

# CSCE 313

## Embedded Systems Programming

---

2003/1/22

### Spring 2003 - Lectures 4 & 5

### Micro-Computer Architecture

© 2003 Dr. James P. Davis

Some figures: Tanenbaum, 4<sup>th</sup> ed. © 1999 Prentice Hall Publishers, Inc.  
MacKenzie © 1996 Prentice Hall Publishers, Inc.

---

## Lecture 4 - Outline

---

### ● Objectives.

- ✓ We want to cover the basic structures of a microcomputer architecture.
- ✓ These notions (structure and semantics) are common to most microcomputer architectures, but we'll start with general properties, then specialize to the 68000.
- ✓ Our interest is in developing an understanding of how to express architecture and program design intent using assembler language, starting from a conceptual view, thus allowing program intent to be expressed in a largely language-independent manner.
- ✓ We will use several graphical notations as aids in this process (UML Use Case and Activity diagrams, ASM diagrams for CDFGs, Block diagrams for SFG decomposition). Most of our programs may be single threaded, but I'd like to get you into multi-threaded embedded systems design—since this is most of what we do in industry.

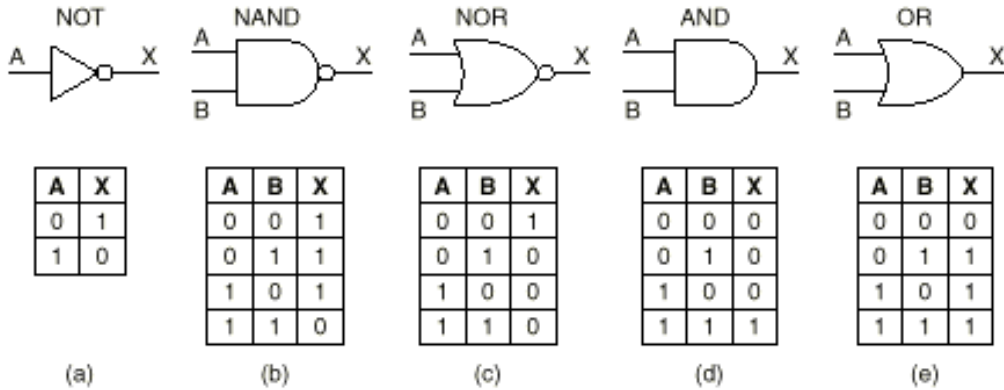
### ● Concepts.

- ✓ Function units – These perform specific functions in the context of the architecture. They can be *combinational logic* (output is fully dependent on the input and the logic function performed) or *sequential logic* (output is dependent on the input, but stores its value until signaled to latch a new value on its input).
- ✓ Interfaces – This defines how we talk to the functional unit, in terms of input and output signals, but also the timing constraints on these function block interfaces.

# Function Units – Gate Devices-1

Tanenbaum © 1999 Prentice Hall Publishing

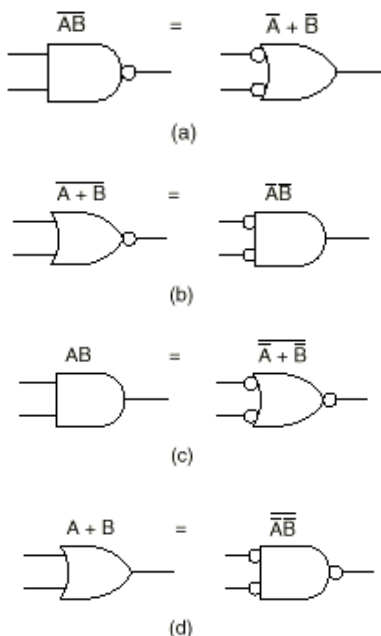
- Symbols and Truth tables for 5 basic gate-level logic gates: NOT, NAND, NOR, AND, OR.



© 2002 Dr. James P. Davis Page 3

# Function Units – Gate Devices-2

Tanenbaum © 1999 Prentice Hall Publishing



- Equivalent circuit functions and Boolean expressions:
  - ✓ (a) NAND, (b) NOR, (c) AND, (d) OR.
  - ✓ Often, we need to take a function and represent it using a different form of expression, or in different circuit realization.
  - ✓ This might be done for device technology support, or to improve speed, area or power usage in the design.
  - ✓ The expressions for these equivalent circuits can be derived using the Boolean identities.



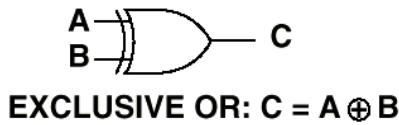
© 2002 Dr. James P. Davis Page 4

# Function Units – Gate Devices-3

- XOR Logic Function

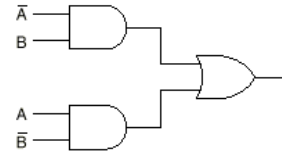
- ✓ There is generally no XOR gate in a VLSI technology library. It is realized using other gate structures.
- ✓ The Truth table for XOR and 3 different circuits that realize the function for two input signals.

Tanenbaum © 1999 Prentice Hall Publishing

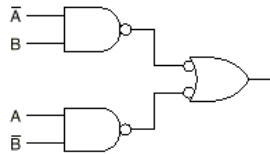


A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

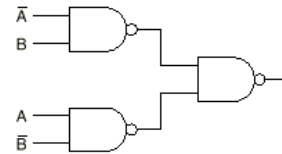
(a)



(b)



(c)



(d)

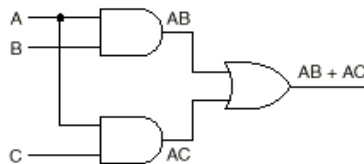


# Function Units – Gate Devices-4

Tanenbaum © 1999 Prentice Hall Publishing

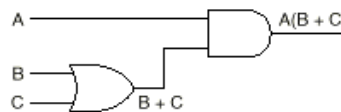
- Equivalent functions:

- ✓ It is possible to create equivalent circuits to implement the same logic function, as shown in this example.



A	B	C	AB	AC	AB + AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

(a)



A	B	C	A	B + C	A(B + C)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

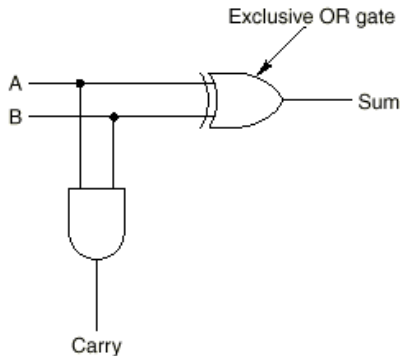
(b)



# Function Units – 1-bit Half Adder

Tanenbaum © 1999 Prentice Hall Publishing

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



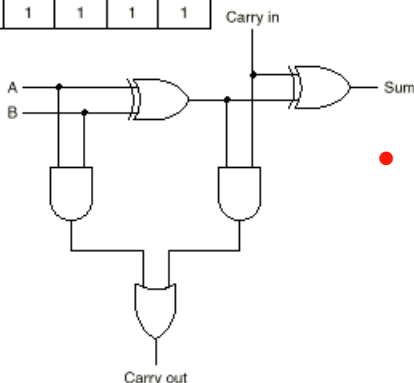
- Half Adder circuit:
  - ✓ This circuit takes two single-bit inputs and adds them together to produce a Sum as output.
  - ✓ The adder also generates a Carry signal, which can be used to connect to another adder element.
  - ✓ Multiple adders are connected together to implement a multi-bit adder circuit, and more complex arithmetic functions.
- Text Reference:
  - ✓ MacKenzie, pp. 25-28.
  - ✓ Note that my schematic is structured slightly differently, but it is the same circuit on p. 26.

© 2002 Dr. James P. Davis Page 7

# Function Units – 1-bit Full Adder

Tanenbaum © 1999 Prentice Hall Publishing

A	B	Carry in	Sum	Carry out
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

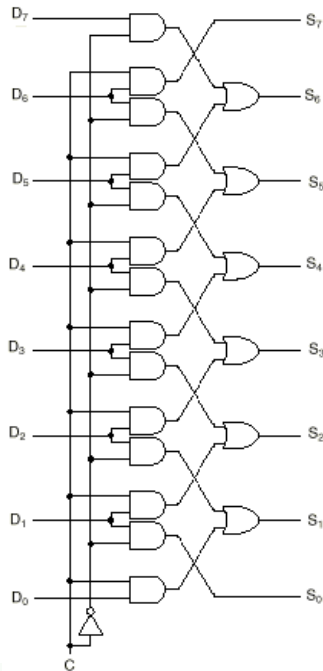


- Full Adder circuit:
  - ✓ This circuit takes two single-bit inputs and adds them together to produce a Sum as output.
  - ✓ The Full Adder also has a Carry (called a Carry Out) like the Half Adder.
  - ✓ The Full Adder also has a Carry In signal, allowing Carry Out from earlier adder stage to be connected.
  - ✓ This allows a multi-bit, multi-stage adder circuit to be constructed.
- Text Reference:
  - ✓ MacKenzie, pp. 25-28.
  - ✓ Note that my schematic is structured differently, but it is the same circuit on p. 26.

© 2002 Dr. James P. Davis Page 8

# Function Units – Left & Right Logical Shift

Tanenbaum © 1999 Prentice Hall Publishing

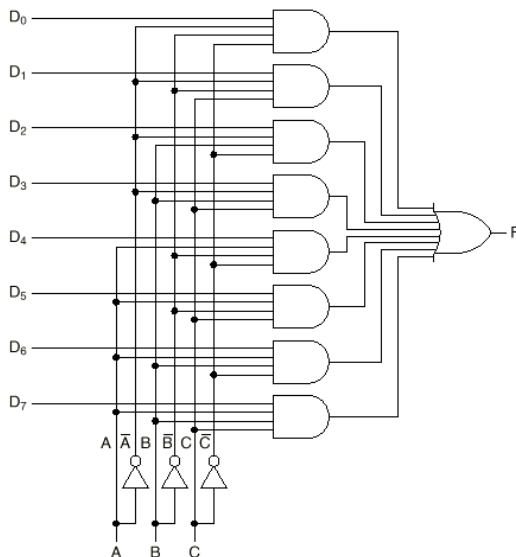


- 1-bit Left-Right Shifter (LSHFT, RSHFT):
  - ✓ The n-bit input signal (D0 – D7) is shifted either Left or Right by one bit.
  - ✓ The shifted value is propagated to the output (S0 – S7).
  - ✓ The bit-width of the input must equal the bit-width of the output.
  - ✓ If we shift Left (downward in the figure), the value on input signal D0 is lost.
  - ✓ If we shift Right (upward in the figure), the value on input D7 is lost.

© 2002 Dr. James P. Davis Page 9

# Function Units – 8:1 Multiplexer

Tanenbaum © 1999 Prentice Hall Publishing

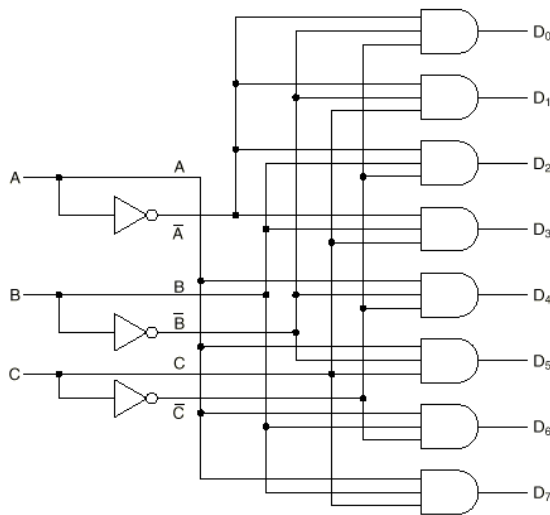


- 8-input Multiplexer (MUX):
  - ✓ MUX allows the signal value of one of its data inputs (D0 – D7) to pass to the output F.
  - ✓ The selection of the signal to be passed is controlled by SELECT lines (A, B, C).
  - ✓ The number of select lines, n, is based on a power of 2 for the number of inputs, m. So, if we have m inputs, we'll need n select lines so that  $2^n = m$ .
  - ✓ The MUX inputs and output must be the same width, and the SELECT lines are 1-bit each.

© 2002 Dr. James P. Davis Page 10

# Function Units – 3:8 Decoder

Tanenbaum © 1999 Prentice Hall Publishing



- Text Reference
  - ✓ MacKenzie, pp. 28-29.
  - ✓ Application: address decoding.

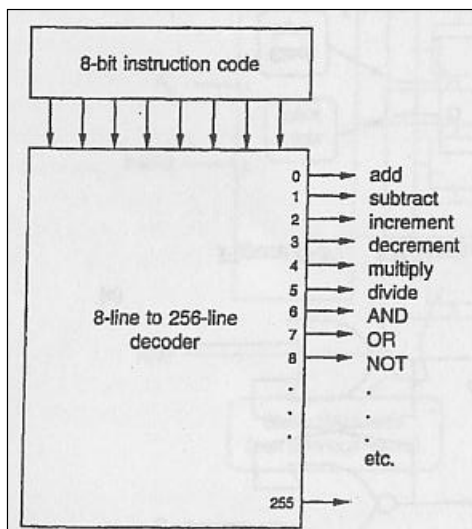


- 3:8 Decoder ( $n$  to  $2^n$  DECO):
  - ✓ DECO takes a binary encoded input of  $n$  data bits, and decodes it into individual data output lines, where one output ( $D_0 - D_7$ ) is enabled, depending on whether the encoded value corresponds to the data line number.
  - ✓ A Decoder input with  $n$  lines means we can encode  $2^{**}n$  possible binary encoded values.
  - ✓ With  $2^{**}n$  possible encoded values on the input, we'll need exactly  $n$  output lines, one for each possible encoded input value.
  - ✓ The output line corresponding to the decoded value will be enabled.

© 2002 Dr. James P. Davis Page 11

# Function Units – Decoder Application

MacKenzie © 1995 Prentice Hall Publishing



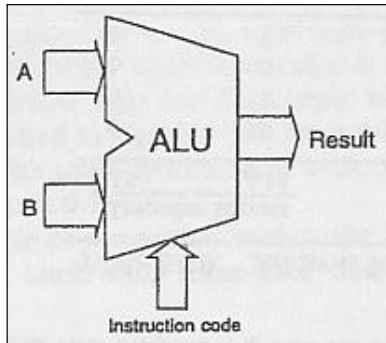
- 8:256 Instruction Decoder
  - ✓ We input a binary encoded instruction value, from an 8-bit instruction code (the "opcode").
  - ✓ The Decoder output is a set of up to 256 individually decoded control signals.
  - ✓ These signals could be used to enable certain blocks of logic in the CPU to carry out specific arithmetic or logical functions, given some set of staged input data brought in from elsewhere in the block.
  - ✓ The application of input to output of the Decoder would generally be under the control of a synchronizing system clock signal.



© 2002 Dr. James P. Davis Page 12

# Function Units – The ALU

MacKenzie © 1995 Prentice Hall Publishing



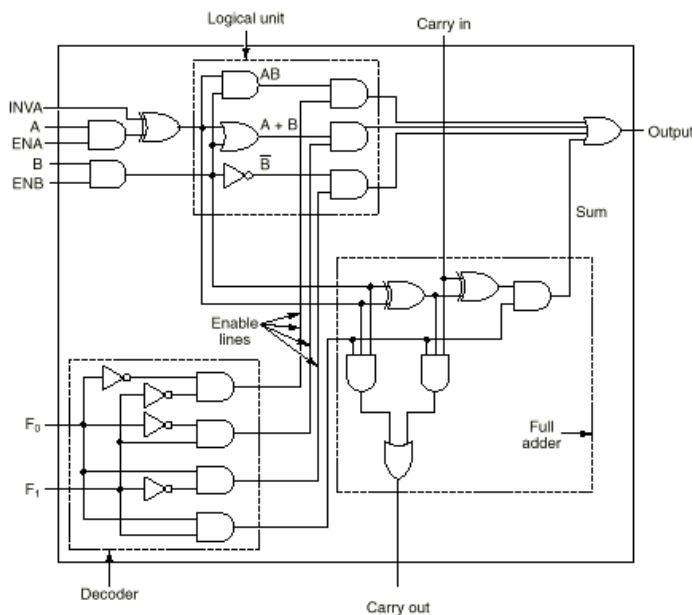
- The Arithmetic Logic Unit (ALU)
  - ✓ The ALU provides many functions that are selectable using control signals.
  - ✓ The ALU has an arithmetic circuit, a logic circuit, and decoding logic to determine which function to select.
  - ✓ Only one ALU function can be selected at a time.
  - ✓ ALUs are integral component of a CPU (they provide most of the function for the “Execute” cycle).



© 2002 Dr. James P. Davis Page 13

# Function Units – The ALU

Source: Tanenbaum, 4<sup>th</sup> Edition, 1999.



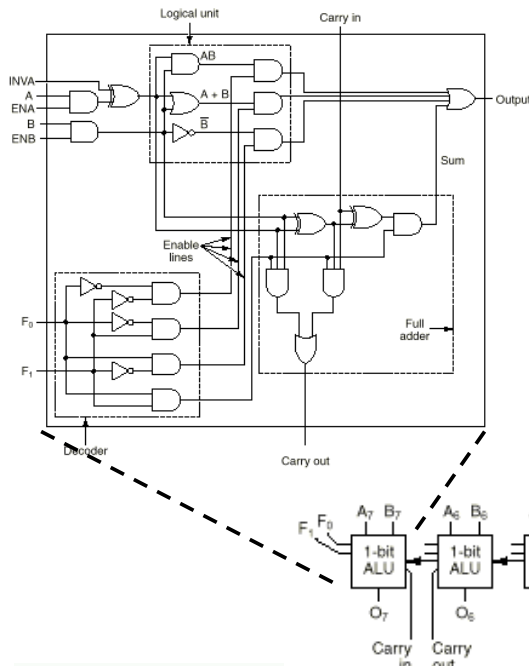
- 1-bit Arithmetic Logic Unit (ALU)
  - ✓ The ALU function is provided as a single-bit operation, built up of gate-level combinational logic.
  - ✓ The multi-bit ALU is created as a result of combining many single-bit ALUs in a cascaded configuration.
  - ✓ This ALU can be analyzed by first constructing a truth table to obtain the Boolean equations.



© 2002 Dr. James P. Davis Page 14

# Function Units – The ALU

Source: Tanenbaum, 4<sup>th</sup> ed. © 1999, Prentice-Hall



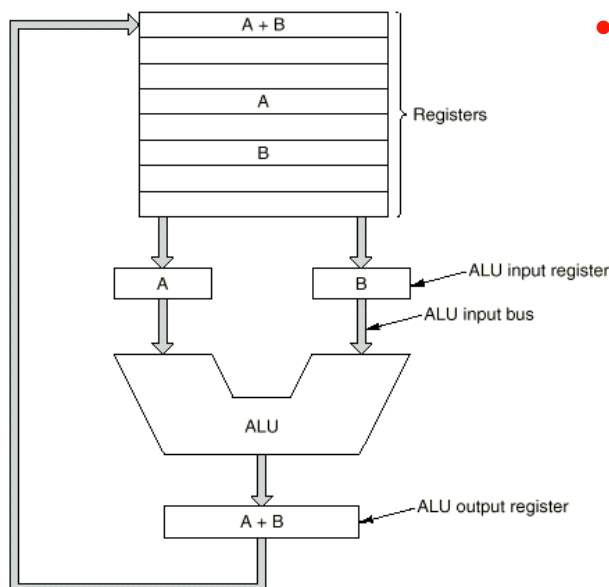
## Arithmetic Logic Unit (ALU)

- ✓ Provides the arithmetic, logic and other data functions in a single package.
- ✓ ALU functions are selectable using control signals.
- ✓ Individual gate-level elements are cascaded together to form an ALU of the computer's word length.
- ✓ NOTE: The ALU incorporates the Full Adder model.



© 2002 Dr. James P. Davis Page 15

# Function Unit – The ALU in the CPU



## The ALU Resource:

- ✓ Carries out the core of the Execution of program instructions involving operations on data.
- ✓ The ALU resources in the CPU must be scheduled according to operations on each CPU clock cycle.
- ✓ Program instructions stored in memory and fetched, in sequence, refer to the operands on which computation is performed by the ALU.
- ✓ Results of ALU execution are stored in Registers, to be later written back to system memory.

Source: Tanenbaum, 4<sup>th</sup> ed. © 1999, Prentice-Hall



© 2002 Dr. James P. Davis Page 16