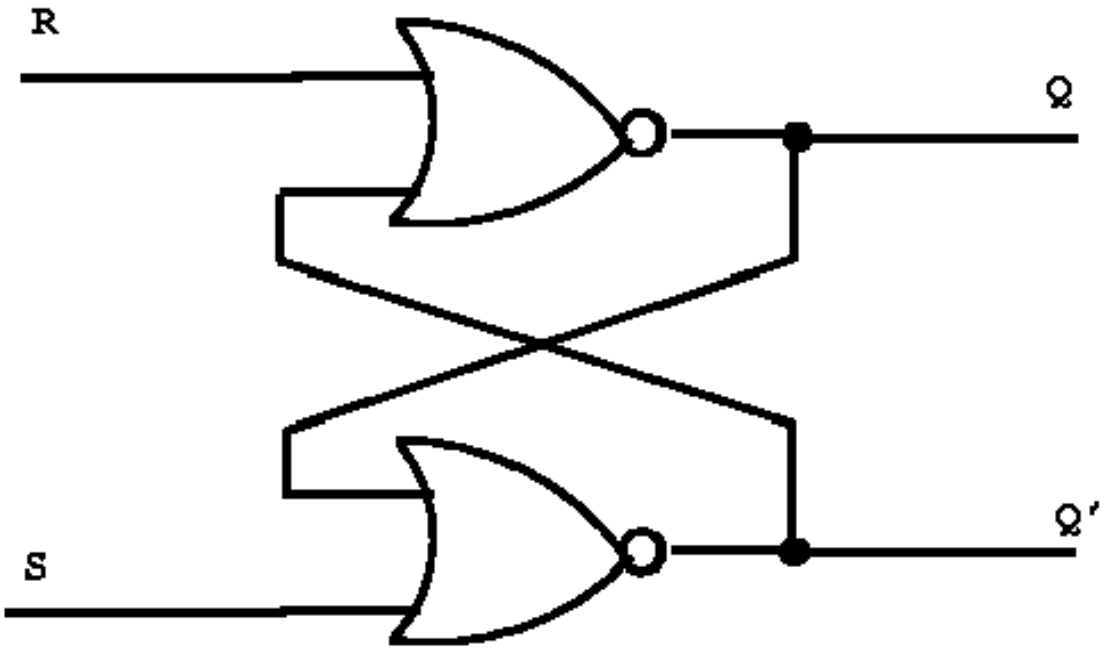
Means inverts



Uni-directional (in this case left to right)

RS Flip Flop

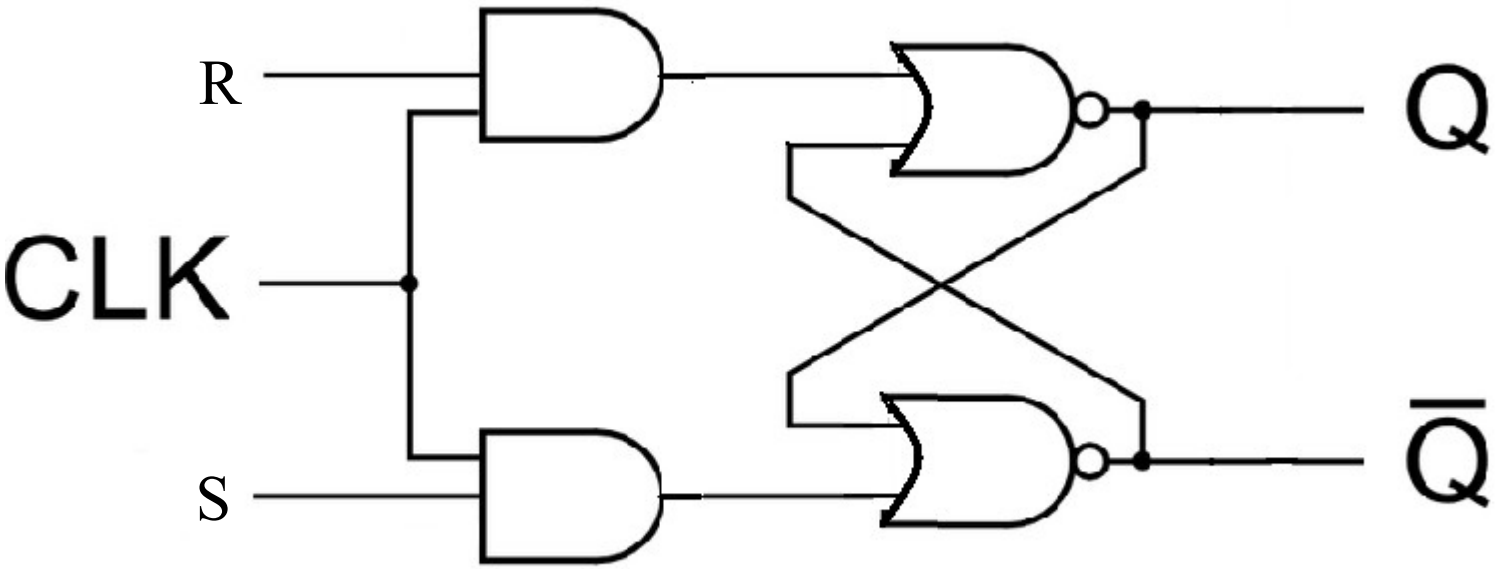
Signal stabilization wait



Where is the bit saved?

The Mechanics of Saving

R & S = 0 then Nothing changes



Similar clocked mechanism on this side.

Clocked RS Flip Flop

Example

What does the circuit of a full 2-byte read/write memory look like?

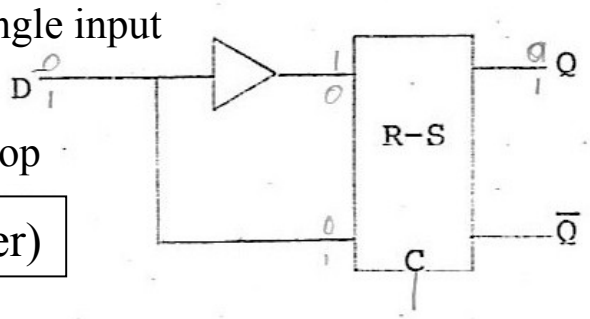
Hint: Flip-flops and incremental development

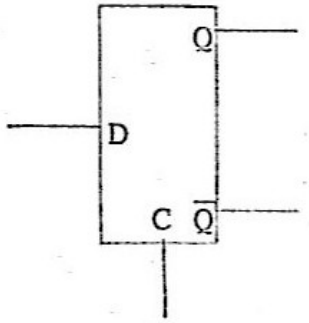
Basic Flip Flop Circuits

Single input

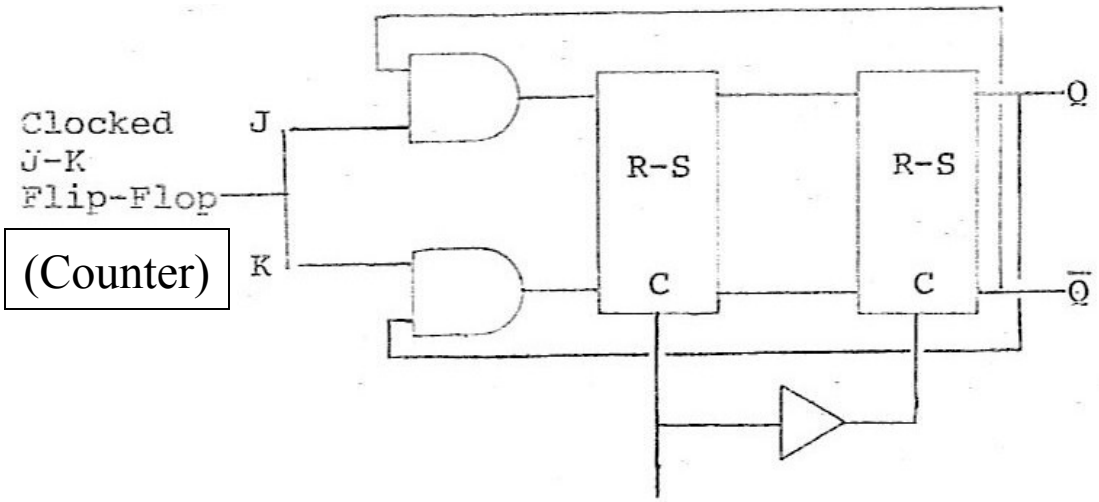
D Flip-Flop

(Register)

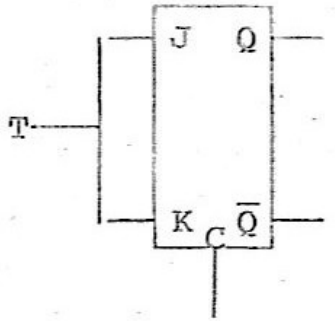




Basic Flip Flop Circuits

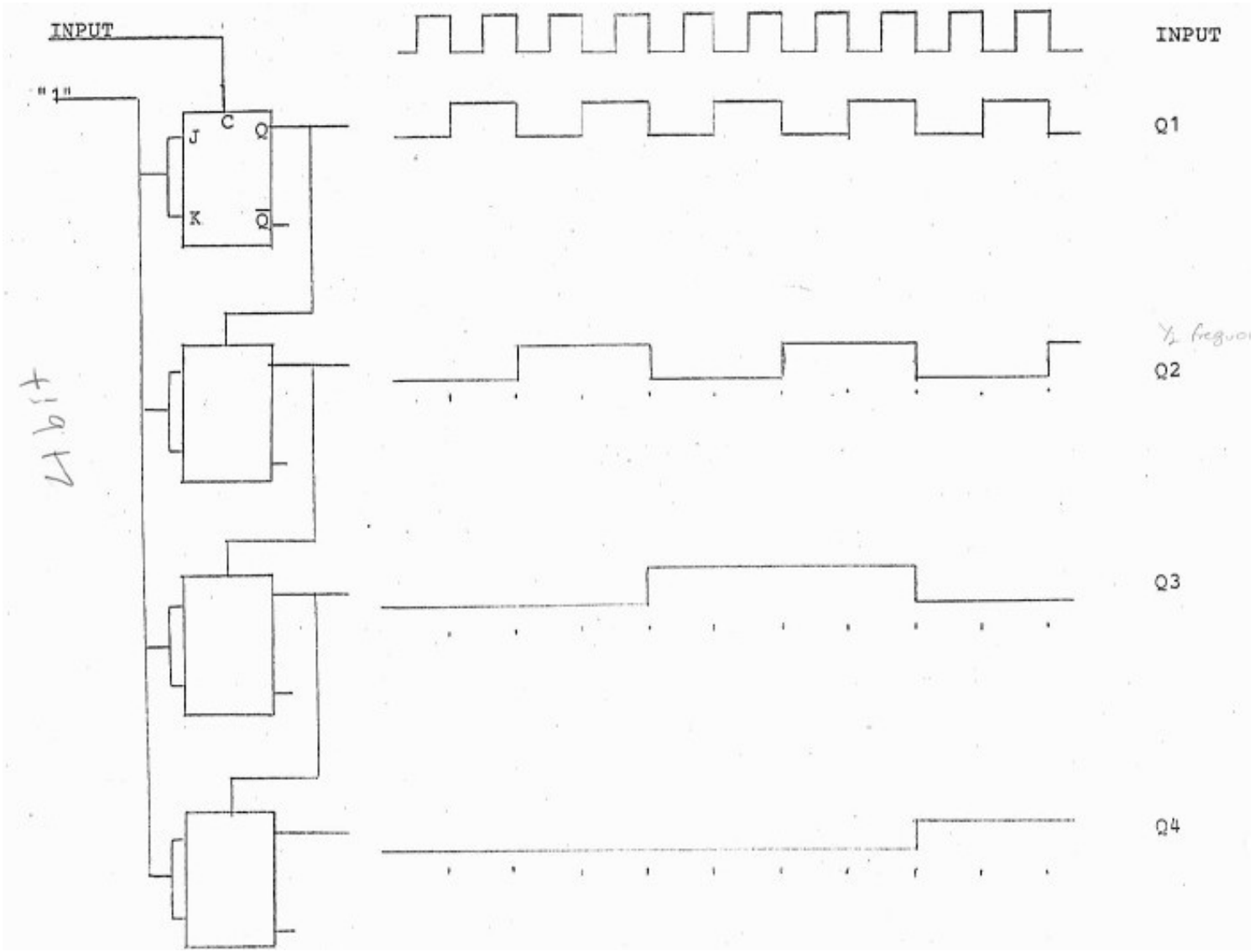


(Counter)



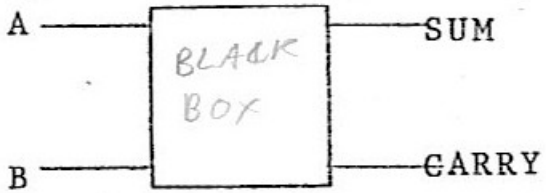
Arbitrary JPL engineer assignment of letters: J mean set and K mean reset;
J=1,K=0 SET; J=0,K=1 RESET; J=1,K=1 TOGGLE; J=0,K=0 NOTHING.

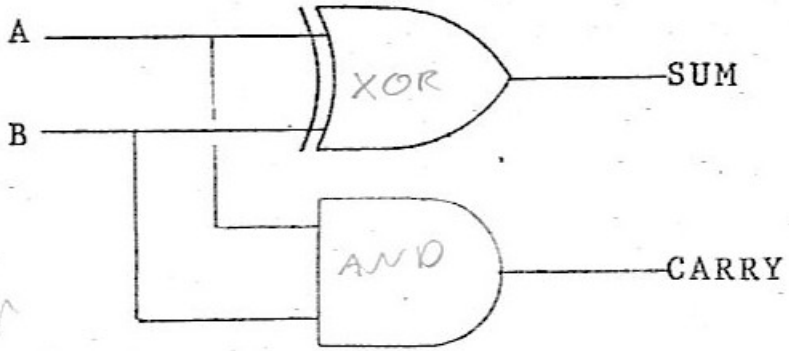
Binary Counter



Basic Adder

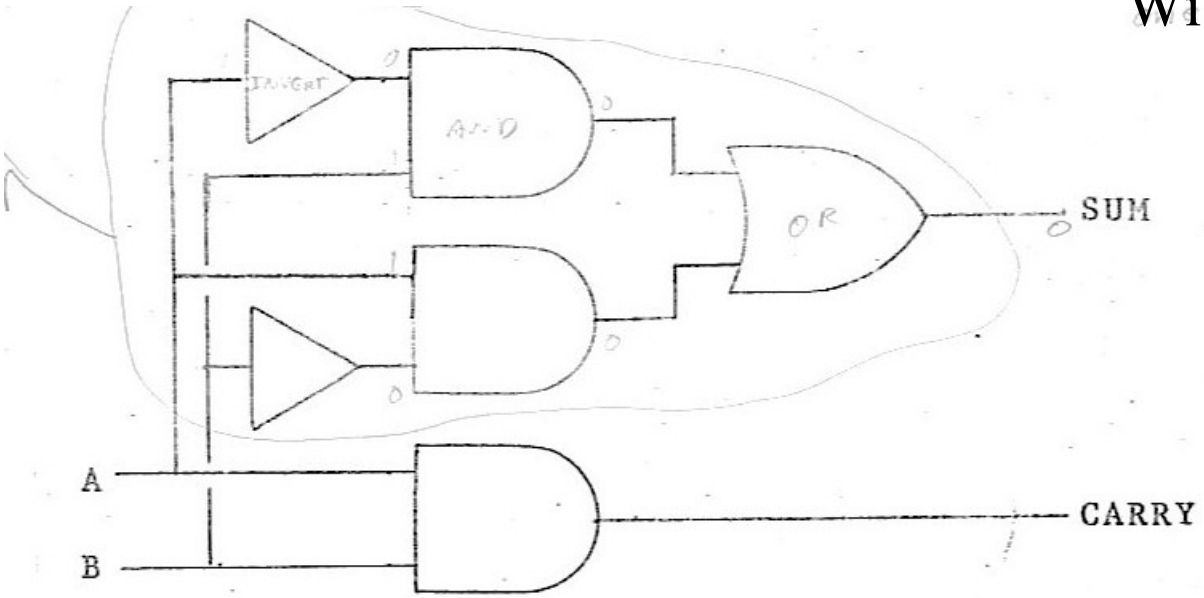
HALF-ADDER 2 bits in, Sum, Carry out



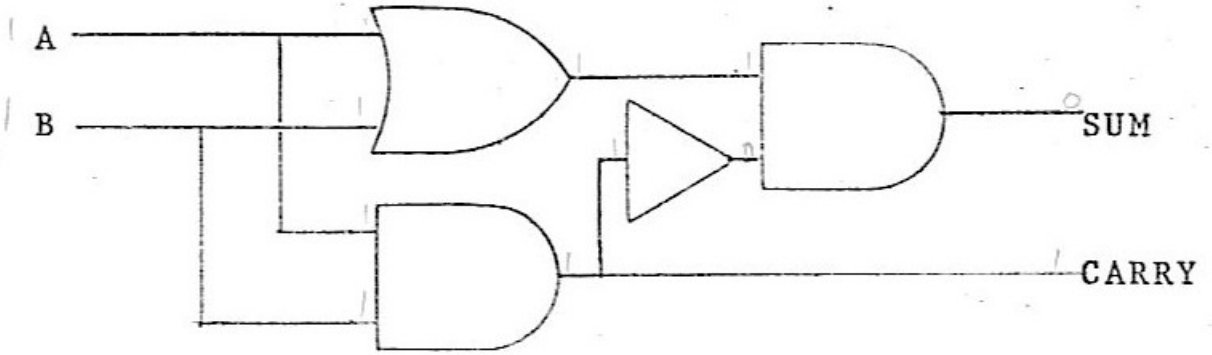


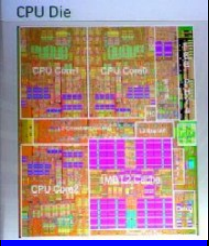
Basic Adder (Half-Adder)

Without XOR



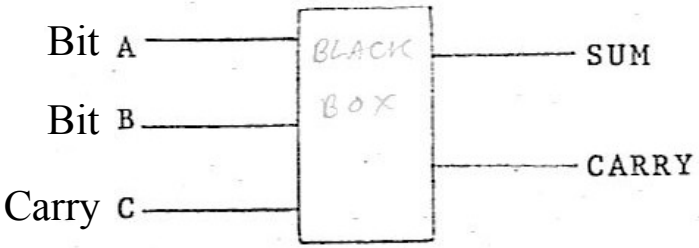
Nicer



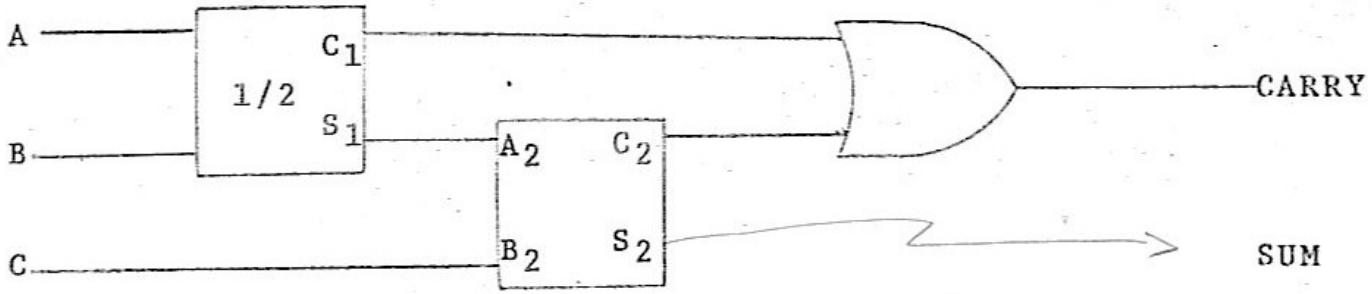


Full Adder

FULL ADDER 3 bits in, Sum,Carry out

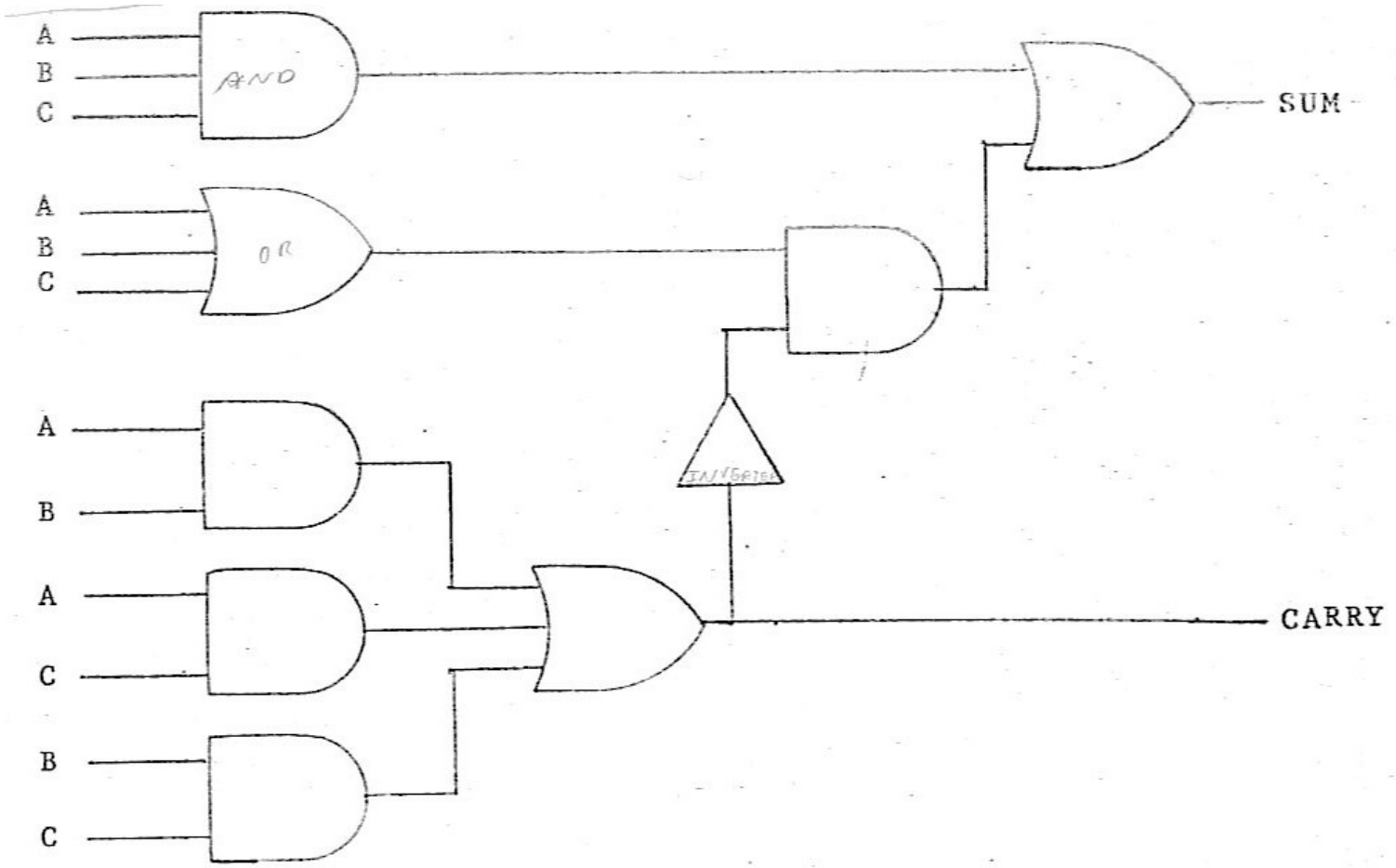


Use two half adders and an OR circuit



Full Adder

Faster execution



COMP 273
Introduction to Computer Systems

An ALU

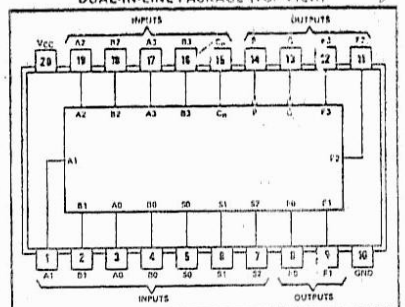
TTL
MSI

TYPE SN74S381
ARITHMETIC LOGIC UNIT/FUNCTION GENERATOR
BULLETIN NO. DL-S 7412124, MARCH 1974

PIN DESIGNATIONS

DESIGNATION	PIN NOS.	FUNCTION
A3, A2, A1, A0	17, 19, 1, 3	WORD A INPUTS
B3, B2, B1, B0	16, 18, 2, 4	WORD B INPUTS
S2, S1, S0	7, 6, 5	FUNCTION-SELECT INPUTS
C_n	15	CARRY INPUT FOR ADDITION, INVERTED CARRY INPUT FOR SUBTRACTION
F3, F2, F1, F0	12, 11, 9, 8	FUNCTION OUTPUTS
$\overline{P}$	14	INVERTED CARRY PROPAGATE OUTPUT
$\overline{G}$	13	INVERTED CARRY GENERATE OUTPUT
V_{CC}	20	SUPPLY VOLTAGE
GND	10	GROUND

N
DUAL-IN-LINE PACKAGE (TOP VIEW)



logic: see function table

- A Fully Parallel 4-Bit ALU in 20-Pin Package for 0.300-Inch Row Spacing
- Ideally Suited for High-Density Economical Processors
- Parallel Inputs and Outputs and Full Look-Ahead Provide System Flexibility
- Arithmetic and Logic Operations Selected Specifically to Simplify System Implementation:
  - A Minus B
  - B Minus A
  - A Plus B
  - and Five Other Functions
- Schottky-Clamped for High Performance
  - 16-Bit Add Time . . . 29 ns Typ Using Look-Ahead
  - 32-Bit Add Time . . . 34 ns Typ Using Look-Ahead

FUNCTION TABLE

SELECTION			ARITHMETIC/LOGIC
S2	S1	S0	OPERATION
L	L	L	CLEAR
L	L	H	B MINUS A
L	H	L	A MINUS B
L	H	H	A PLUS B
H	L	L	A ⊕ B
H	L	H	A + B
H	H	L	AB
H	H	H	PRESET

H = high level, L = low level

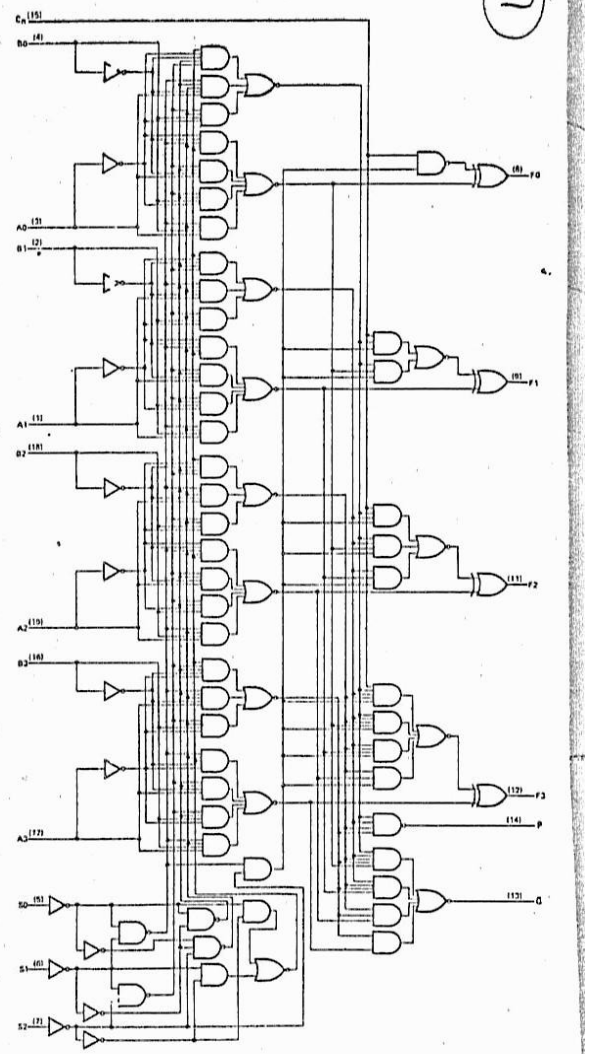
description

The SN74S381 is a Schottky TTL arithmetic logic unit (ALU)/function generator that performs eight binary arithmetic/logic operations on two 4-bit words as shown in the function table. These operations are selected by the three function-select lines (S0, S1, S2). A full carry look-ahead circuit is provided for fast, simultaneous carry generation by means of two cascade outputs ($\overline{P}$ and $\overline{G}$) for the four bits in the package. The method of cascading SN54182/SN74182 or SN54S182/SN74S182 look-ahead carry generators with these ALU's to provide multi-level full carry look-ahead is illustrated under typical applications data for the '182 and 'S182. The typical addition times shown above illustrate the short delay time required for addition of longer words when full look-ahead is employed. The exclusive-OR, AND, or OR function of two Boolean variables is provided without the use of external circuitry. Also, the outputs can be either cleared (low) or preset (high) as desired.

CHIP CIRCUITRY

ARITHMETIC LOGIC UN

functional block diagram and schematics of inputs and outputs



ALU Questions

- What do we need to assume about our data to build an ALU?
- How would the assumptions impact the design of the ALU?
- In general, how would a 4-bit integer ALU that does addition and subtraction look like? What do you think it would need?
- How would the ALU design change when upgrading a 4-bit integer adding and subtracting ALU to an 8-bit version?

General Circuit Questions

- Construct a circuit that implements a truth table
- Construct a simple 3 bit increment by 1 circuit (do not use adder)
- 2's complement circuit?
- Construct a circuit that picks 1 bit out of an 2-bit bus given a 1-bit positive binary integer number
  - What about 4-bit bus & 2-bit binary number

Designing gateway networks

EXAMPLE

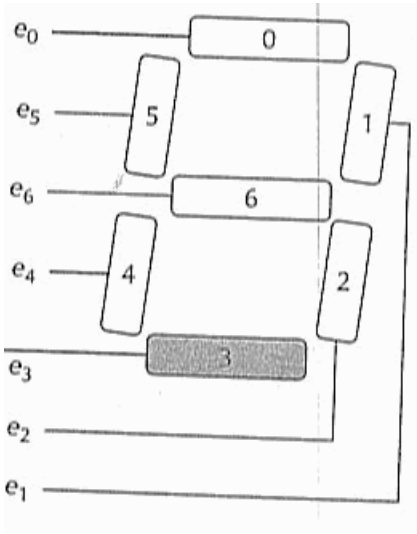
Binary Code Decimal Example

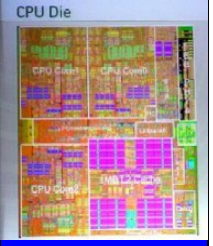
Example 1.2: BCD-to-seven-segment decoder Figure 1.7 shows how the decimal digits 0–9 might appear on a seven-segment display device. Design logic circuits to generate the enable signals that cause the segments to be lit or darkened, given a 4-bit binary representation of the decimal digit (binary-coded decimal or BCD code) to be displayed as input.

Figure 1.7 Seven-segment display of decimal digits. The three open segments may be optionally used. The digit 1 can be displayed in two ways, with the more common right-side version shown.

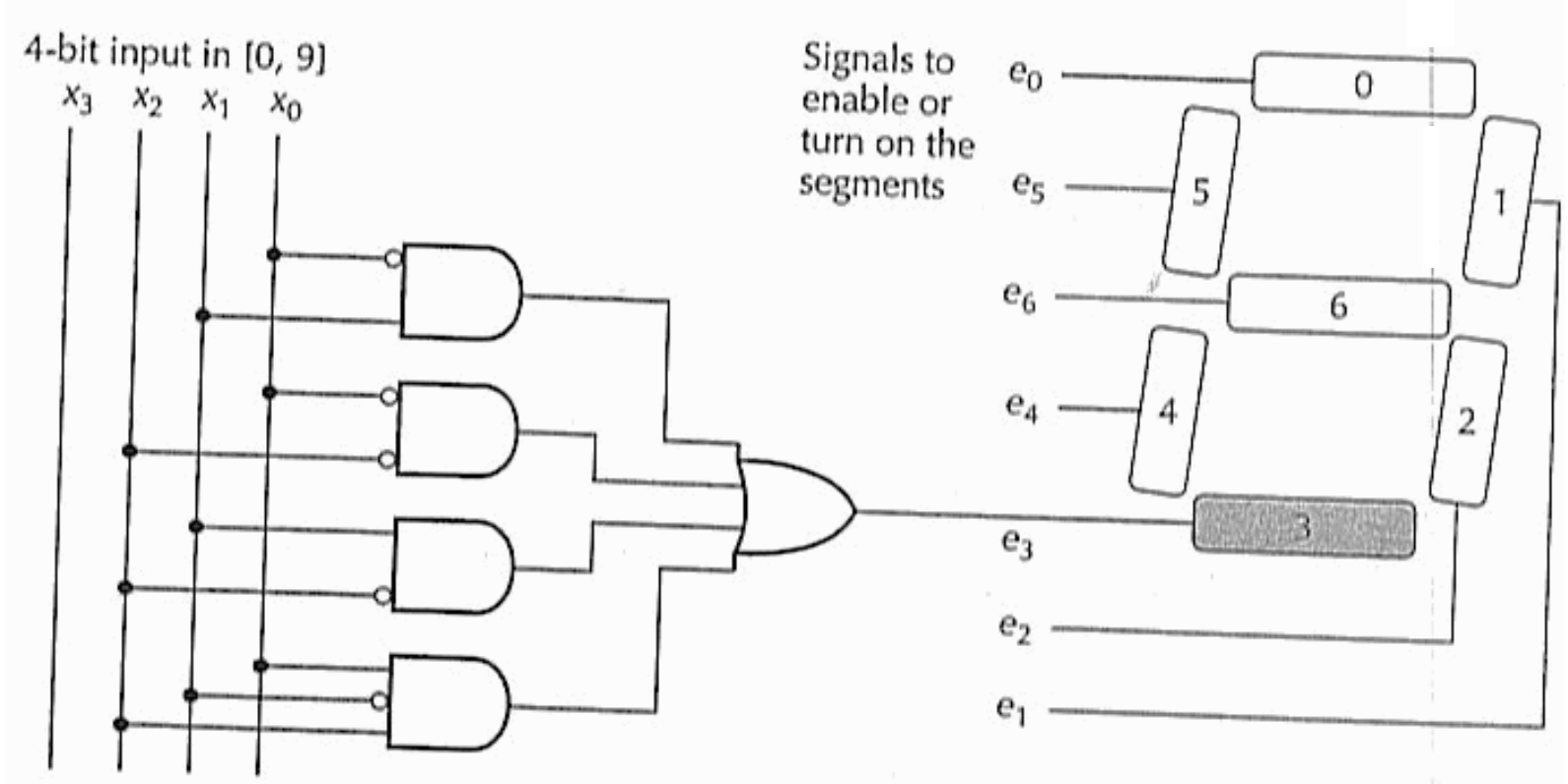
Working Truth Table

DIGIT	INPUT: abcd	OUTPUT: 0123456
0	0000	1111110
1	0001	0110000
2	0010	etc
3	0011	
4	0100	
5	0101	
6	0110	
7	0111	
8	1000	
9	1001	





Now Construct the Circuit

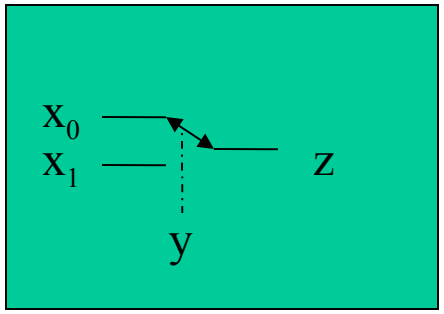


Part 2

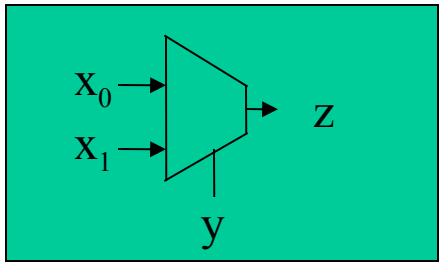
Useful (off-the-shelf) Combinatorial Networks

Multiplexers

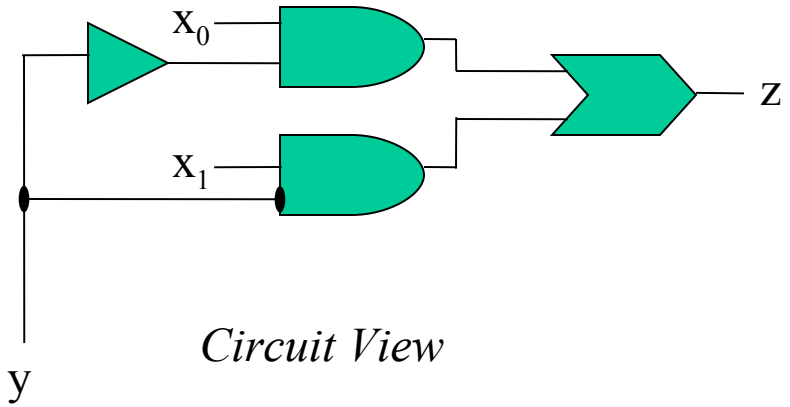
Definition: 2^a-to-1 mux has 2^a inputs $x_o,\ldots x_{n-1}$ and a single output z. A selector address 'a' composed of input signals $y_0,\ldots y_{a-1}$ selects which x_i signal passes through z.



Switch View



Symbolic View

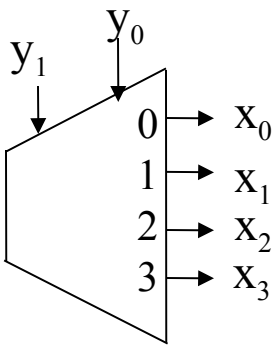


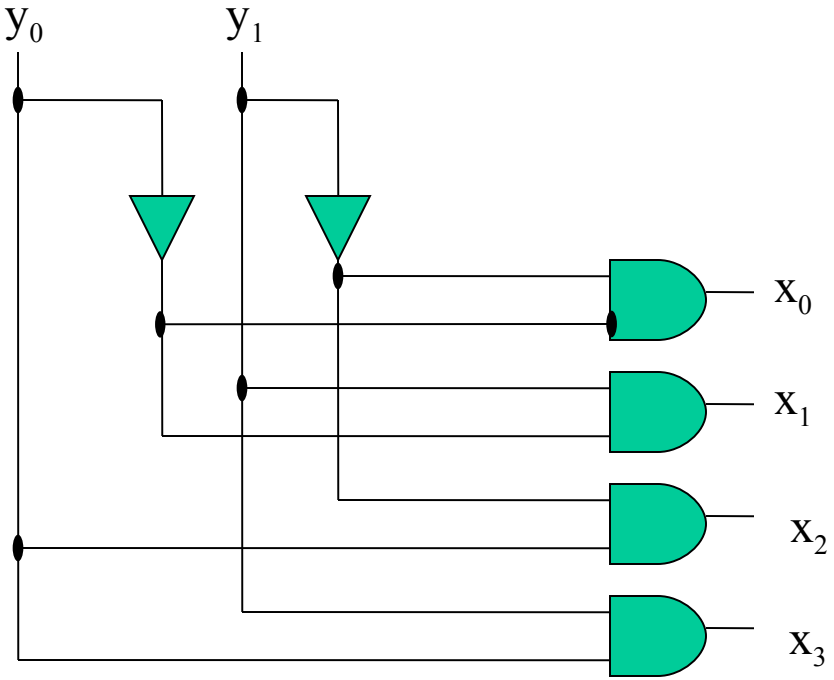
Circuit View

Multiplexers and networks?

Decoders

Definition: a-to-2^a decoder asserts one and only one of its 2^a output lines.



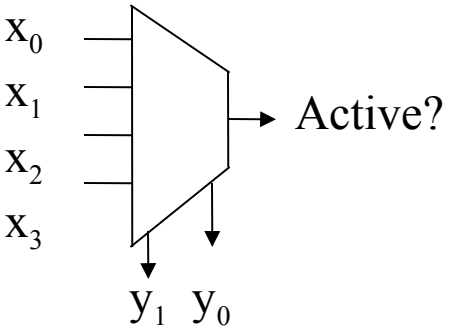


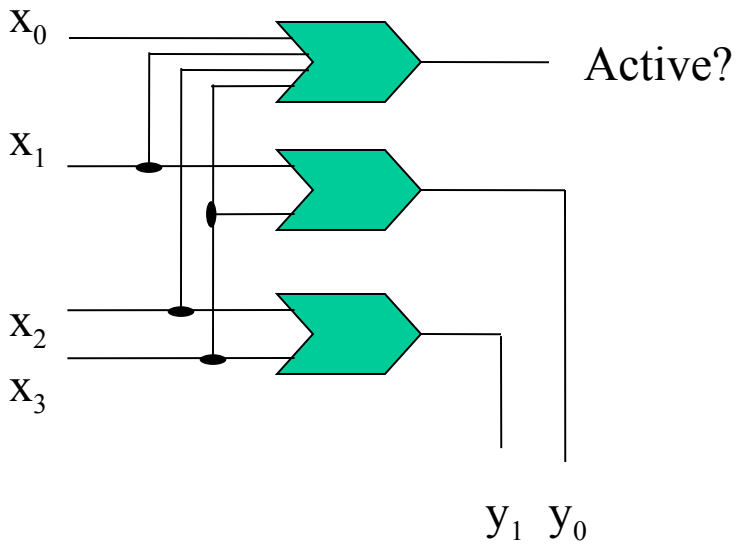
2-4 Decoder
Binary number triggers
1 output line

Turn on a switch…
The CPU Sequencer...

Encoders

Definition: Outputs an a-bit binary number, y_i, equal to the index of the single 1 signal among its 2^a inputs, $x_0 \ldots x_{a-1}$.



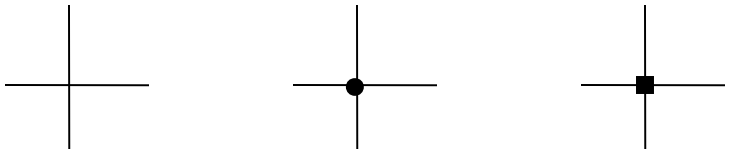


Active is used to tell us if x_0 is ON as compared with circuit turned off.

Hardware error codes…

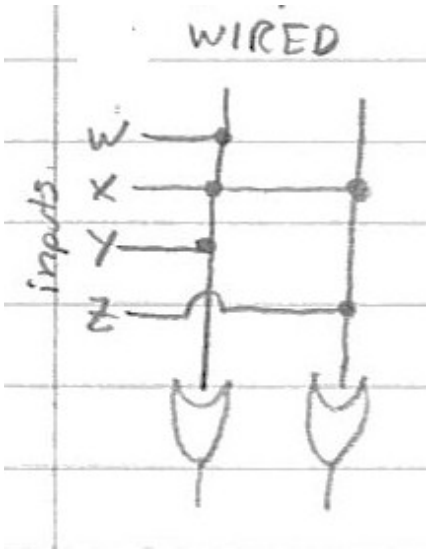
Programmable Combinatorial Parts
(PLA, PROM, ROM)

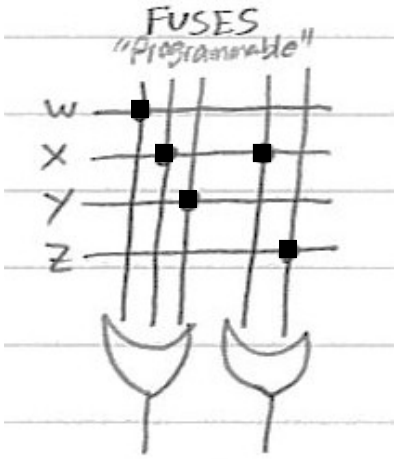
Type 1: VIA fuses that can be “blown” given sufficient current.
Type 2: VIA anti-fuse that “set” given sufficient current.



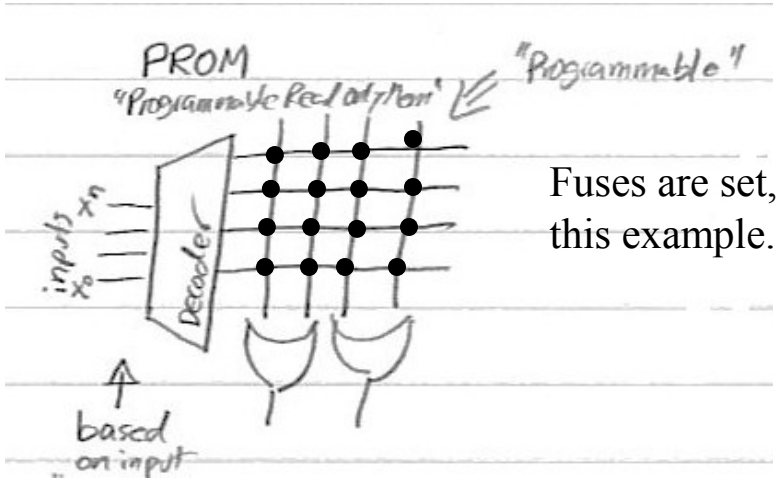
Note

ROM & PROM

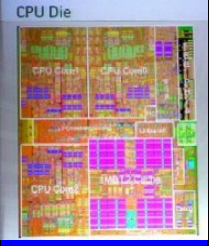




Fuses not blown



Fuses are set, not yet so in this example. Suggestions?



PLA & PAL

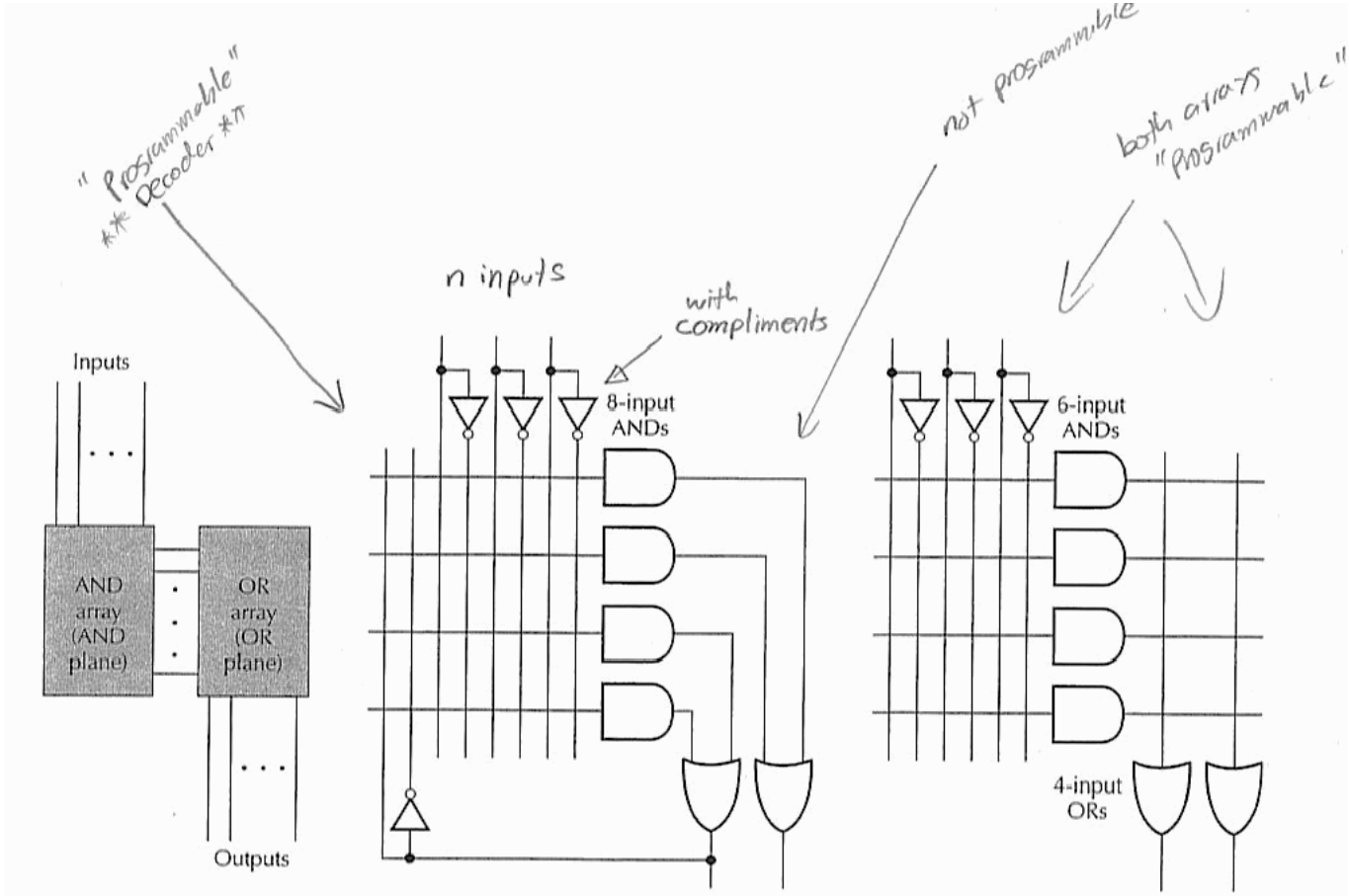


Figure 1.13 Programmable combinational logic: general structure and two classes known as PAL and PLA devices. Not shown is PROM with fixed AND array (a decoder) and programmable OR array.

PAL = Programmable Array Logic

PLA = Programmable Logic Array

PLA Question

Given a 2 bit binary number, use a PLA that sends out a single bit signal for each binary value: 00, 01, 10, 11.

(Like a decoder)

More Questions…

- Assume a temperature is stored in a register as a binary number, assume an alarm is on when it receives an input signal. Set the alarm when the temperature is at or above 500.
- Numeric key-pad operation with num-lock feature (home, up, pg-up, left, nothing, right, end, down, pg-dn, ins, del, / * - + and enter).
- Robot obstacle avoidance circuit: motor on/off, left on/off, right on/off, forward sensor, left sensor, and right sensor. Write a circuit that will cause the robot to follow the wall on its left.
  - What do we need? (equipment, input, output)