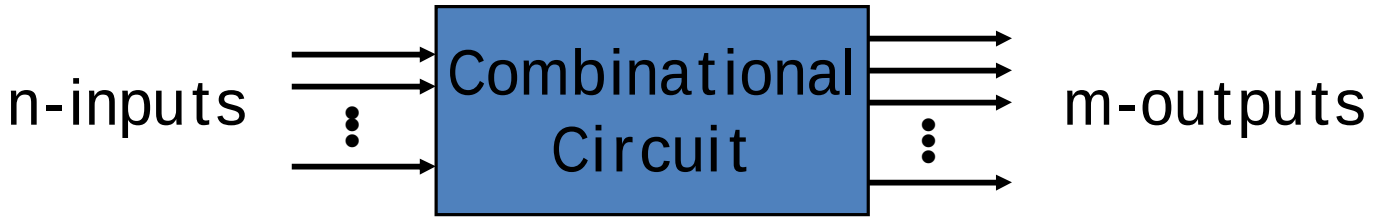
Combinational Circuits

Combinational Circuits

- A combinational circuit consists of logic gates whose outputs, at any time, are determined by combining the values of the inputs.
- For *n* input variables, there are 2^n possible binary input combinations.
- For each binary combination of the input variables, there is one possible output.

Combinational Circuits (cont.)

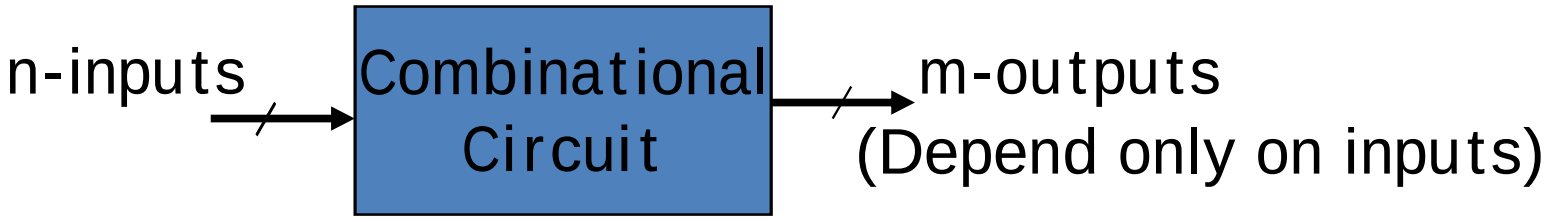
- Hence, a combinational circuit can be described by:
  1. A truth table that lists the output values for each combination of the input variables, or
  2. *m* Boolean functions, one for each output variable.



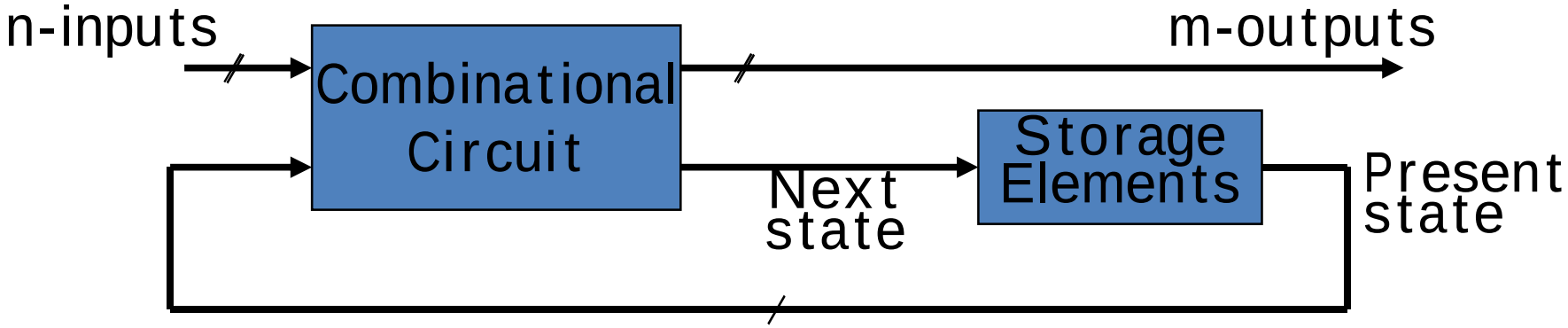
Combinational vs. Sequential Circuits

- Combinational circuits are memory-less. Thus, the output value depends ONLY on the current input values.
- Sequential circuits consist of combinational logic as well as memory elements (used to store certain circuit states). Outputs depend on BOTH current input values and previous input values (kept in the storage elements).

Combinational vs. Sequential Circuits



Combinational Circuit



Sequential Circuit

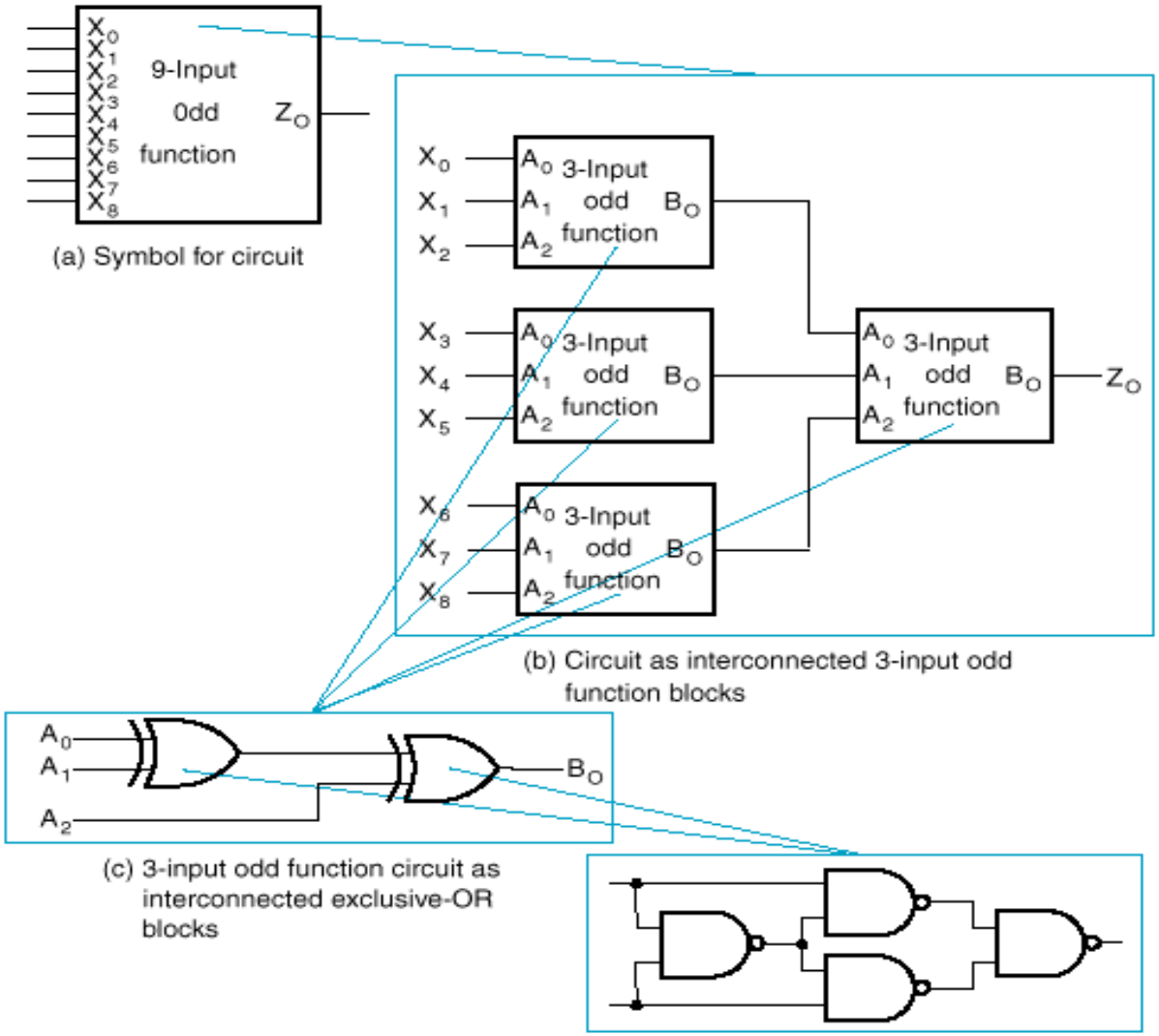
Important Design Concepts

- Modern digital design deals with various methods and tools that are used to design and verify complex circuits and systems.
- Concepts:
  - Design Hierarchy
  - Computer-Aided-Design (CAD) tools
  - Hardware Description Languages (HDLs)

Design Hierarchy

- ***"Divide-and-Conquer"*** approach used to cope with the challenges of designing complex circuits and systems (many times in the order of millions of gates).
- Circuit is broken into ***blocks***, repetitively.

Design Hierarchy
Example: 9-input odd function (for counting # of 1 in inputs)



Why is Hierarchy useful?

- Reduces the complexity required to design and represent the overall schematic of the circuit.
- Reuse of blocks is possible. Identical blocks can be used in various places in a design, or in different designs.

Reusable Functions and CAD

- Whenever possible, we try to decompose a complex design into common, *reusable* function blocks
- These blocks are
  - verified and well-documented
  - placed in libraries for future use

Integrated Circuits

- Integrated circuit (a chip) is a semiconductor crystal (most often silicon) containing the electronic components for the digital gates and storage elements which are interconnected on the chip.
- Terminology - Levels of chip integration
  - *SSI* (*small-scale integrated*) - fewer than 10 gates
  - *MSI* (*medium-scale integrated*) - 10 to 100 gates
  - *LSI* (*large-scale integrated*) - 100 to thousands of gates
  - *VLSI* (*very large-scale integrated*) - thousands to 100s of millions of gates

Technology Parameters

- Specific gate implementation technologies are characterized by the following parameters:
  - *Fan-in* – the number of inputs available on a gate
  - *Fan-out* – the number of standard loads driven by a gate output
  - *Cost for a gate* - a measure of the contribution by the gate to the cost of the integrated circuit
  - *Propagation Delay* – The time required for a change in the value of a signal to propagate from an input to an output
  - *Power Dissipation* – the amount of power drawn from the power supply and consumed by the gate

Chip Design Styles

- Full custom - the entire design of the chip down to the smallest detail of the layout is performed
  - Expensive, its timing and power is hard to analyze
  - only for dense, fast chips with high sales volume
- Standard cell - blocks have been design ahead of time or as part of previous designs
  - Intermediate cost
  - Less density and speed compared to full custom
- Gate array - regular patterns of gate transistors that can be used in many designs built into chip - only the interconnections between gates are specific to a design
  - Lowest cost
  - Less density compared to full custom and standard cell
  - Prototype design
  - The base of FPGA

Cell Libraries

- *Cell* - a pre-designed primitive block
- *Cell library* - a collection of cells available for design using a particular implementation technology
- *Cell characterization* - a detailed specification of a cell for use by a designer

Cell Library Based Design Procedure

1. Specification
   - Write a specification for the circuit if one is not already available
2. Formulation
   - Derive a truth table or initial Boolean equations that define the required relationships between the inputs and outputs, if not in the specification
3. Optimization
   - Draw a logic diagram or provide a netlist for the resulting circuit using ANDs, ORs, and inverters

Cell Library Based Design Procedure

4. Technology Mapping
   - Map the logic diagram to the implementation technology selected
   - Map to CMOS
5. Evaluation
   - Evaluate the timing and power

Design Example

1. Specification
   - BCD to Excess-3 code converter
   - Transforms BCD code for the decimal digits to Excess-3 code for the decimal digits
   - BCD code words for digits 0 through 9: 4-bit patterns 0000 to 1001, respectively
   - Excess-3 code words for digits 0 through 9: 4-bit patterns consisting of 3 (binary 0011) added to each BCD code word

Design Example (continued)

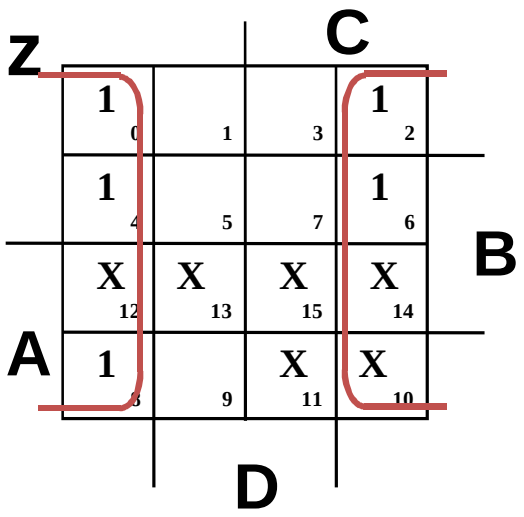
2. Formulation

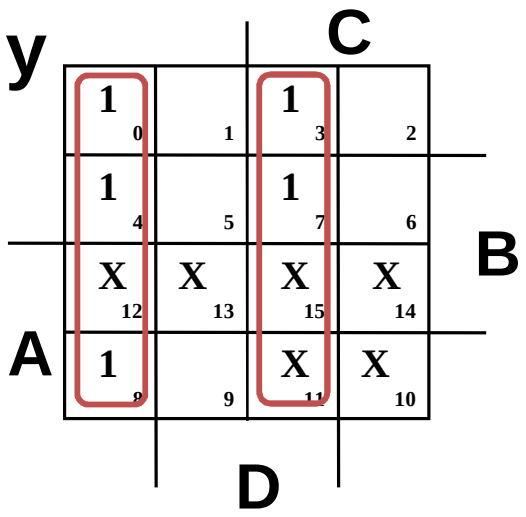
- Conversion of 4-bit codes can be most easily formulated by a truth table
- Variables
  - BCD: A,B,C,D
- Variables
  - Excess-3 W,X,Y,Z
- Don't Cares
  - BCD 1010 to 1111

Input BCD A B C D	Output Excess-3 WXYZ
0 0 0 0	0 0 1 1
0 0 0 1	0 1 0 0
0 0 1 0	0 1 0 1
0 0 1 1	0 1 1 0
0 1 0 0	0 1 1 1
0 1 0 1	1 0 0 0
0 1 1 0	1 0 0 1
0 1 1 1	1 0 1 0
1 0 0 0	1 0 1 1
1 0 0 1	1 0 1 1

Design Example (continued)

3. Optimization
   a. K-maps



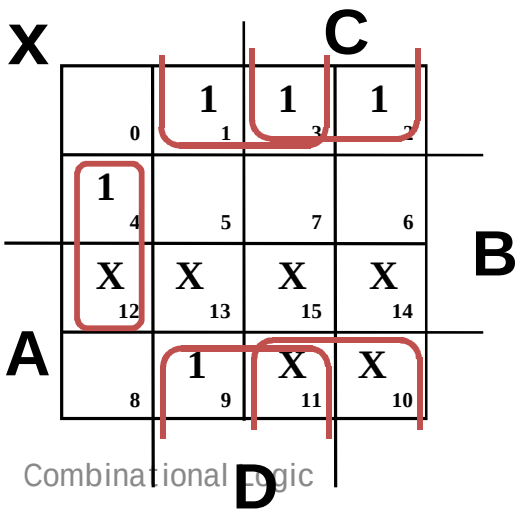


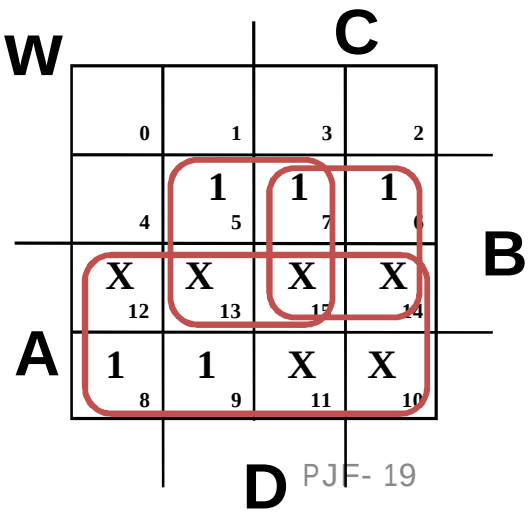
$W = A + BC + BD$

$X = B\overline{C} + B\overline{D} + \overline{B}\,\overline{C}\,\overline{D}$

$Y = CD + \overline{C}\,\overline{D}$

$Z = \overline{D}$





Design Example (continued)

3. Optimization (continued)

   Multiple-level using transformations

   $W = A + BC + BD$

   $X = \overline{B}C + D\overline{B} + B\overline{C}\,\overline{D}$

   $Y = CD + \overline{C}\,\overline{D}$

   $Z = \overline{D}$

   – Perform extraction, finding factor:

   $T_1 = C + D$

   $W = A + BT_1$

   $X = \overline{B}T_1 + B\overline{C}\,\overline{D}$

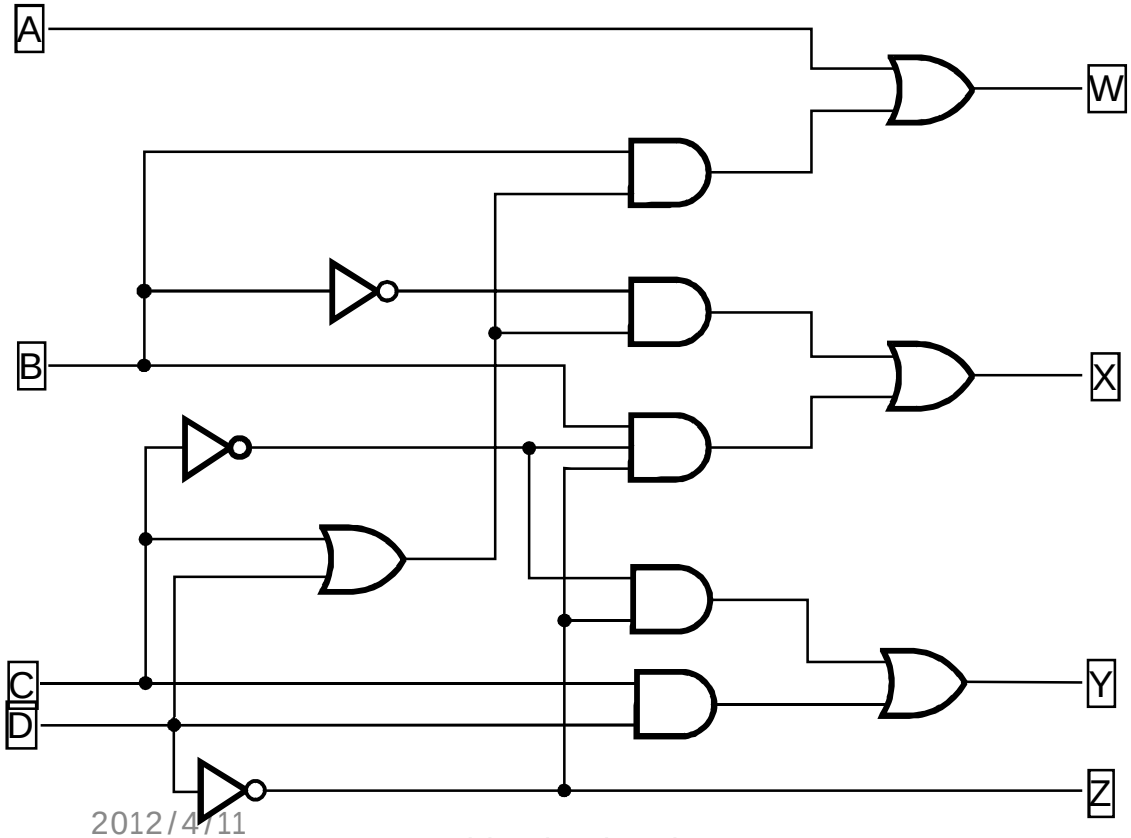
   $Y = CD + \overline{C}\,\overline{D}$

   $Z = \overline{D}$

Design Example (continued)

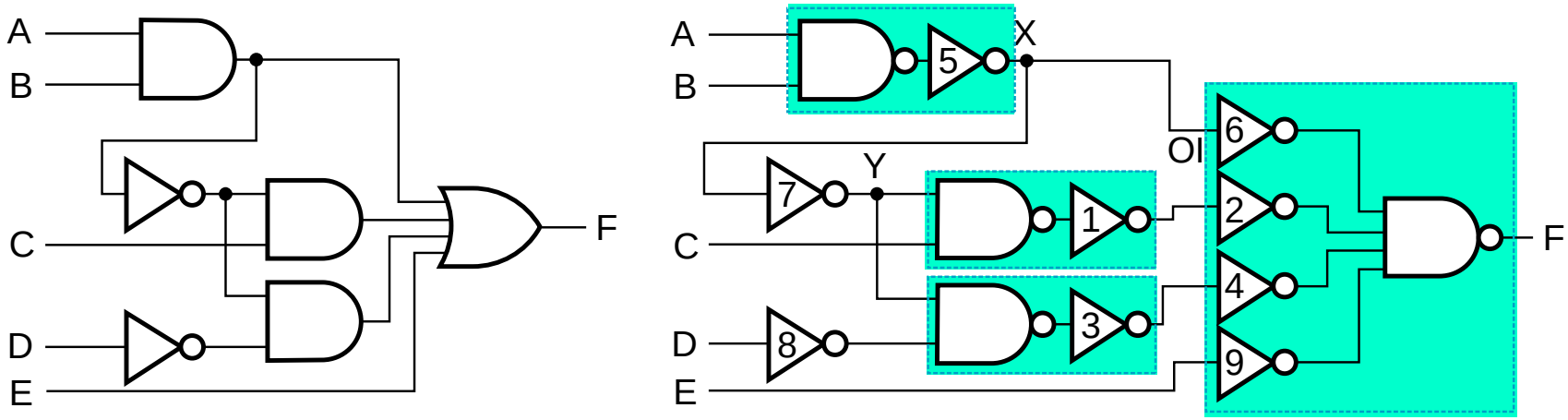
4. Technology Mapping

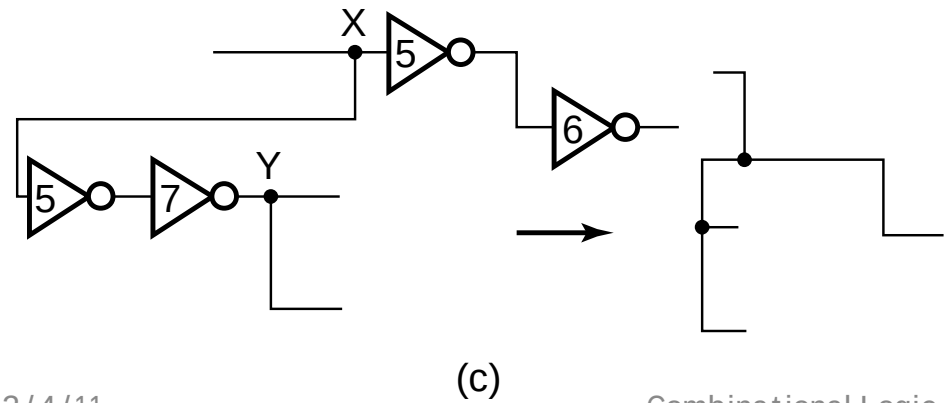
Map with a library containing inverters and 2-input NAND, and then map it to a CMOS based circuit

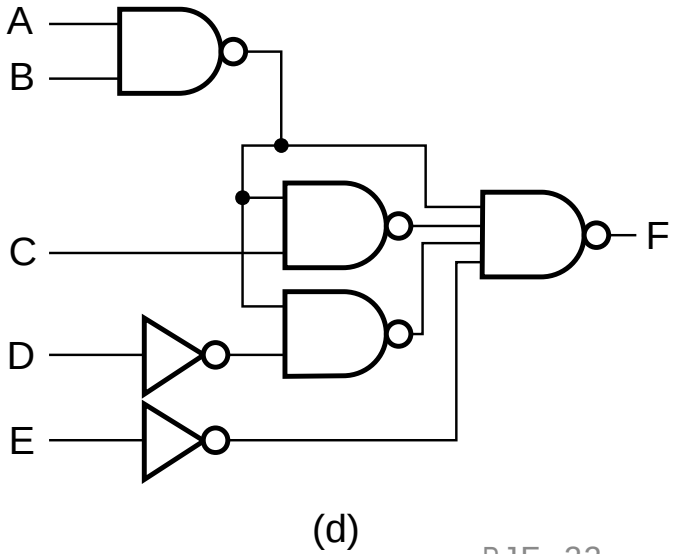


Z

Technology Mapping Example



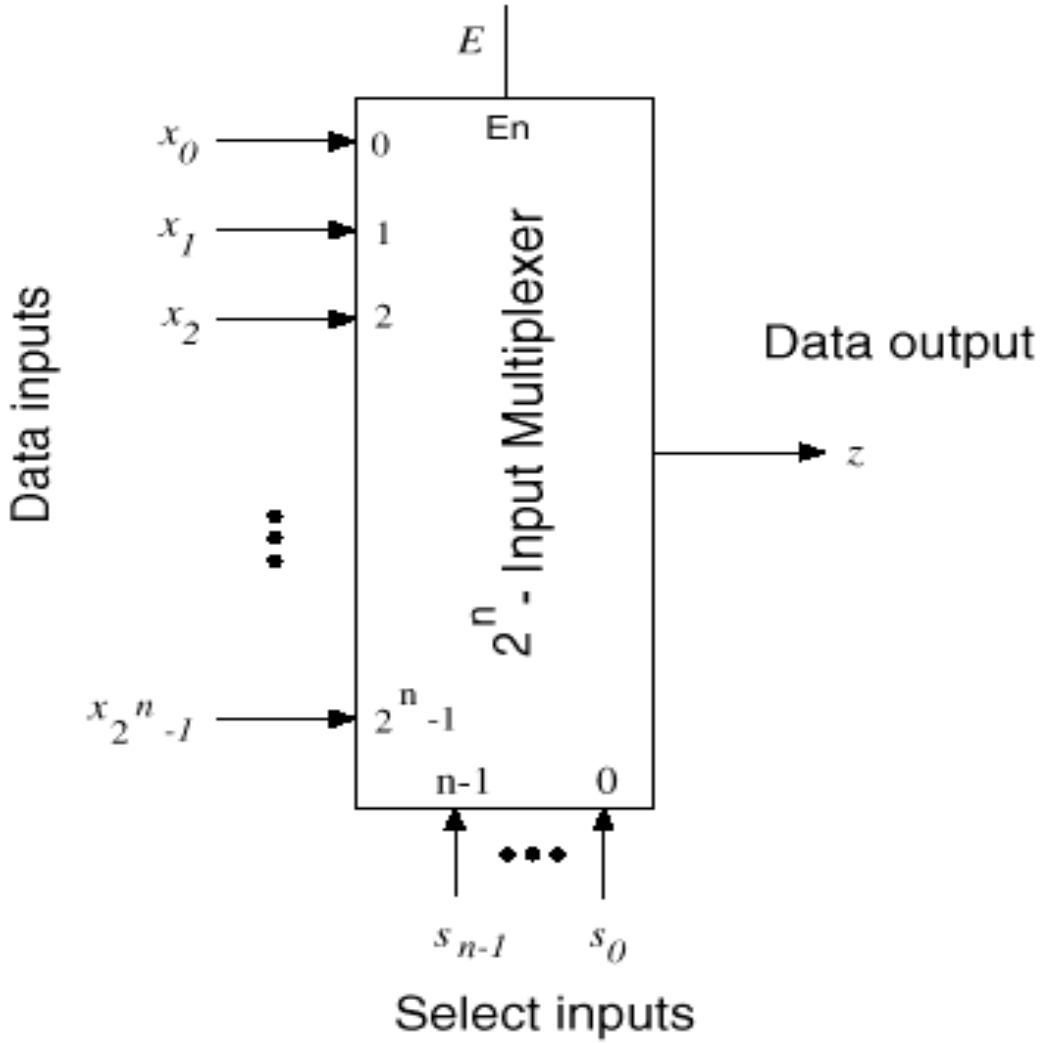




Transmission Gate Based Design - Multiplexer

- " Selects" binary information from one of many input lines and directs it to a single output line.
- Also know as the "selector" circuit,
- Selection is controlled by a particular set of inputs lines whose # depends on the # of the data input lines.
- For a 2^n-to-1 multiplexer, there are 2^n data input lines and n selection lines whose bit combination determines which input is selected.

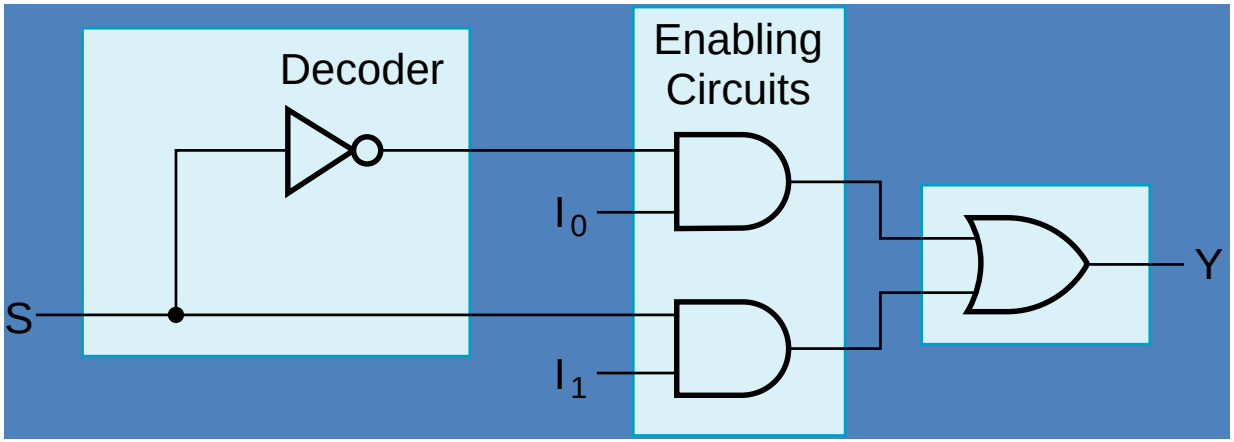
Multiplexer (cont.)



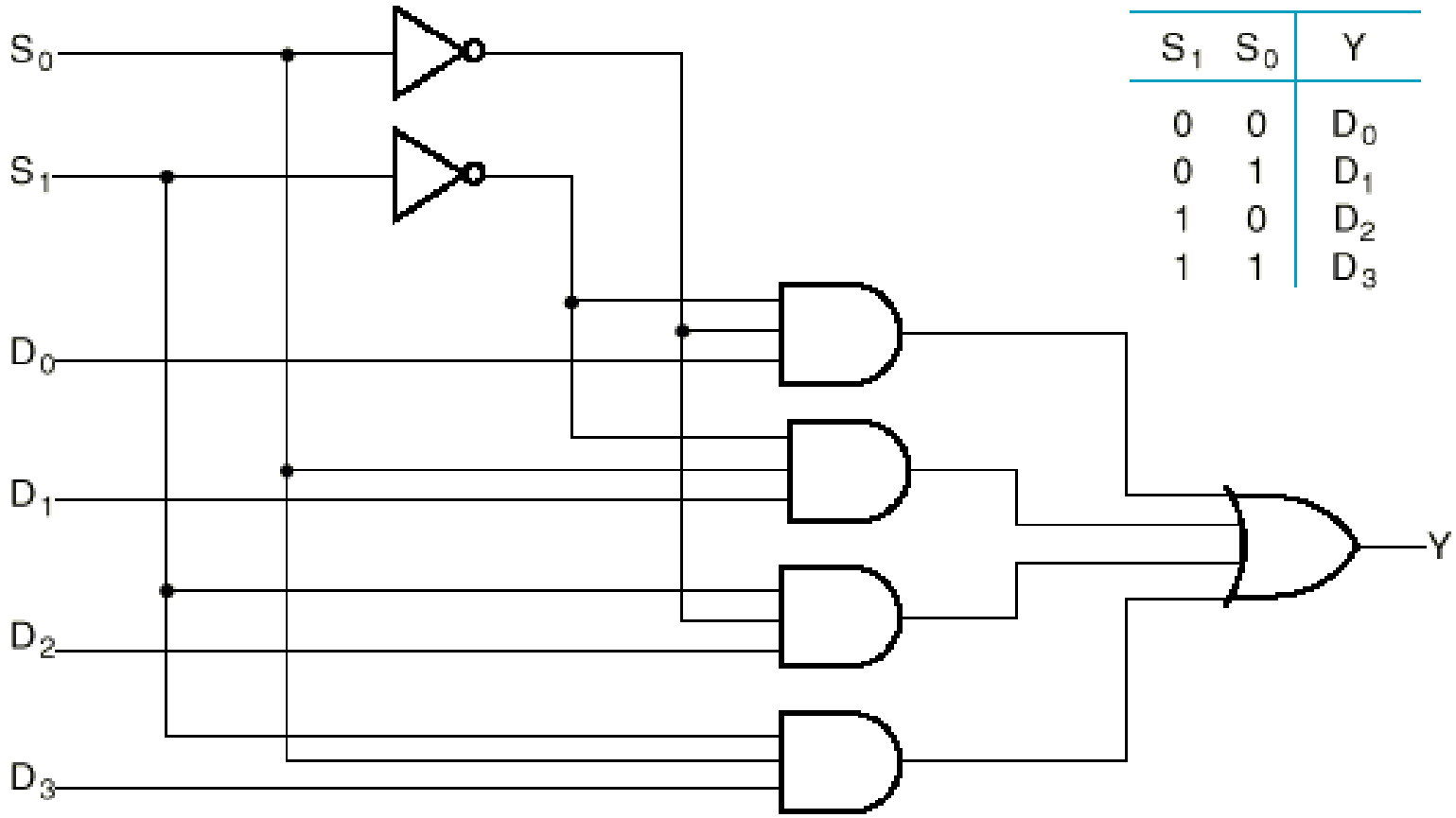
2-to-1-Line Multiplexer

- Since $2 = 2^1$, $n = 1$
- The single selection variable S has two values:
  - S = 0 selects input I_0
  - S = 1 selects input I_1
- The equation:

  $Y = S' I_0 + S I_1$
- The circuit:



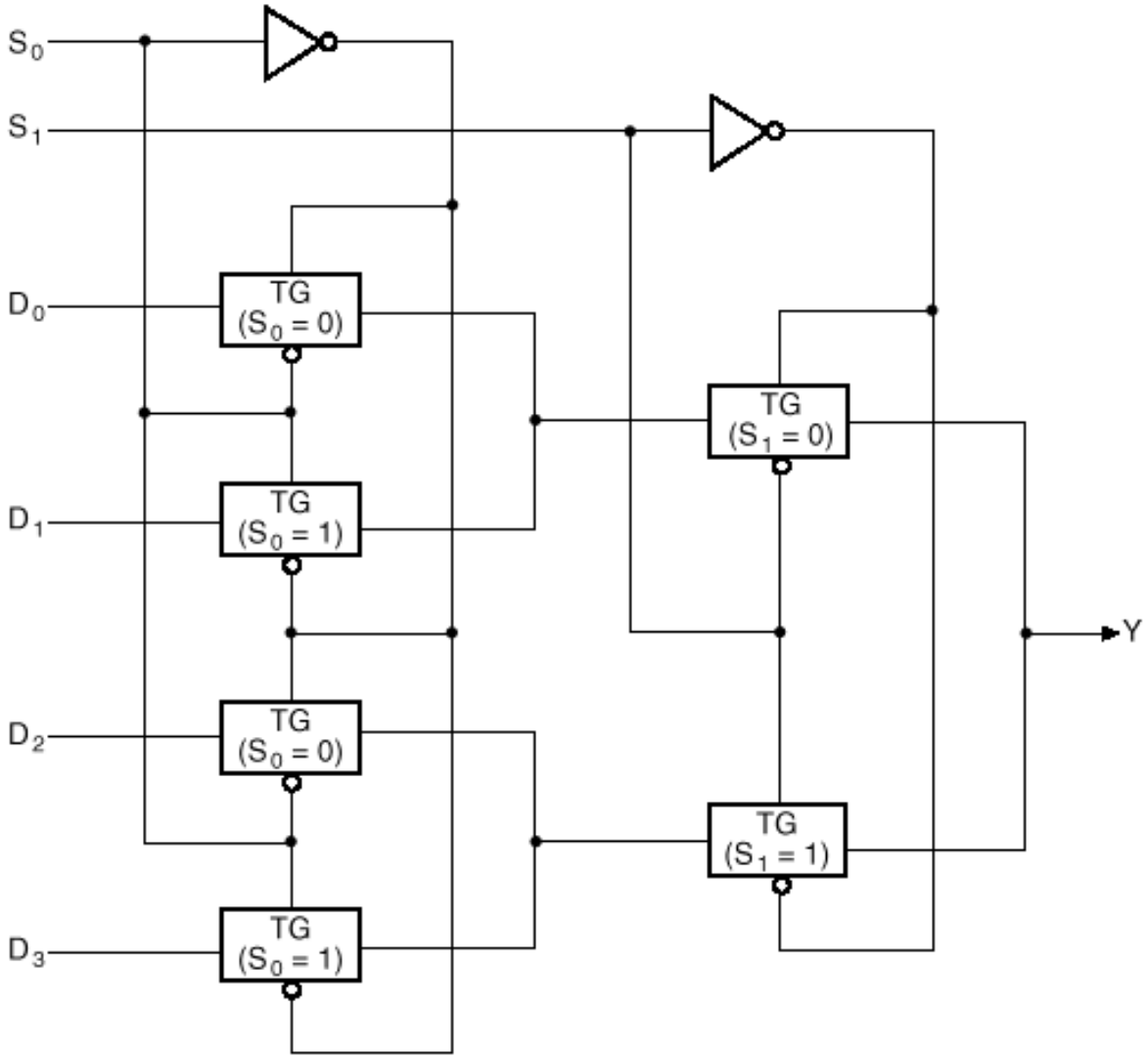
Example: 4-to-1 MUX - Cell Library Based Design



Function table

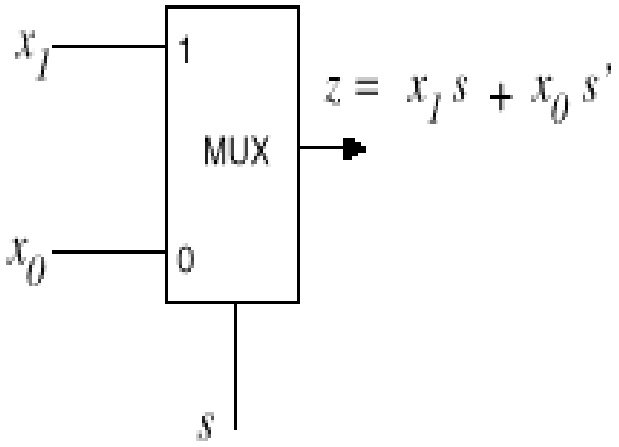
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

4–to–1-Line Multiplexer using Transmission Gates

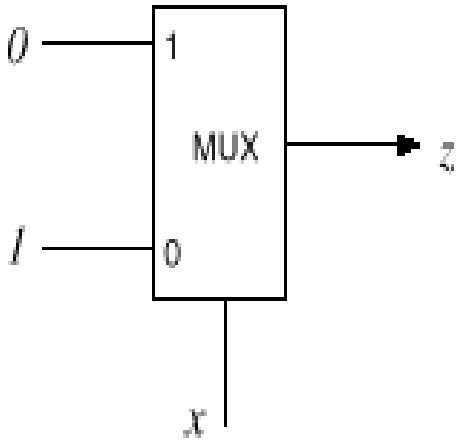


MUX as a Universal Gate

- We can construct AND and NOT gates using 2-to-1 MUXs. Thus, 2-to-1 MUX is a universal gate.

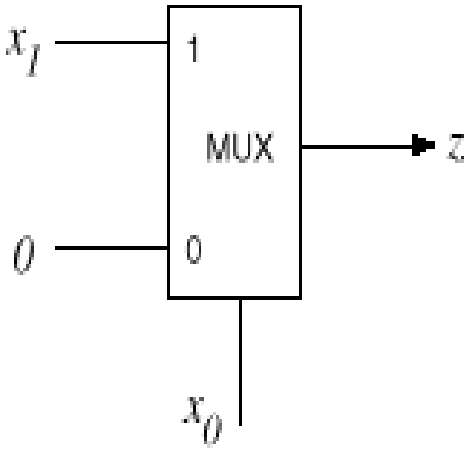


NOT



$z = 0x + 1x' = x'$

AND



$z = x_1x_0 + 0x_0' = x_1x_0$

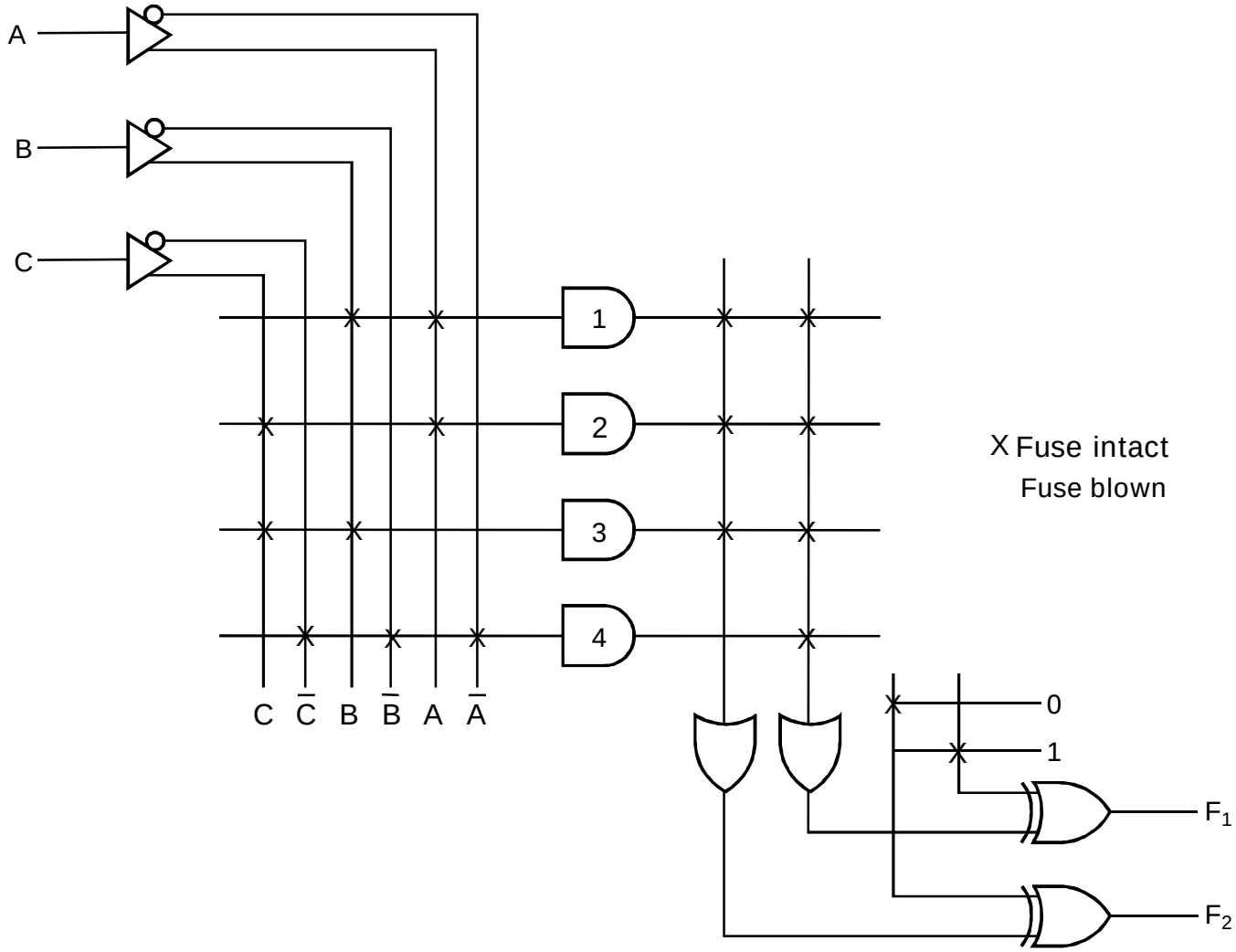
Programmable Logic Array

- The set of functions to be implemented is first transformed to product terms
- Since output inversion is available, terms can implement either a function or its complement

Programmable Logic Array Example

- To implement
  - F1= A'B'C+A'BC'+AB'C'=(AB+AC+BC+A'B'C')'
  - F2=AB+AC+BC

Programmable Logic Array Example



Summary

- Combinational Circuit
- Chip Design styles
  - Full-custom design
  - Cell library based design
  - Programmable Logic Array