# SI100B EE Part Project • Tutorial 4

## Load & Store Datapath

@Yunxiang He, heyx1@shanghaitech.edu.cn
@Quanyu Chen, chenqy2022@shanghaitech.edu.cn
@Yongzhi Zhang, zhangyz2022@shanghaitech.edu.cn
@Chaofan Li, lichf1@shanghaitech.edu.cn
@Qihan Ding, dingqh@shanghaitech.edu.cn

# Contents

1. Interaction with Memory ->
2. I-Type Load Datapath ->
3. S-Type Store Datapath ->
4. Grading Policy ->

# 1. Interaction with Memory

# Assembly Instructions—lw(Load word)

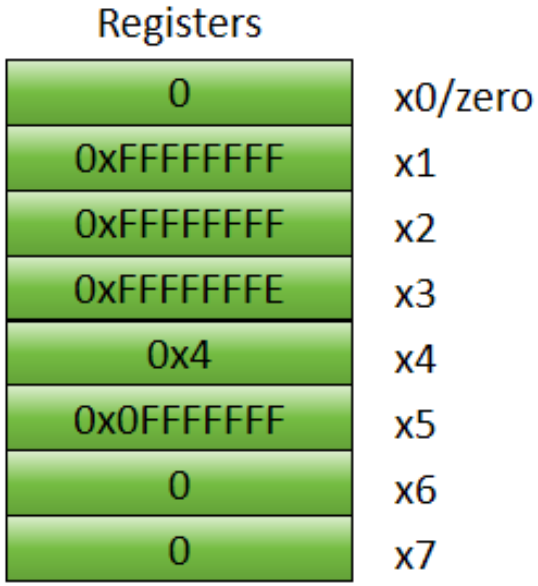
$$\mathbf{lw}\ \mathrm{rd},\ \overbrace{\mathrm{imm}(\mathrm{rs1})}^{\mathrm{OffsetBase}}:$$

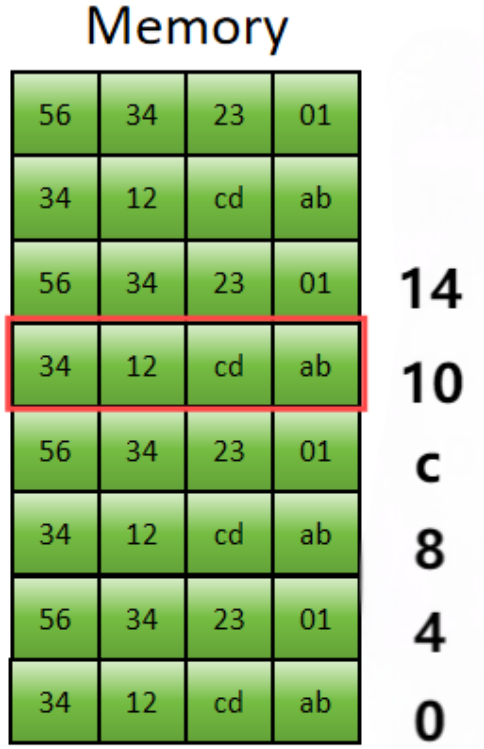
$$\mathrm{addr.} = (\mathrm{x}[\mathrm{rs1}]) + \mathrm{imm}$$

**Example:** $\mathrm{lw\ x1,\ 12(x4)}$

$$\mathrm{addr.} = 4 + 12 = (10)_{\mathrm{HEX}}$$

- `lw` loads a word, in other words, 4 bytes.



# Assembly Instructions—sw(Store word)

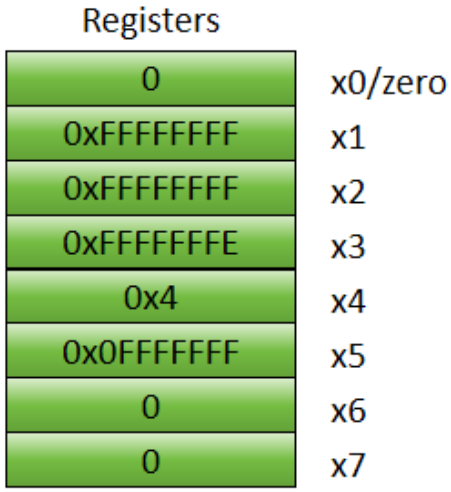
$$\mathbf{sw}\ \mathrm{rs2},\ \overbrace{\mathrm{imm}(\mathrm{rs1})}^{\mathrm{OffsetBase}}:$$

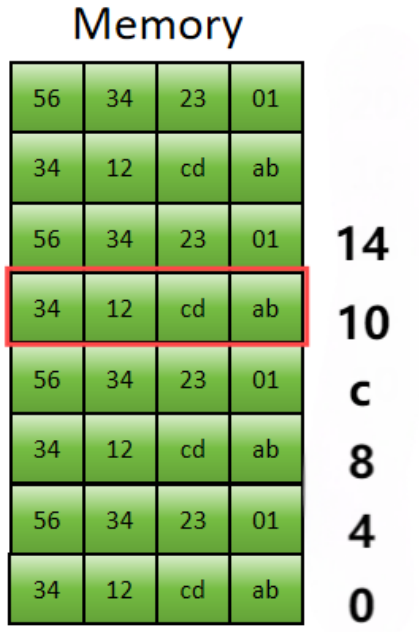
$$\mathrm{addr.} = (\mathrm{num\ in\ rs1}) + \mathrm{imm}$$

**Example:** $\mathrm{sw\ x1},\ 12(\mathrm{x4})$

$$\mathrm{addr.} = 4 + 12 = (10)_{\mathrm{HEX}}$$

- `sw` stores a word, in other words, 4 bytes.



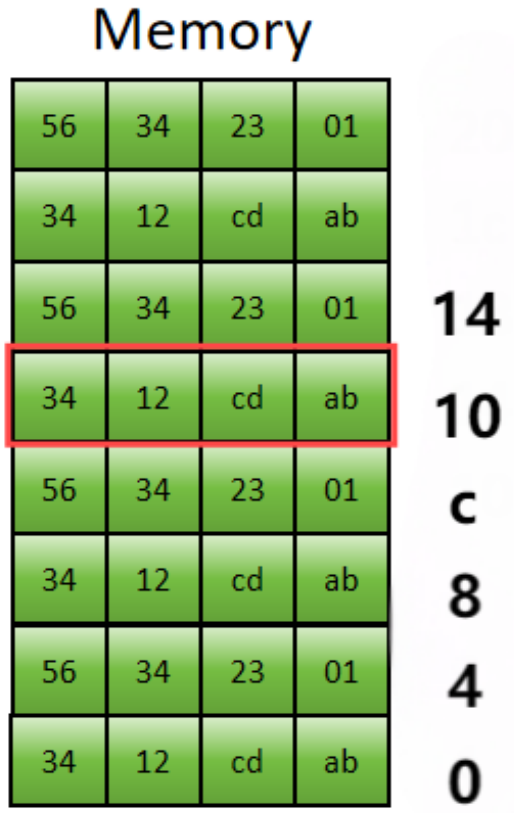
# Other Load/Store Instructions

- `lh`, `lhu`, `lb`, `lbu`
  - `h`: half-word, 2 bytes
  - `b`: byte, 1 byte
  - `u`: unsigned, no sign extension
- `sh`, and `cyb`……?

These instructions are similar to `lw` and `sw`, but with different data sizes.

We don't require you to implement all of them, but you can try to implement them by yourself.

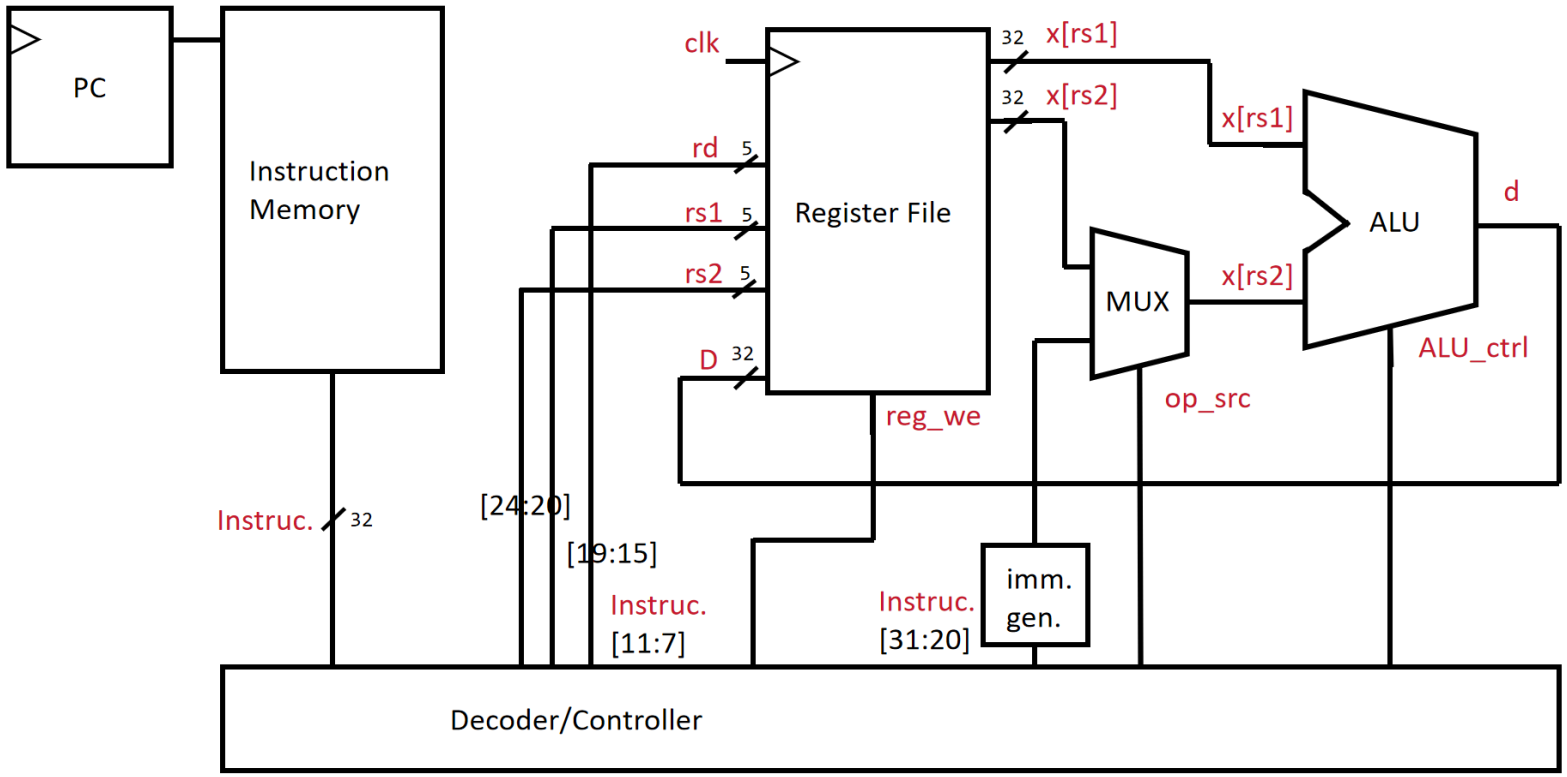
> “ If you wan't to implement them, you may know big-endian and little-endian's difference.



# 2. I-Type Load Datapath

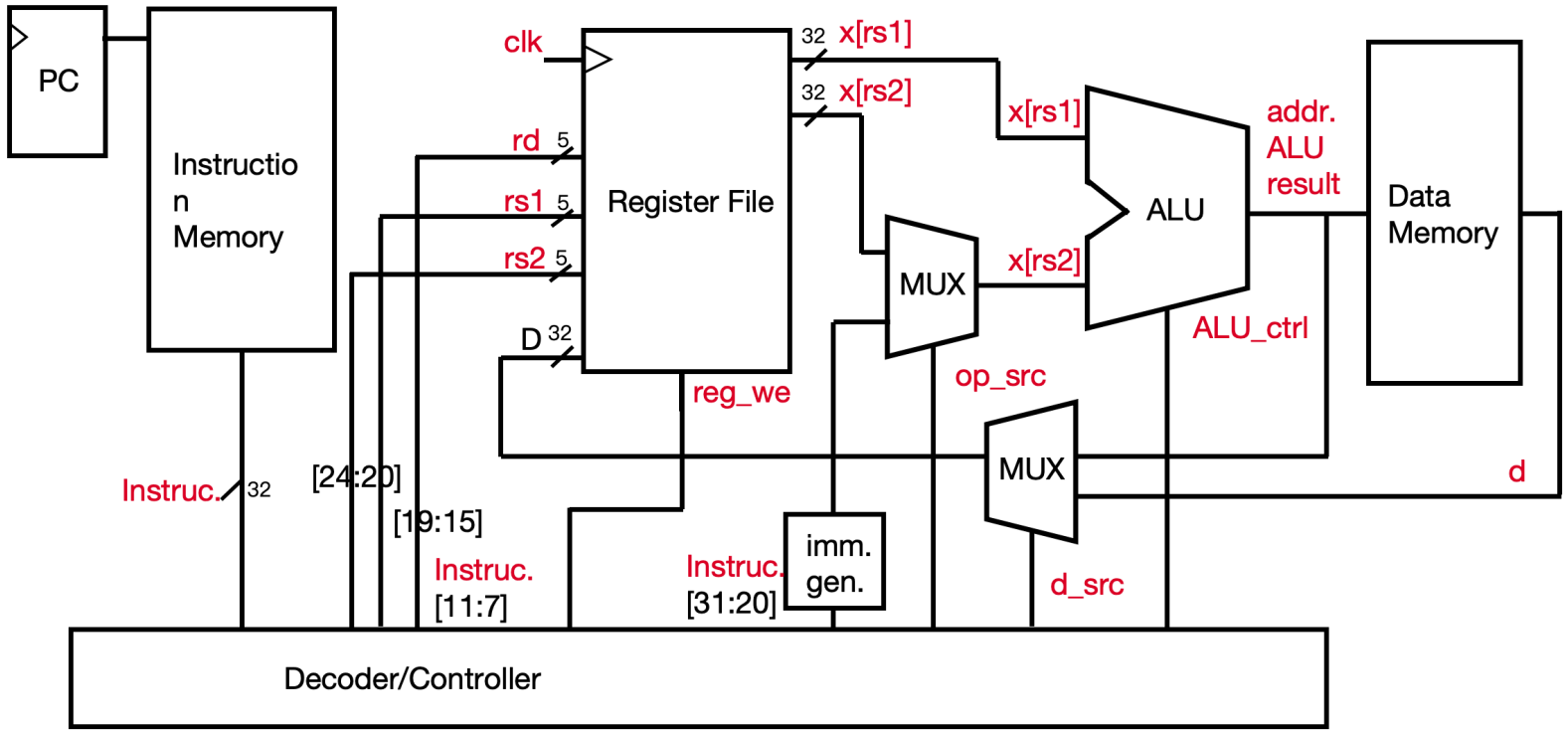
# 2. I-Type Load Datapath

## Datapath before



# 2. I-Type Load Datapath

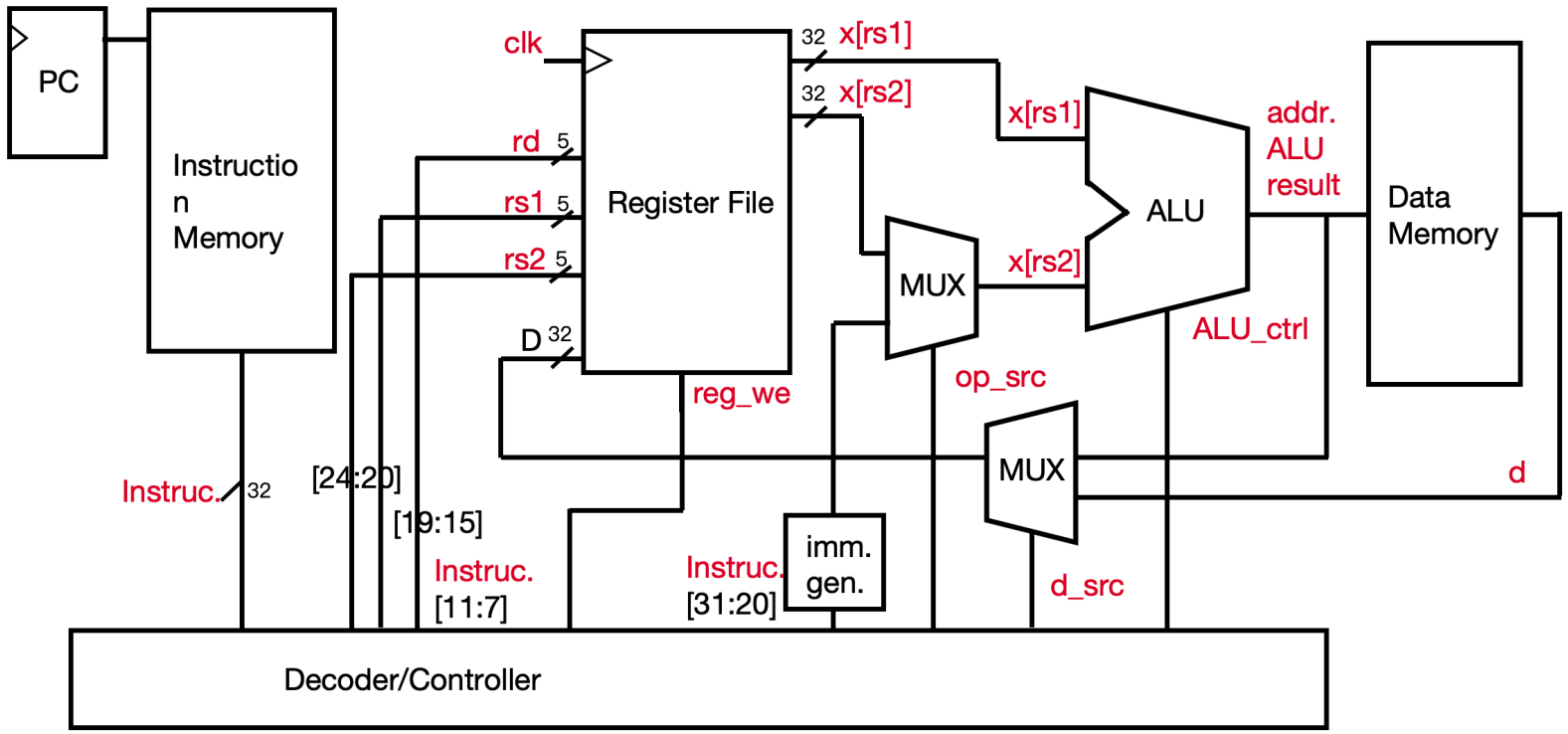
## Datapath with load instruction support



# 3. S-Type Store Datapath

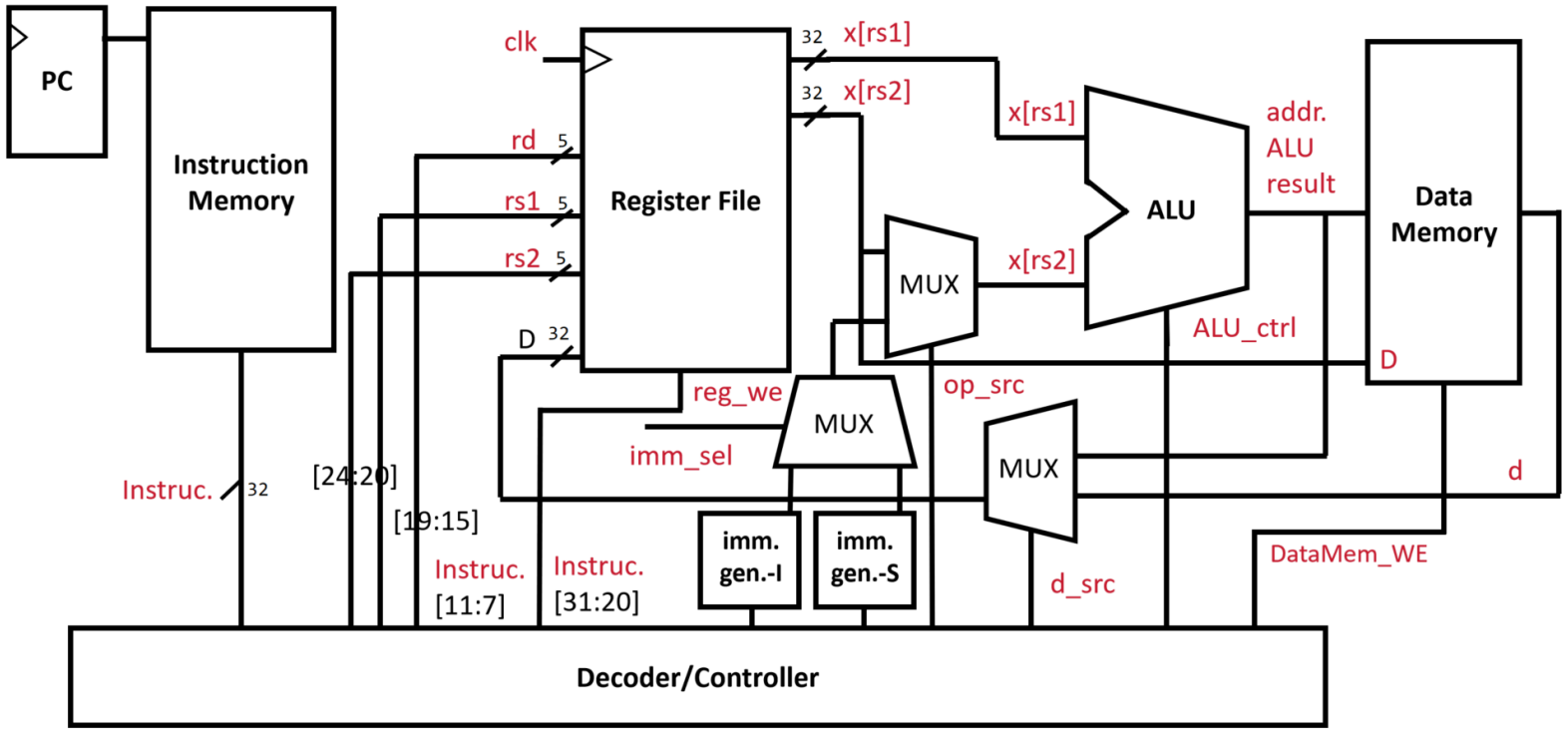
# 3. S-Type Store Datapath

## Datapath with load instruction support



# 3. S-Type Store Datapath

## Datapath with store instruction support



# 4. Grading Policy

# Grading Policy - Basic

**通过OJ测试意味着你能够获得一部分基础分数**

1. **Datapath**（30%）+ **Controller**（30%）
   - 寄存器堆、ALU、程序计数器、指令存储器、数据存储器等的实现；译码电路的实现。
   - **通过 Online Judge 测试，则可以获得 Datapath 的 30% 分数**；否则将在面对面 Check 的时候给出分数。
2. **Assembly Program**（20%）
   - 编写简单的汇编程序，转换为机器码，加载到指令存储器中。
   - 在 面对面 Check 的时候，**现场展示你们的程序和程序 miniCPU 的运行情况以获得分数。**
3. **Report**（20%）
   - 你需要使用 LaTeX 编写实验报告，详细介绍 miniCPU 的设计、实现、运行情况等。
   - **实验报告的分数会在面对面 Check 的时候结合你的表现一同给出**。

# Grading Policy

我们会根据完成情况为每部分电路打分，给分细节可作如下参考：

1. 未能完全实现合格的电路：给分少于总分的 40%。
2. 实现全部电路，设计了覆盖率一般的测试：给分至多为总分的 80%
3. 实现全部电路，设计了高覆盖率的测试：给分可达总分的 100%

# Grading Policy - Bonus

- 如果你实现了电路，我们会在面对面 Check 的时候，检查你们的实现。
  - 你们需要**自己设计合适的测试用例，以证明你们的实现是正确的**。否则，我们可能会认为你们的实现是不完整的，从而你们的加分也会受到影响。
- 如果你是通过调研和学习来完成 Bonus 部分的，我们会在面对面 Check 的时候，询问你们的调研和学习情况。
  - **请准备好相应的材料，以证明你们的调研和学习是有意义的**。
- **每个 Bonus 会获得额外 5 分钟的面对面 Check 时间，以展示你们的实现和调研成果**。
- 即使你们的实现不完整，如果在报告中展现了你们的尝试和思考，并做出有意义的尝试和分析，我们也会给予一定的加分。
  - 如果你想实现具体的 Bonus 部分电路，我们建议你首先实现 Branch 和 Jump 部分，然后再考虑其他部分。
  - 否则，我们建议你以完成相关的学习和调研为主，不要纠结于具体的实现（因为 Logisim 本身是一个效率比较低的工程工具）。

# SI100B EE Part Project · Tutorial 4

@ Presented by **Yunxiang He**
@ ShanghaiTech University