



Computer Hardware Engineering (IS1200)

Computer Organization and Components (IS1500)

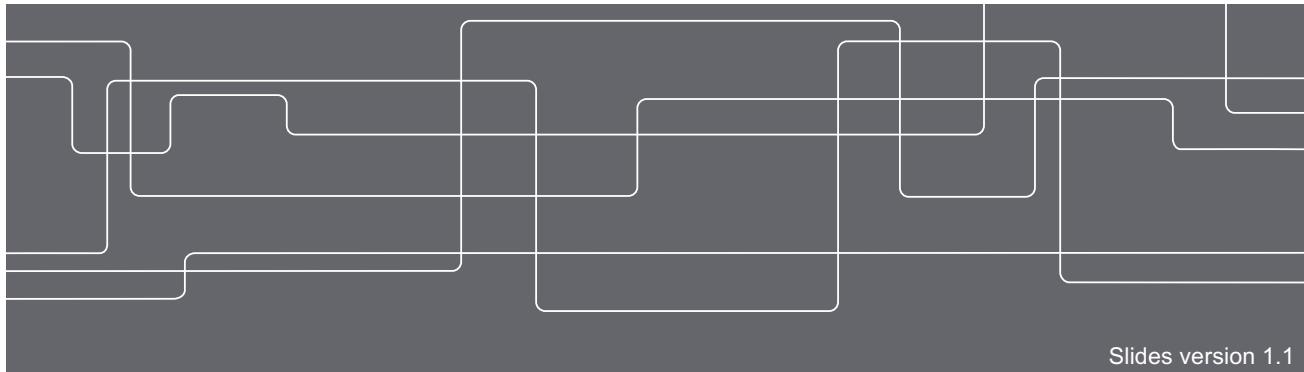
Spring 2019

Lecture 8: Sequential Logic

Note: This lecture is optional for IS1200 (for review only)

David Broman

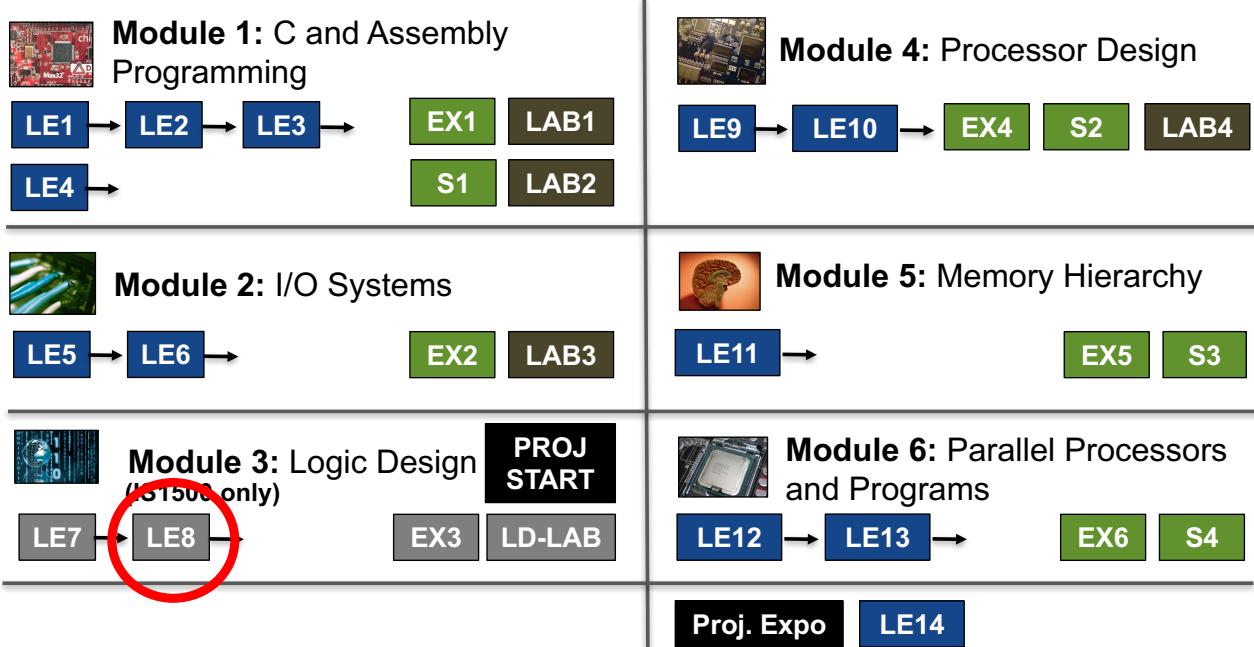
Associate Professor, KTH Royal Institute of Technology



2



Course Structure



David Broman
dbro@kth.se

Part I
Bistability and
Latches

Part II
Flip-Flops, Registers,
and Register Files

Part II
Synchronous Sequential Circuits
and Finite State Machines



Agenda

Part I

Bistability and Latches



Part II

Flip-Flops, Registers, and Register Files



Part III

Synchronous Sequential Circuits and Finite State Machines



David Broman
dbro@kth.se

Part I
Bistability and
Latches

Part II
Flip-Flops, Registers,
and Register Files

Part II
Synchronous Sequential Circuits
and Finite State Machines



Part I

Bistability and Latches



<http://www.publicdomainpictures.net/view-image.php?image=8284&picture=stapla-de-stenar&large=1>

Acknowledgement: The structure and several of the good examples are derived from the book "Digital Design and Computer Architecture" (2013) by D. M. Harris and S. L. Harris.

David Broman
dbro@kth.se



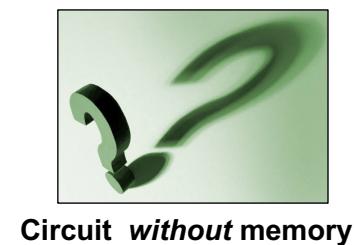
Part I
Bistability and
Latches

Part II
Flip-Flops, Registers,
and Register Files

Part II
Synchronous Sequential Circuits
and Finite State Machines

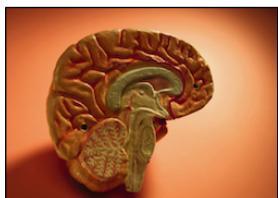
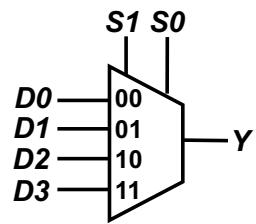
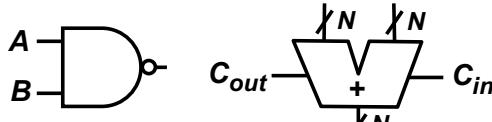


Combinational vs. Sequential Logic



Combinational Logic Design (previous lecture)

- Output depends *only* on the input.
- There is *no memory*.



Circuit with memory

Sequential Logic Design (this lecture)

- Depends on *both* current and prior input values.
- As a consequence, sequential logic *has memory*.
- Today, we will learn about:

Latches

Flip-Flops

Registers

We will discuss other kinds of memories in course module 5: *Memory hierarchy*.

David Broman
dbro@kth.se



Part I
Bistability and
Latches

Part II
Flip-Flops, Registers,
and Register Files

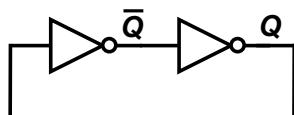
Part II
Synchronous Sequential Circuits
and Finite State Machines



Bistability

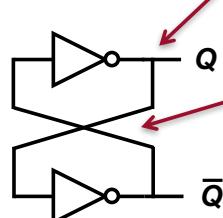
6
E

What is the difference
between these two circuits?



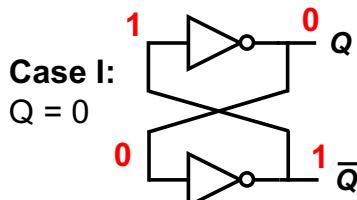
Answer: None, but
both circuits
contain a *cycle*.

What is the value of Q?

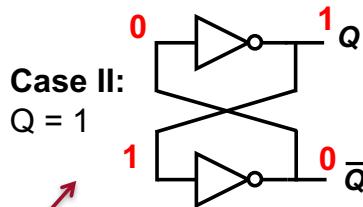


Notation convention:
T-connections
connect, four way
connections do not.

Analysis by considering two cases:



Both cases are stable.
The circuit is **bistable**.



This circuit has 2
stable **states**. Hence,
it is a memory that
can store **1 bit** of
information.

Problem: We cannot
decide what to store
(there is no input)

David Broman
dbro@kth.se

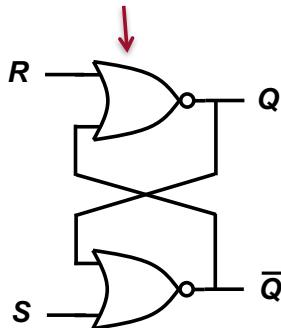


Part I
Bistability and
Latches

Part II
Flip-Flops, Registers,
and Register Files

Part II
Synchronous Sequential Circuits
and Finite State Machines

What is the behavior of this circuit? Analyze the 4 cases for inputs S and R .

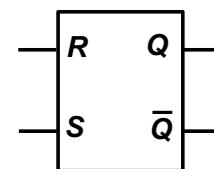


S is the SET signal and R is the RESET signal.

If S and R are zero, the circuit “remembers” the previous Q value, called Q_{pre} . We have a memory...

S	R	Q	\bar{Q}
0	0	Q_{pre}	\bar{Q}_{pre}
0	1	0	1
1	0	1	0
1	1	0	0

An SR latch can be implemented using different gates. This is the abstract symbol for an SR latch.



Problem 1. The awkward case $S=1$, $R=1$ results in that both Q and \bar{Q} are zero.

Problem 2. Mixes the issues of *what* and *when* updates are made. It is hard to design large circuits this way.

David Broman
dbro@kth.se

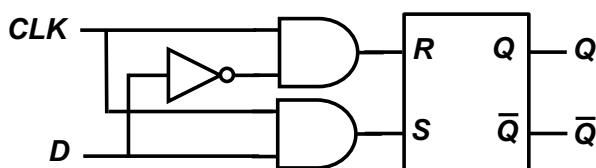


Part I
Bistability and
Latches

Part II
Flip-Flops, Registers,
and Register Files

Part II
Synchronous Sequential Circuits
and Finite State Machines

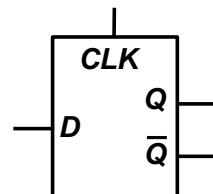
The **D latch** solves the problems with the SR latch. It has one data input D , and a clock input CLK .



CLK	D	Q	\bar{Q}
0	?	Q_{pre}	\bar{Q}_{pre}
1	0	0	1
1	1	1	0

The symbol **?** means “don’t care”. It is used to simplify truth tables (we can skip one row in this case). Sometimes a symbol **X** or **D** is used to describe “don’t care”.

Symbol describing a D latch:



Also called a **transparent latch** or **level-sensitive latch**.

- $CLK = 1$, the latch is *transparent* (D flows through to Q).
- $CLK = 0$, the latch is *opaque* (the latch blocks data from flowing through).

David Broman
dbro@kth.se



Part I
Bistability and
Latches

Part II
Flip-Flops, Registers,
and Register Files

Part II
Synchronous Sequential Circuits
and Finite State Machines



Part II

Flip-Flops, Registers, and Register Files



Acknowledgement: The structure and several of the good examples are derived from the book "Digital Design and Computer Architecture" (2013) by D. M. Harris and S. L. Harris.

David Broman
dbro@kth.se

Part I
Bistability and
Latches



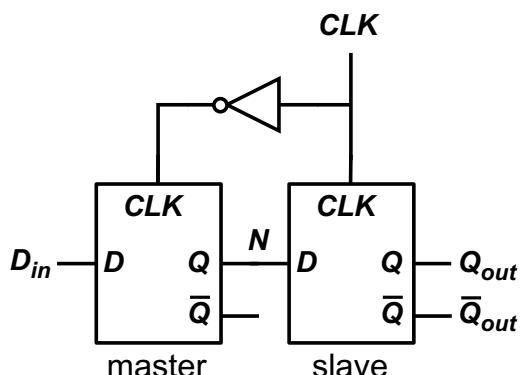
Part II
Flip-Flops, Registers,
and Register Files

Part II
Synchronous Sequential Circuits
and Finite State Machines



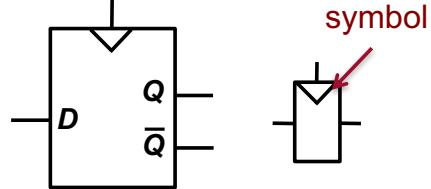
D Flip-Flop

A flip-flop is **edge-triggered** and not level-triggered.



These symbols describe D Flip-Flops.

Condensed
symbol



The **D flip-flop** (the standard flip-flop) copies D to Q on the **rising edge**, and remembers its state all other times.

Case I: CLK = 0

The master is transparent and the slave is opaque. D_{in} flows to N.

Case II: CLK = 1

The slave is transparent and the master is opaque. N flows to Q_{out} .

David Broman
dbro@kth.se

Part I
Bistability and
Latches

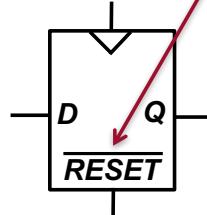
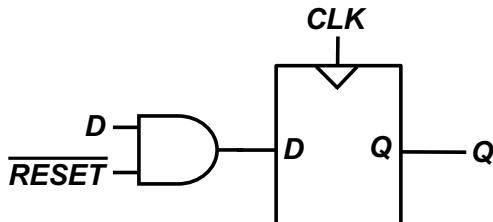


Part II
Flip-Flops, Registers,
and Register Files

Part II
Synchronous Sequential Circuits
and Finite State Machines

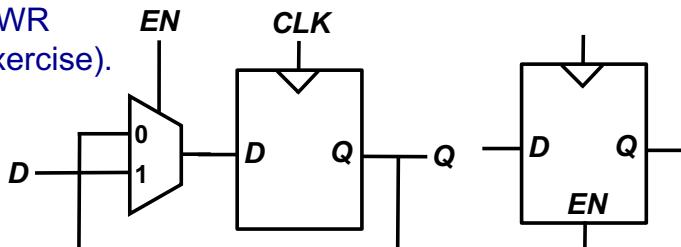
Resettable and Enabled Flip-Flops

A **resettable flip-flop** resets the flip-flop to 0 when a reset signal is active.



The line above the signal name shows that the reset signal is **active low**: The reset is active on 0.

An **enabled flip-flop** has an input *EN*. Its state changes only when *EN* = 1 and there is a raising clock edge (Called WR in the exercise).



True or False physical Q/A

"This resettable flip-flop is **synchronously resettable**, meaning that it resets on a rising clock edge. It is not **asynchronously resettable** where the reset is independent of the clock."

Answer: True



David Broman
dbro@kth.se

Part I
Bistability and
Latches

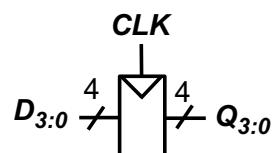
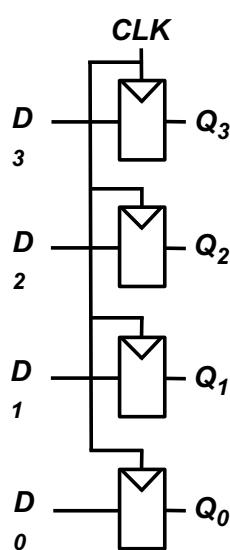
Part II
Flip-Flops, Registers,
and Register Files

Part II
Synchronous Sequential Circuits
and Finite State Machines

Register

An **N-bit register** consists of N flip-flops that share the same clock input.

4-bit register built out of D flip-flops (using condensed symbol notation).



Abstract form of a 4-bit register.

Note that registers can also have enable signals, reset signals etc.

David Broman
dbro@kth.se

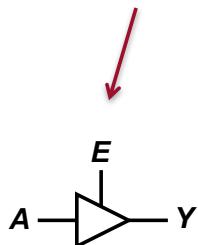
Part I
Bistability and
Latches

Part II
Flip-Flops, Registers,
and Register Files

Part II
Synchronous Sequential Circuits
and Finite State Machines

Output Enable Register

Recall the **tristate** buffer with floating value (Z)

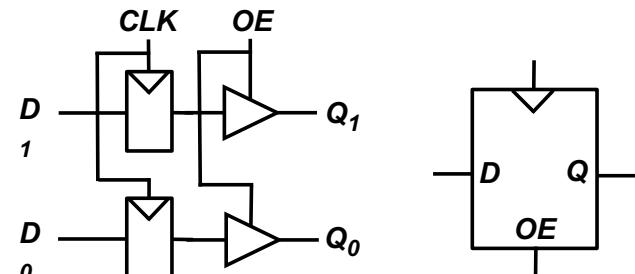


E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

Exercise:

Create a 2-bit register that has an **output enable (OE)** input signal. If OE = 0 then Q is floating, else it outputs the registers' state.

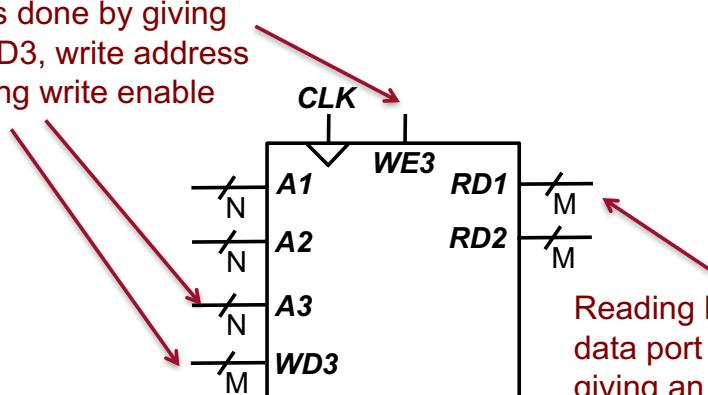
Answer:



Register File (1/2)

A **register file** can be used to read and write data using an address.

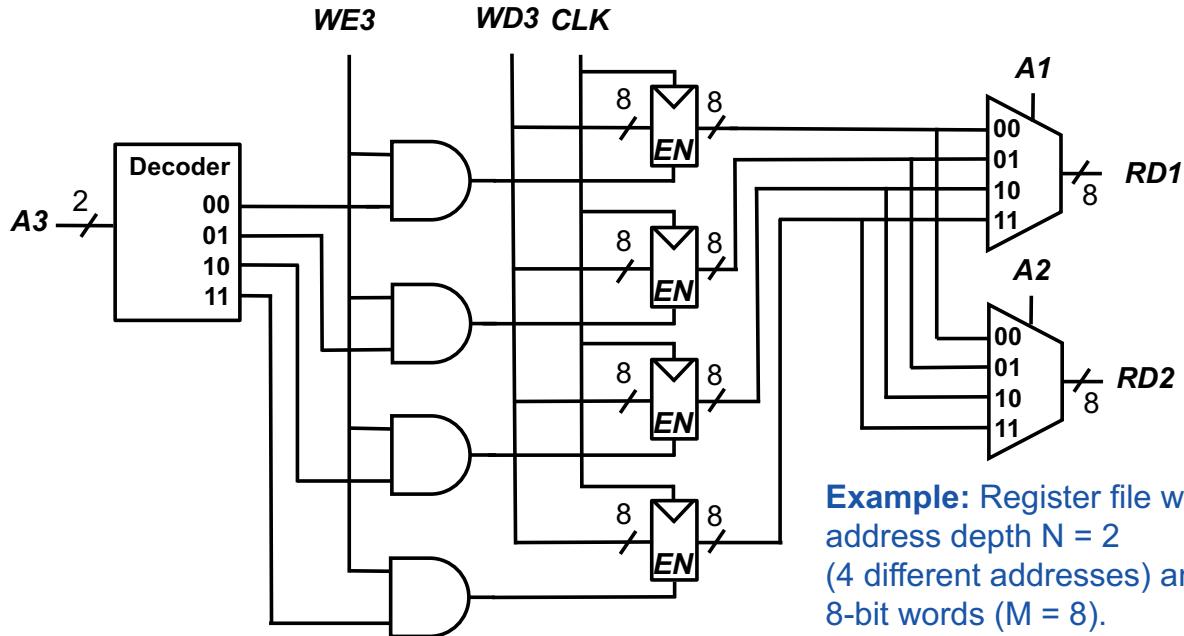
Writing **M** bits is done by giving write data to WD3, write address to A3, and setting write enable WE3 to 1.



Reading **M** bits from read data port RD1 is done by giving an **N**-bit read address to A1. Same for the second read port (RD2 and A2).

This is a **multi-ported** register file. Two read ports and one write port. Reads and writes can be done in parallel.

Register File (2/2)



David Broman
dbro@kth.se

Part I
Bistability and
Latches

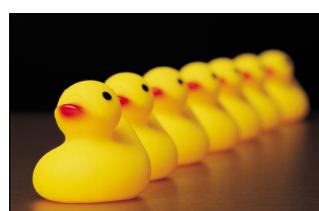
Part II
Flip-Flops, Registers,
and Register Files

Part II
Synchronous Sequential Circuits
and Finite State Machines



Part III

Synchronous Sequential Circuits and Finite State Machines



Acknowledgement: The structure and several of the good examples are derived from the book "Digital Design and Computer Architecture" (2013) by D. M. Harris and S. L. Harris.

David Broman
dbro@kth.se

Part I
Bistability and
Latches

Part II
Flip-Flops, Registers,
and Register Files

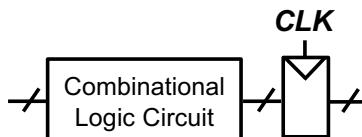
Part II
Synchronous Sequential Circuits
and Finite State Machines

It is hard to analyze large asynchronous circuits that contain cycles.

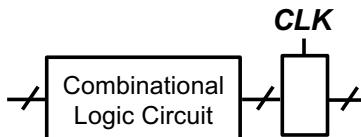
Solution: Design **synchronous sequential circuits**, also called designing at the **register-transfer level (RTL)**, which means that

- combinational logic is combined with registers
- states are only updated on clock edges

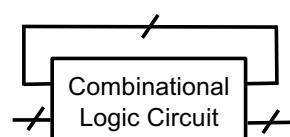
Which of the following circuits are using **RTL design / sequential synchronous logic?**



Yes, with no feedback



No, because of latch.



No, has a cycle without register in the path

David Broman
dbro@kth.se

Part I
Bistability and
Latches

Part II
Flip-Flops, Registers,
and Register Files

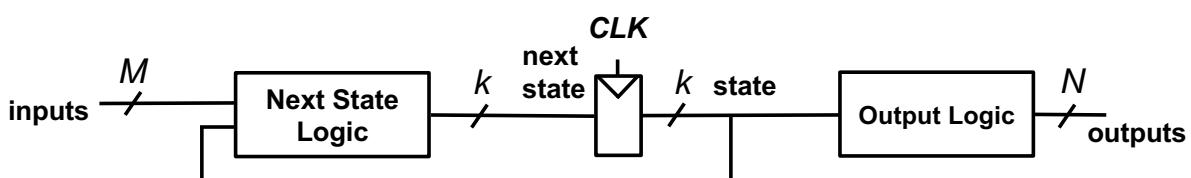
Part II
Synchronous Sequential Circuits
and Finite State Machines



Finite State Machines (FSMs)

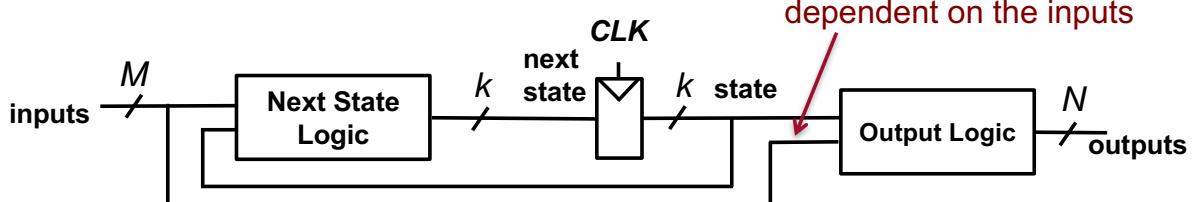
Synchronous Sequential Logics can be defined as FSMs

Moore Machine (M inputs, N outputs, k states)



Mealy Machine (M inputs, N outputs, k states)

Outputs can be directly dependent on the inputs



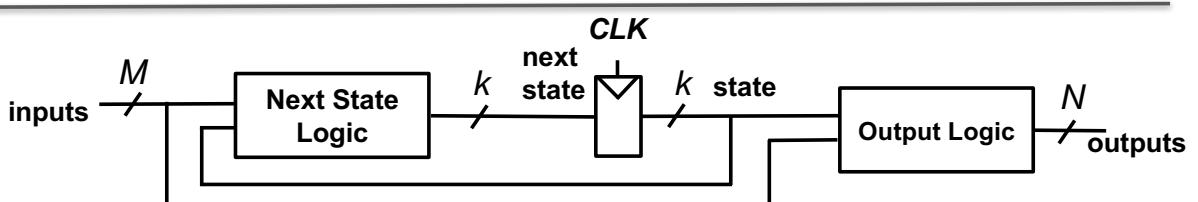
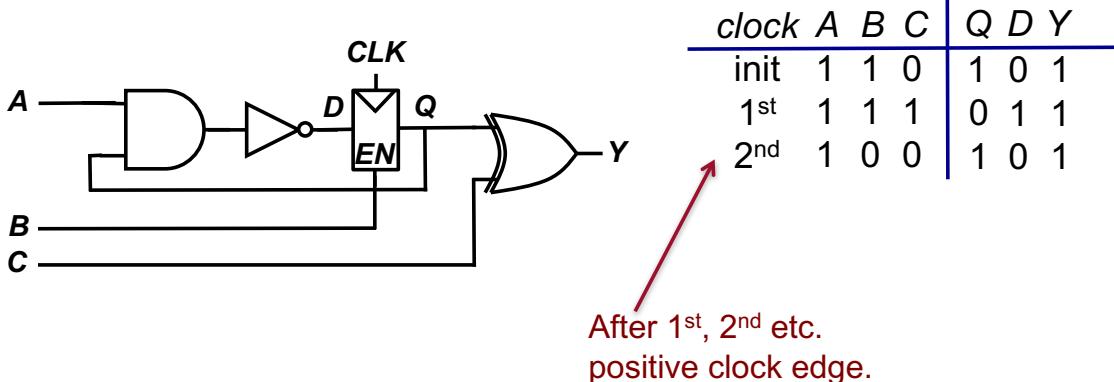
David Broman
dbro@kth.se

Part I
Bistability and
Latches

Part II
Flip-Flops, Registers,
and Register Files

Part II
Synchronous Sequential Circuits
and Finite State Machines

A Simple Mealy Machine Example



David Broman
dbro@kth.se

Part I
Bistability and
Latches

Part II
Flip-Flops, Registers,
and Register Files

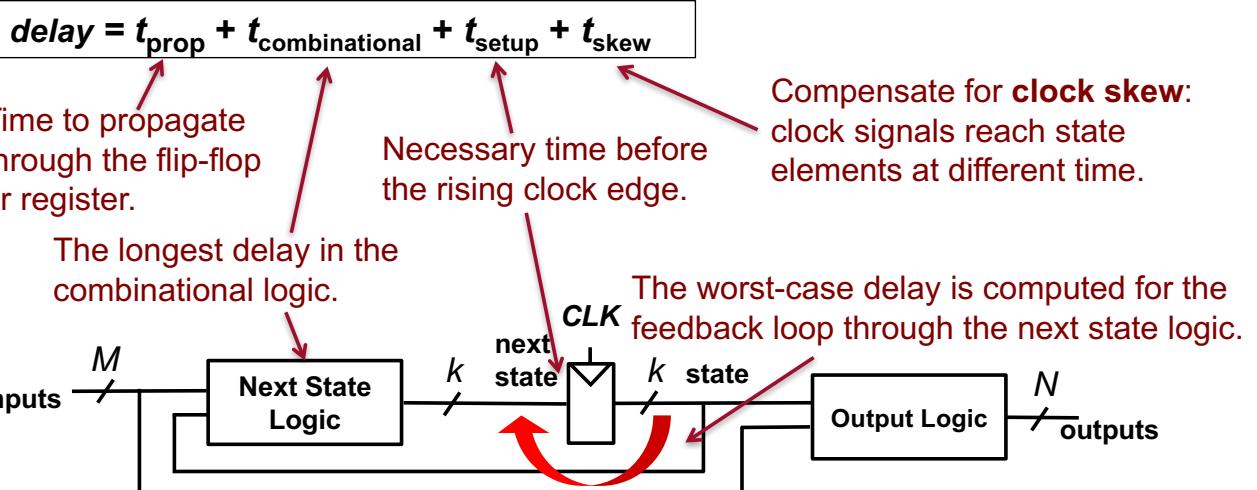
Part II
Synchronous Sequential Circuits
and Finite State Machines

Edge-Triggered Timing Methodology

How fast can we run a circuit?

The clock period must be longer than the worst-case of delays in the circuit.

A **race** may occur when the values of state elements depend on the relative speed of logic elements in the circuit.



David Broman
dbro@kth.se

Part I
Bistability and
Latches

Part II
Flip-Flops, Registers,
and Register Files

Part II
Synchronous Sequential Circuits
and Finite State Machines



Why not skip...
...fumble with the bags? 😊



David Broman
dbro@kth.se

Part I
Bistability and
Latches

Part II
Flip-Flops, Registers,
and Register Files

Part II
Synchronous Sequential Circuits
and Finite State Machines



Reading Guidelines



Module 3: Logic Design

Lecture 7: Combinational Logic Design

- H&H Chapters 1.5, 2.1-2.4, 2.6, 2.8-2.9

Lecture 8: Sequential Logic Design

- H&H Chapters 3.1-3.3 (not 3.2.7),
3.4.1-3.4.3, 5.2.1-5.2.2, 5.5.5

Reading Guidelines

See the course webpage
for more information.

David Broman
dbro@kth.se

Part I
Bistability and
Latches

Part II
Flip-Flops, Registers,
and Register Files

Part II
Synchronous Sequential Circuits
and Finite State Machines



Summary

Some key take away points:

- **Combinational Logic vs. Sequential Logic:** Combinational logic has *no* memory, whereas sequential logic *includes* memory.
- **Latches vs. Flip-Flops vs. Registers vs. Register File**
Flip-flops are edge triggered, registers combine flip-flops, and register files uses addresses to access data.
- **Synchronous vs. Asynchronous Logic Design**
Synchronous Design makes design work easier.
- **Mealy vs. Moore Machines**
Both are finite state machines (FSMs). Moore machines depend only on the state, whereas Mealy machines depend on the state and the input.



Thanks for listening!

David Broman
dbro@kth.se

Part I
Bistability and
Latches

Part II
Flip-Flops, Registers,
and Register Files

Part II
Synchronous Sequential Circuits
and Finite State Machines