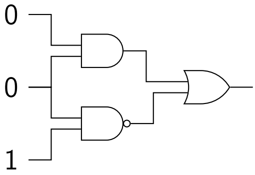
Example circuit



What is the output of this circuit?

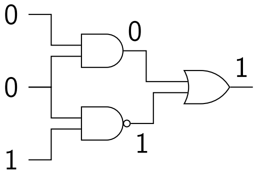
A. 0

B. 1

C. not defined

Vote at m.socrative.com. Room number 415439.

Example circuit

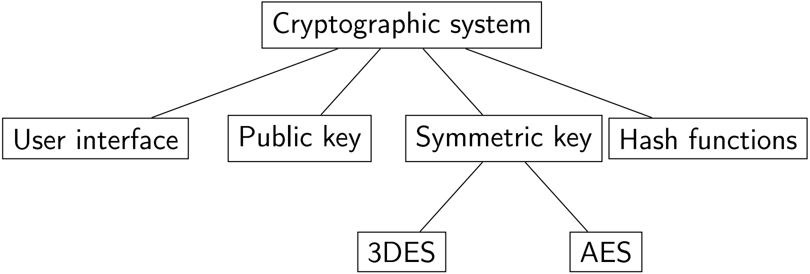


What is the output of this circuit?

B. 1

Abstraction

Example: Top-down design - cryptographic system



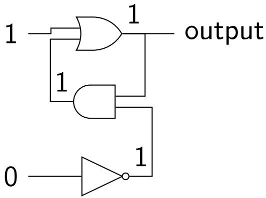
Abstraction

Things at higher levels need not know how things at lower levels function, only how to use them.

Interface, modularity, and modelling give:

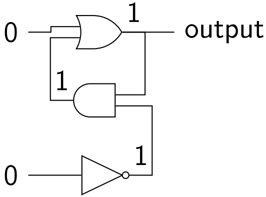
- Structure — divide up work
- Independence between modules
  (can re-implement without changing the rest)
- Ability to analyze
- Increased innovation, productivity
  (don't need to re-invent the wheel)

Flip flop



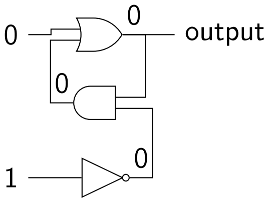
Note that this is stable.
Keeps same output until temporary outside pulse.
Can store a bit.

Flip flop



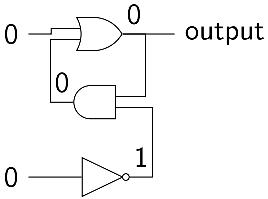
Note that this is stable.

Flip flop



Note that this is stable.

Flip flop



Note that this is stable.
But two different stable outputs are possible with input (0,0).

Flip flops can be implemented differently. Fig. 1.5, p. 36.
Abstraction: know input/output effect —
don't care about implementation.

Hexadecimal Notation

To shorten bit strings for humans:

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Storage technology

capacitors on chips??? — changes!!!

dynamic memory — need to refresh data, it dissipates
non-volatile memory — doesn't lose data if power lost

Memory:

byte — 8 bits

0 1 0 1 1 0 0 1

high-order bit

low-order bit

most significant bit

least significant bit

Storage technology

Main memory

- words = cells — fixed size
  8, 16, 24, 32, 64 bits
- words have addresses - count from 0
- can use consecutive words if need more bits for value
- can access words in any order random access memory (RAM)
- get value of word — read or load
- place value of word — write or store

Storage technology

Main memory

- size — power of 2 — addresses fixed length (usually)
  - $2^{10} = 1024$ bytes $=$ 1 kilobyte — 1 KB
  - 4096 bytes $=$ 4 KB
  - $2^{20} = 1,048,576$ bytes $=$ 1 megabyte — 1MB
  - $2^{30} = 1,073,741,824$ bytes $=$ 1 gigabyte — 1GB
  - $2^{40} = 1,099,511,627,776$ bytes $=$ 1 terabyte — 1TB
- Some people use these terms for powers of 10.

Storage technology

Mass (secondary) storage

- disk, CD's, magnetic tapes
- flash memory (SSD — solid-state disks, SD — secure digital, SDHC — high capacity)
- CD $\rightarrow$ DVD $\rightarrow$ Blu-ray
  similar technologies (optical) — more capacity
- on-line vs. off-line — human intervention
- mechanical, slower (except flash memory)
- disk
  - often several in layers — space for heads
  - read/write heads above tracks
  - cylinder — tracks on top of each other

Storage technology

Mass (secondary) storage

- disk
  - sector — arc of a track
    - files stored as physical records = sectors
      vs. logical records (fields, keys)
    - each contains same number of bits
      (512 or 1024 bits, for example)
    - within a group of tracks, each contains same number of sectors — having different groups, with fewer sectors toward middle is zone bit recording
    - locations of tracks and sectors marked magnetically during formatting

Storage technology

Secondary storage

- flash memory
  - cameras, cell phones, etc.
  - not mechanical
  - not dynamic
  - hard to erase or rewrite a few locations often
  - intensive writing reduces lifespan

Text

Text — characters (symbols) — standards

- ASCII — appendix A
- EBCDIC
- BCD
- Unicode — implemented by different character encodings
  - UTF–8 — one byte for ASCII, up to 4 bytes
  - UCS–2 — older, 16 bit codes
  - UTF–16 — extends UCS–2, two 16-bit code units

Integers

Integers

- Base 10 — $234 = 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0 = \sum_{i=0}^{2} d_i \cdot 10^i$
  Generally $d_{k-1} \ldots d_1 d_0 = \sum_{i=0}^{k-1} d_i \cdot 10^i$.
- Base 2 — $11101100 =$
  $1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = \sum_{i=0}^{7} b_i \cdot 2^i$
  Generally $b_{k-1} \ldots b_1 b_0 = \sum_{i=0}^{k-1} b_i \cdot 2^i$.

Integers

Algorithm to find binary representation:

procedure convert(value):
{ Input: **integer** value }
{ Output: **char string** str }

str := λ
remainder := value **mod** 2
str := remainder || str
quotient := value **div** 2

while quotient $\neq$ 0
begin
remainder := quotient **mod** 2
str := remainder || str
quotient := quotient **div** 2
end
return(str)

Numbers

- Adding binary — unsigned integers can get extra bit
- fractions: $101.011 = 5\frac{3}{8}$

Two's complement

two's complement, 32 bits common

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7
1000	-8

Two's complement

- sign bit — high order bit
- $+x, \quad -x$ — same low order bits to first 1
  complement after that
- addition — same as before (2+(-5))
- subtraction — How?

Two's complement

- sign bit — high order bit
- $+x, \quad -x$ — same low order bits to first 1
  complement after that
- addition — same as before (2+(-5))
- subtraction — create negative and add
- Overflow — $3 + 7 = ?$

Two's complement

- sign bit — high order bit
- $+x, \quad -x$ — same low order bits to first 1
  complement after that
- addition — same as before (2+(-5))
- subtraction — create negative and add
- Overflow — $3+7=1010=-6$
  2,147,483,646 OK without overflow in 32-bit
  overflow bit can be checked

Excess notation

with 4 bits, bias 8 (p.62)

1111	7
1110	6
1101	5
1100	4
1011	3
1010	2
1001	1
1000	0
0111	-1
0110	-2
0101	-3
0100	-4
0011	-5
0010	-6
0001	-7
0000	-8

How do you get value?

Excess notation

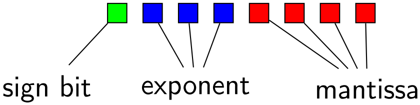
with 4 bits, bias 8 (p.62)

1111	7
1110	6
1101	5
1100	4
1011	3
1010	2
1001	1
1000	0
0111	-1
0110	-2
0101	-3
0100	-4
0011	-5
0010	-6
0001	-7
0000	-8

subtract bias to get value

Floating point

Textbook doesn't use implicit leading bit (you should)



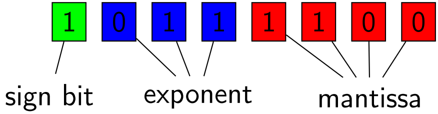
exponent — excess notation, bias 4

111	3
110	2
101	1
100	0
011	-1
010	-2
001	-3
000	-4

p. 63

Floating point

Textbook doesn't use implicit leading bit (you should)



mantissa — implicit leading bit

It is really 5 bits, with the first bit 1. $1100 \rightarrow 1.1100$

sign — negative

exponent — $011 \rightarrow -1$

$-(1.11 \cdot 2^{-1}) = -\frac{7}{8}$

Floating point

$$1\frac{1}{8}$$

mantissa — 1.001 $\rightarrow$ 0010
exponent — 0 $\rightarrow$ 100
result — 01000010

How do we represent $2\frac{5}{8}$? (Note: can't in book.)

A. 01010101
B. 00101010
C. 01011101
D. 00111010

Vote at m.socrative.com. Room number 415439.

Floating point

$$1\frac{1}{8}$$

mantissa — 1.001 $\rightarrow$ 0010
exponent — 0 $\rightarrow$ 100
result — 01000010

How do we represent $2\frac{5}{8}$? (Note: can't in book.)

[A.] 01010101

Floating point

$$4\tfrac{5}{8} \; = \; 100.101$$
$$\text{exponent} = 2 \;\; \rightarrow \;\; 110$$

result — 01100010

Last bit is truncated. $4\frac{5}{8} = 4\frac{1}{2}$?
$(4\frac{1}{2} + \frac{1}{8}) + \frac{1}{8} = 4\frac{1}{2}$?
$4\frac{1}{2} + (\frac{1}{8} + \frac{1}{8}) = 4\frac{3}{4}$?

Truncation errors and reducing them
— numerical analysis

Floating point

What about $\frac{1}{3}$ and $\frac{1}{10}$.

A. $\frac{1}{3}$ and $\frac{1}{10}$ both require truncation.
B. $\frac{1}{3}$ requires truncation, but not $\frac{1}{10}$
C. $\frac{1}{10}$ requires truncation, but not $\frac{1}{3}$.
D. Neither $\frac{1}{3}$ nor $\frac{1}{10}$ require truncation.

Vote at m.socrative.com. Room number 415439.

Floating point

What about $\frac{1}{3}$ and $\frac{1}{10}$.

[A.] $\frac{1}{3}$ and $\frac{1}{10}$ both require truncation.

Images

Bit map — scanner, video camera, etc.

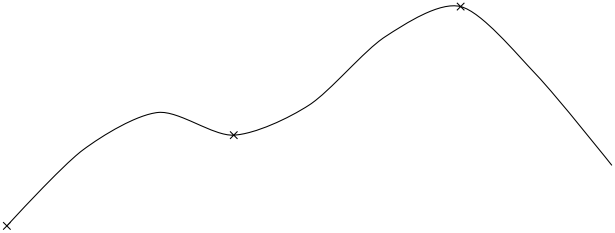
- image consists of dots — pixels
- 0 — white; 1 — black
- colors — use more bits —
  - red, green, blue components
  - 3 bytex per pixel
  - example: 1024×1024 pixels
  - megapixels (how many millions of pixels)
  - need to compress

Images

Vector techniques — fonts for printers

- scalable to arbitrary sizes
- image = lines and curves
- poorer photographic quality

Sound



Sounds waves

- sample amplitude at regular intervals — 16 bits
  -8000/sec — long distance telephone
  -more for music
- Musical Instrument Digital Interface — MIDI
  -musical synthesizers, keyboards, etc.
  -records directions for producing sounds (instead of sounds)
    -what instrument, how long