# Course Structure

## Module 1: C and Assembly Programming

<table>
<thead>
<tr>
<th>LE1</th>
<th>LE2</th>
<th>LE3</th>
<th>EX1</th>
<th>LAB1</th>
<th>S1</th>
<th>LAB2</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>LE4</th>
</tr>
</thead>
</table>

## Module 2: I/O Systems

<table>
<thead>
<tr>
<th>LE5</th>
<th>LE6</th>
<th>EX2</th>
<th>LAB3</th>
</tr>
</thead>
</table>

## Module 3: Logic Design

<table>
<thead>
<tr>
<th>LE7</th>
<th>LE8</th>
</tr>
</thead>
</table>

## Module 4: Processor Design

<table>
<thead>
<tr>
<th>LE9</th>
<th>LE10</th>
<th>EX4</th>
<th>S2</th>
<th>LAB4</th>
</tr>
</thead>
</table>

## Module 5: Memory Hierarchy

<table>
<thead>
<tr>
<th>LE11</th>
<th>EX5</th>
<th>S3</th>
</tr>
</thead>
</table>

## Module 6: Parallel Processors and Programs

<table>
<thead>
<tr>
<th>LE12</th>
<th>LE13</th>
<th>EX6</th>
<th>S4</th>
</tr>
</thead>
</table>

**Proj. Expo**

<table>
<thead>
<tr>
<th>LE14</th>
<th>LE14</th>
</tr>
</thead>
</table>

---

David Broman  
Associate Professor, KTH Royal Institute of Technology  

**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

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.
**Combinational vs. Sequential Logic**

**Combinational Logic Design (previous lecture)**
- Output depends *only* on the input.
- There is *no 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-Flips
  - Registers

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

### Part I

- Bistability and Latches

### Part II

- Flip-Flops, Registers, and Register Files
- Synchronous Sequential Circuits and Finite State Machines

---

**Bistability**

What is the difference between these two circuits?

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

What is the value of Q?

Analysis by considering two cases:

**Case I:**
- Q = 0
- Q = 1

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

**Case II:**
- Q = 0
- Q = 1

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)*

---

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

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

If S and R are zero, the circuit "remembers" the previous Q value, called Q_pre. We have a memory...

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>Q_pre</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Q_pre</td>
<td>Q_pre</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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

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 Q_bar are zero.

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

D Latch

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

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).
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.

D Flip-Flop

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

These symbols describe D Flip-Flops. 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 $Q_{out}$. 

Part I
Bistability and Latches
Part II
Flip-Flops, Registers, and Register Files
Part II
Synchronous Sequential Circuits and Finite State Machines

David Broman
dbro@kth.se
Resettable and Enabled Flip-Flops

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

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).

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

True or False physical Q/A

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

Answer: True

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.
Output Enable Register

**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:**

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

<table>
<thead>
<tr>
<th>E</th>
<th>A</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Z</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Z</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**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)

Example: Register file with address depth $N = 2$ (4 different addresses) and 8-bit words ($M = 8$).

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.
Register-Transfer Level - Synchronous Sequential Circuits

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

Finite State Machines (FSMs)

Synchronous Sequential Logics can be defined as FSMs

Moore Machine \((M\text{ inputs}, N\text{ outputs}, k\text{ states})\)

Mealy Machine \((M\text{ inputs}, N\text{ outputs}, k\text{ states})\)

Outputs can be directly dependent on the inputs
A Simple Mealy Machine Example

```
<table>
<thead>
<tr>
<th>clock</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Q</th>
<th>D</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1st</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2nd</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
```

After 1\(^{st}\), 2\(^{nd}\) etc. positive clock edge.

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.

\[
delay = t_{\text{prop}} + t_{\text{combinational}} + t_{\text{setup}} + t_{\text{skew}}
\]

- Time to propagate through the flip-flop or register.
- The longest delay in the combinational logic.
- Necessary time before the rising clock edge.
- Compensate for clock skew: clock signals reach state elements at different time.
- The worst-case delay is computed for the feedback loop through the next state logic.

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? 😊

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.
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 use 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.

<table>
<thead>
<tr>
<th>Part I</th>
<th>Part II</th>
<th>Part II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bistability and Latches</td>
<td>Flip-Flops, Registers, and Register Files</td>
<td>Synchronous Sequential Circuits and Finite State Machines</td>
</tr>
</tbody>
</table>

Thanks for listening!