Design of Digital Systems II

Introduction

Moslem Amiri, Václav Přenosil

Embedded Systems Laboratory
Faculty of Informatics, Masaryk University
Brno, Czech Republic

amiri@mail.muni.cz
prenosil@fi.muni.cz

September, 2012

Analog vs. Digital

- **Analog** systems process time-varying signals that can take on any value across a continuous range of voltage, current, ...
  - So do **digital** systems; the difference is a digital signal is modeled as taking on only one of two discrete values, 0 and 1
- Reasons to favor digital circuits over analog ones
  - *Reproducibility of results*
    - Given the same inputs, a digital circuit always produces the same results
    - Outputs of an analog circuit vary with temperature, power supply voltage, component aging, ...
  - *Ease of design*
    - Digital design is logical; no math skills and no insights about operation of capacitors, transistors, ... are needed
  - *Flexibility and functionality*
    - E.g., using a digital circuit that scrambles recorded voice so that anyone with key can decipher and hear it undistorted
  - *Programmability*
    - Much of digital design is carried out today by writing programs in *hardware description languages (HDLs)*
    - HDLs allow both structure and function of a digital circuit to be modeled

Analog vs. Digital

- Reasons to favor digital circuits over analog ones (continued)
  - *Speed*
    - Today, transistors can switch in less than 10 picoseconds
    - A device can examine its inputs and produce an output in less than a nanosecond
  - *Economy*
    - Digital circuits provide a lot of functionality in a small space
    - Circuits that are used repetitively can be integrated into a single chip and mass-produced at a very low cost
  - *Steadily advancing technology*
    - Technology for a digital system always gets faster, cheaper, or otherwise better

Digital Devices

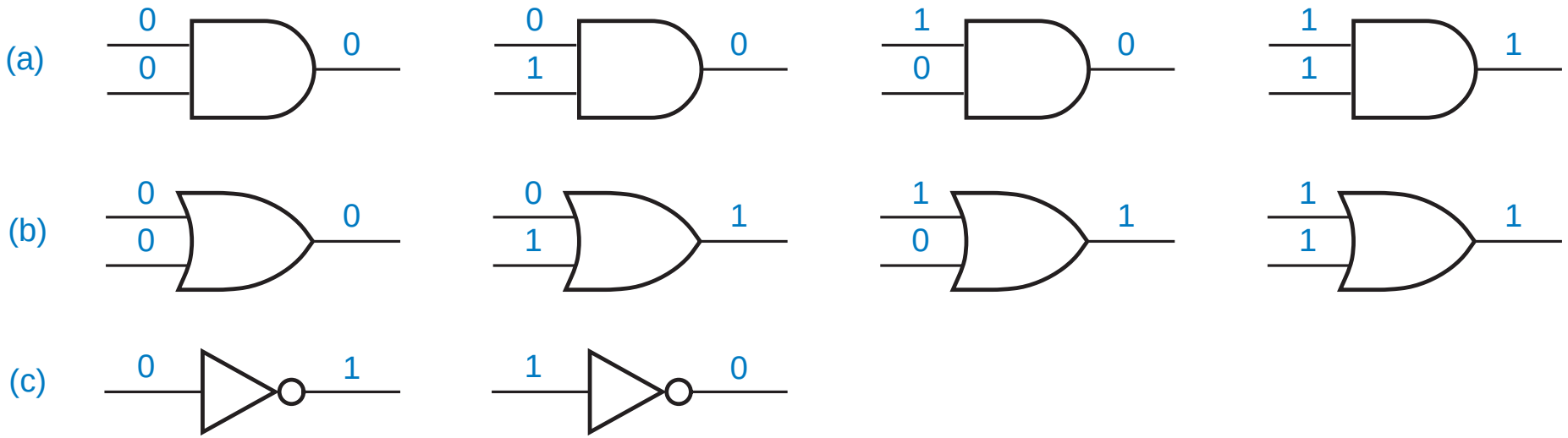


Figure 1: Digital devices: (a) AND gate; (b) OR gate; (c) NOT gate or inverter.

- Any digital function can be realized using just three kinds of gates shown in Fig. 1
- A gate is a **combinational circuit** because its output depends only on current combination of input values

Digital Devices

- A **flip-flop** is a device that stores either a 0 or 1
  - **State** of a flip-flop is the value that it currently stores
  - Stored value can be changed only at certain times determined by a clock input
  - New value may depend on flip-flop's current state and its control inputs
- A digital circuit that contains flip-flops is a **sequential circuit** because its output at any time depends not only on its current input but also on past sequence of inputs

Electronic Aspects of Digital Design

- Digital circuits deal with analog voltages and currents and are built with analog components
  - *Digital abstraction* allows analog behavior to be ignored in most cases, so circuits can be modeled as if they really did process 0s and 1s
- Digital abstraction
  - To associate a range of analog values with 0 or 1
- Noise margin
  - A gate's output can be corrupted by this much noise and still be correctly interpreted at inputs of other gates

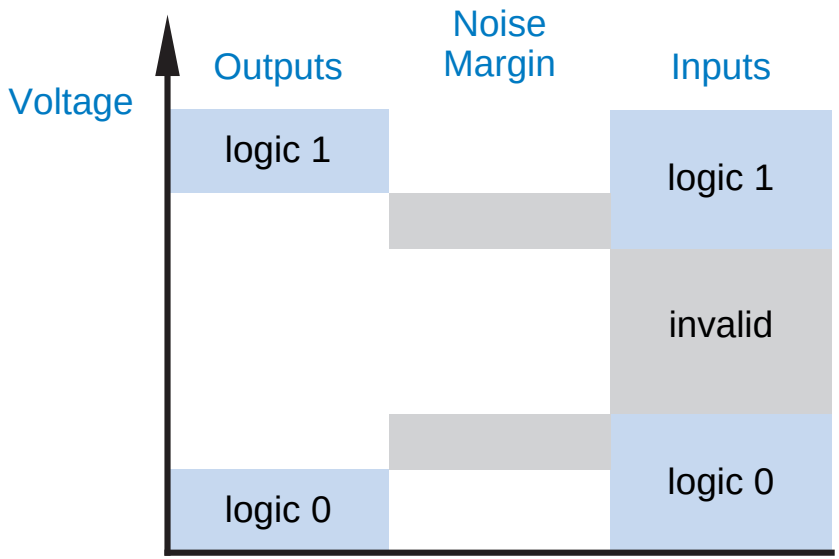


Figure 2: Logic values and noise margins.

Software Aspects of Digital Design

- In *computer-aided design*, various software tools improve designer's productivity and help to improve correctness and quality of designs
- Important examples of software tools for digital design
  - *Schematic entry*
    - Allows schematic diagrams to be drawn on-line instead of with paper and pencil
    - Checks for common, easy-to-spot errors, e.g., shorted outputs, signals that don't go anywhere, ...
  - *HDLs*
    - Are used to design anything from individual function modules to large, multichip digital systems
  - *HDL text editors, compilers, and synthesizers*
    - Text editor is used to write an HDL program
    - HDL compiler checks it for syntax and related errors
    - Synthesizer creates a corresponding circuit realization that is targeted to a particular hardware technology
    - Before synthesis, designer runs HDL program on a simulator to verify behavior of design

Software Aspects of Digital Design

- Important examples of software tools for digital design (continued)
  - *Simulators*
    - Once first chip is built, it's very difficult to debug it by probing internal connections, or to change gates and interconnections
    - Simulators help predict electrical and functional behavior of a chip, allowing most bugs to be found before chip is fabricated
  - *Simulators*
    - Used in overall design of systems with many individual components
    - However, it's easier to make changes in components and interconnections on a printed-circuit board
  - *Test benches*
    - Environments to simulate and test HDL-based digital designs
    - A set of programs are built around HDL programs to automatically exercise them, checking both their functional and timing behavior
  - *Timing analyzers and verifiers*
    - Automate task of drawing timing diagrams and specifying and verifying timing relationships between different signals in a complex system
  - *Word processors*
    - HDL-specific text editors are useful for writing source code, but word processors can be used to create documentation

Integrated Circuits

- **Integrated circuit (IC)**
  - A collection of one or more gates fabricated on a single silicon chip
  - An IC is initially part of a larger, circular **wafer**, containing dozens to hundreds of replicas of same IC
  - All of IC chips on wafer are fabricated at the same time
  - Each piece (IC chip) is called a **die**
  - Each die has **pads** around its periphery, i.e., electrical contact points, so wires can be connected later
  - After wafer is fabricated, dice are tested in place on wafer using tiny, probing pins that contact pads, and defective dice are marked
  - Then wafer is sliced up to produce individual dice, and marked ones are discarded
  - Each good die is mounted in a package, its pads are wired to package pins, packaged IC is subjected to a final test, and shipped to a customer

- **Small-scale integration (SSI)** ICs
  - Contain equivalent of 1 to 20 gates
  - Are largely supplanted by programmable logic devices (PLDs)
  - Are still sometimes used as *glue* to tie together larger-scale elements in complex systems

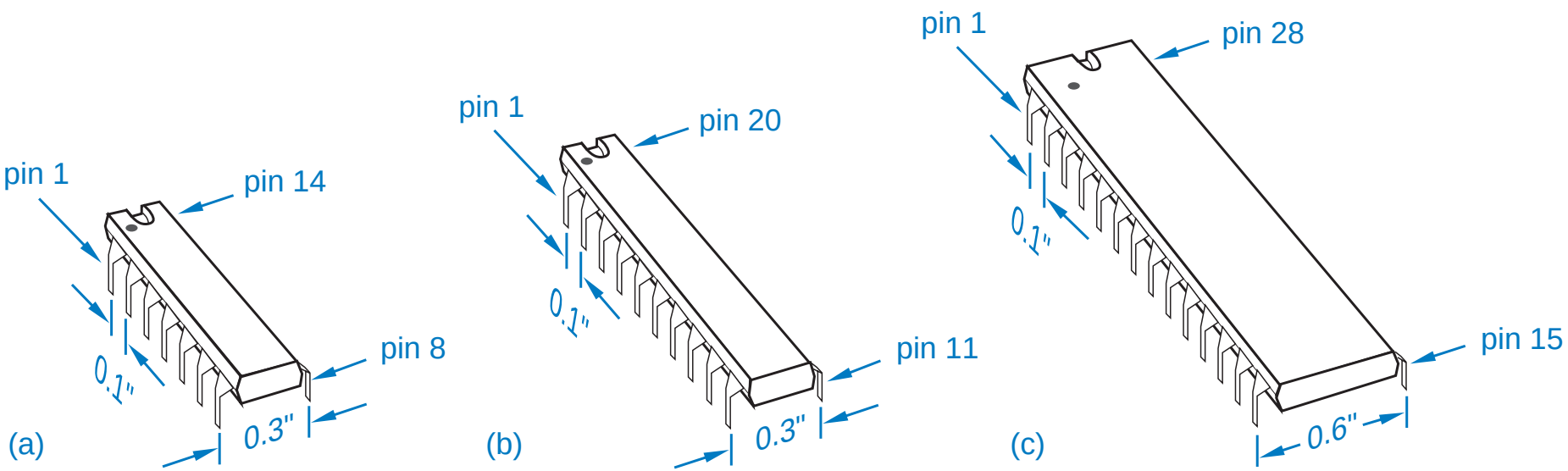


Figure 3: Dual inline pin (DIP) packages: (a) 14-pin; (b) 20-pin; (c) 28-pin.

Integrated Circuits

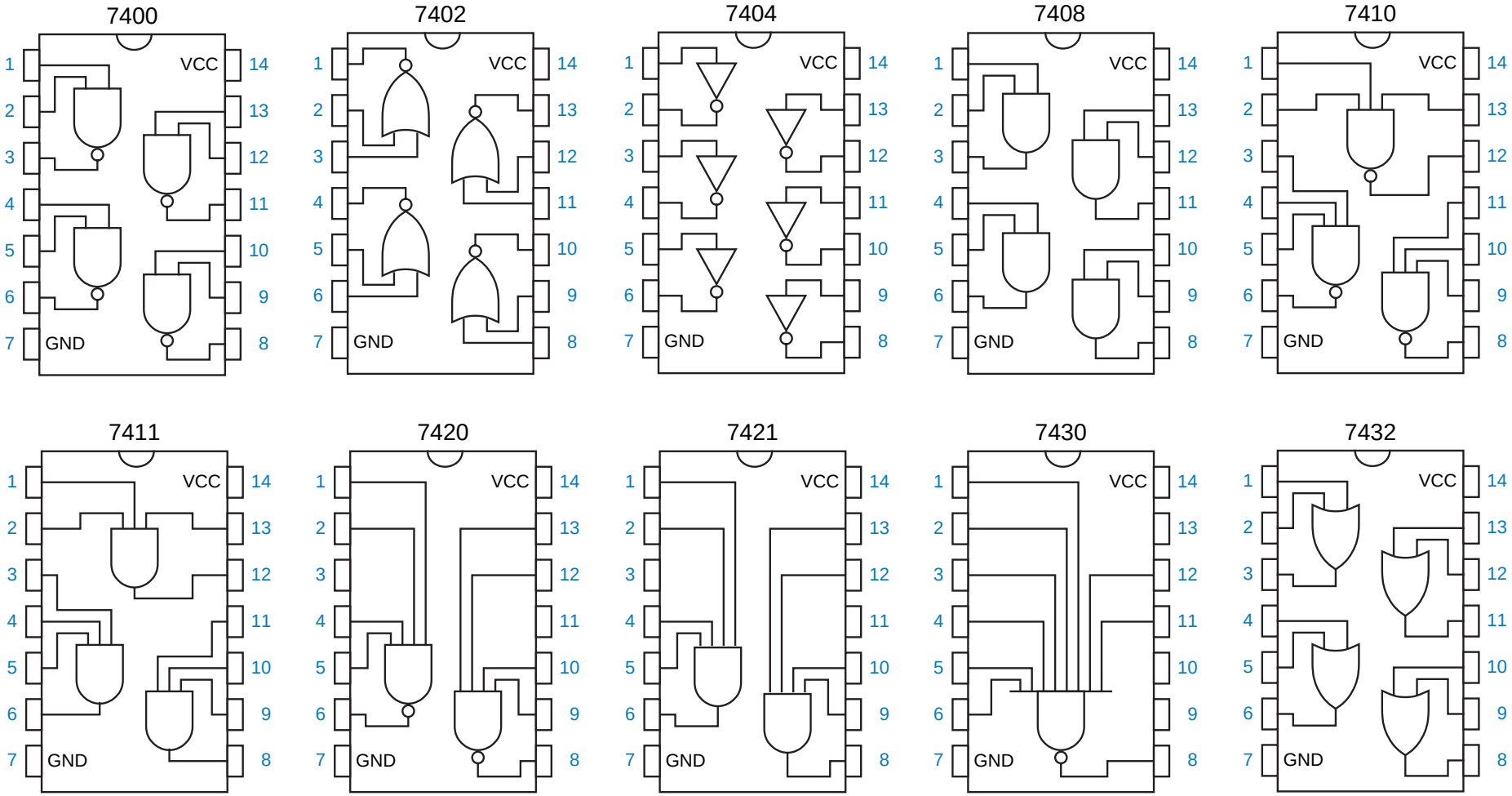


Figure 4: Pin diagrams for a few 7400-series SSI ICs.

Integrated Circuits

- **Medium-scale integration (MSI)** ICs
  - Contain equivalent of about 20 to 200 gates
  - Typically contain a functional building block, such as a decoder, register, or counter
  - Even though use of discrete MSI ICs has declined, equivalent building blocks are used extensively in design of larger ICs
- **Large-scale integration (LSI)** ICs
  - Contain equivalent of 200 to 1,000,000 gates or more
  - LSI parts include small memories, microprocessors, programmable logic devices, and customized devices
- **Very large-scale integration (VLSI)** ICs
  - Contain over a few million transistors
  - E.g., today's most microprocessors, memories, larger programmable logic devices and customized devices

Programmable Logic Devices

- There are a wide variety of ICs that can have their logic function *programmed* into them after they are manufactured
  - Most of them can be *reprogrammed* (for fixing bugs)
- **Programmable logic arrays (PLAs)**
  - Historically, first programmable logic devices
  - Contained a two-level structure of AND and OR gates with user-programmable connections
- **Programmable array logic (PAL) devices**
  - PLA structure was enhanced and PLA costs were reduced with introduction of PAL devices
  - Today, such devices are generically called **programmable logic devices (PLDs)**
  - PLDs are MSI of programmable logic industry
- **Complex PLD (CPLD)**
  - To design larger PLDs for larger applications, for technical reasons, basic two-level AND-OR structure of PLDs could not be scaled to larger sizes
  - IC manufacturers devised CPLD architectures to achieve required scale
  - CPLD is a collection of multiple PLDs and a programmable interconnection structure, all on the same chip

Programmable Logic Devices

- **Field-programmable gate array (FPGA)**
  - While CPLDs were being invented, other IC manufacturers took a different approach to scaling size of PLDs
  - Compared to a CPLD, an FPGA contains a much larger number of smaller individual logic blocks
  - FPGA provides a large, distributed interconnection structure

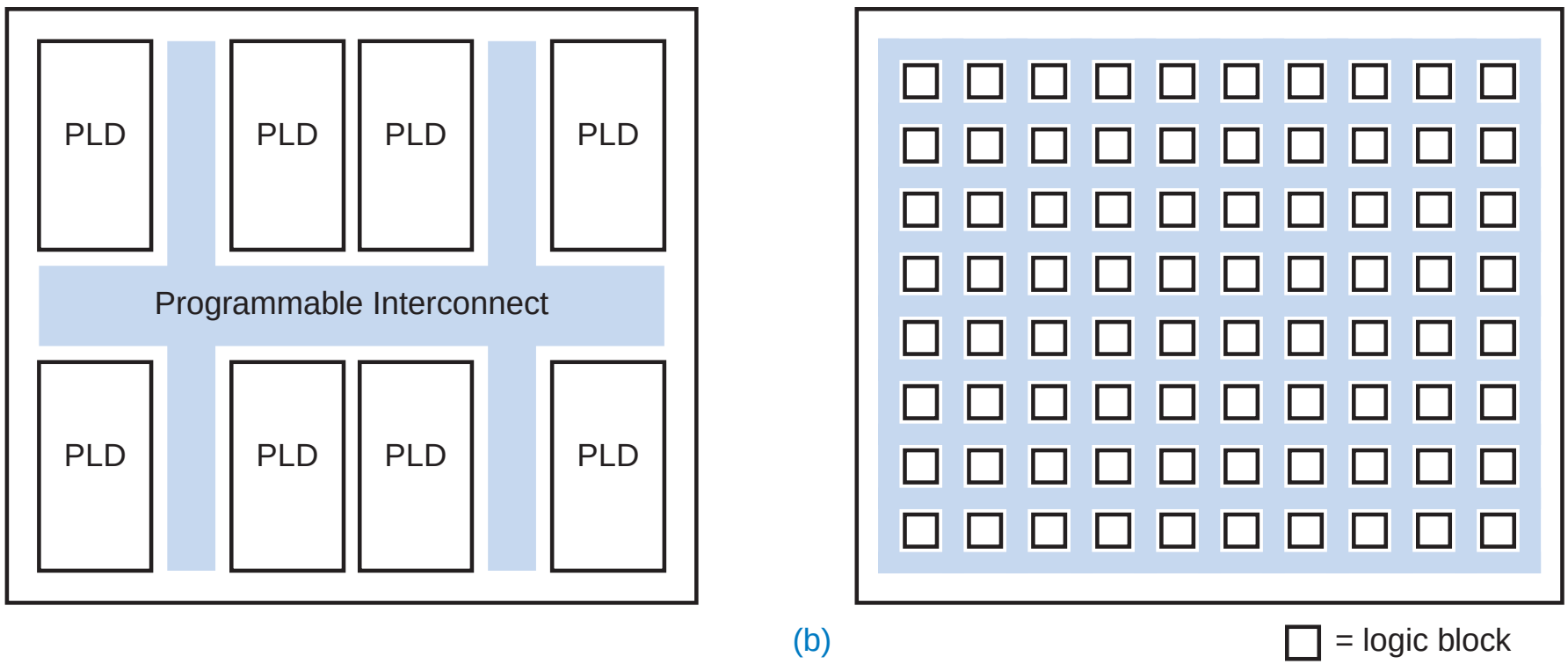


Figure 5: Large PLD scaling approaches: (a) CPLD; (b) FPGA.

Application-Specific ICs (ASICs)

- **ASICs or semicustom ICs**
  - Chips designed for a particular, limited product or application
  - Reduce total component and manufacturing cost of a product by reducing chip count, physical size, and power consumption
  - Provide higher performance
- **Nonrecurring engineering (NRE) cost** for an ASIC design can exceed cost of a discrete design by $10,000 to $500,000 or more
  - NRE charges are paid to IC manufacturer and others responsible for designing internal structure of chip, creating tools such as metal masks, developing tests, and making first few sample chips
  - An ASIC design makes sense if NRE cost is offset by per-unit savings
- **Custom LSI chip**
  - A chip whose functions, internal architecture, and detailed transistor-level design is tailored for a specific customer
  - Its NRE cost is very high, $500,000 or more
  - I.e., chips that have general commercial application like microprocessors, or high sales volume in a specific application like a digital watch chip

Application-Specific ICs (ASICs)

- **Standard cells**
  - Developed to reduce NRE charges
  - Libraries of standard cells include commonly used MSI and LSI functions
  - In a standard-cell design, designer interconnects functions like as in a multichip MSI/LSI design
  - Custom cells are created only if absolutely necessary
  - All of cells are then laid out on chip, optimizing layout to reduce propagation delays and minimize chip size
  - NRE cost for a standard-cell design is $250,000 or more
- **Gate array**
  - Developed to reduce NRE charges even further
  - Is an IC whose internal structure is an array of gates whose interconnections are initially unspecified
  - Designer specifies gate types and interconnections
  - Even though chip design is ultimately specified at this very low level, designer works with *macrocells*, the same high-level functions used in multichip MSI/LSI and standard-cell design
  - Software expands high-level design into a low-level one

Application-Specific ICs (ASICs)

- **Standard-cell vs. gate-array design**
  - Macrocells and chip layout of a gate array are not as highly optimized as those in a standard-cell design, so chip may be 25% or more larger and therefore may cost more
  - Not possible to create custom cells in gate-array approach
  - A gate-array design can be finished faster and at lower NRE cost, ranging from $10,000 to $100,000

Printed-Circuit Boards (PCBs)

- An IC is mounted on a PCB that connects it to other ICs in a system
- Multilayer PCBs have copper wiring etched on multiple, thin layers of fiberglass that are laminated into a single board
- **PCB traces**
  - Individual wire connections
  - Are 10 to 25 mils (1 mil = 1/1000 inch) wide in typical PCBs
  - In **fine-line** PCB technology, traces are 3 mils wide with 3-mil spacing between adjacent traces
  - If higher connection density is needed, more layers are used
- **Surface-mount technology (SMT)**
  - Instead of having long pins of DIP packages that poke through board and are soldered to underside, leads of SMT IC packages are bent to make flat contact with top surface of PCB
  - First, a *solder paste* is applied to contact pads on PCB using a stencil
  - Then, SMT components are placed on pads
  - Finally, entire assembly is passed through an oven to melt solder paste, which then solidifies when cooled

Printed-Circuit Boards (PCBs)

- Surface-mount tech. coupled with fine-line PCB tech.
  - Allows extremely dense packing of ICs and other components on a PCB
  - Saves space
  - Minimizes transmission-line effects
  - Minimizes speed-of-light limitations
- **Multichip modules (MCMs)**
  - Developed to satisfy the most stringent requirements for speed and density
  - IC dice are not mounted in individual plastic or ceramic packages
  - IC dice for a high-speed subsystem (e.g., a processor and its cache memory) are bonded directly to a substrate that contains required interconnections on multiple layers
  - MCM is sealed and has its own external pins for power, ground, and just those signals that are required by system that contains it

Digital-Design Levels

- Digital design can be carried out at several different levels of representation and abstraction
  - Sometimes it is needed to go up or down a level or two to get the job done
  - Industry and most designers are moving to higher levels as circuit density and functionality increase
  - Lowest level = device physics and IC manufacturing processes
    - Won't be discussed in this course
  - Transistor level design $\Longrightarrow$ logic design using HDLs
    - Will be discussed in this course
    - Level of functional building blocks is center of our discussion
  - Highest level = computer design and overall system design
    - Won't be discussed in this course

Digital-Design Levels: Example

- Multiplexer

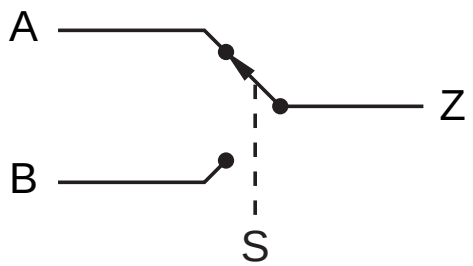


Figure 6: Switch model for multiplexer function.

- For some functions it is advantageous to optimize them by designing at transistor level
  - Multiplexer is such a function
  - Multiplexer can be designed in CMOS technology using specialized transistor circuit structures called *transmission gates*
    - Using this approach, mux can be built with just six transistors
    - Any other approach requires at least 14 transistors

Digital-Design Levels: Example

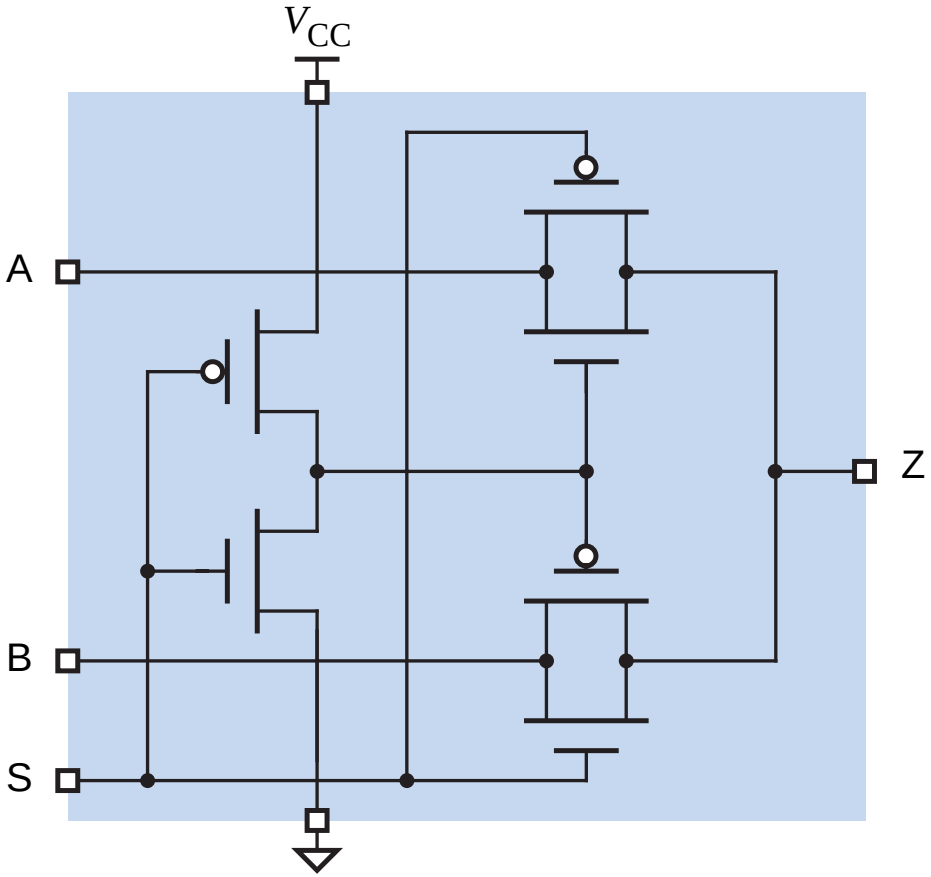


Figure 7: Multiplexer design using CMOS transmission gates.

Digital-Design Levels: Example

- **Truth table**
  - Is used to describe logic function
  - Traditional logic design methods use Boolean algebra and minimization algorithms to derive an optimal two-level AND-OR equation from truth table
- For Tab. 1

$$Z = S'.A + S.B \tag{1}$$

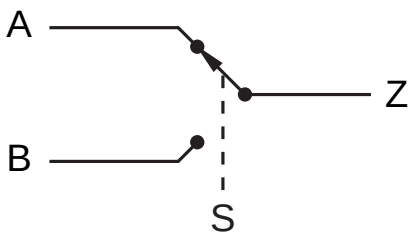


Table 1: Truth table for the multiplexer function.

S	A	B	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Digital-Design Levels: Example

- Going one step further, (1) can be converted into a set of logic gates, as shown in Fig. 8
  - This circuit requires 14 transistors

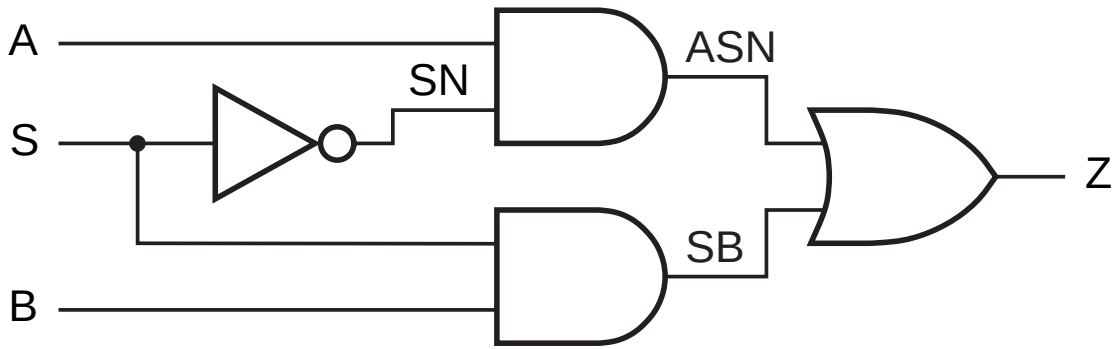


Figure 8: Gate-level logic diagram for multiplexer function.

Digital-Design Levels: Example

- Multiplexer is a very commonly used function
  - Most digital logic technologies provide predefined multiplexer building blocks
  - E.g., 74x157 is an MSI chip that performs multiplexing on two 4-bit inputs simultaneously

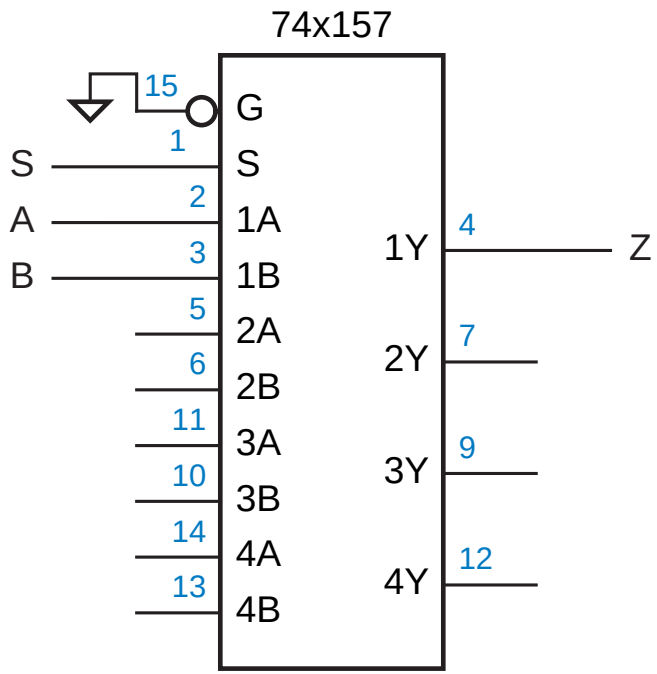


Figure 9: Logic diagram for a multiplexer using an MSI building block.

Digital-Design Levels: Example

- We can also realize multiplexer function as part of a PLD
  - HDLs allow us to specify logic functions using Boolean equations like (1)
  - An HDL's higher-level language elements can create a more readable program

Table 2: ABEL program for the multiplexer.

```
module chap1mux
title 'Two-input multiplexer example'
CHAP1MUX device 'P16V8'

A, B, S      pin 1, 2, 3;
Z            pin 13 istype 'com';

equations

when S == 0 then Z = A;  else Z = B;

end chap1mux
```

Digital-Design Levels: Example

- VHDL and Verilog are even higher-level languages than ABEL
  - They can be used to specify multiplexer function in a way that is very flexible and hierarchical

Table 3: VHDL program for the multiplexer.

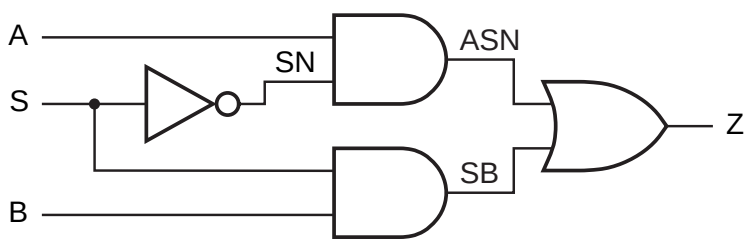
```
library IEEE;
use IEEE.std_logic_1164.all;

entity Vchap1mux is
    port ( A, B, S: in  STD_LOGIC;
           Z:       out STD_LOGIC );
end Vchap1mux;

architecture Vchap1mux_arch of Vchap1mux is
begin
  Z <= A when S = '0' else B;
end Vchap1mux_arch;
```

Digital-Design Levels: Example

- Input/output definitions (entity) and internal realization (architecture) are separate in VHDL
  - Easy to define alternate realizations of functions
  - An alternate, structural architecture for multiplexer is shown in Tab. 4

Table 4: "Structural" VHDL program for the multiplexer.

```
architecture Vchap1mux_gate_arch of Vchap1mux is
signal SN, ASN, SB: STD_LOGIC;
  -- required component declarations have been omitted
  --   for brevity in this example.
begin
  U1: port map INV (S, SN);
  U2: port map AND2 (A, SN, ASN);
  U3: port map AND2 (S, B, SB);
  U4: port map OR2 (ASN, SB, Z);
end Vchap1mux_gate_arch;
```



Digital-Design Levels: Example

- VHDL is powerful enough to define operations that model functional behavior at transistor level
  - Won't be explored in this course
- Verilog syntax is somewhat C-like
  - Like C, Verilog is less picky about variable and type definition
    - E.g., in Tab. 5 all of variables default to being 1-bit wires
  - Unlike VHDL, Verilog does not require separate definitions of entity and architecture
  - Verilog provides a means for defining functions structurally as in VHDL example of Tab. 4

Table 5: Verilog program for the multiplexer.

```
module Vrchap1mux(A, B, S, Z);
  input A, B, S;
  output Z;

   assign Z = (S==0) ? A : B;
endmodule
```

References

John F. Wakerly, *Digital Design: Principles and Practices (4th Edition)*, Prentice Hall, 2005.