MODULE 3: Boolean Algebra and Digital Logic

Lecture 3.4
Combinational Logic

Prepared By:
- Scott F. Midkiff, PhD
- Luiz A. DaSilva, PhD
- Kendall E. Giles, PhD

Electrical and Computer Engineering
Virginia Tech

Lecture 3.4 Objectives

- Describe the function of a binary adder and a “FullAdder” building block and show how this block can be used to construct a ripple-carry adder
- Describe the function and design of a multiplexer
- Describe the function and design of a demultiplexer
- Describe the function and design of a decoder

Combinational Logic Examples

- We consider four example combinational logic circuits
  - Ripple-carry adder (and full adder cell)
  - Multiplexer
  - Demultiplexer
  - Decoder
- All are commonly used in a variety of digital systems, including processors

Ripple-Carry Adder

- Need an adder for positive and negative integers
- Functional requirements – the adder shall:
  - Add two 4-bit signed numbers using two's complement representation
  - Produce a 4-bit signed sum, using two's complement representation, and an overflow indicator
- Note that there may be other requirements
  - Delay (affects clock rate)
  - Cost (to fabricate)
  - Reliability (mean time to failure)
  - Testability coverage (ability to test for hardware faults)

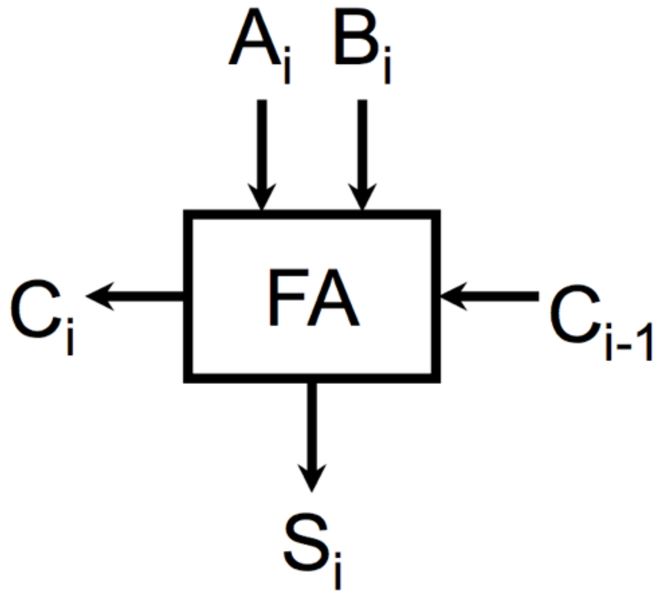
Decomposition of Binary Addition

- Procedure used to add two binary numbers

0100	Carry (C)
1011	A
0010	B
1101	Sum (S)

- At each bit in the addition, two operand bits (A_i and B_i) and a previous carry (C_{i-1}) generate a sum bit (S_i) and carry bit for next column (C_i)
- One 1-bit adder (full adder) performs the above addition
- *N*-bit adder requires *N* such units

Full Adder Building Block (for one bit)

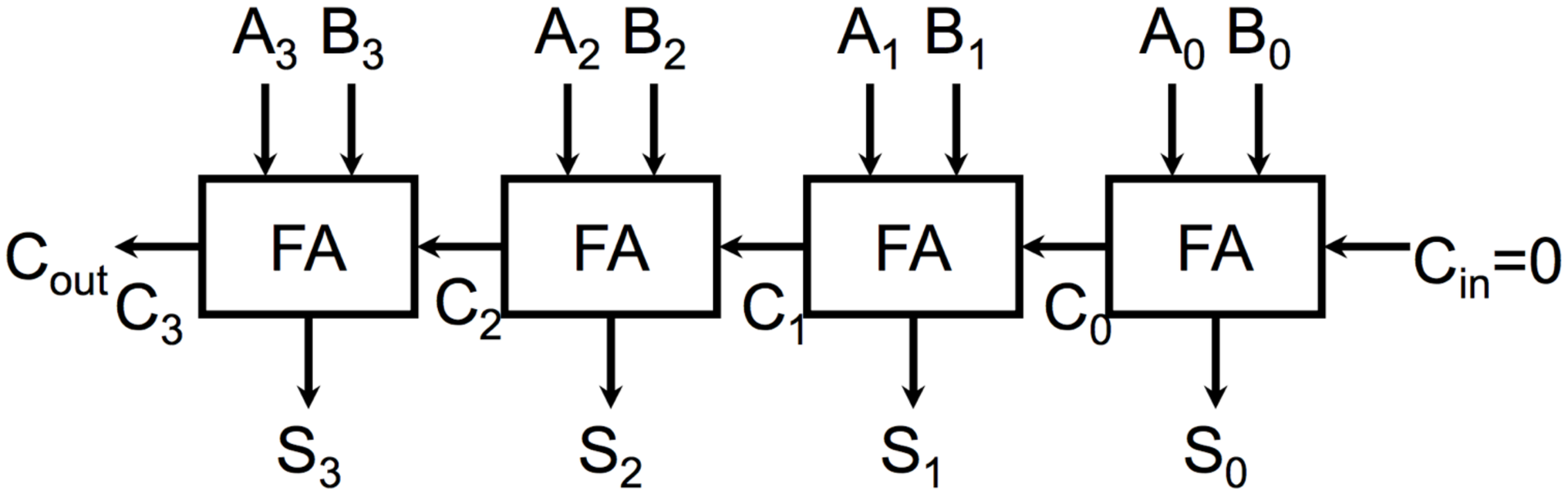


$$S_i = \overline{A_i}\,\overline{B_i}\,C_{i-1} + \overline{A_i}\,B_i\,\overline{C_{i-1}} + A_i\,\overline{B_i}\,\overline{C_{i-1}} + A_i\,B_i\,C_{i-1} = A_i \oplus B_i \oplus C_{i-1}$$

$$C_i = A_i B_i + A_i C_{i-1} + B_i C_{i-1}$$

4-Bit Ripple-Carry Adder

- 4-bit ripple-carry adder is built using four full adders (*N*-bit adder is built using *N* full adders)



$$OVF = C_3 \oplus C_2$$

CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

- Draw the Full Adder Building Block for 1 bit and write out the logic formulas for the outputs S_i and C_i.

If you have any difficulties, please review the lecture video before continuing.

Truth Table for the Full Adder

- Full adder's logical specification is shown
- Outputs: S_i and C_i
- Inputs: A_i, B_i, and C_{i-1} Overflow logic
- Overflow occurs if and only if the last two carries differ
- For 8-bit adder, last carries are C_6 and C_7, so $OVF=C_7 \oplus C_6$

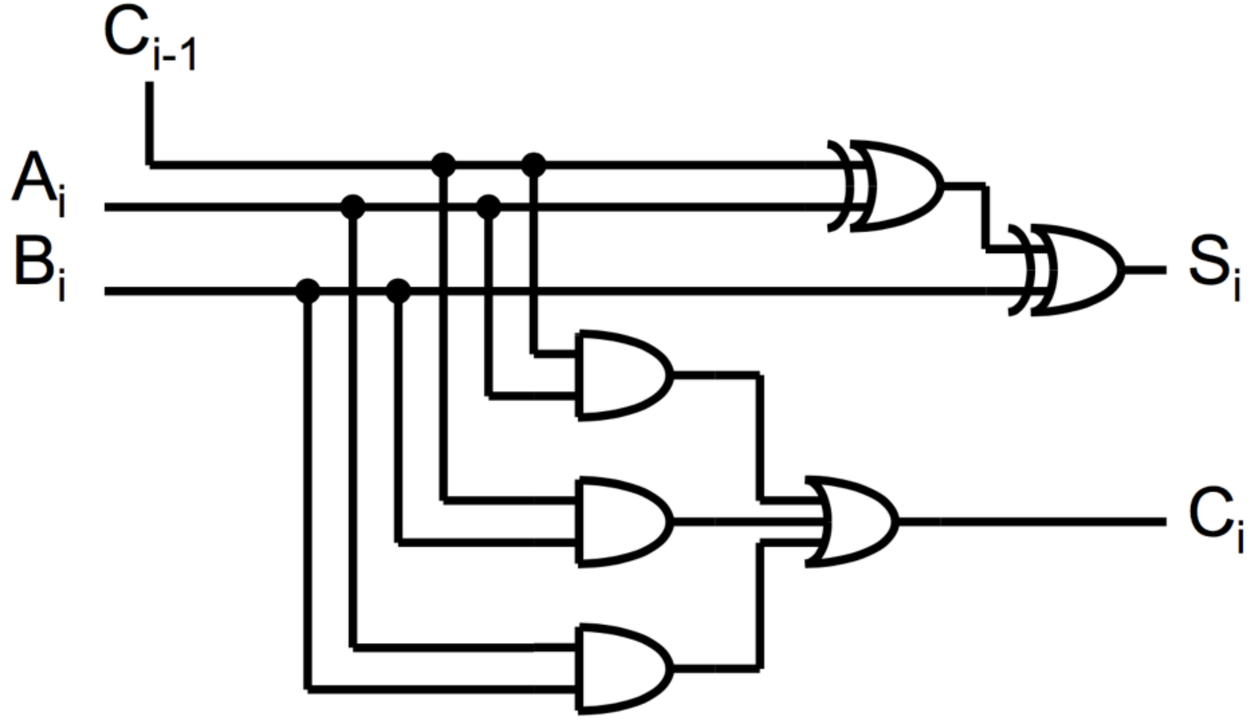
A_i	B_i	C_{i-1}	S_i	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Synthesis of the Full Adder

$$S_i = \overline{A}_i\overline{B}_iC_{i-1} + \overline{A}_iB_i\overline{C}_{i-1} + A_i\overline{B}_i\overline{C}_{i-1} + A_iB_iC_{i-1}$$

$$= A_i \oplus B_i \oplus C_{i-1}$$

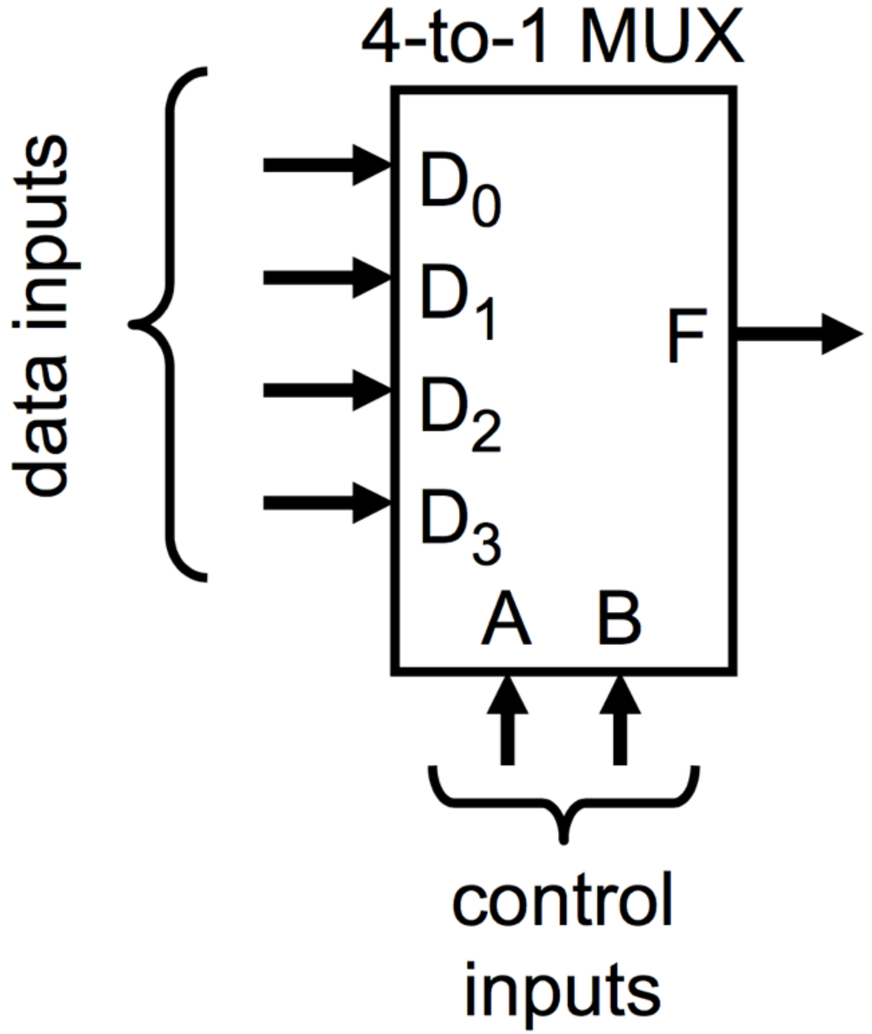
$$C_i = A_iB_i + A_iC_{i-1} + B_iC_{i-1}$$



Multiplexers

- Basic function of a multiplexer (MUX) is to select from several data inputs to create one output based on the value of control lines
- Used to select from several possible data sources to store a value in a register
- Used to select from different data lines in a multiplexed communications system

4-to-1 Multiplexer



A	B	F
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

"Truth table" describing the behavior of a 4-to-1 multiplexer

General Structure of Multiplexers

- A 2^N-to-1 multiplexer has:
  - N control lines
  - 2^N inputs lines
  - 1 output line
  - Optionally, one or more enable lines
- N control lines can represent 2^N different values, 0 through 2^N-1, inclusive
- Each value represented by the control lines selects a different input

CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

- Draw the diagram of a 4-to-1 Multiplexer and write out the “truth table” that describes the multiplexer’s behavior.

If you have any difficulties, please review the lecture video before continuing.

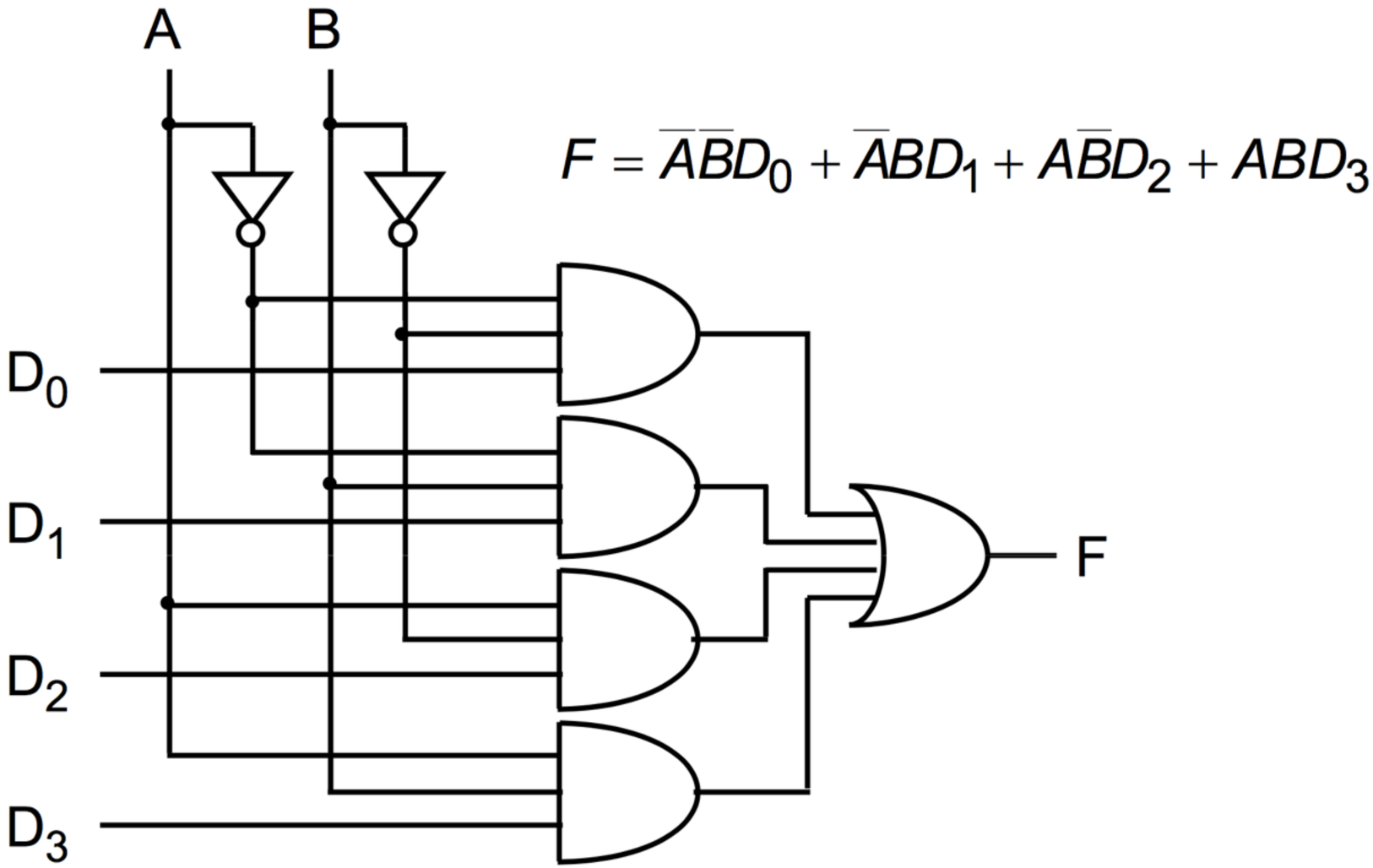
Implementation of a Multiplexer

- A MUX is a combinational logic unit and can be built from standard logic gates
- For example, a Boolean expression for the 4-to-1 MUX can be derived based on the truth table for the MUX

A	B	F
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

$$F = \overline{A}\,\overline{B}D_0 + \overline{A}BD_1 + A\overline{B}D_2 + ABD_3$$

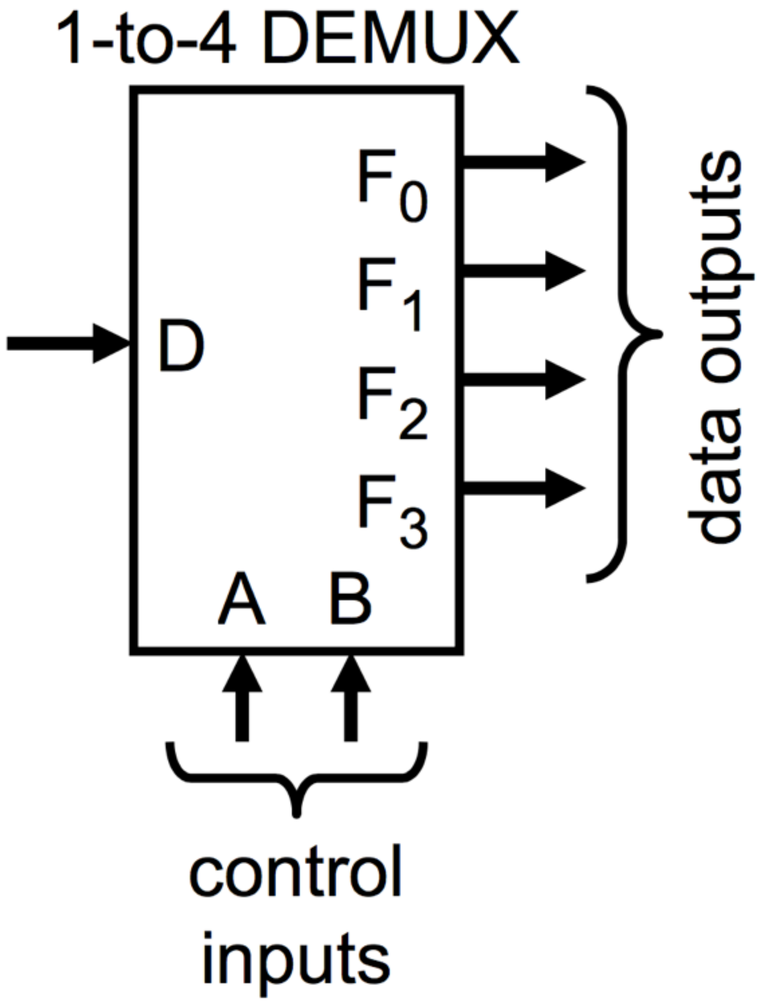
Implementation of a Multiplexer



Demultiplexers

- Basic function of a demultiplexer (DEMUX) is to output a input data value to just one of several data output signals based on the value of control lines
- A demultiplexer is the inverse of the multiplexer function
- Used to move the value on one data line to one of a set of different data output lines

1-to-4 Demultiplexer

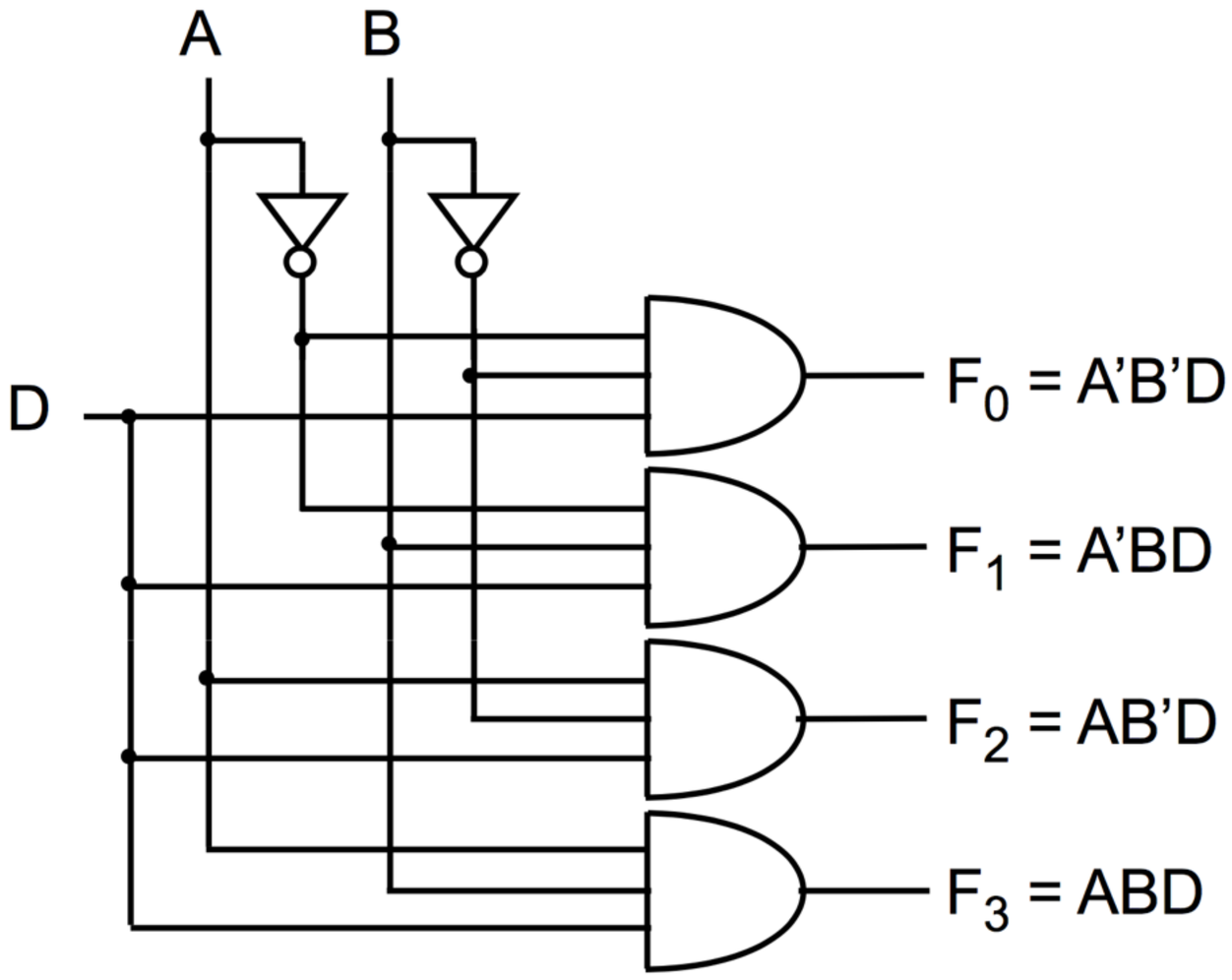


D	A	B	F_0	F_1	F_2	F_3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

General Structure of Demultiplexers

- A 1-2^N demultiplexer has:
  - N control lines
  - 1 input line
  - 2^N output lines
  - Optionally, one or more enable lines
- N control lines can represent 2^N different values, 0 through 2^N-1, inclusive
- Each value represented by the control lines selects a different output

Implementation of a Demultiplexer



CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

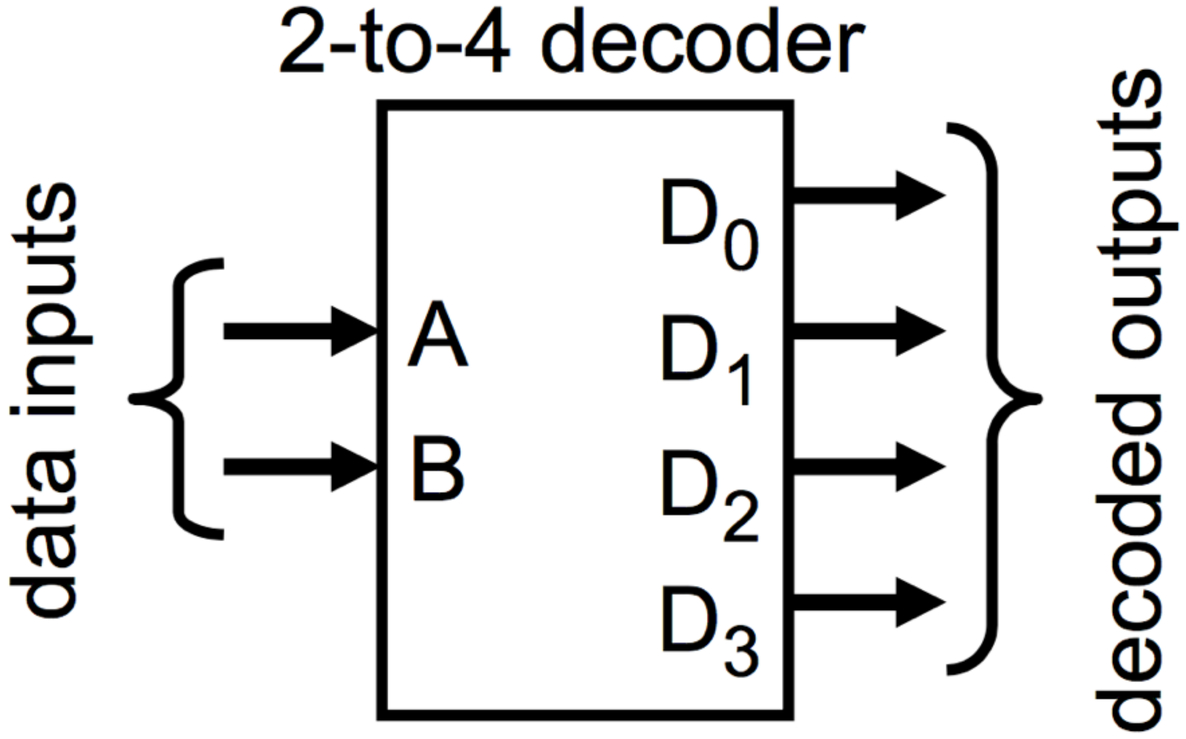
- Draw the diagram of a 1-to-4 Demultiplexer and write out the “truth table” that describes the demultiplexer’s behavior.

If you have any difficulties, please review the lecture video before continuing.

Decoders

- A decoder converts an encoded value on N input lines into a true value on one and only one of 2^N output lines
- Logical encoding is translated to a spatial location (separate signal)

2-to-4 Decoder



A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

General Structure of Decoders

- An N-to-2^N decoder has:
  - N input lines
  - 2^N output lines
  - Optionally, one or more enables lines
- N input lines can represent 2^N different values, 0 through 2^N-1, inclusive
- The corresponding output line is activate (logic 1), others are inactive (logic 0)

Implementation of a 2-to-4 Decoder (1)

- A decoder is a combinational logic unit and can be built from standard logic gates
- Multiple outputs, so the combination logic unit implements multiple functions
- Each output function

A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

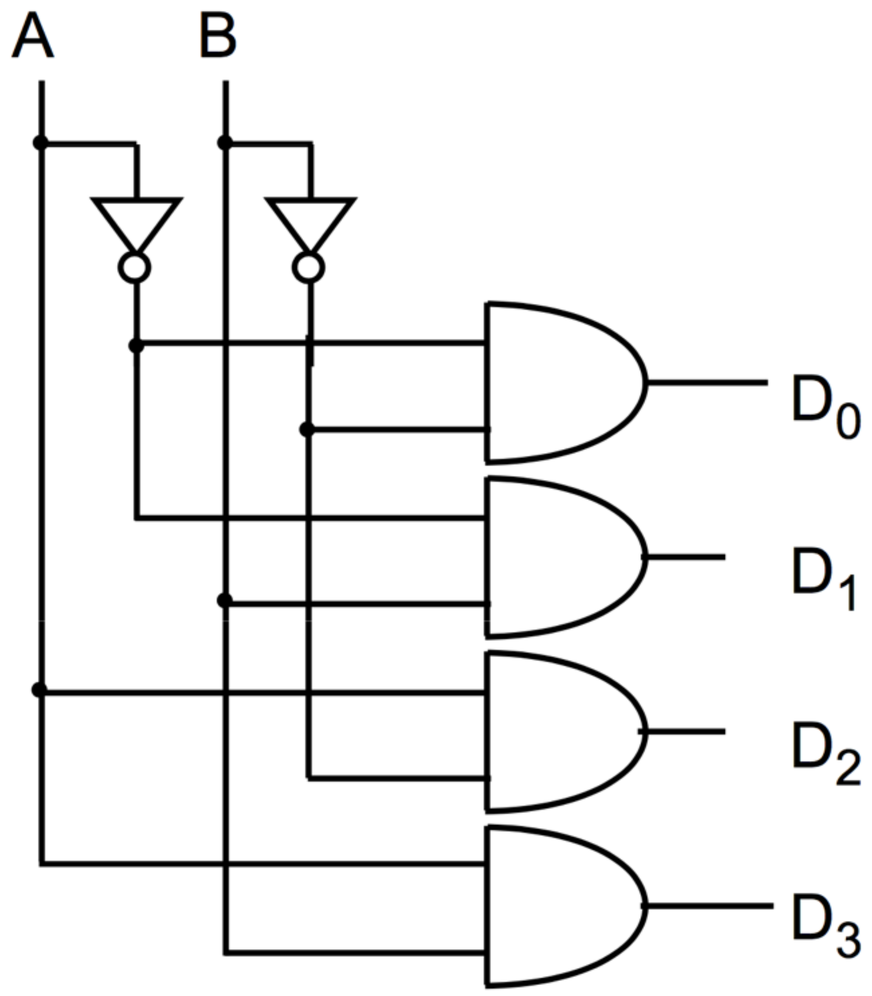
$$D_0 = \overline{A}\,\overline{B}$$

$$D_1 = \overline{A}B$$

$$D_2 = A\overline{B}$$

$$D_3 = AB$$

Implementation of a 2-to-4 Decoder (2)



$$D_0 = \overline{A}\,\overline{B}$$

$$D_1 = \overline{A}B$$

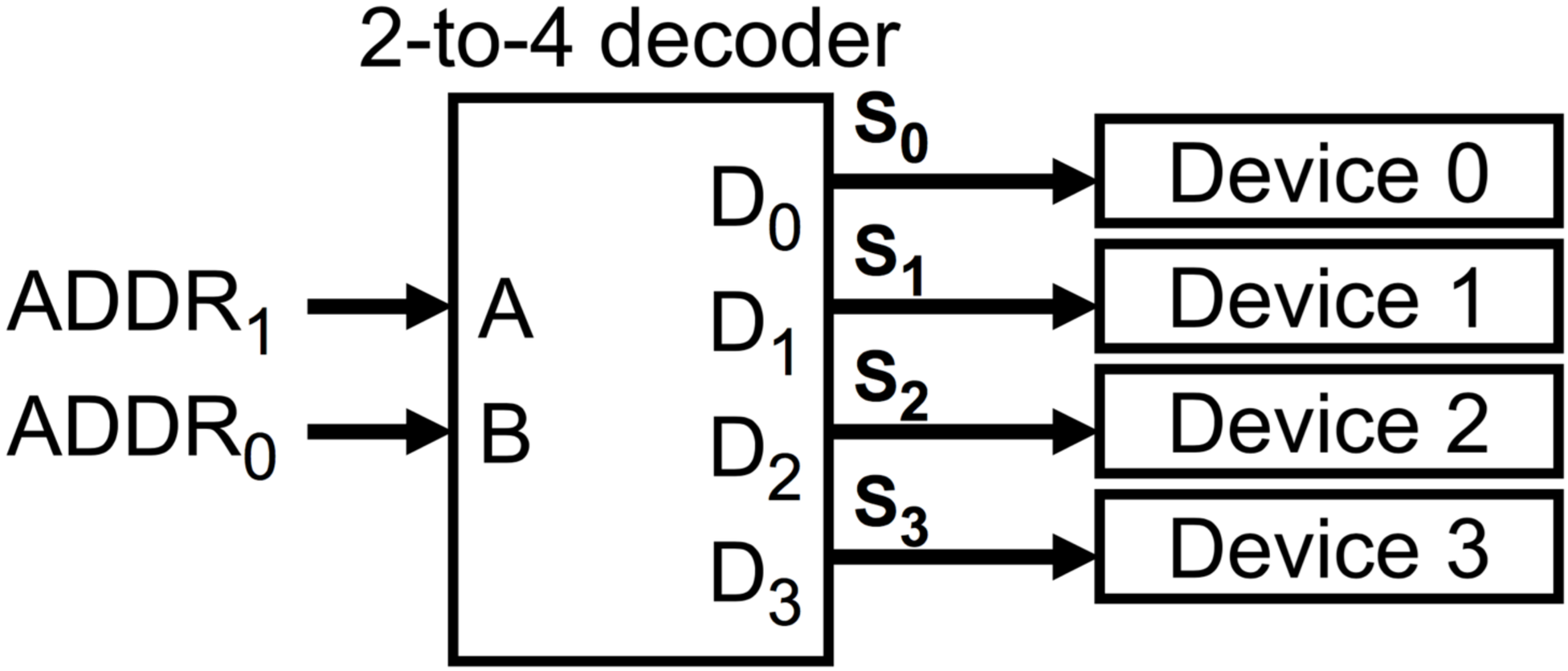
$$D_2 = A\overline{B}$$

$$D_3 = AB$$

Memory Select Example (1)

- Suppose we have four memory devices
- We select memory devices 0, 1, 2, and 3 using signals S_0, S_1, S_2, and S_3, respectively
- At any given time, we want to select exactly one of the devices based on bits in a memory address, $ADDR_0$ and $ADDR_1$
- A decoder can be used to "decode" the bits $ADDR_0$ and $ADDR_1$ to generate select lines
- Note that other address lines would be used to select specific locations within each memory device

Memory Select Example (2)



CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

- Draw the diagram of a 2-to-4 Decoder and write out the “truth table” that describes the decoder’s behavior.

If you have any difficulties, please review the lecture video before continuing.

Summary

- An *N*-bit ripple-carry adder is constructed using *N* full adder cells
- Full adder takes two bits and a carry as inputs
- Full adder generates a sum bit and a carry as outputs
- Multiplexers are used to select or merge data, while demultiplexers separate data
- Can be used in conjunction for time-division multiplexing
- A decoder translates N inputs to 2^N outputs
- Decoders can be used to select one out of multiple devices based on an address

MODULE 3: Boolean Algebra and Digital Logic

Lecture 3.4
Combinational Logic

Prepared By:
- Scott F. Midkiff, PhD
- Luiz A. DaSilva, PhD
- Kendall E. Giles, PhD

Electrical and Computer Engineering
Virginia Tech