Computer Systems
Lecture 22

Overview

- Building computers from logic
- Digital systems
- Digital electronic circuits
- Boolean operations and Boolean gates
- Truth tables for basic logic operations
- Boolean circuits
- Selector circuits

Building computers from logic

- Computer systems may be described at different levels of understanding.
- At the **architectural** level the computer is described as a machine for executing instructions.
  - This level is most appropriate for understanding how programs are executed.

- What about the layers **below** the architectural model?

Engineering level

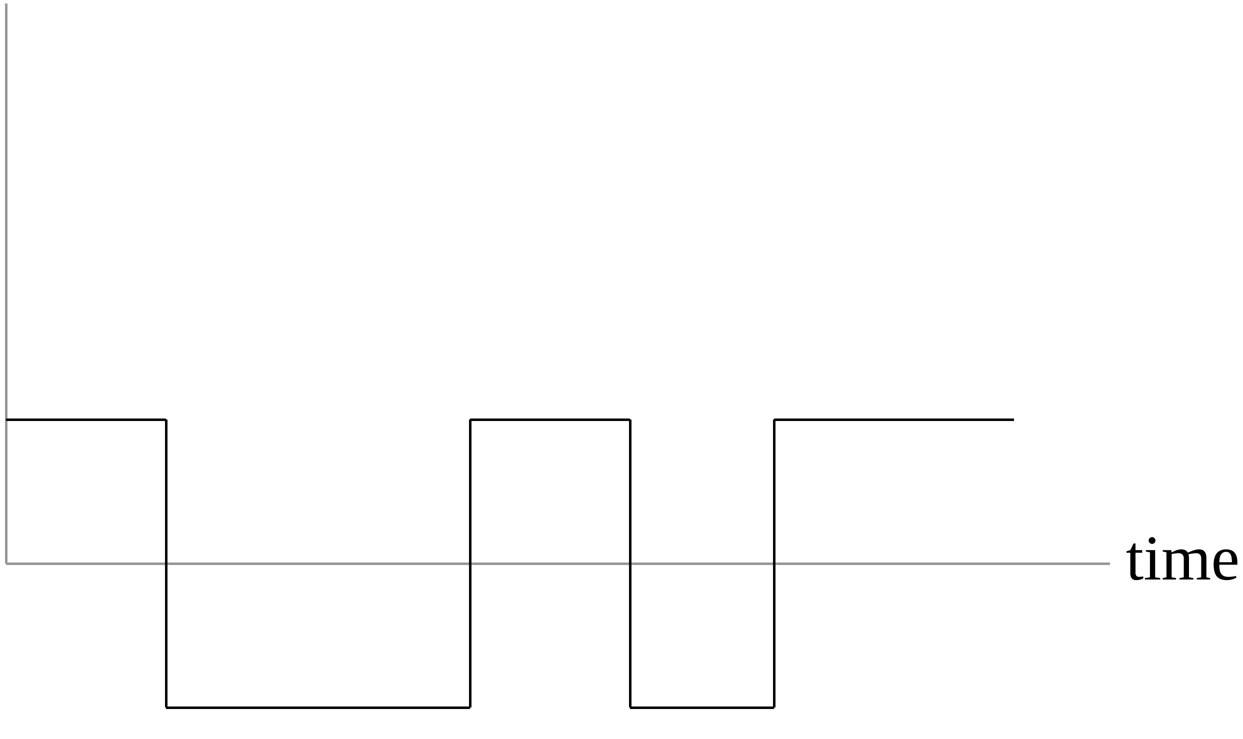
- Engineering model of the computer represents the machine as a complex electrical circuit.
- Within the circuit there are a large number of physical connections, along each of which a current may flow during the operation of the machine.
  - Presence of a current is used to represent transmission of the binary digit **1**, while
  - Absence of a current represents a value **0**.

Digital Systems

- Digital systems based on electronic circuitry
  - 1s and 0s, or on and off
  - Each 1 or 0 is called a bit; or binary digit
  - Computers use digital data representation

Digital Signal

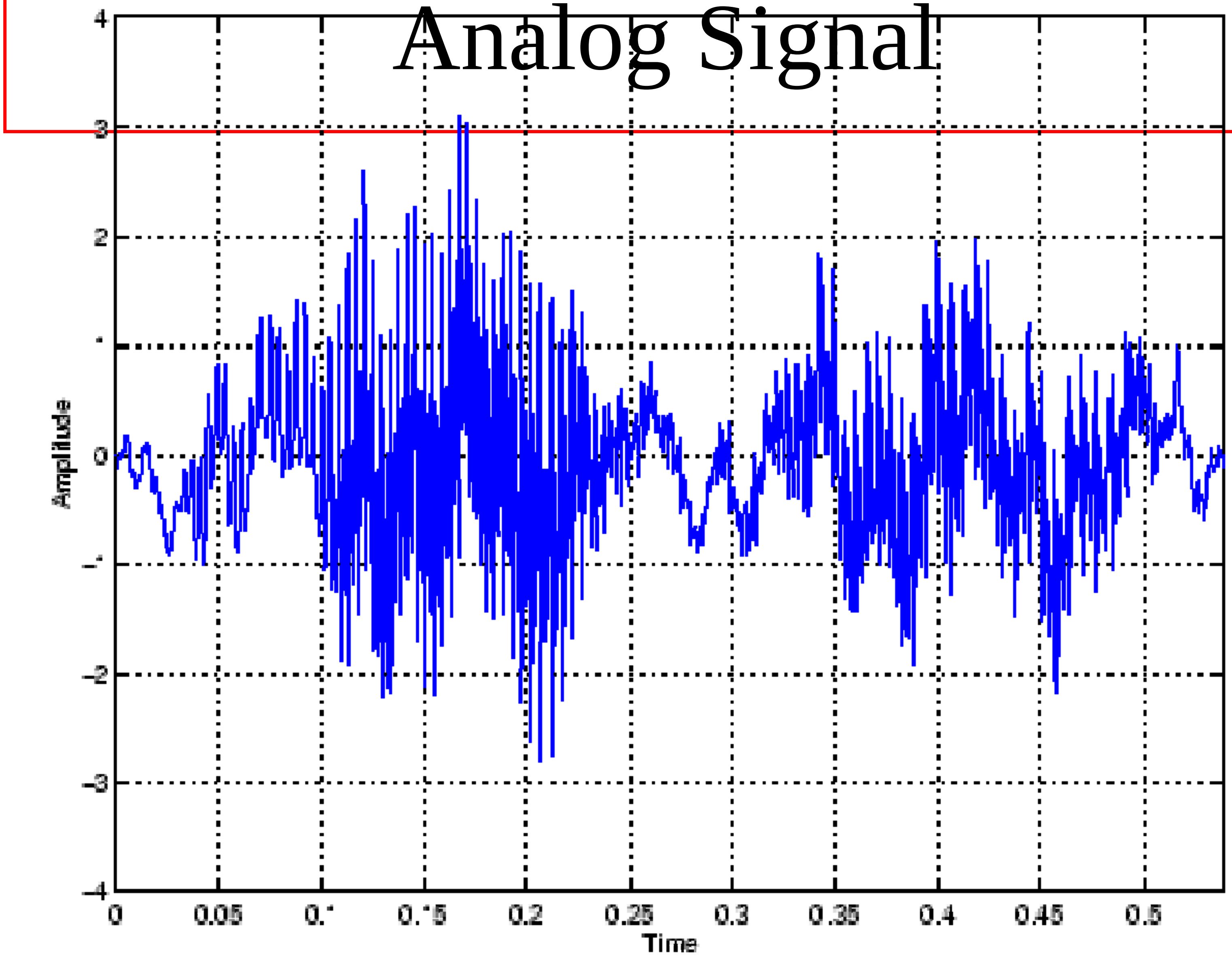
- 1001011…



Digital electronic circuit

- This kind of circuit is called **digital electronic** circuit, because the relevant characteristic is the presence or absence of current (digit 1 or 0), rather than the amount of current flowing.

Analog Signal



Analog Systems

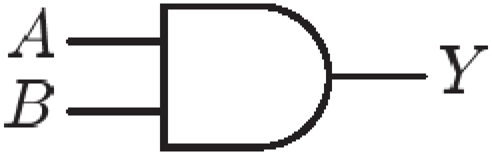
- Analog
  - – continuously variable values, along a range, such as
    - temperature and
    - pressure values
  - – traditional analog recording devices are
    - humidity recorders,
    - mercury thermometers, and
    - pressure gauges
  - – standard telephone lines transmit analog signals

Boolean Operations and Boolean Gates

- All operations that computers perform may be defined in terms of **basic boolean functions**, **operating on bits**.
- The digital electronic circuits and their components can be built from the devices implementing basic boolean operations – **boolean gates** (logic gates).

Boolean Gates

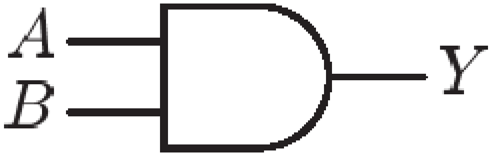
- Consider the circuit device called **AND gate**:



- Two of these, labelled *A* and *B*, represents electronic **inputs** to the component, along each of which a current may
  - flow (boolean value 1) or may
  - not flow (boolean value 0).
- Third connection, labelled *Y* represents the **output** of the component.

AND gate

- In the case of AND gate current will flow in *Y* if and only if a current flows in the input *A* **and** in the input *B*.



- One may write *Y* = *A* **and** *B*.
- Thus, this component implements logic (boolean) operation **AND**.

Truth tables for basic logic operations

A B	A **and** B
0 0	0
0 1	0
1 0	0
1 1	1

A B	A **or** B
0 0	0
0 1	1
1 0	1
1 1	1

A	**not** A
0	1
1	0

A B	A **xor** B
0 0	0
0 1	1
1 0	1
1 1	0

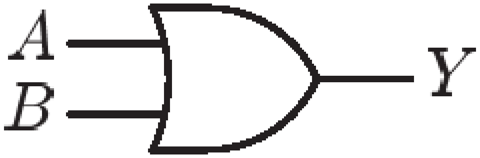
A B	A **=** B
0 0	1
0 1	0
1 0	0
1 1	1

Alternative notations

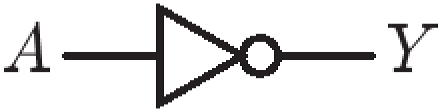
- **and**: $\wedge$, **&**
- **or**: $\vee$
- **xor**: |
- **not**: $\neg$, -

Boolean gates

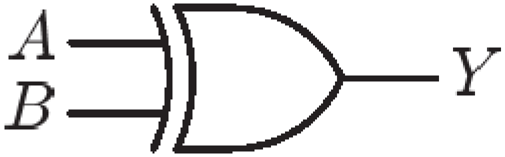
- Other standard logic gate drawing representations:



OR gate

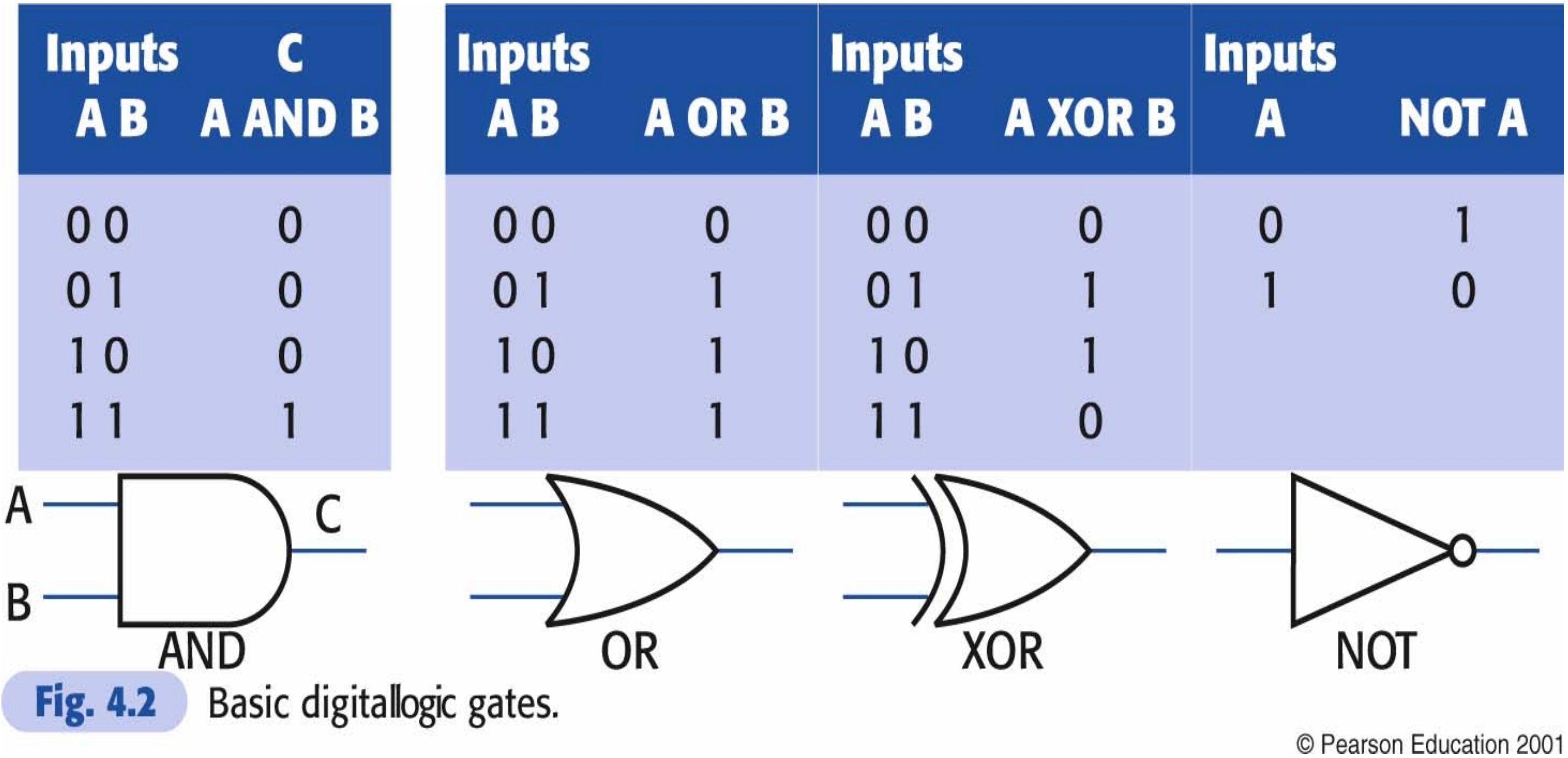


NOT gate



XOR gate

Basic logic gates and their truth tables



Inputs A B	C A AND B
0 0	0
0 1	0
1 0	0
1 1	1

Inputs A B	A OR B
0 0	0
0 1	1
1 0	1
1 1	1

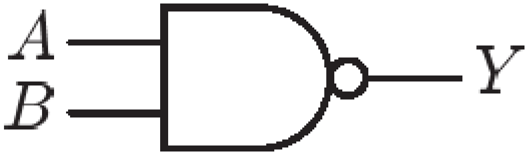
Inputs A B	A XOR B
0 0	0
0 1	1
1 0	1
1 1	0

Inputs A	NOT A
0	1
1	0

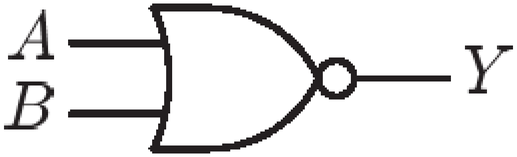
Fig. 4.2 Basic digitallogic gates.

© Pearson Education 2001

More boolean gates



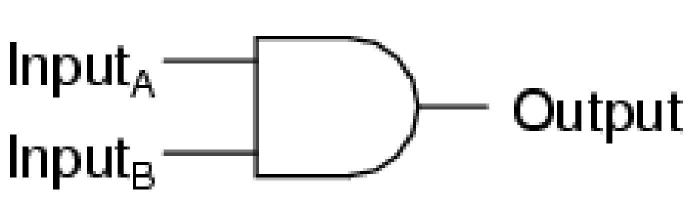
NAND gate, *Y* = **not** (*A* **and** *B*)



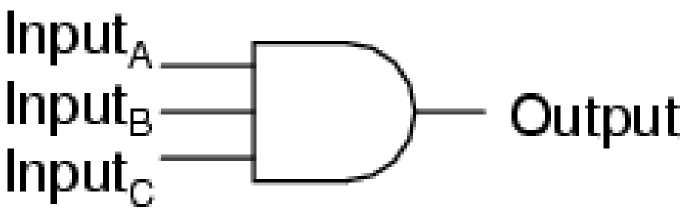
NOR gate, *Y* = **not** (*A* **or** *B*)

Three-input gates

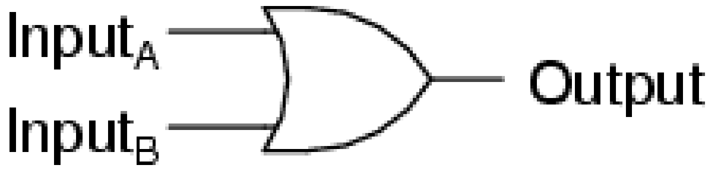
2-input AND gate



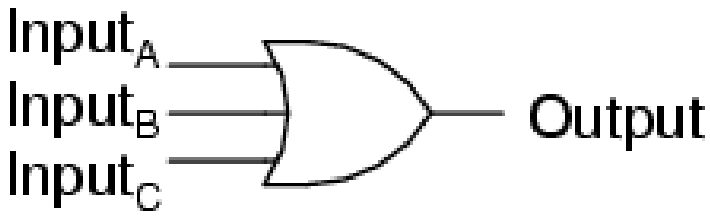
3-input AND gate



2-input OR gate



3-input OR gate



Boolean circuits

- Elementary boolean gates can be combined into **boolean circuits**, implementing more complex **boolean functions** (operations).
- In fact, any boolean function can be implemented with this set of basic boolean gates.

Examples of circuits

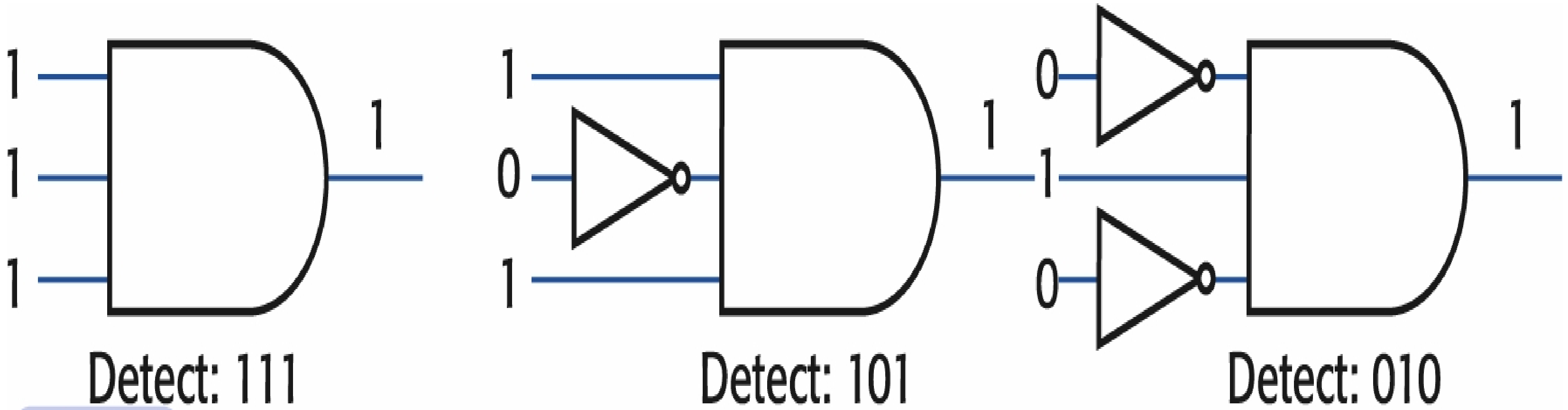


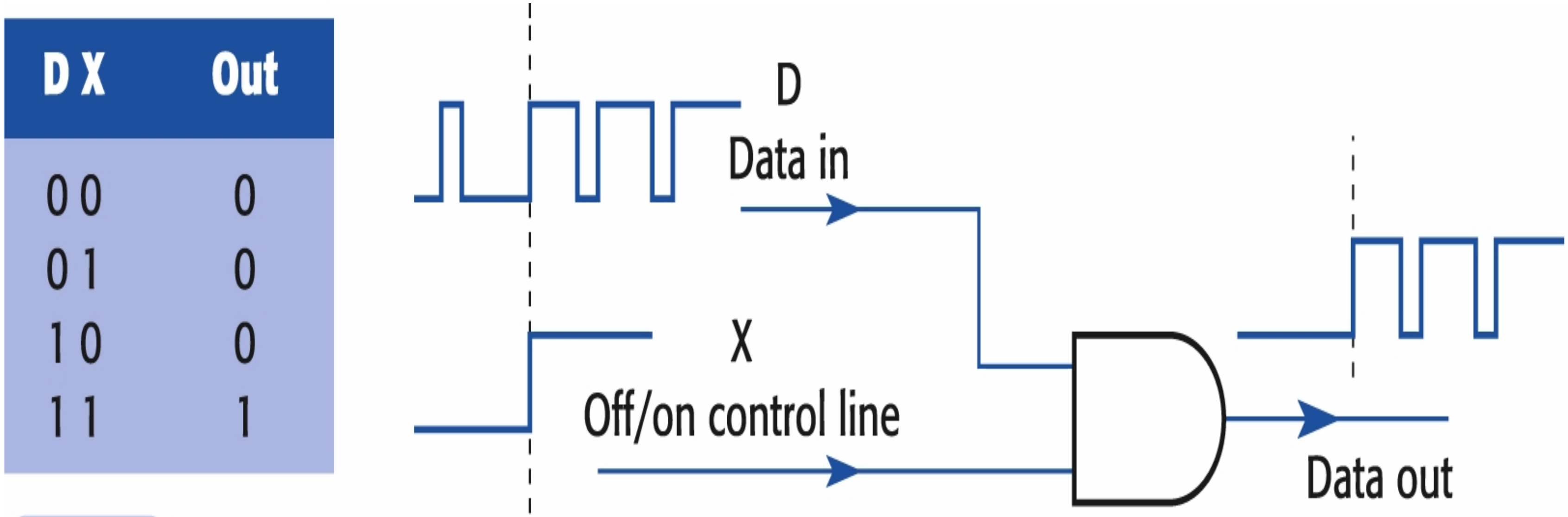
Fig. 4.3 Using AND gatestodetect specific bit patterns.

© Pearson Education 2001

Exercise

- Draw Boolean circuits to implement detectors that detect the presence of the following inputs
  - 000
  - 110

Data flow control circuit - Filter

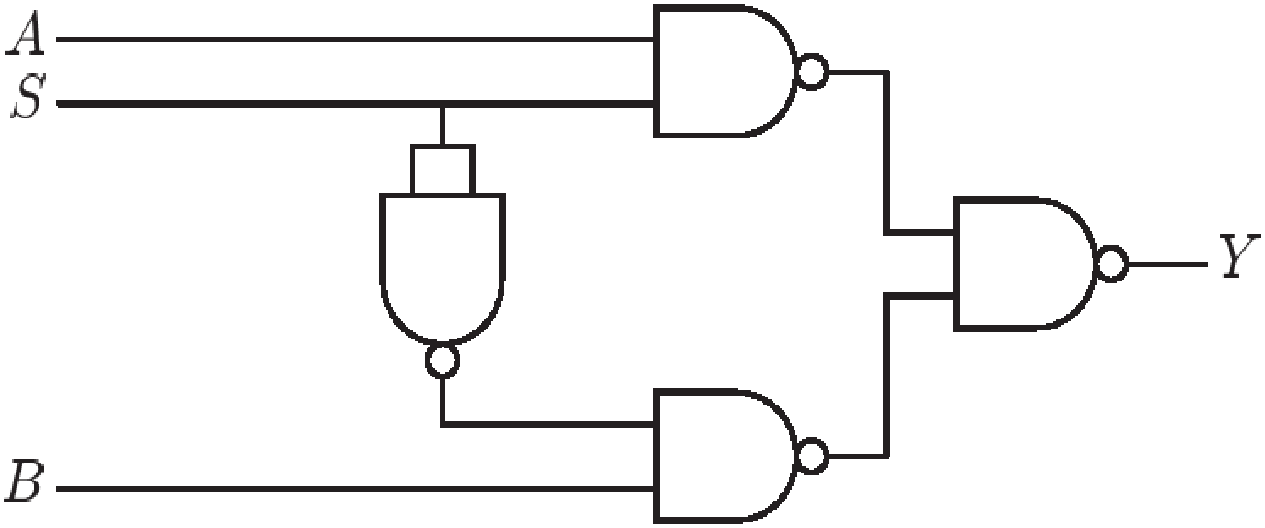


D	X	Out
0	0	0
0	1	0
1	0	0
1	1	1

Fig. 4.4 Data flow control circuit.

© Pearson Education 2001

Selector circuit



If S = 1 then Y = A

if S = 0 then Y = B

Ex. Build the truth table for the selector circuit and verify its functions.

Readings

- [Wil06] Chapter 4, sections 4.1 - 4.2.