MODULE 3: Boolean Algebra and Digital Logic

Lecture 3.7
Sequential Logic

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

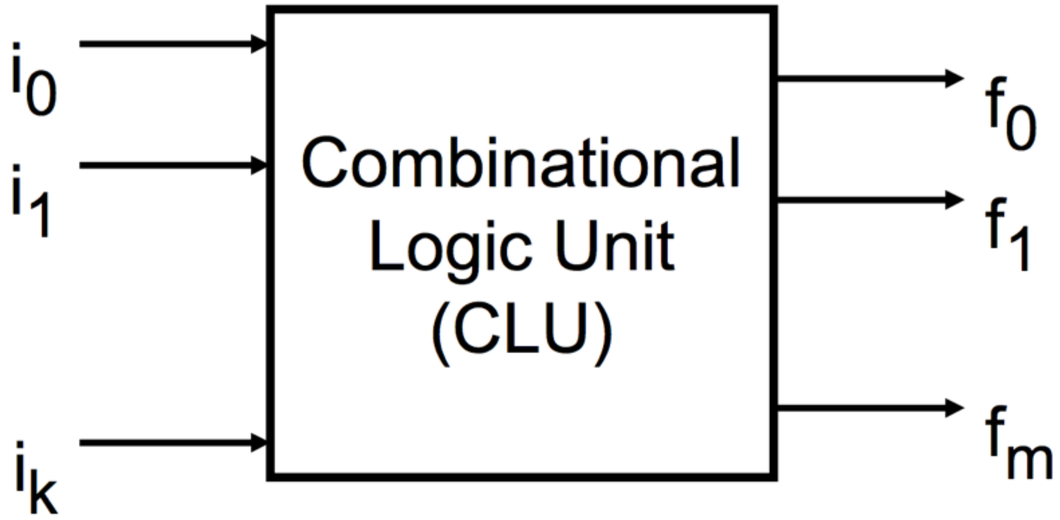
Electrical and Computer Engineering
Virginia Tech

Lecture 3.7 Objectives

- Define the difference between combinational and sequential logic
- Describe the operation of a finite state machine (FSM), including the role of inputs, outputs, and state bits
- Describe the operation of a simple register built from D flip-flops
- Describe some functions of simple counters

Combinational Logic

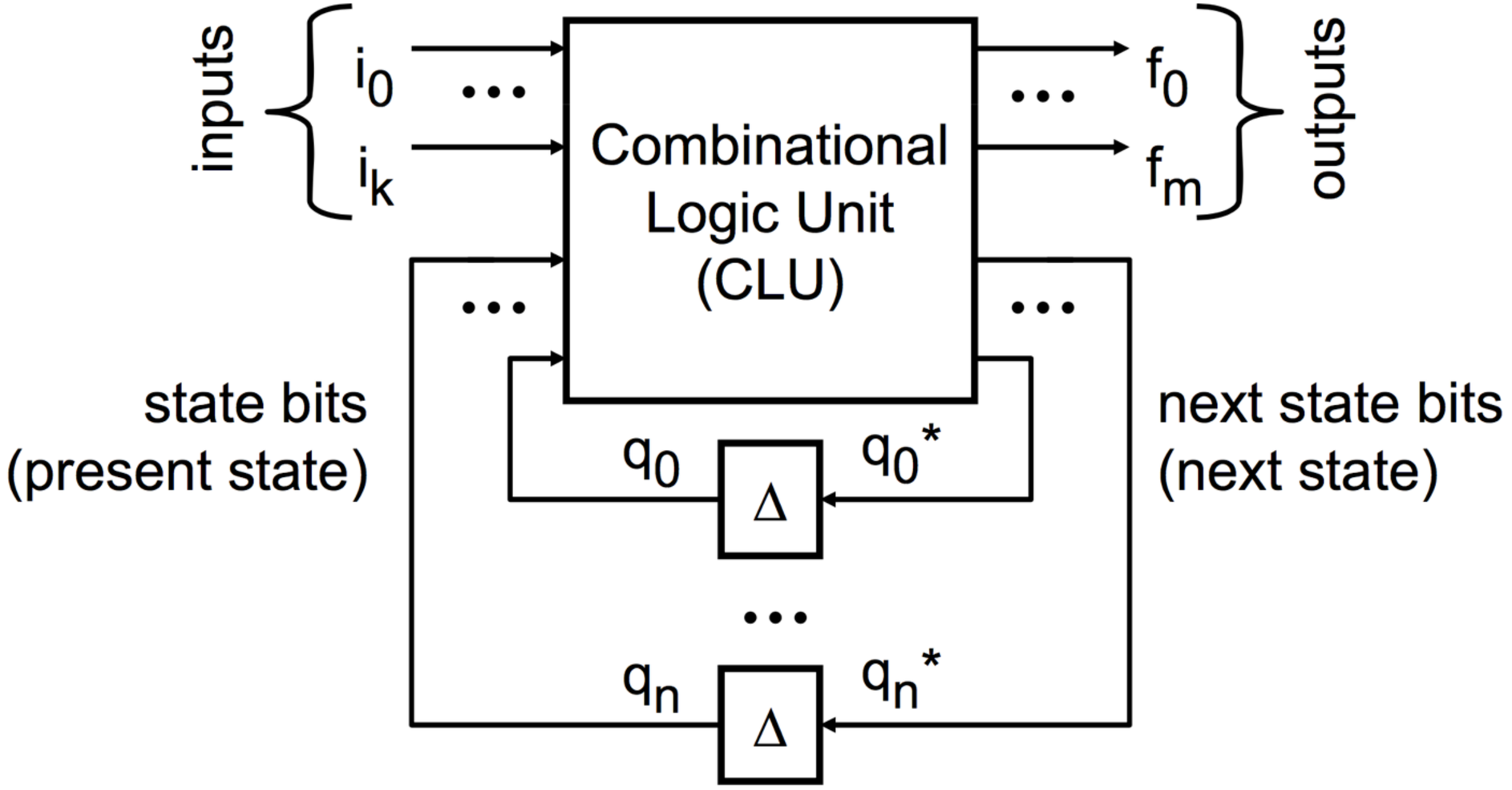
- The outputs of a combinational logic unit (CLU) depend only on the current inputs
  - $f_j(i_0, i_1, \ldots, i_k)$ for j=0,...,m
- There is no memory and, thus, no way to "remember" past inputs or state information



Sequential Logic

- A sequential logic unit, or finite state machine (FSM), maintains state information
- The “outputs” of a finite state machine are:
  - The outputs
  - The next state of the FSM, represented as state bits
- The outputs and the next state depend on:
  - The present inputs
  - The present state
- Typically, a complete digital system, like a processor, is a finite state machine — often with many possible states

Finite State Machine Model (1)



Finite State Machine Model (2)

- Outputs depend on inputs and present state bits ("Mealy" model) or just the present state bits ("Moore" model)
  - Mealy model: $f_j\ (i_0,..., i_k, q_0, ..., q_n)$
  - Moore model: $f_j\ (q_0, ..., q_n)$
- Next state bits depend on inputs and present state bits
  - $q_j^*(i_0, \ldots, i_k, q_0, \ldots, q_n)$
- Memory is required to store the state bits
  - Usually accomplished by flip-flops
  - Each flip-flop stores one bit of information
  - The next state bit is stored in the flip-flop, usually synchronized by a clock signal

CHECK POINT

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

- Draw and label the model of a finite state machine

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

State Transition Diagrams (1)

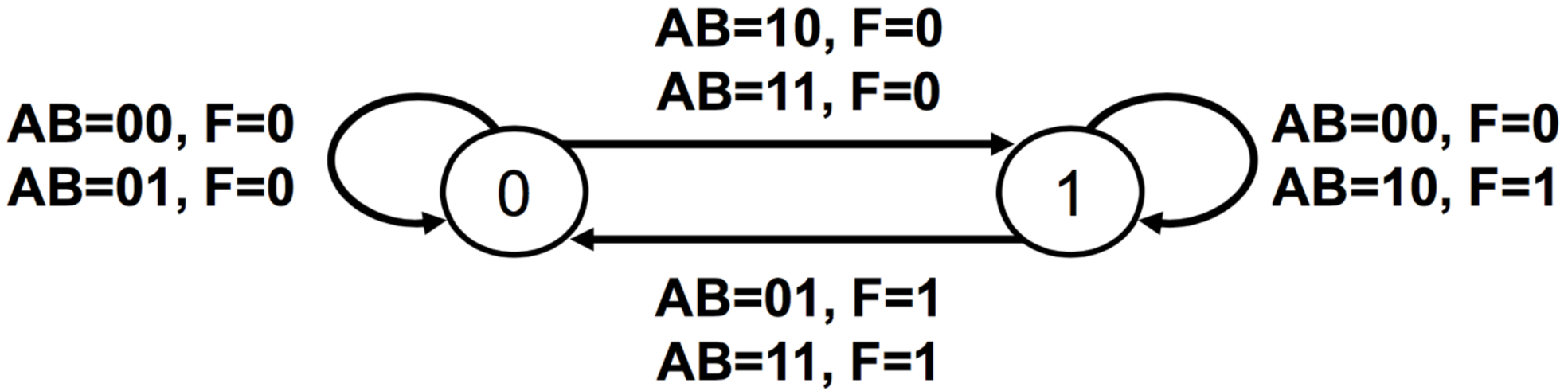
- A state transition diagram is a graph representation of the behavior of the FSM
- Graph nodes
  - Nodes represent the states
  - Label is the present state (for example, value of Q_t)
- Graph edges or arcs
  - Directed edges indicate a transition from one state to another
  - Label is the input condition causing the transition, e.g. values of A and B

State Transition Diagrams (2)

- Specifying outputs
  - On the edges for a Mealy machine (depends on state and input)
  - On the nodes for a Moore machine (depends only on state)

Example State Transition Diagram

- Two nodes, for $Q_t = 0$ and $Q_t = 1$
- For each node, a transition is specified for each possible combination of inputs A and B
- An output is specified for each present state and each input combination since this is a “Mealy machine”



Example Sequential Circuits

- Let's consider two examples of widely used sequential circuits
  - A register to store *N* bits of information
  - A counter used to generate a sequence of *N*-bit values (for example, representing 0, 1, 2, 3, 0, 1, 2, 3...)

Register Functions (1)

- Registers are collections of flip-flops to hold a group of N bits as an N-bit word
- Need one flip-flop for each of the N bits
  - For example, a 1-byte (8-bit) register requires 8 flip-flops
- Standard operations for an N-bit register
  - LOAD or WRITE: Store N input bits into the register
  - ENABLE or READ: Drive the register contents using tri-state buffers

Register Functions (2)

- Some registers provide conversion to or from serial data streams
  - Parallel-to-serial conversion
  - Serial-to-parallel conversion

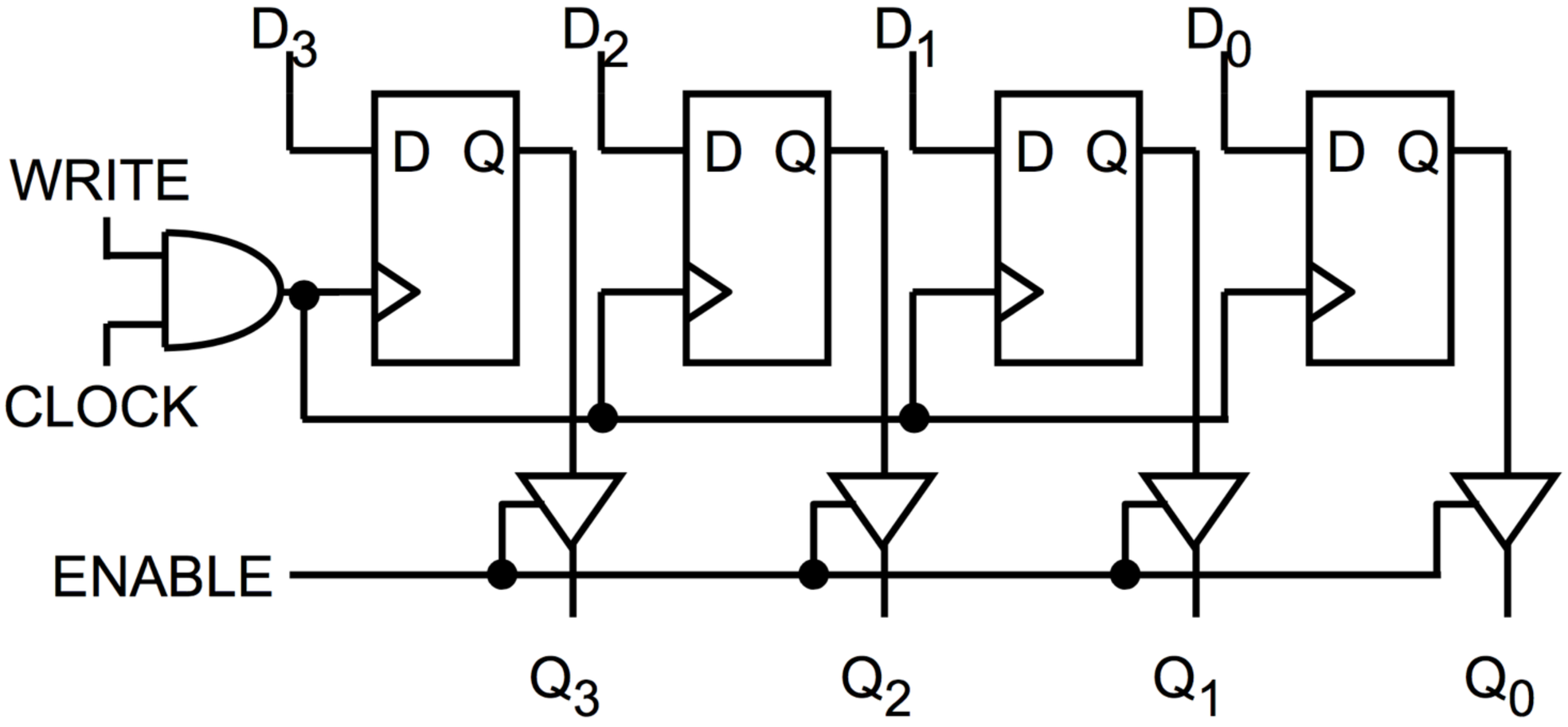
CHECK POINT

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

- Describe the relationship between registers and flip-flops

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

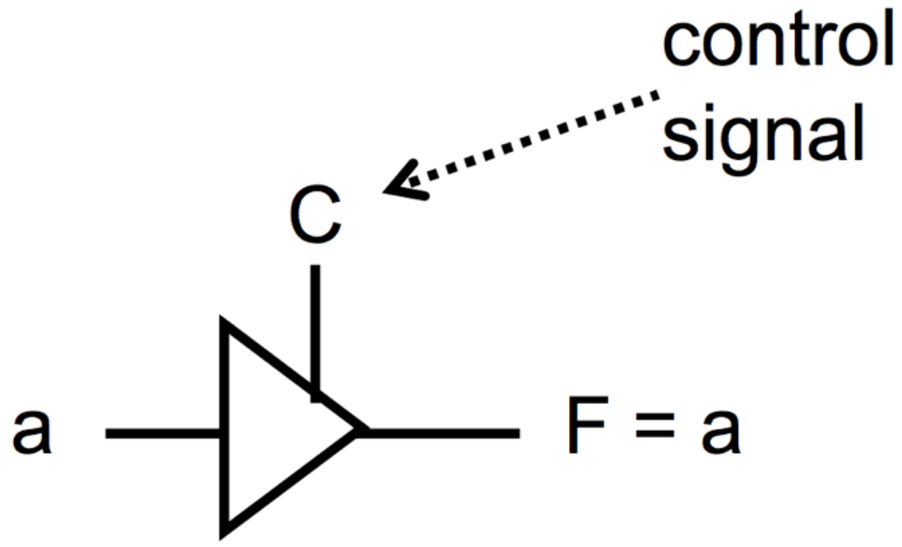
Simple Four-Bit Register



Tri-State Buffers

- Tri-state buffers are special logic gates that can effectively disable the output, so it goes into a “high impedance” state
  - Output can be enabled, and is 0 (LOW) or 1 (HIGH)
  - Output can be disabled, denoted ∅ (or “Z”)
- Used to implement registers and busses

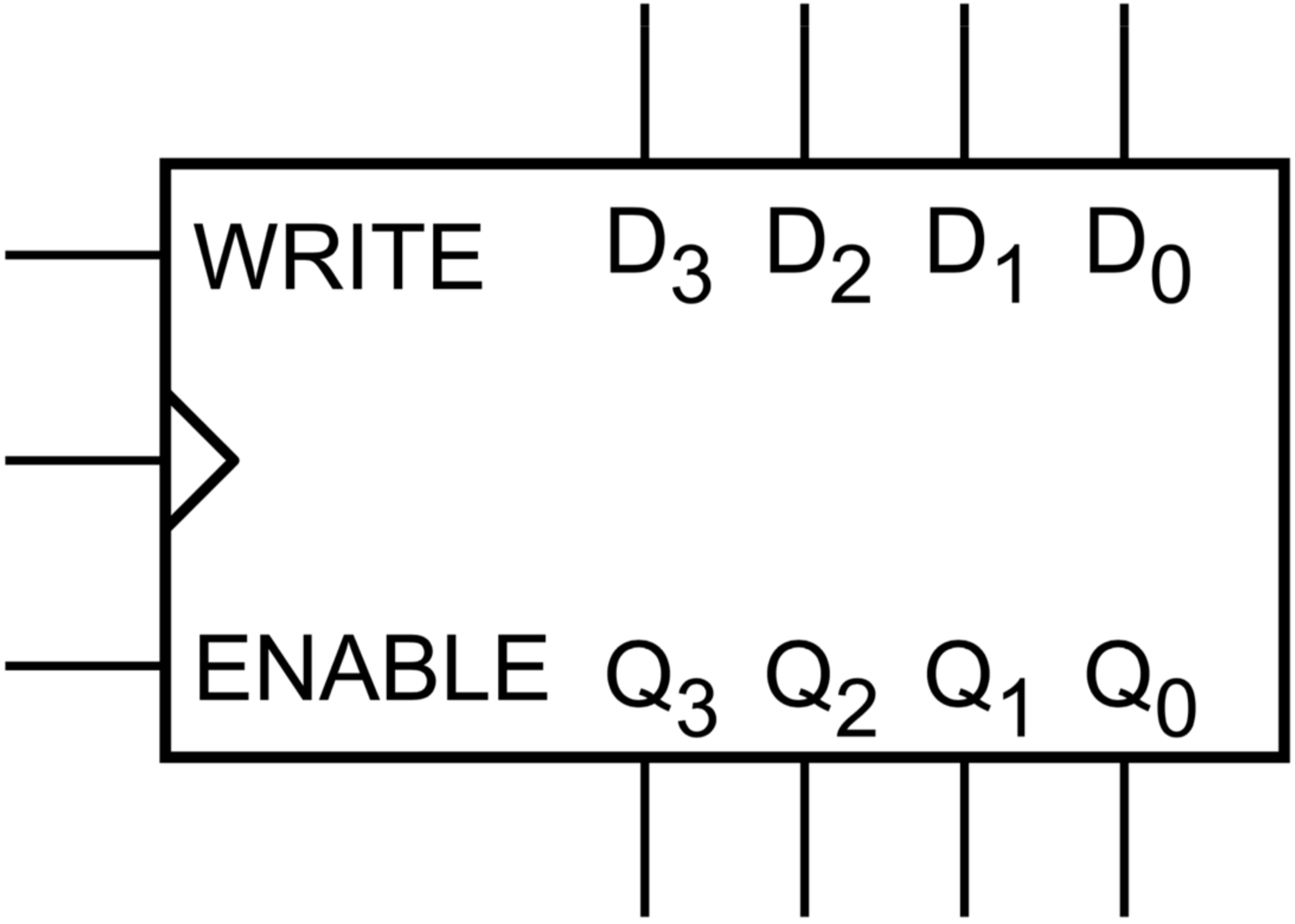
C	a	F=a⊕b
0	0	∅
0	1	∅
1	0	0
1	1	1



Operation of Four-Bit Register

- Register is loaded (written) when WRITE=1 on rising clock edge
  - Inputs D_3, D_2, D_1, D_0 stored in D flip-flops
  - This is a “parallel load” operation
- Register outputs are enabled when ENABLE=1
  - Contents of D flip-flops driven on Q_3, Q_2, Q_1, Q_0 by tri-state buffer
  - Output is “tri-stated” (in high-impedance mode) when ENABLE=0

Block Diagram of Four-Bit Register



Counters (1)

- Counters are specialized sequential circuits that produce a repeating sequence of outputs
  - Example: 000, 001, 010, 011, 100, 101, 110, 111, 000, 001, ...
  - Note that this sequence represents 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, ...

Counters (2)

- Inputs
  - The only required input is a CLOCK (at least for a synchronous counter)
  - Other inputs may reset the counter to some known value, enable/disable the counter to pause the count, or cause the count to go up or down
- Outputs
  - The sequence of n-bit values (n=3 for the previous example sequence)

Common Counters

- N-bit UP Counters
  - Example (N=3): 000, 001, 010, 011, 100, 101, 110, 111, 000, …
- N-bit DOWN Counters
  - Example (N=3): 000, 111, 110, 101, 100, 011, 010, 001, 000, …
- N-bit UP/DOWN Counters
  - Require an UP/DOWN control input
  - Behave like an UP counter if UP is active
  - Behave like a DOWN counter if DOWN is active
- Counters that produce special codes, like Gray Codes, are useful for some applications
  - In a Gray Code sequence, just one bit changes for each transition

CHECK POINT

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

- Draw and label the block diagram of a 4-bit register
- Describe some functions of simple counters

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

Summary (1)

- Sequential logic is used to realize finite state machines
  - Output depends on present input and the prior inputs
  - FSM has “memory” so that information about the prior inputs is captured in the present state bits
  - Output is a function of the present state or of the present state and present inputs
  - Next state is a function of the present state and the present inputs
- The behavior of a finite state machine can be represented using a state transition diagram (or other mechanisms)

Summary (2)

- Registers use N flip-flops to store an N-bit word
- Simple registers can be loaded or read in parallel, i.e. all bits at once
- Counters are a special type of sequential circuit or finite state machine that produce an output sequence
- Counters are commonly used to control some sequence of events or operations

MODULE 3: Boolean Algebra and Digital Logic

Lecture 3.7
Sequential Logic

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

Electrical and Computer Engineering
Virginia Tech