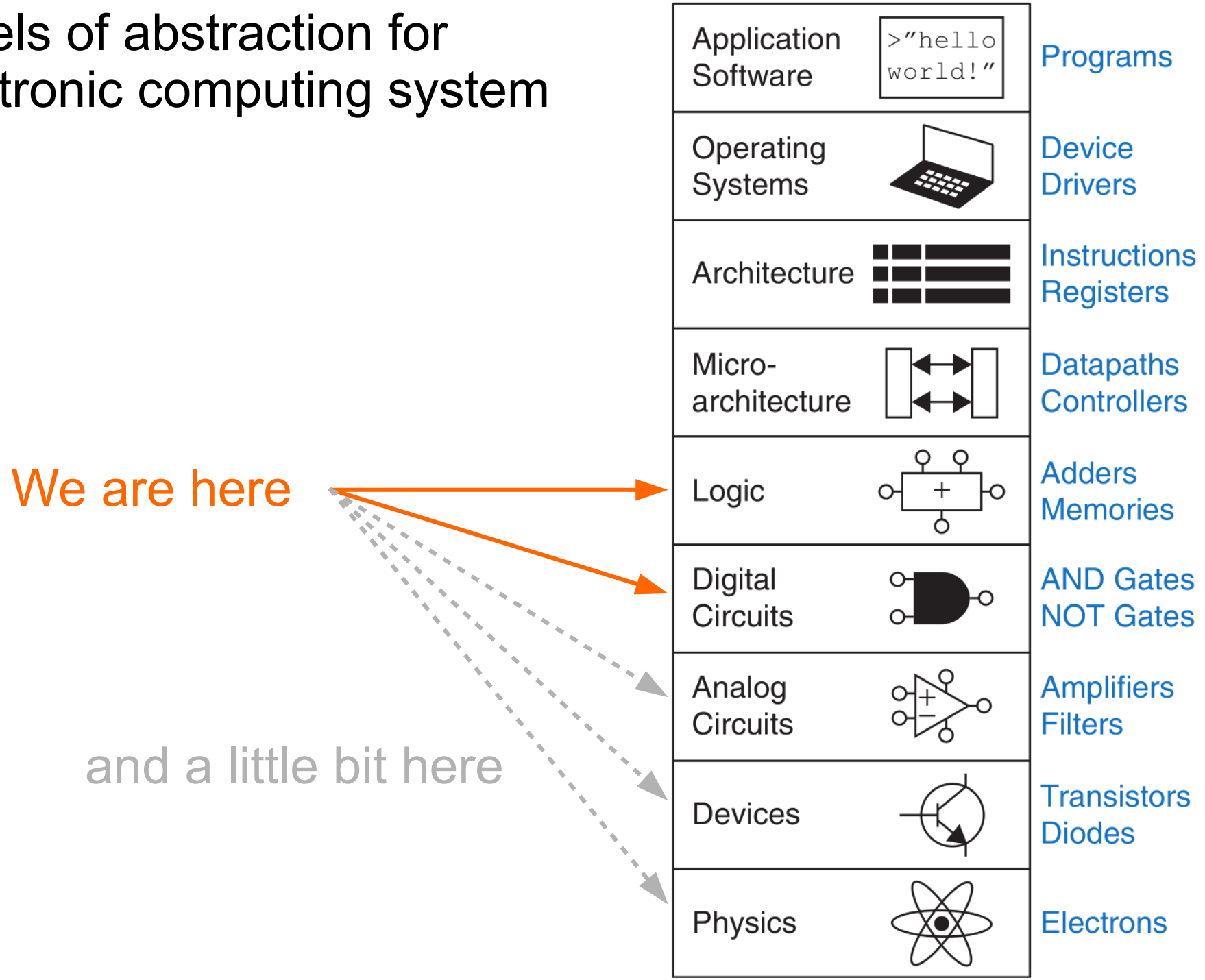
Digital Logic Design

Lecture 1

FPGA, analog and digital signals, numbers

Levels of abstraction for electronic computing system



Programmable logic devices (PLDs)

– user-configurable integrated circuits that can be customized (i.e., programmed) to perform the specific logic operation

Altera MAX CPLD

Altera Cyclone FPGA

Programmable logic devices (PLDs)

They contain many logic gates plus advanced sequential logic functions inside a single package.

This internal digital logic is not configured to perform any particular function.

PLD software is used for programming the logic that needs to be implemented.

The compiled program is then used as an input to a programming process that electronically alters the internal PLD connections (synthesizes) to make it function specifically as required.

What can PLDs be used for

– prototyping ASIC designs

– the physical implementation of new algorithms

– communications devices and software-defined radios

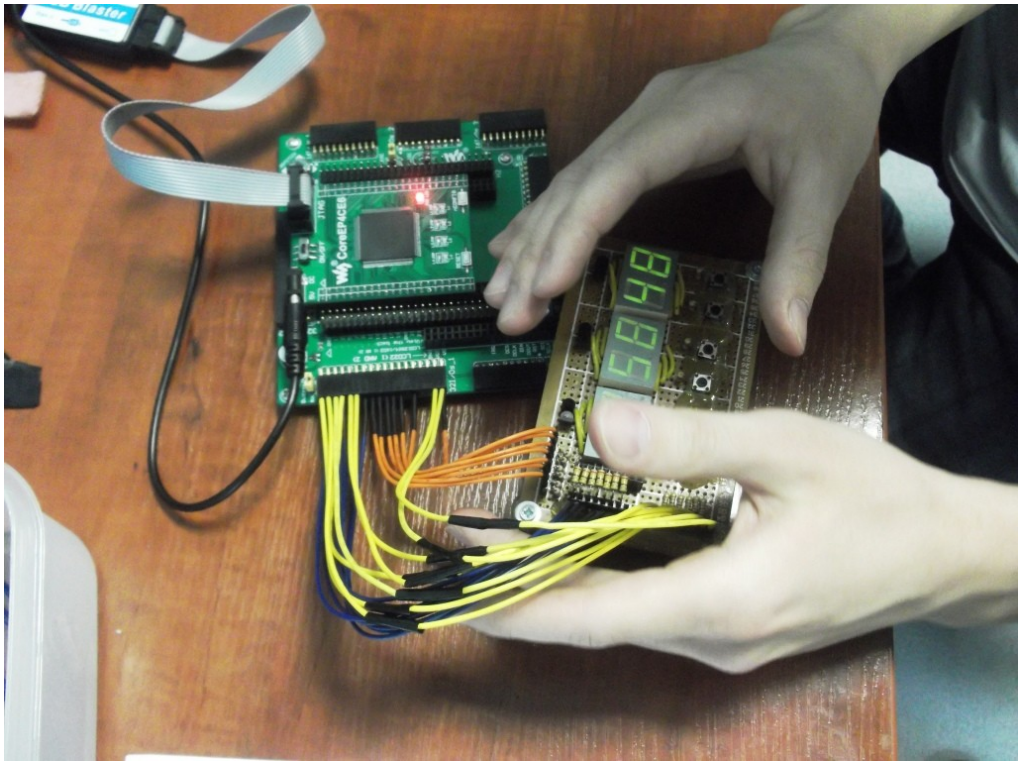
– radars, images, and other digital signal processing applications

– system-on-chip components that contain both hardware and software elements

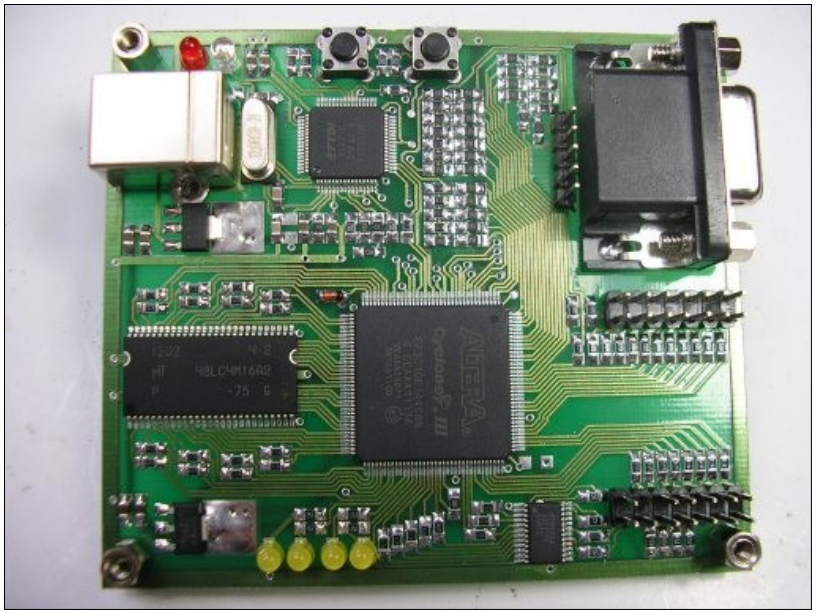
Various companies are currently building huge FPGA-based reconfigurable computing engines for tasks ranging from hardware simulation to cryptography analysis to discovering new drugs.

FPGA

Altera Cyclone IV

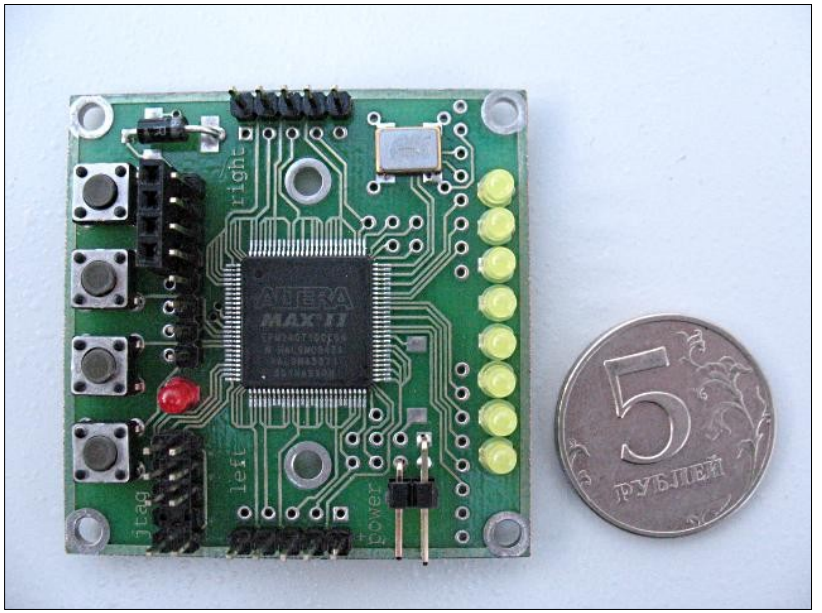


Altera Cyclone III



CPLD

Altera
MAX II



http://www.altera.com/literature/lit-cyc3.jsp

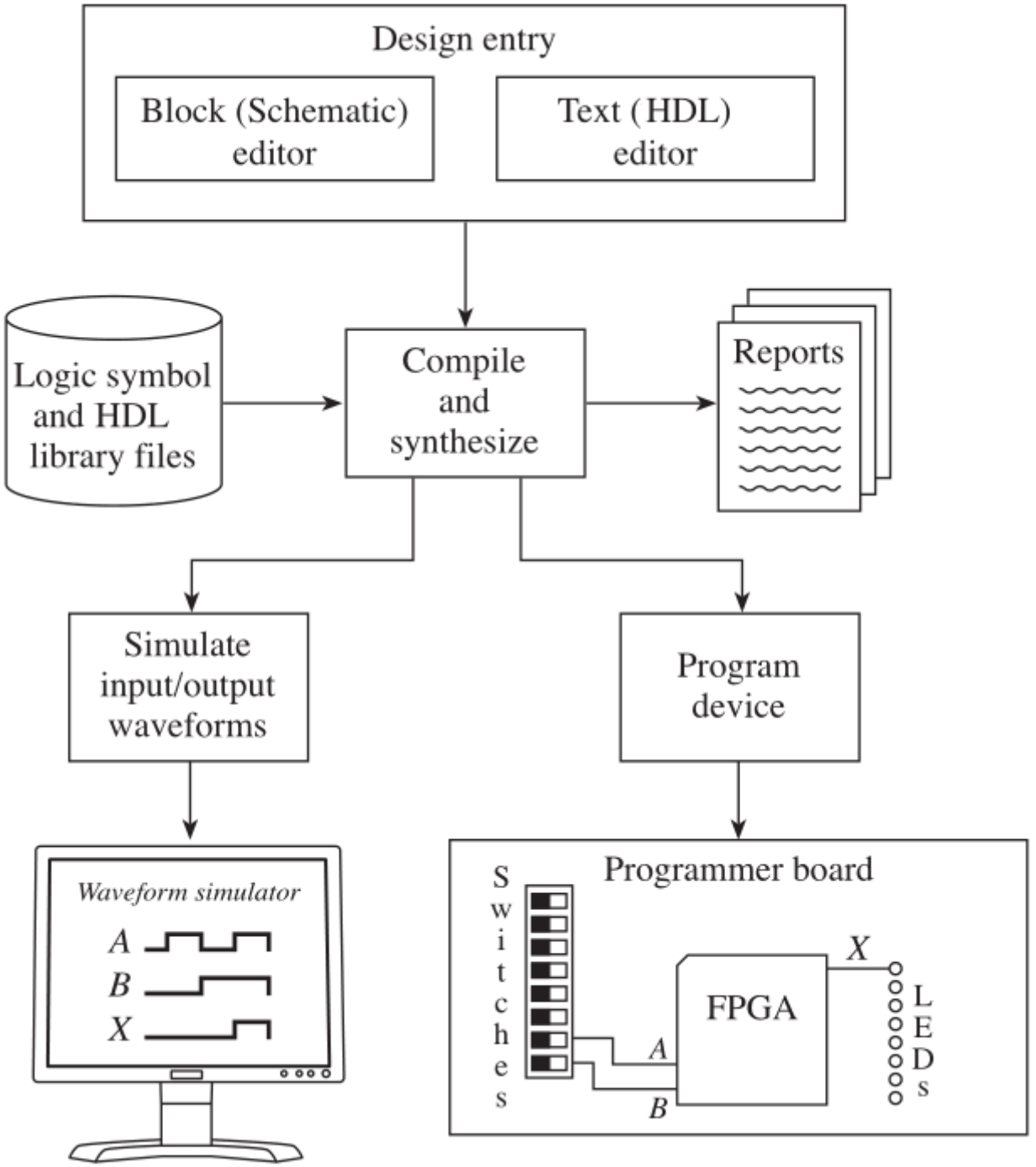
http://www.altera.com/literature/lit-max2.jsp

FPGA design flow

HDL - hardware description language

HDLs:

- VHDL
- Verilog
- AHPL
- Bluespec



Software

Icarus Verilog

http://iverilog.icarus.com/

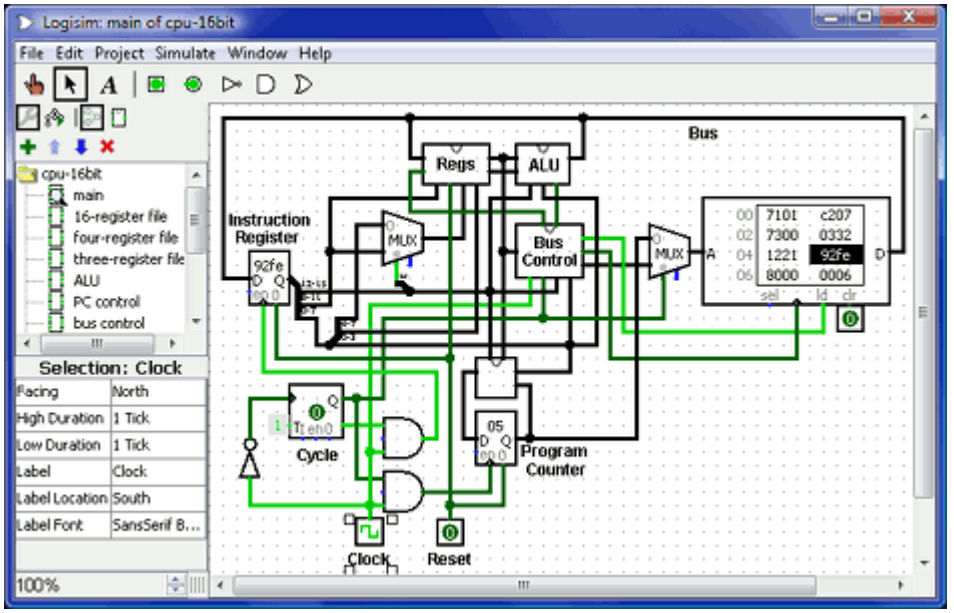
Icarus Verilog for Windows: http://bleyer.org/icarus/

Icarus Verilog is a Verilog simulation and synthesis tool.

http://gtkwave.sourceforge.net/

GTKWave is a wave viewer

Software



Logisim

http://www.cburch.com/logisim/

– an educational tool for designing and simulating digital logic circuits

References

Books

1) David Harris, Sarah Harris – Digital Design and Computer Architecture

2) William Kleitz – Digital electronics. A practical approach with VHDL

3) Clive Maxfield – Bebop to the Boolean Boogie. An Unconventional Guide to Electronics

4) Roger Woods, John McAllister, Gaye Lightbody, Ying Yi – FPGA-based implementation of signal processing systems

Web

http://twanclik.free.fr/electricity/electronic/pdfdone5/Digital.Logic.And.Microprocessor.Design.With.VHDL.pdf

http://www.fpga4fun.com/

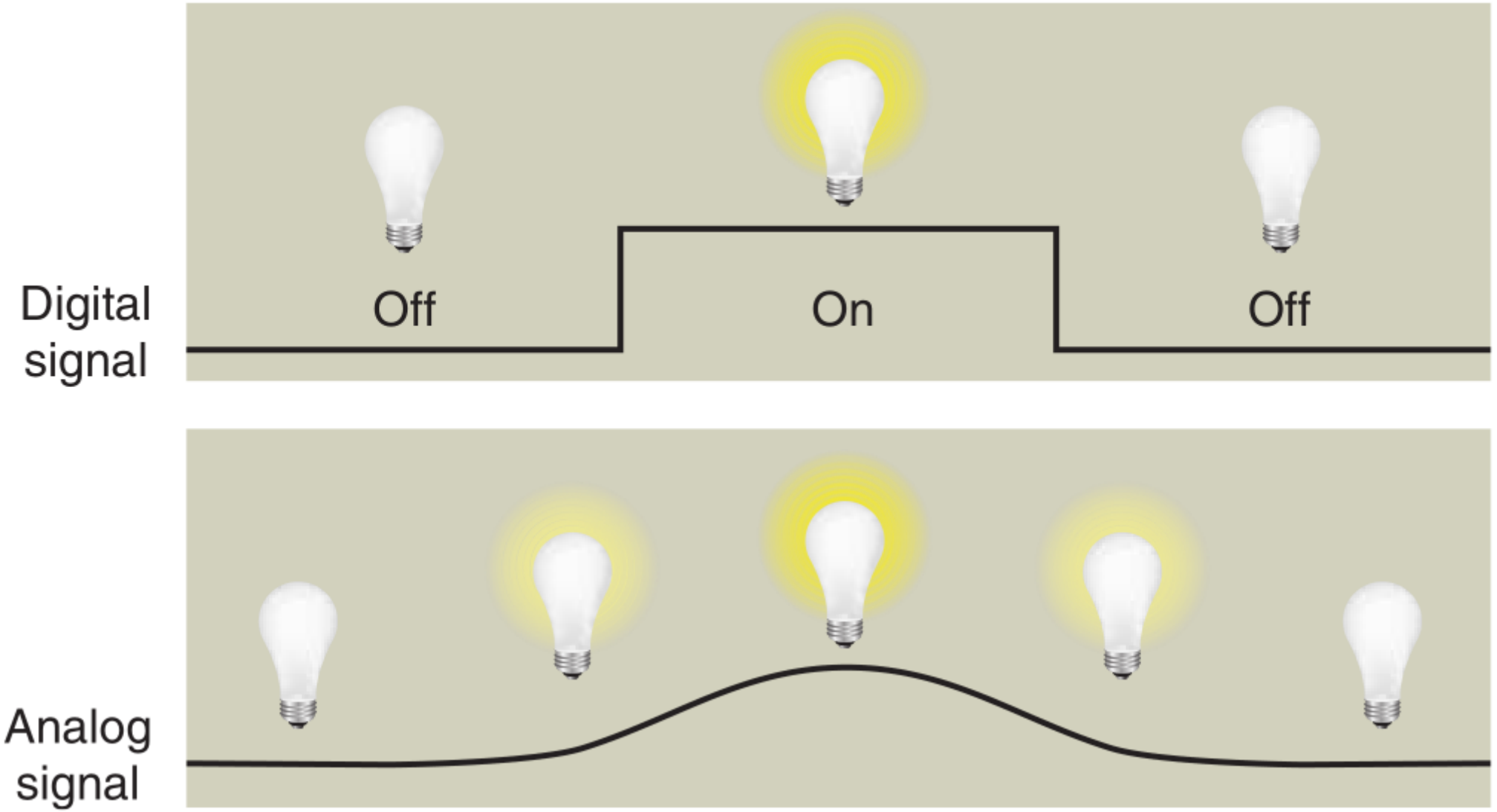
https://www.altera.com/support.html

http://www.terasic.com.tw/en/

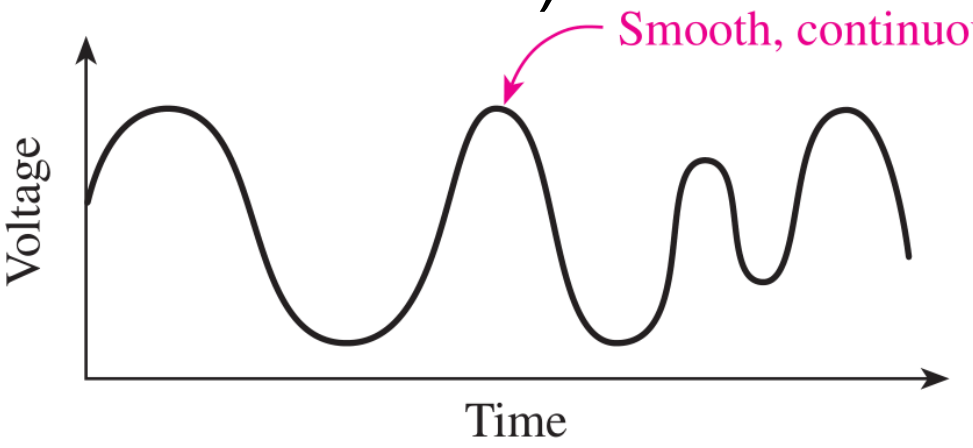
http://www.learnabout-electronics.org

https://marsohod.org/ (in Russian)

Analog versus digital



Analog → continuously varying magnitudes (infinite amount of them)

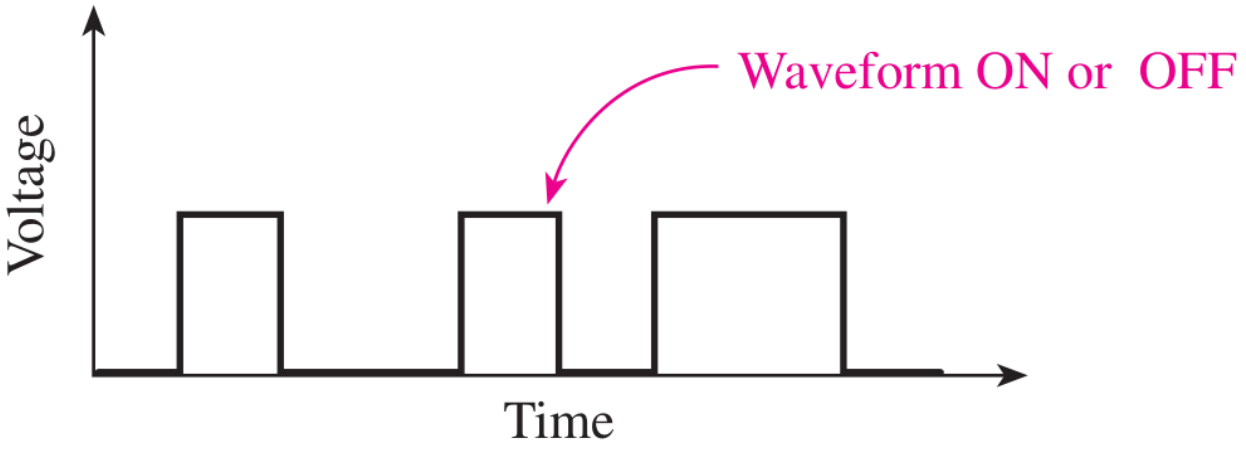


Physical quantities in our world are analog

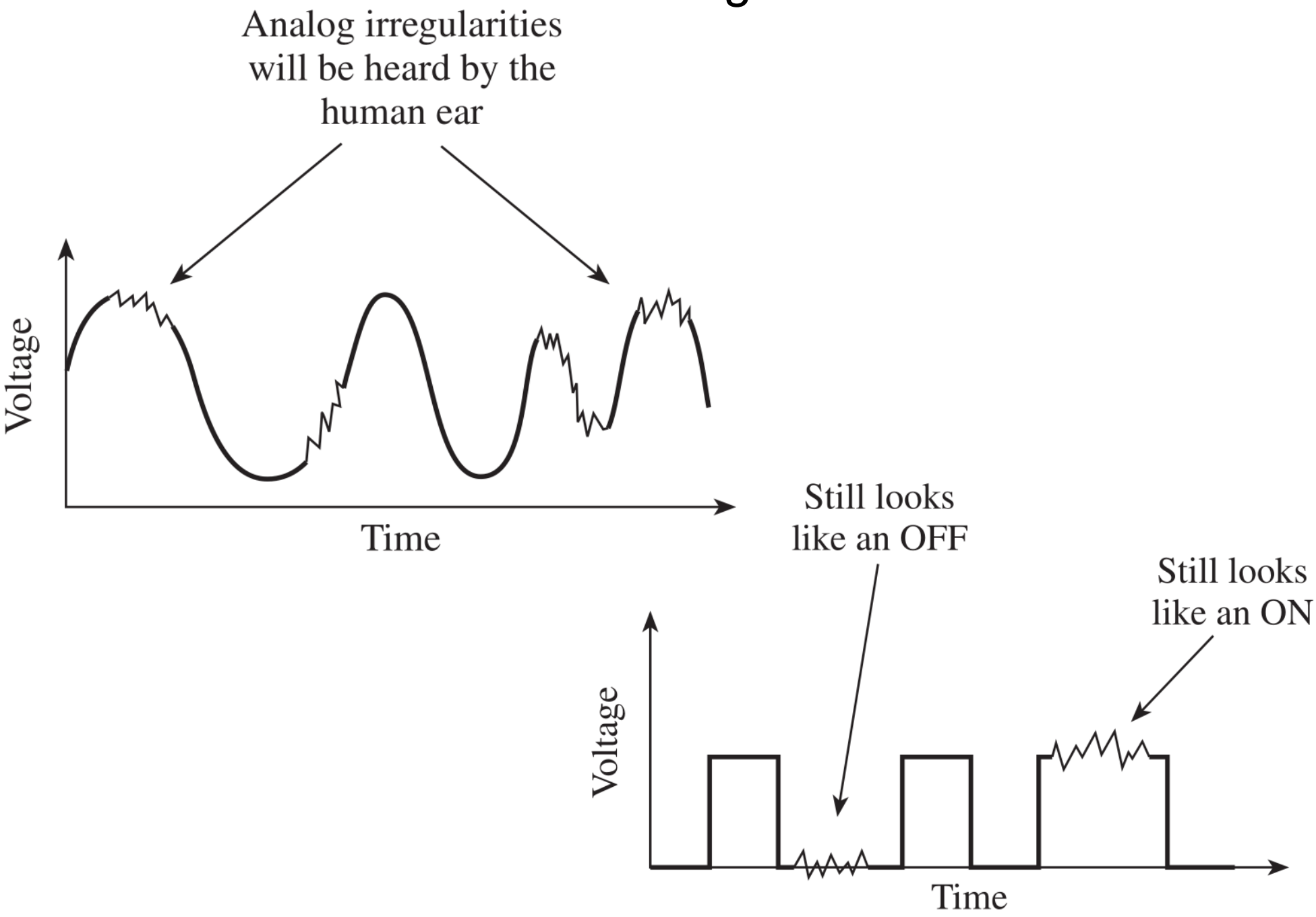
- temperature
- pressure
- velocity, and position

Digital → discrete signals, e.g. ON and OFF states,

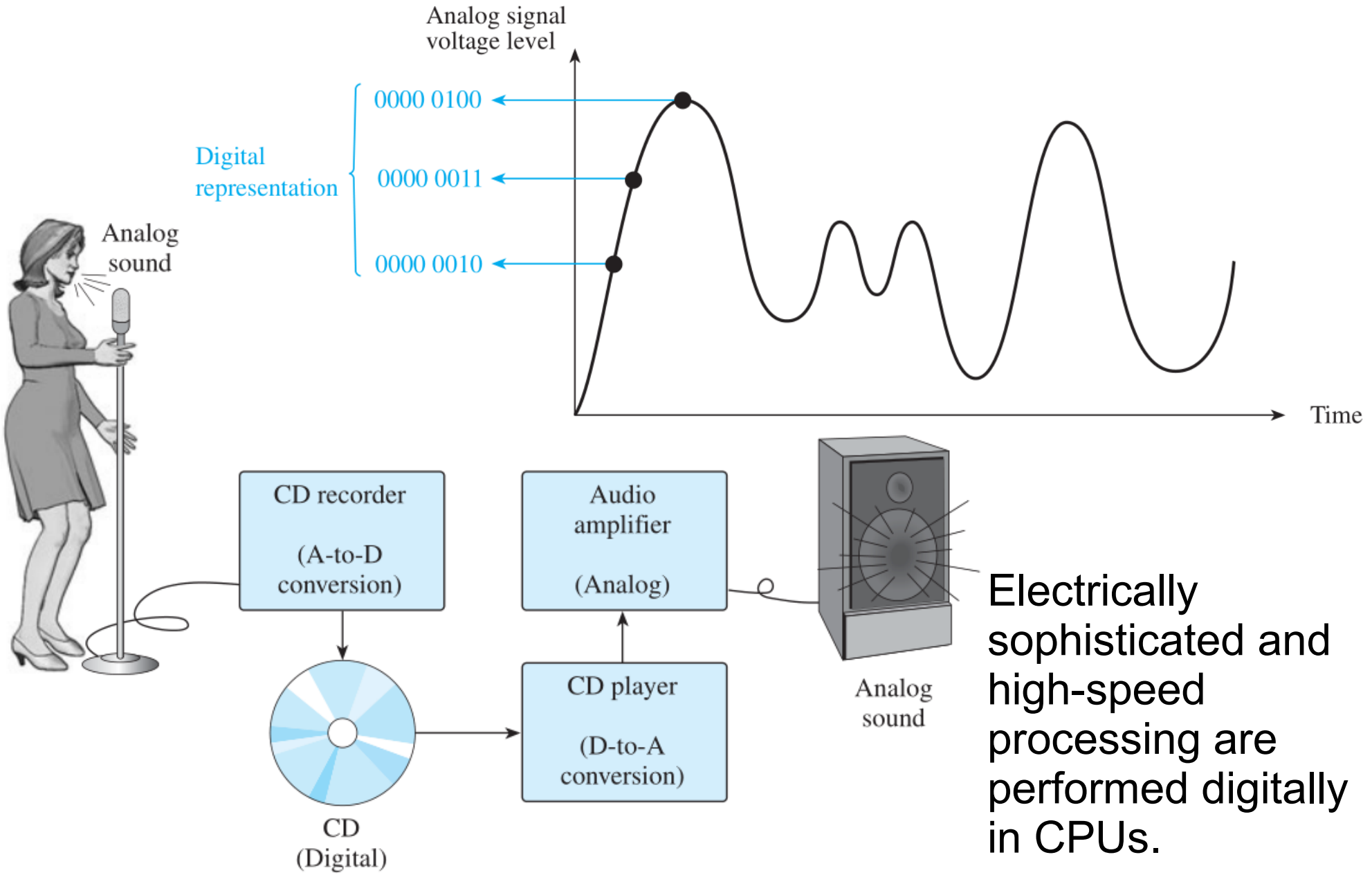
Instead of dealing with the infinite span and intervals of analog voltage levels, we use ON or OFF voltages.



Digitization eliminates noise



Analog-to-digital and digital-to analog conversion (ADC & DAC)

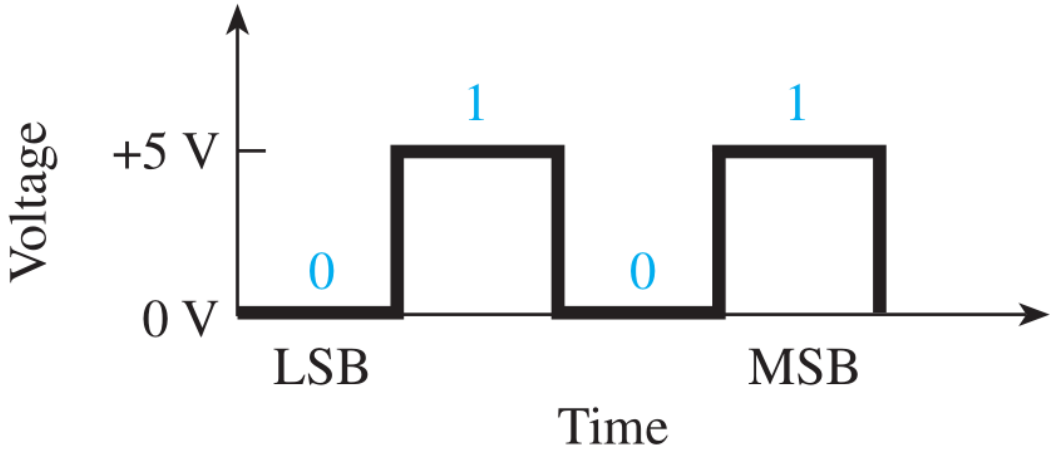


Electrically sophisticated and high-speed processing are performed digitally in CPUs.

Digital signals

A digital signal is made up of a series of 1s and 0s that represent numbers, letters, symbols, or control signals.

A timing diagram:



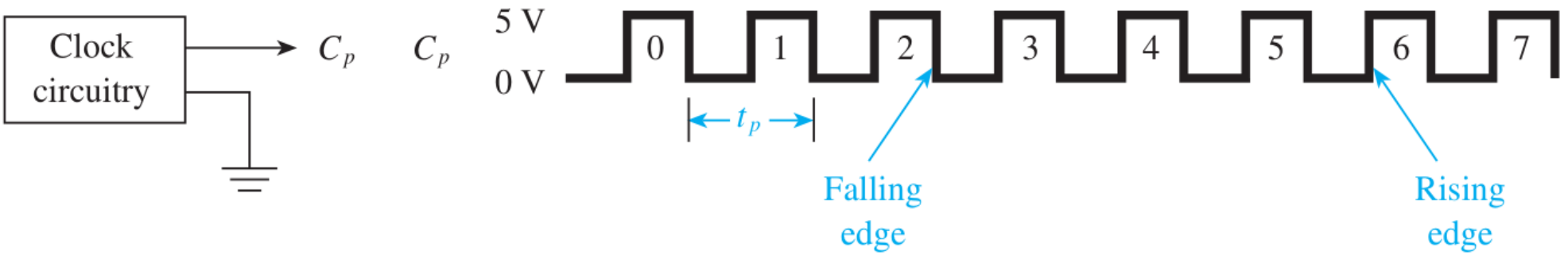
Digital systems respond to the digital state (0 or 1), not the actual voltage levels.

If the voltage levels were not exactly 0V and +5V the digital circuitry would still interpret it as the 0 state and 1 state and respond identically.

Clock waveform timing

Most digital signals require precise timing.

Special clock and timing circuits are used to produce clock waveforms to trigger the digital signals at precise intervals.



$$t_p = \frac{1}{f}$$

The frequency of the clock waveform is the reciprocal of the clock period

Transmission rate

Digital communications concerns itself with the transmission of bits (1s and 0s)

The rate, or frequency, at which they are transmitted is given in bits-per-second (bps)

Standard transmission speed for a PC's serial port (labeled COM on Windows-based machines) is 115 kbps

The USB version 1.1 standard called for 12 Mbps transmission speeds

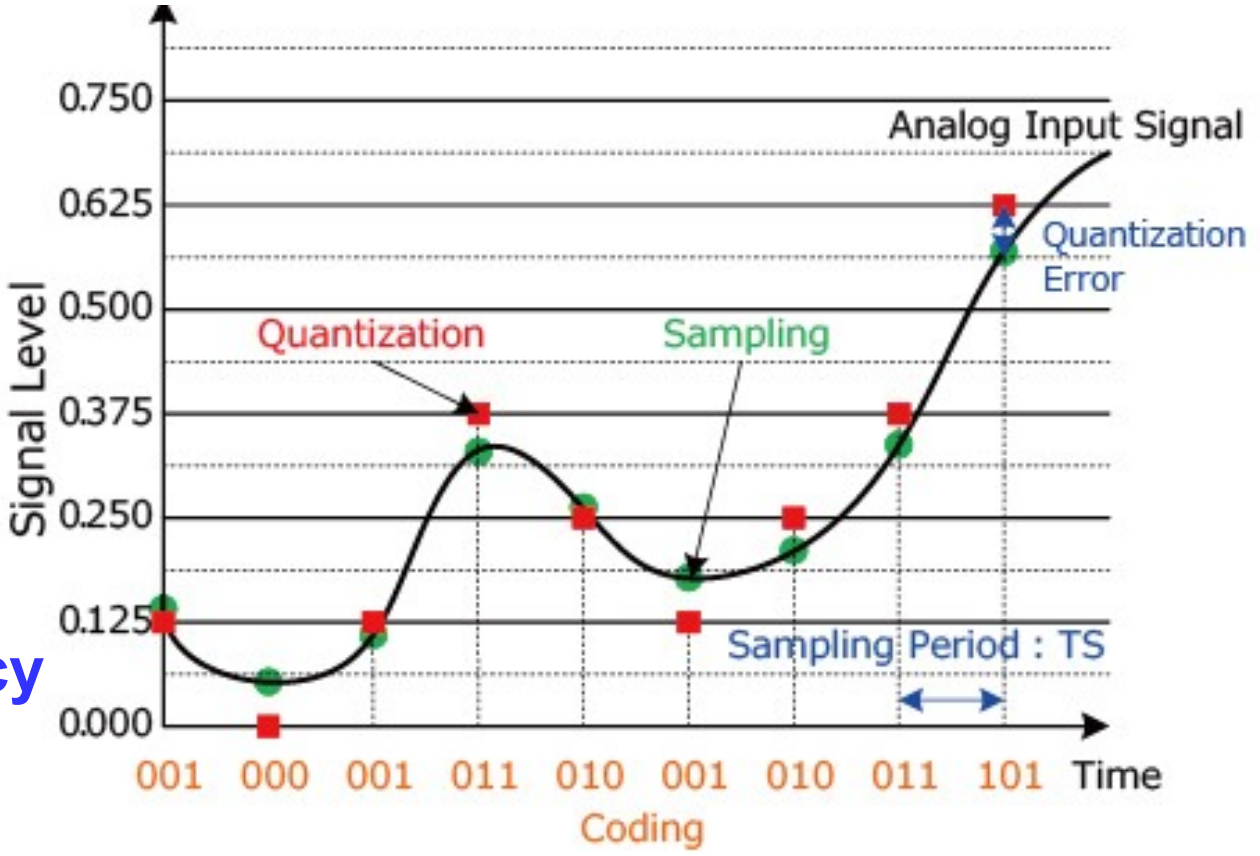
USB 2.0 specifies 480 Mbps and version 3.0 can transmit at speeds up to 5 Gbps

Analog to digital conversion involves a series of steps, including **sampling**, **quantization**, and **coding**.

Sampling is taking amplitude values of the analog signal at discrete time intervals $T_s = 1/F_s$

T_s – **sampling period**

F_s – **sampling frequency**

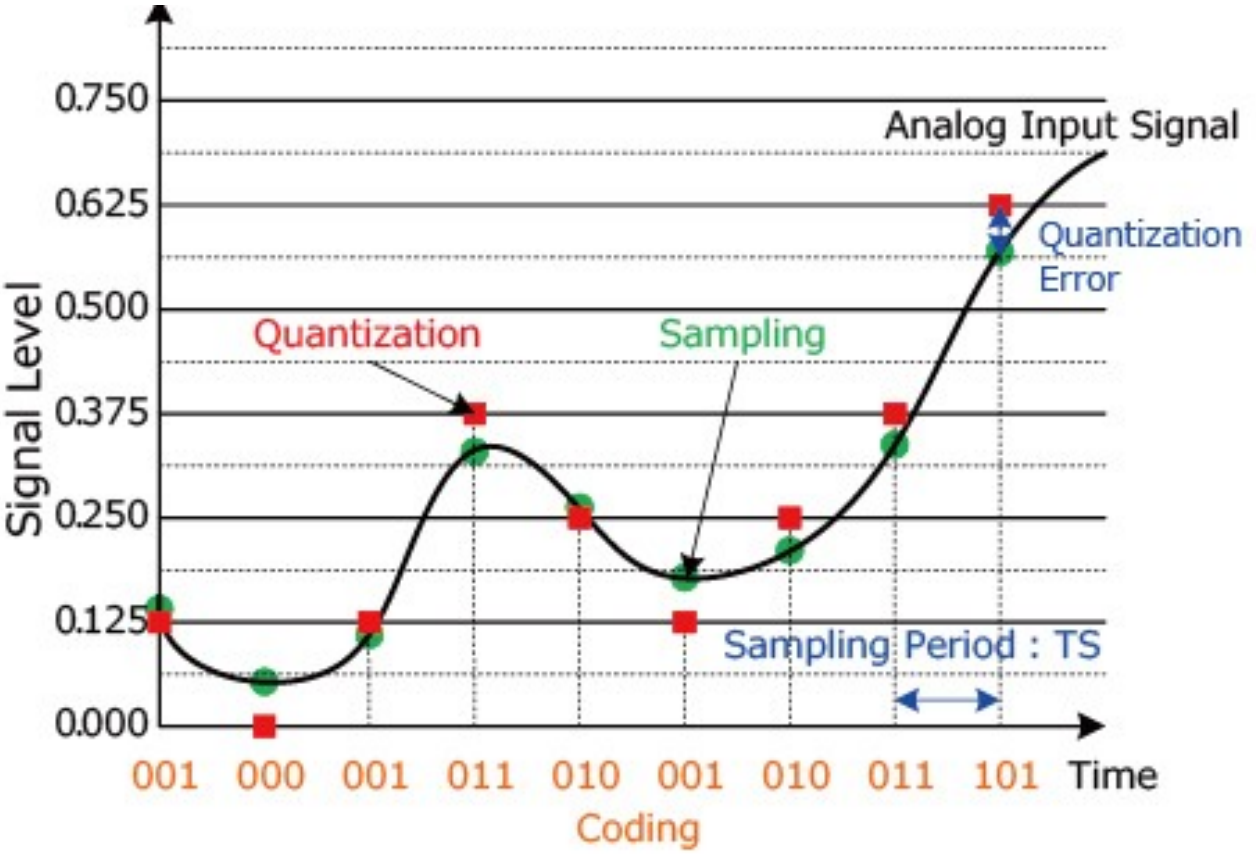


Quantization – assigning a numerical value to each sampled amplitude value from a range of possible values covering the entire amplitude range.

Coding – representation of the amplitude values by binary numbers

The signal is normally **voltage**.

These voltage values may represent, that is be converted from, some other unit for example fluid level or temperature by a sensor.



Two conversions take place in the system:

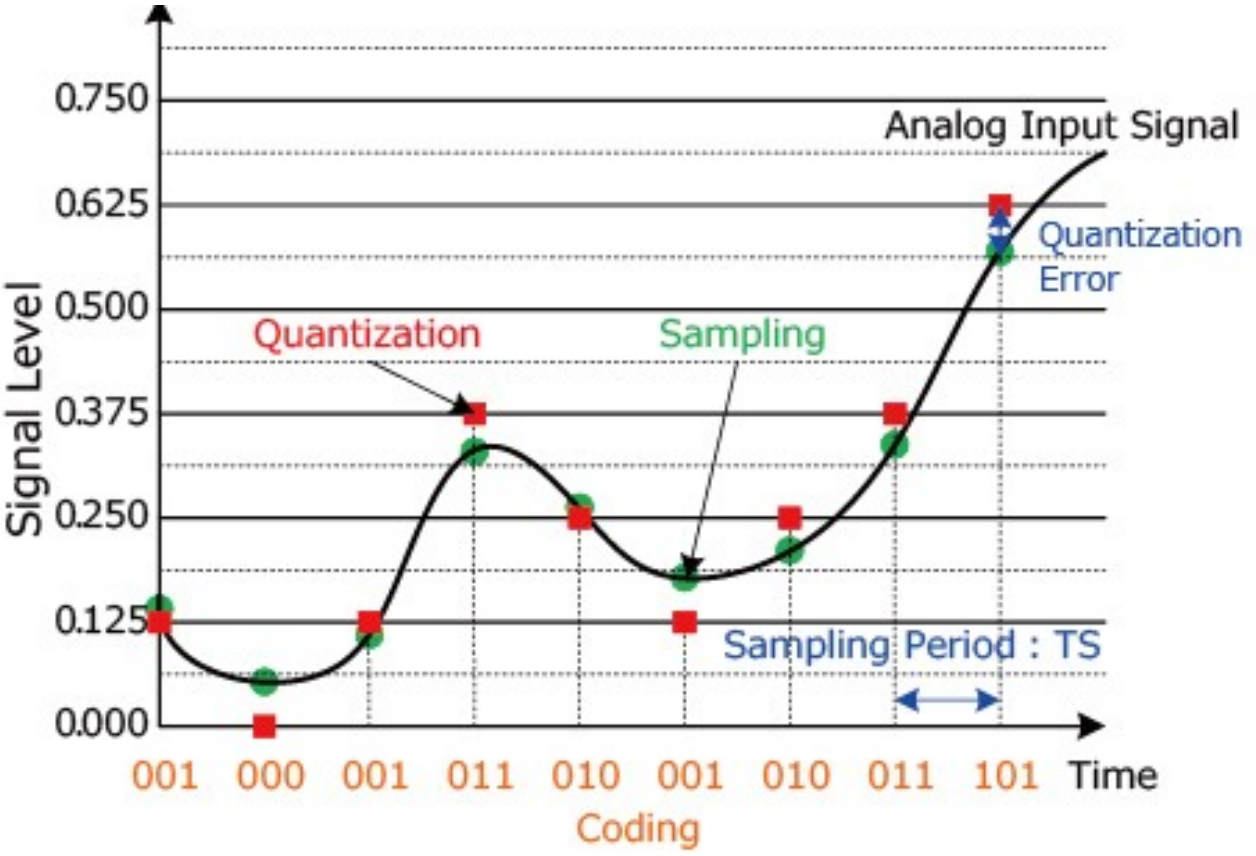
Physical quantity → voltage → binary number

If we convert an analog signal to digital with an *n*-bit analog-to-digital converter, then the number of digital values we can represent is $N = 2^n$

The step in volts between two consecutive digital values is called the **resolution** of the converter.

It is equal to

$$\Delta = \frac{U_{max} - U_{min}}{N}$$



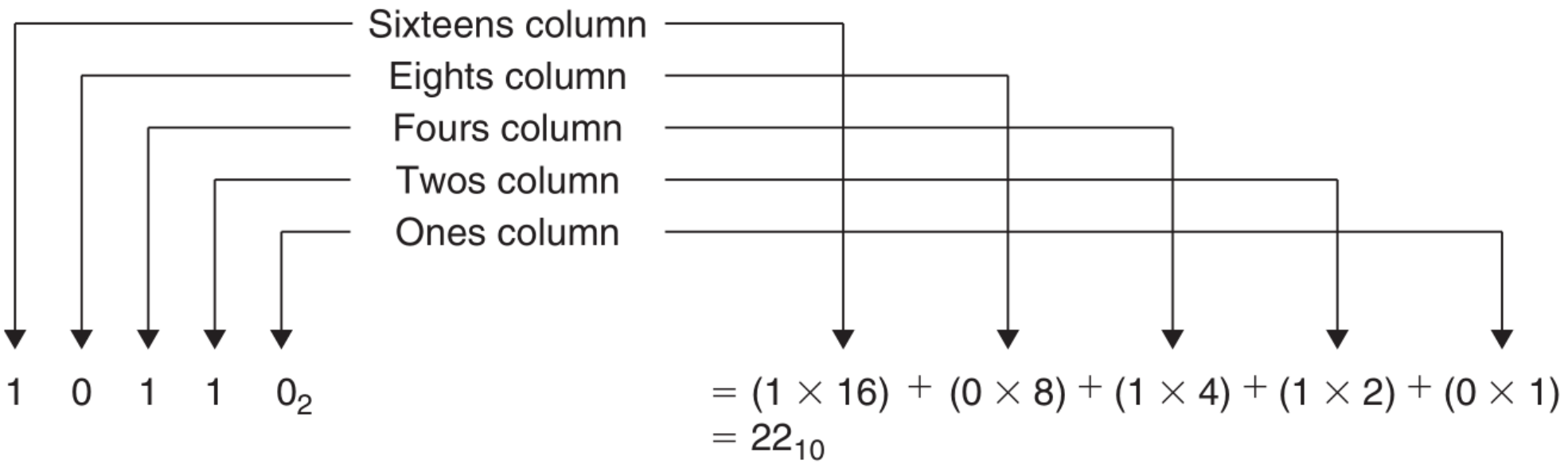
U_{max}, U_{min} – the largest and the smallest signal values

The largest **quantization error** is $\Delta/2$

Binary (base-2 or radix-2)

Digital systems are constructed out of logic gates that can only represent two states

Thus, computers are obliged to make use of a number system comprising only two digits



$$10110 = 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0 = 22_{10}$$

Other positional numeral systems

Ternary (base-3)

$$111112_3 = 1\times3^5 + 1\times3^4 + 1\times3^3 + 1\times3^2 + 1\times3^1 + 2\times3^0 = 365$$

quaternary (base-4)

duodecimal or dozenal (base-12)

hexadecimal (base-16)

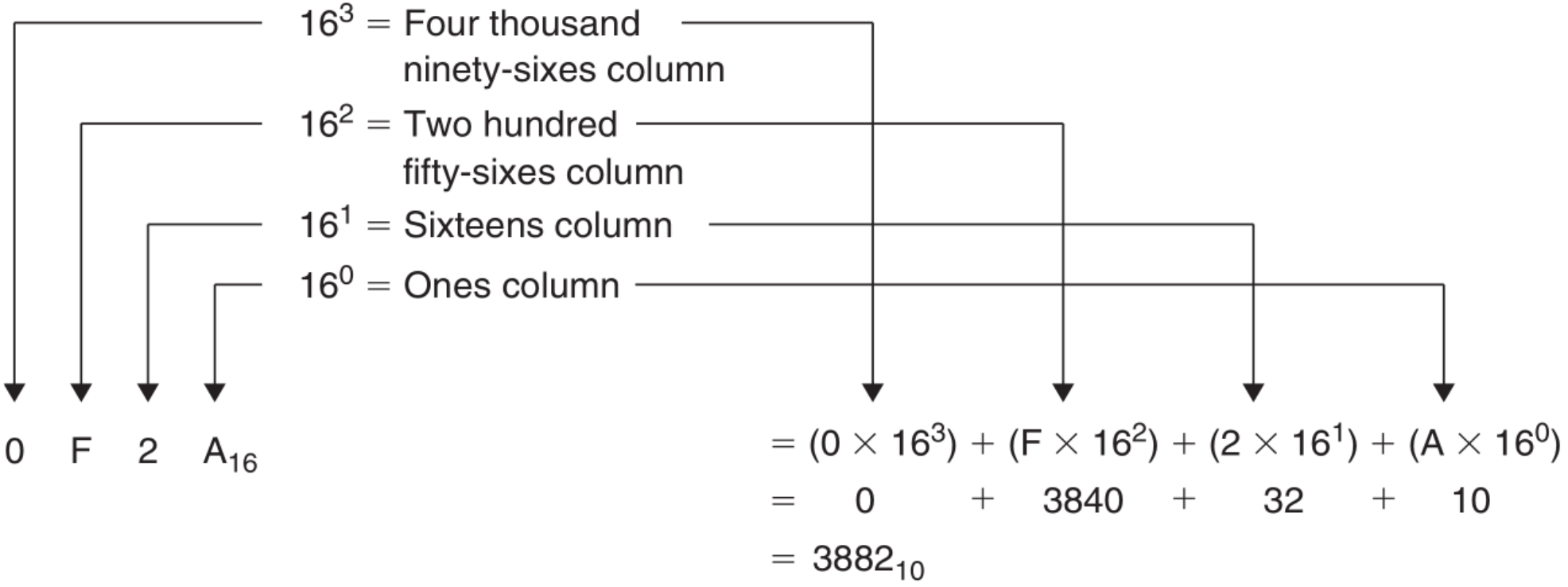
senary or heximal (base-6)

vigesimal (base-20)

octal (base-8)

sexagesimal (base-60)

Hexadecimal (base-16)

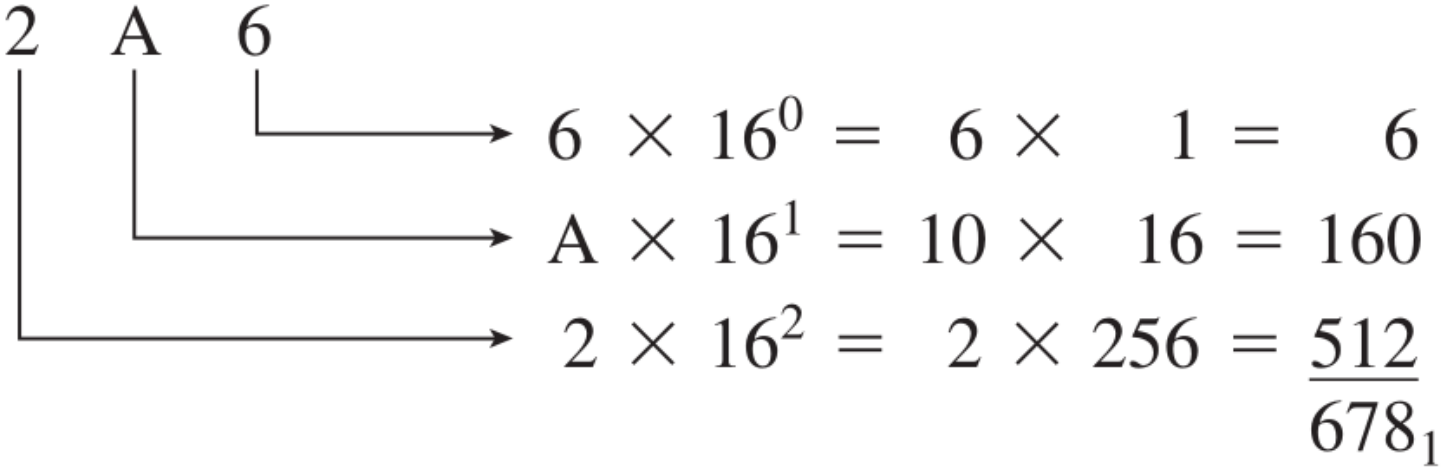


The hex system is a method of grouping bits to simplify entering and reading the instructions or data present in digital computer systems

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	0001 0000	1 0
17	0001 0001	1 1
18	0001 0010	1 2
19	0001 0011	1 3
20	0001 0100	1 4

Convert 01101101 to hex $\underbrace{0\ 1\ 1\ 0}_{6}\ \underbrace{1\ 1\ 0\ 1}_{D}{}_2 = 6D_{16}$

Convert A9 to binary $\overbrace{1\ 0\ 1\ 0}^{A}\ \overbrace{1\ 0\ 0\ 1}^{9} = 10101001_2$



$$\overbrace{0010}^{2}\ \overbrace{1010}^{A}\ \overbrace{0110}^{6} = 2 + 4 + 32 + 128 + 512 = 678_{10}$$

Convert 151 to hex

$$151 \div 16 = 9 \quad \text{remainder} \quad 7 \quad \text{(LSD)}$$
$$9 \div 16 = 0 \quad \text{remainder} \quad 9 \quad \text{(MSD)}$$
$$151_{10} = 97_{16}$$

Convert 498 to hex

$$498 \div 16 = 31 \quad \text{remainder} \quad 2 \quad \text{(LSD)}$$
$$31 \div 16 = 1 \quad \text{remainder} \quad 15 \quad (= \text{F})$$
$$1 \div 16 = 0 \quad \text{remainder} \quad 1 \quad \text{(MSD)}$$
$$498_{10} = 1 \quad \text{F} \quad 2_{16}$$

Octal (base-8)

Decimal	Binary	Octal
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	10
9	1001	11
10	1010	12

The octal numbering system is a method of grouping binary numbers in groups of three.

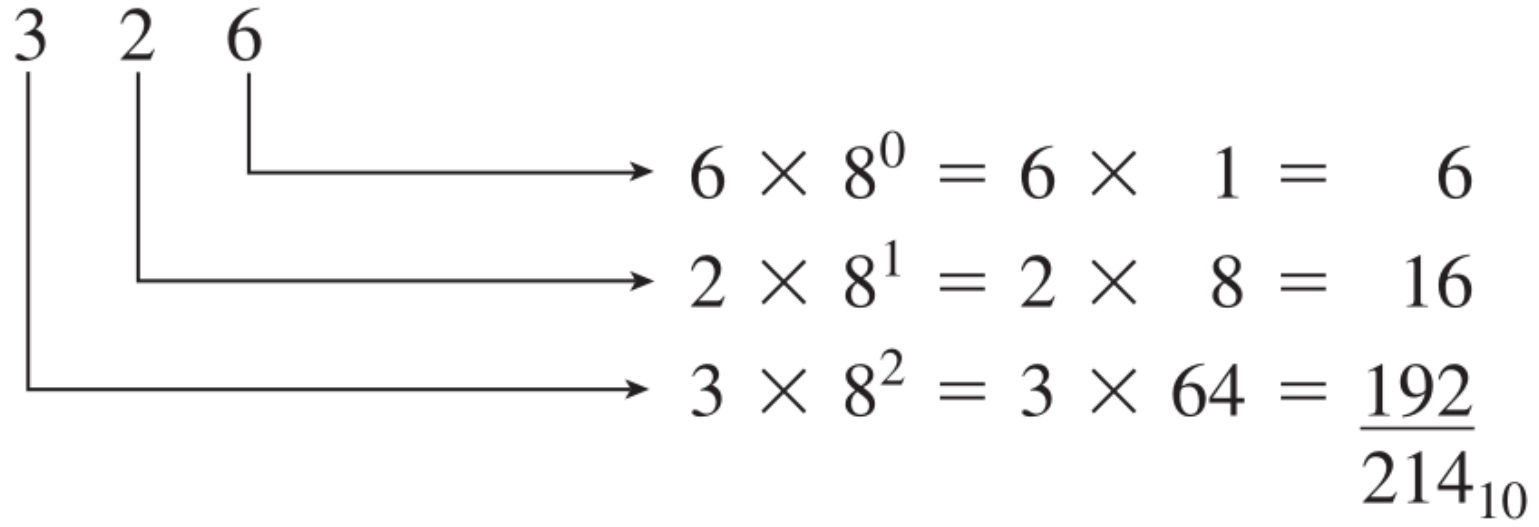
Convert 011101 to octal

$$\underbrace{0\ 1\ 1}_{3}\quad\underbrace{1\ 0\ 1}_{5} = 35_8$$

Convert 624 octal to binary

$$\overbrace{1\ 1\ 0}^{6}\quad\overbrace{0\ 1\ 0}^{2}\quad\overbrace{1\ 0\ 0}^{4} = 1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0_2$$

Convert 326 octal to decimal



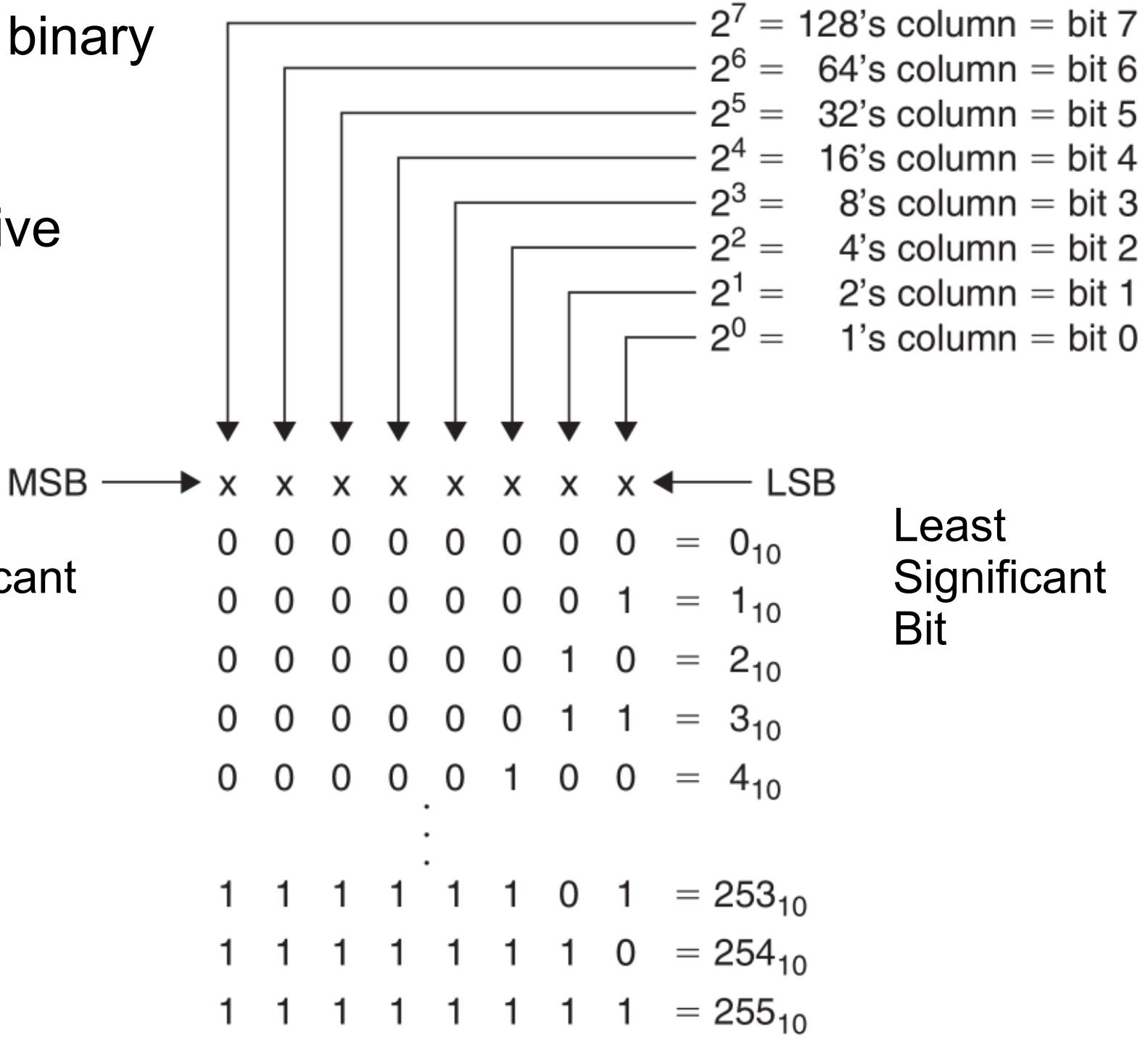
Convert 486 decimal to octal

$$\left.\begin{aligned} 486 \div 8 &= 60 \quad \text{remainder } 6 \\ 60 \div 8 &= 7 \quad \text{remainder } 4 \\ 7 \div 8 &= 0 \quad \text{remainder } 7 \end{aligned}\right\} \; 746_8$$

$$486_{10} = 746_8$$

Unsigned binary numbers

only positive values



Most Significant Bit

Least Significant Bit

Binary addition

```
  0 0 1 1 1 0 0 1
+ 0 0 0 1 1 0 1 0
= ---------------
              1
```

(a) Bit 0, $1 + 0 = 1_2$

```
  0 0 1 1 1 0 0 1
+ 0 0 0 1 1 0 1 0
= ---------------
            1 1
```

(b) Bit 1, $0 + 1 = 1_2$

```
  0 0 1 1 1 0 0 1
+ 0 0 0 1 1 0 1 0
= ---------------
          0 1 1
```

(c) Bit 2, $0 + 0 = 0_2$

```
  0 0 1 1 1 0 0 1
+ 0 0 0 1 1 0 1 0
= ---------------
        0 0 1 1
```

(d) Bit 3, $1 + 1 = 10_2$

```
  0 0 1 1 1 0 0 1
+ 0 0 1 1 1 0 1 0
= ---------------
      1 0 0 1 1
```

(e) Bit 4, $1 + 1 + \text{carry\_in} = 11_2$

```
  0 0 1 1 1 0 0 1
+ 0 1 1 1 1 0 1 0
= ---------------
      0 1 0 0 1 1
```

(f) Bit 5, $1 + 0 + \text{carry\_in} = 10_2$

```
  0 0 1 1 1 0 0 1
+ 0 1 1 1 1 0 1 0
= ---------------
    1 0 1 0 0 1 1
```

(g) Bit 6, $0 + 0 + \text{carry\_in} = 1_2$

```
  0 0 1 1 1 0 0 1
+ 0 1 1 1 1 0 1 0
= ---------------
  0 1 0 1 0 0 1 1
```

(h) Bit 7, $0 + 0 = 0_2$

57_{10}

$+ \; 26_{10}$

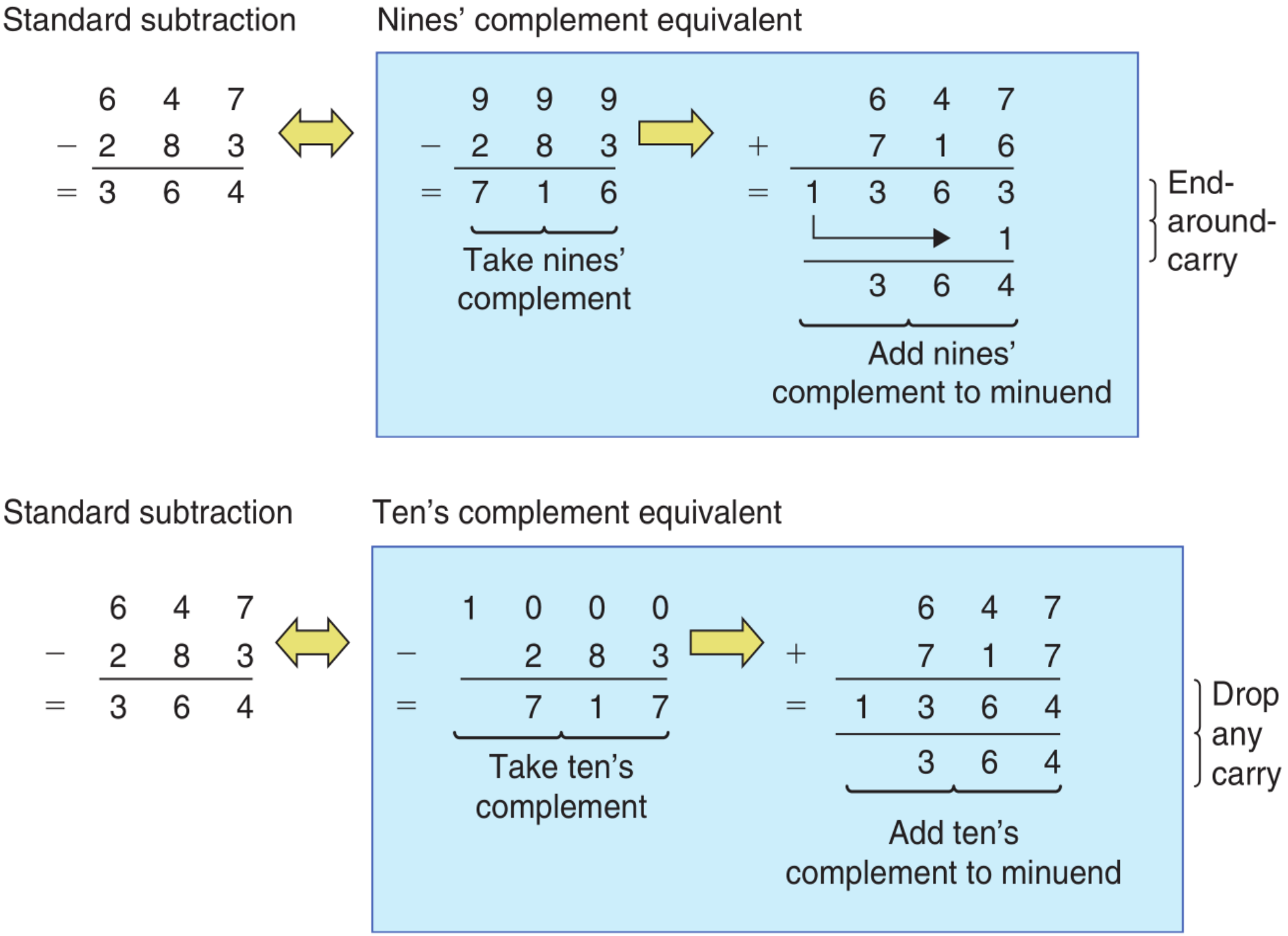
$= 83_{10}$

Complements

There are 2 forms of **complement** associated with every number system

– the radix complement

– the diminished radix complement


Standard subtraction
Nines' complement equivalent
6 4 7
− 2 8 3
= 3 6 4
9 9 9
− 2 8 3
= 7 1 6
Take nines' complement
+ 6 4 7
7 1 6
= 1 3 6 3
1
3 6 4
Add nines' complement to minuend
End-around-carry
Standard subtraction
Ten's complement equivalent
6 4 7
− 2 8 3
= 3 6 4
1 0 0 0
− 2 8 3
= 7 1 7
Take ten's complement
+ 6 4 7
7 1 7
= 1 3 6 4
3 6 4
Add ten's complement to minuend
Drop any carry

Ones' complement binary subtraction

Standard subtraction

$$\begin{array}{rcccccccc} & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ - & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ \hline = & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{array}$$

$\Leftrightarrow$ $57_{10} - 30_{10} = 27_{10}$

$\Updownarrow$

Ones' complement equivalent

$$\begin{array}{rcccccccc} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ - & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ \hline = & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}$$

Take ones' complement

$\Rightarrow$

$$\begin{array}{rcccccccc} & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ + & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ \hline & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ & & & & & & & & & 1 \\ \hline & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{array}$$

End-around-carry

Add ones' complement to minuend

Two's complement binary subtraction

Standard subtraction

$$\begin{array}{rcccccccc} & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ - & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ \hline = & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{array} \quad \Leftrightarrow \quad 57_{10} - 30_{10} = 27_{10}$$

Two's complement equivalent

$$\begin{array}{rcccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ - & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ \hline = & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}$$

Take two's complement

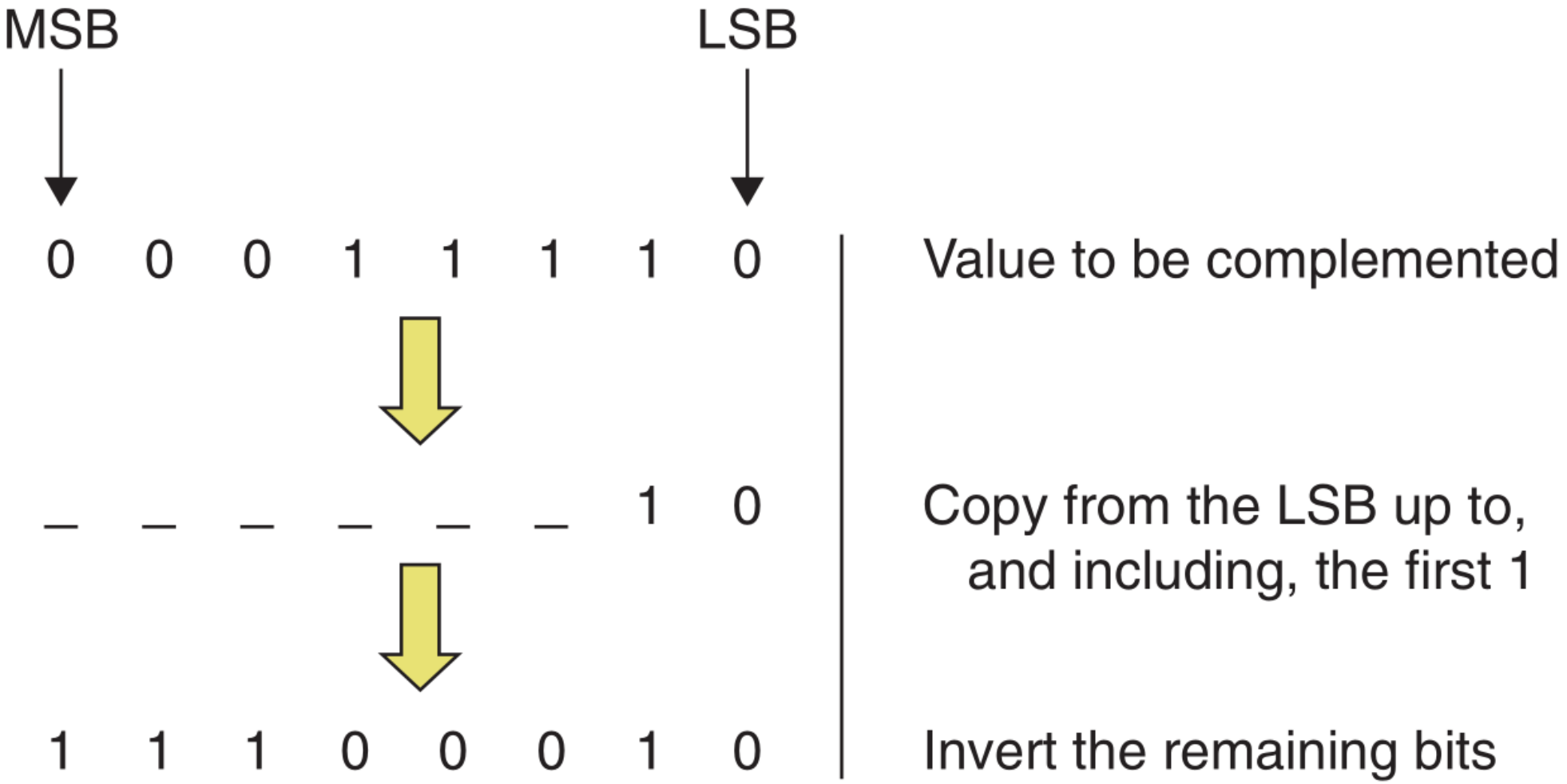
$$\Rightarrow \quad \begin{array}{rcccccccc} & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ + & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ \hline & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{array}$$

Drop any carry

Add two's complement to minuend

A shortcut for generating a two's complement:

- Start with the LSB of the value to be complemented
- Copy each bit up to and including the first 1
- Invert the remaining bits



Another shortcut for generating a two's complement:

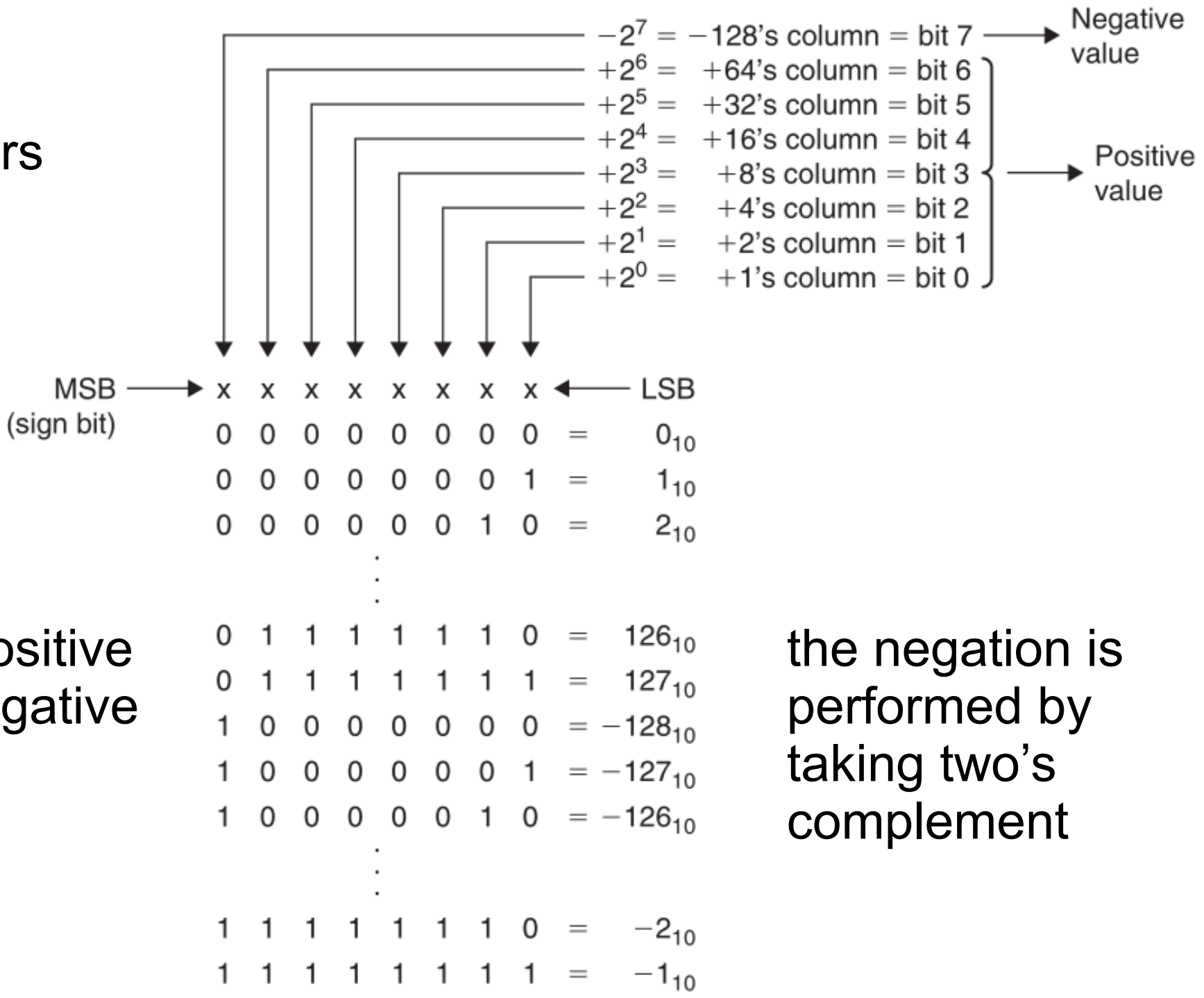
1) flip all the bits

$1 \rightarrow 0$

$0 \rightarrow 1$

2) add 1

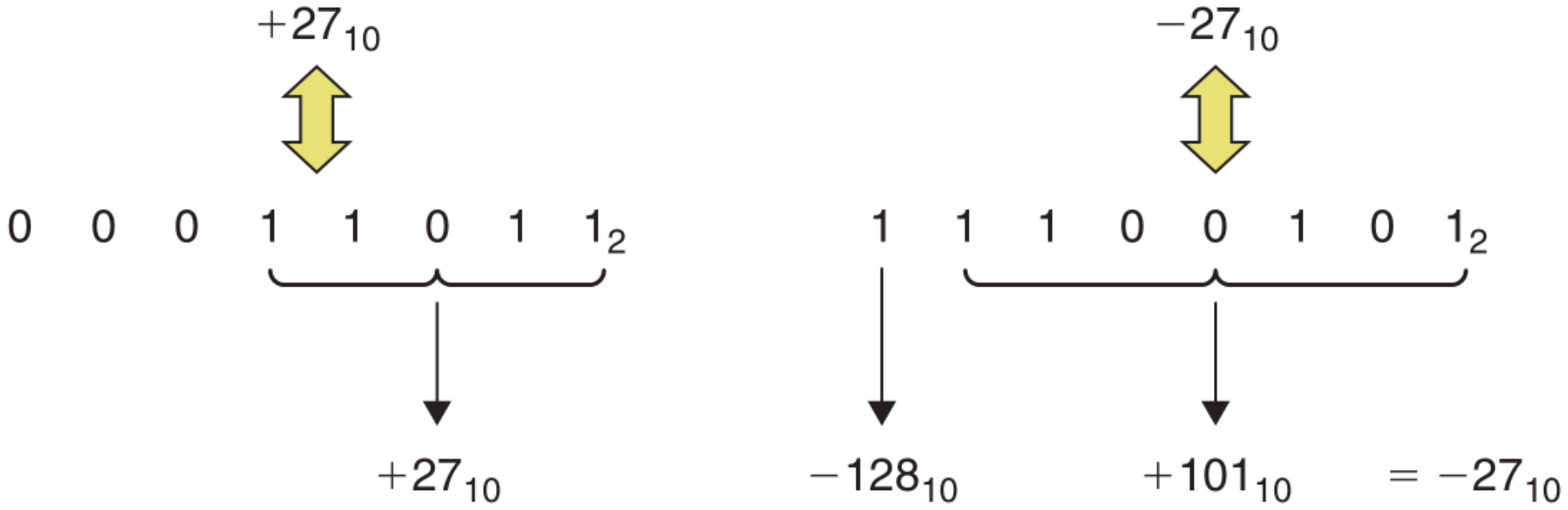
Signed binary numbers



both positive and negative values

the negation is performed by taking two's complement

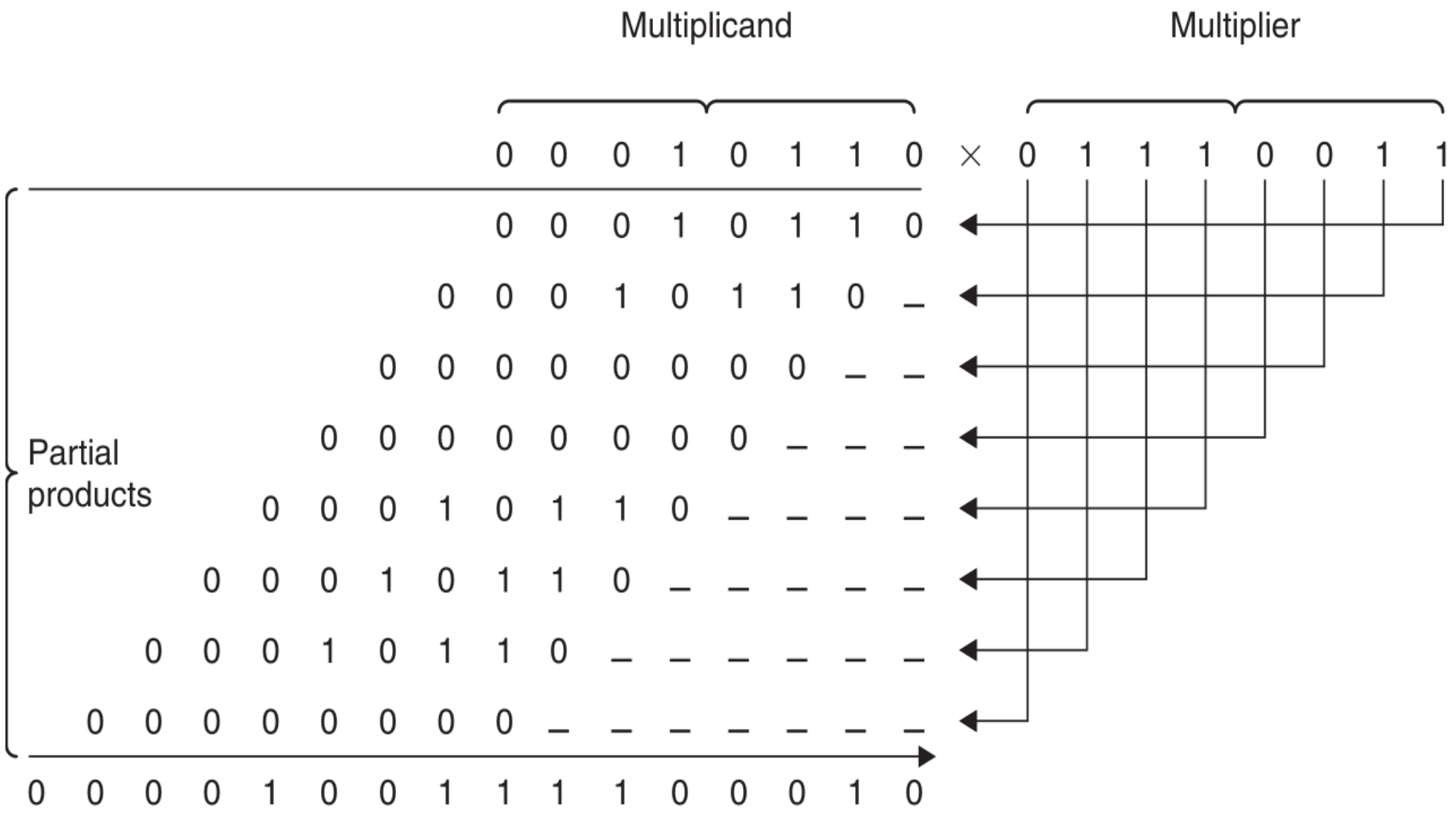
The negation is performed by taking two's complement



Decimal sign-magnitude		Signed binary
57		0 0 1 1 1 0 0 1
+ 30	⇔	+ 0 0 0 1 1 1 1 0
= 87		0 1 0 1 0 1 1 1
57		0 0 1 1 1 0 0 1
+ −30	⇔	+ 1 1 1 0 0 0 1 0
= 27		0 0 0 1 1 0 1 1
−57		1 1 0 0 0 1 1 1
+ 30	⇔	+ 0 0 0 1 1 1 1 0
= −27		1 1 1 0 0 1 0 1
−57		1 1 0 0 0 1 1 1
+ −30	⇔	+ 1 1 1 0 0 0 1 0
= −87		1 0 1 0 1 0 0 1

Binary multiplication via a shift-and-add technique

$$22 \times 115 = 2530$$



The shift-and-add technique works only with unsigned binary values.

This can be overcome by taking the two's complement of any negative values before feeding them into the multiplier.

If the signs of the two values are the same, both positive or both negative, then no further action need be taken.

If the signs are different, then the result returned from the multiplier must be negated by transforming it into its two's complement.

There are several ways to construct a multiplier based on the shift-and-add technique.

In one implementation, all of the partial products are generated simultaneously and then added together.

This requires a lot of logic gates, but the resulting multiplication is fast.

As another option, we can cycle round generating each of the partial products one-at-a-time and adding them into an accumulated result.

This cuts down on the number of logic gates required, but the resulting multiplication is slow.

Binary-coded-decimal (BCD)

The BCD system is used to represent each of the 10 decimal digits as a 4-bit binary code.

This code is useful for outputting to displays that are always numeric (0 to 9), such as those found in digital clocks or digital voltmeters.

Convert 496 to BCD

$$\overbrace{0100}^{4}\ \overbrace{1001}^{9}\ \overbrace{0110}^{6} = 0100\ \ 1001\ \ 0110_{\text{BCD}}$$

Decimal	Binary	Octal	Hexadecimal	BCD
0	0000	0	0	0000
1	0001	1	1	0001
2	0010	2	2	0010
3	0011	3	3	0011
4	0100	4	4	0100
5	0101	5	5	0101
6	0110	6	6	0110
7	0111	7	7	0111
8	1000	1 0	8	1000
9	1001	1 1	9	1001
10	1010	1 2	A	0001 0000
11	1011	1 3	B	0001 0001
12	1100	1 4	C	0001 0010
13	1101	1 5	D	0001 0011
14	1110	1 6	E	0001 0100
15	1111	1 7	F	0001 0101
16	0001 0000	2 0	1 0	0001 0110
17	0001 0001	2 1	1 1	0001 0111
18	0001 0010	2 2	1 2	0001 1000
19	0001 0011	2 3	1 3	0001 1001
20	0001 0100	2 4	1 4	0010 0000

Hexadecimal addition

Add the two hex digits by working with their decimal equivalents

If the decimal sum is less than 16, write down the hex equivalent

If the decimal sum is 16 or more, subtract 16, write down the hex result in that column, and carry 1 to the next-more-significant column

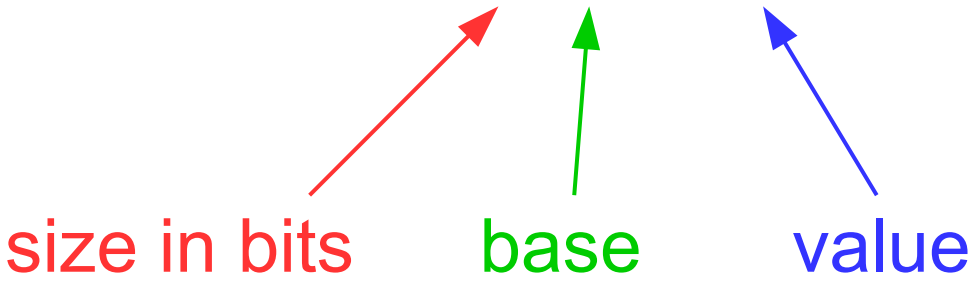
$$\begin{array}{rr} & A\,7\,C\,5 \\ + & 2\,D\,A\,8 \\ \hline & D\,5\,6\,D \end{array}$$

Verilog numbers

Numbers	Bits	Base	Val	Stored
3'b101	3	2	5	101
'b11	?	2	3	000 ... 0011
8'b11	8	2	3	00000011
8'b1010_1011	8	2	171	10101011
3'd6	3	10	6	110
6'o42	6	8	34	100010
8'hAB	8	16	171	10101011
42	?	10	42	00 ... 0101010

Verilog numbers

Format: N'Bvalue



Example:

9'h25

9-bit 25_{16} =

0 0010 0101

'b – binary (base 2)

'o – octal (base 8)

'd – decimal (base 10)

'h – hexadecimal (base 16)

If the size is not given, the number is assumed to have as many bits as the expression in which it is being used.

It is better practice to explicitly give the size.

b[3:0]				g[3:0]			
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Binary

Gray code

Gray code

only a single bit changes when moving between states

Generating a Gray code

Commence with a state of all zeros

Then change the least significant bit that results in a new state

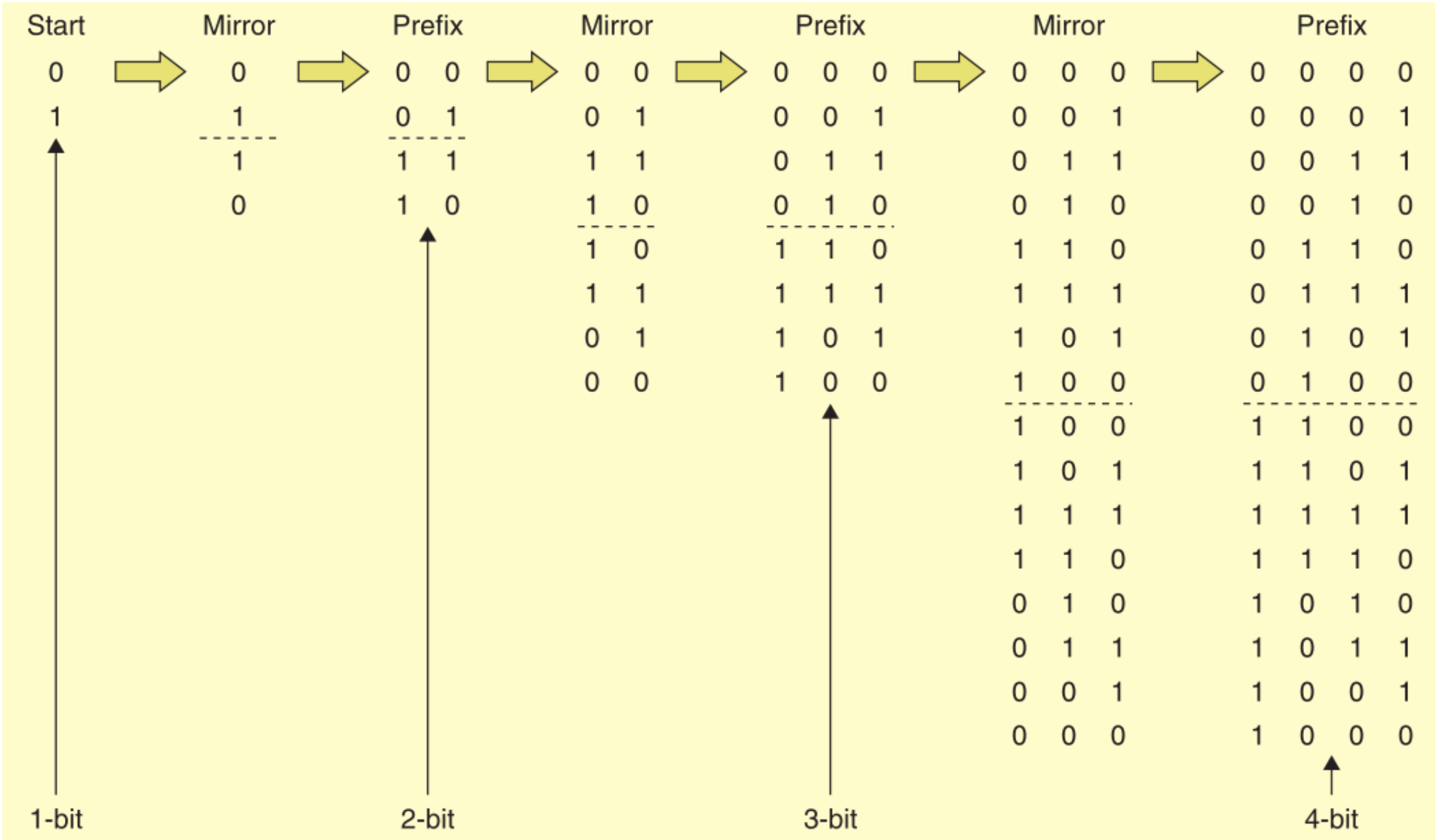
An alternative method of generating a Gray code

1) Commence with the simplest Gray code possible; that is, for a single bit.

2) Create a mirror image of the existing Gray code below the original values.

3) Prefix the original values with 0s and the mirrored values with 1s.

4) Repeat the last two steps until the desired width is achieved.


Start
Mirror
Prefix
Mirror
Prefix
Mirror
Prefix
0
1
0
1
0 0
0 1
1
0
1 1
1 0
0 0
0 1
1 1
1 0
0 0 0
0 0 1
0 1 1
0 1 0
1 1 0
1 1 1
1 0 1
1 0 0
0 0 0 0
0 0 0 1
0 0 1 1
0 0 1 0
0 1 1 0
0 1 1 1
0 1 0 1
0 1 0 0
1 0 0
1 0 1
1 1 1
1 1 0
0 1 0
0 1 1
0 0 1
0 0 0
1 1 0 0
1 1 0 1
1 1 1 1
1 1 1 0
1 0 1 0
1 0 1 1
1 0 0 1
1 0 0 0
1-bit
2-bit
3-bit
4-bit