Project Document

1 Developer Information

1.1 Developers

- **Jiayao Qin**, SID: 12412619, Completed the overall structure design, FSM for main project and all sub systems, all detailed logics for sub systems, and improved the client for advanced features.
- **Ruqi Sun**; SID: 12412620; Completed functional simulation, Arithmetic Logic Unit, Convolution Bonus, Front-end Client, and an LLM application aiming to support Digital Logic teaching.

1.2 Contribution

Two developers contributed equally to this project, each accounts for 50%.

2 Development Plan Schedule

2.1 Plan

Week 13: Complete the overall structure design for the project, and complete Architecture Design Document accordingly.
Week 14: Complete the sub systems for basic requirements.
Week 15: Complete some bonus features, and fix numorous potential problems to improve stability.

2.2 Submit History

See attached file for github submission record.

3 Project Architecture Design Description

The project adopts a highly modular and hierarchical architecture designed for scalability, maintainability, and low coupling. The system is divided into three main layers: Physical Layer, System Integration Layer, and Functional Layer.

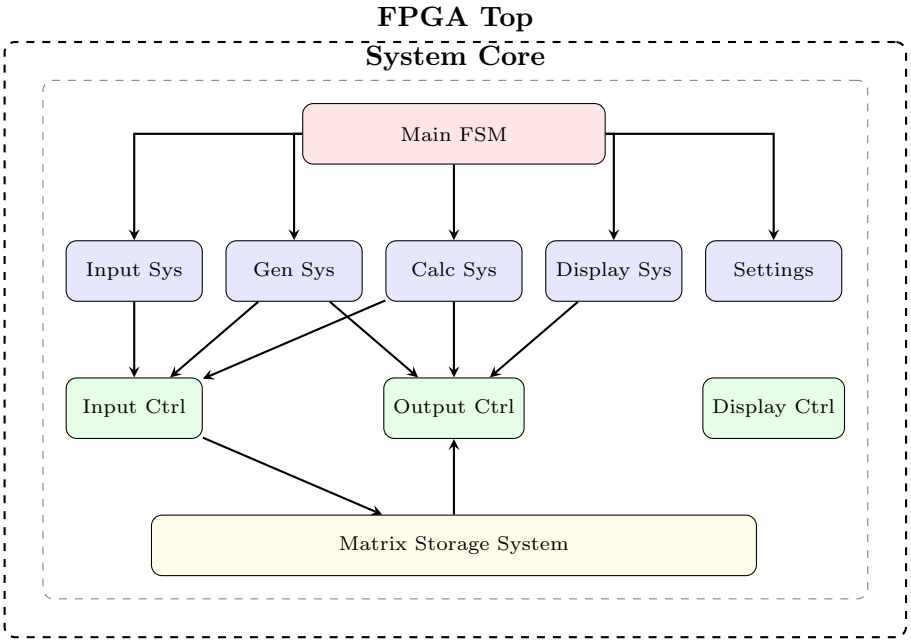

图 1: System Architecture Diagram

3.1 Hierarchical Design Strategy

- **Physical Top Layer (`fpga_top.sv`)**: This module serves as the interface to the physical hardware. It is strictly responsible for signal conditioning, including clock buffering, button debouncing, and mapping physical I/O pins to internal logic signals. No complex business logic resides here.
- **Logical Top Layer (`system_core.sv`)**: This module acts as the system integrator. It instantiates all functional sub-systems and interconnects them. It contains no specific algorithmic logic, ensuring that the system integration remains clean and the coupling between modules is minimized.
- **Functional Layer**: Specific logic is encapsulated within dedicated sub-modules (e.g., `matrix_calc`, `matrix_input`), which operate independently under the coordination of the system core.

3.2 Modular Control & Resource Arbitration

The control logic follows a decentralized pattern:

- **Main FSM (`main_fsm.sv`)**: Manages the high-level system states (e.g., switching between Input, Calculation, and Display modes). It delegates specific control authority to the active sub-system.
- **Sub-System Autonomy**: Each functional module (Input, Gen, Calc, Display, Settings) possesses its own internal Finite State Machine (FSM) to handle specific tasks. Once activated, a sub-system takes control until its task is complete.
- **Resource Controllers**: To manage shared hardware resources efficiently, we introduced dedicated controllers:

  - `output_controller`: Arbitrates access to the UART transmission line, allowing multiple modules to send data without conflict.
  - `input_controller`: Manages write access to the central storage.
  - `seg_controller` & `led_controller`: Centralize the logic for visual feedback, preventing display conflicts.

This design significantly reduced code complexity and made adding new features (like the `settings` module) seamless, as it only required hooking into the existing controller interfaces.

3.3 Structured Organization & Standardization

- **Directory Structure**: Source files are organized by functionality (e.g., `src/calc/`, `src/display_sys/`) rather than a flat directory, improving navigability.
- **Common Library**: Reusable components such as UART drivers, LFSRs, and debounce logic are placed in a `src/common/` directory to promote code reuse.
- **Package Management (`project_pkg.sv`)**: We utilize SystemVerilog packages to centralize global parameters (e.g., matrix dimensions, opcodes) and custom data structures (e.g., `matrix_t`, `matrix_element_t`). This ensures type safety and consistency across the entire project.

3.4 Timing Optimization & Resource Management

To address severe timing violations (WNS -12ns at 100MHz) and resource constraints, we implemented several architectural optimizations:

- **Logic Sequentialization**: In the Convolution module, we replaced a massive single-cycle combinational logic chain with a multi-cycle sequential state machine. This reduced the critical path by spreading the 3x3 kernel accumulation over 9 clock cycles.
- **Pipelining**: We introduced a 2-stage Fetch-Execute pipeline in the Matrix ALU. This isolates the memory read latency from the arithmetic computation, effectively cutting the critical path in half.
- **Algorithm Optimization**: In the LFSR module, we replaced the expensive modulo operator (`%`) with a multiplication-shift algorithm (`(rand * range) >> 8`). This leverages the FPGA's high-speed DSP slices instead of slow division logic.
- **Fanout Control**: For global control signals like `latched_wr_id`, we applied synthesis attributes (`(* max_fanout = 20 *)`) to force register replication, reducing routing delays caused by high fanout.

3.5 Data-Centric Storage Architecture

- **Centralized Storage (`matrix_storage_sys`)**: Matrix data is decoupled from processing logic. The storage system acts as a server, providing read/write ports to other modules.

- **Standardized Interfaces**: Communication between modules relies on standardized handshake signals (`start`, `done`, `ready`) and structured data types, ensuring robust data flow and easier debugging.

4 Bonus Implementation Description

4.1 Output Alignment

To ensure a clean and readable display on the PC client (or serial terminal), we implemented a centralized output formatting module, `matrix_uart_sender.sv`.

- **Centralized Formatting**: Instead of having each module (ALU, Generator, Display) implement its own UART logic and BCD conversion, they all instantiate or share this sender module.
- **Smart Alignment**: The module automatically handles:
  - **Sign Handling**: Detects negative numbers and inserts the minus sign.
  - **BCD Conversion**: Converts binary values to decimal digits for ASCII transmission.
  - **Padding**: Adds dynamic whitespace padding to ensure columns align perfectly, regardless of the number of digits (e.g., aligning "5" with "-123").
  - **Line Endings**: Automatically handles row breaks (`\r\n`) based on the `is_last_col` flag.
- **Mode Indication**: It also supports sending pre-defined mode strings (e.g., "CALC", "INPUT") to notify the client of the current system state.

4.2 Parameter Configuration

We introduced a dedicated `settings_sys` module to allow runtime configuration of system parameters, enhancing flexibility without recompilation.

- **Design Philosophy**: From the early design phase, all functional modules (e.g., `matrix_gen`, `matrix_storage`) were designed with configuration input ports (e.g., `val_max`, `active_limit`) rather than hardcoded constants. Initially, these were tied to default values defined in `project_pkg.sv`.
- **Seamless Integration**: The `settings_sys` module was added as a "plugin" that drives these existing ports. It reads user inputs via switches and buttons, updates the configuration registers, and broadcasts the new values to the rest of the system.
- **Configurable Parameters**:
  - **Generation Range**: Max/Min values for the random matrix generator.
  - **Error Display Time**: Duration for showing error messages on the 7-segment display.
  - **Storage Limit**: The maximum number of matrices the user is allowed to create (limiting resource usage).

4.3 UI Design and Conv Operation

Please see more detailed info of these parts in the vedio.

5 Open Source and AI Usage

5.1 Open Source References

- **UART Module**: The UART communication modules (`uart_rx.sv` and `uart_tx.sv`) located in `src/common/UART/` are adapted from the open-source examples provided by **Alientek**.
  - Source: www.yuanzige.com / http://www.openedv.com
  - These modules provide reliable serial communication capabilities, which we integrated into our system for data exchange with the PC client.

5.2 AI Usage

Tools Used: Gemini, GitHub Copilot
Usage Scenarios:

- **Development:** Provided with the overall structure deign, AI was extensively used to generate code skeletons and boilerplate, significantly accelerating the coding process. It allowed us to focus on high-level logic while AI handled the repetitive implementation details.
- **Debugging:** AI assisted in quickly locating syntax errors and logical bugs. It provided explanations for error messages and suggested potential fixes.
- **Code Comprehension & Commenting:** AI helped in quickly understanding complex code segments and automatically generating comments, improving code readability.
- **Testbench Generation:** We used AI to generate testbenches for various modules. It was particularly effective for creating standard test patterns and edge cases.
- **Documentation:** AI assisted in summarizing project details and generating README files, ensuring documentation was kept up-to-date.
- **Workflow Assistance:** AI guided us in setting up and using unfamiliar workflows, such as configuring `iverilog` for lightweight simulation in VS Code and mastering SystemVerilog syntax.

5.3 AI-Assisted Development: Pros and Cons

Advantages:

1. **Efficiency in Boilerplate Generation:** AI excels at writing repetitive code structures (like FSM templates, module instantiations, and standard port mappings). This allowed us to complete the project foundation much faster than manual coding.

2. **Rapid Learning and Problem Solving:** When encountering new tools (like `iverilog`) or language features (SystemVerilog packages), AI acted as an on-demand tutor, providing examples and explanations that were more direct and context-aware than searching through documentation.

Disadvantages & Solutions:

1. **Unintended Code Modifications (The "Hallucination" Problem):**

   - *Issue:* If prompts are not precise enough, AI might make unintended changes or "hallucinate" logic updates. A specific instance occurred when we had a logic block to divide the CPU clock from 100MHz to 25MHz. In one of the refactoring steps, the AI commented out this critical logic without us noticing. We did not discover this until a very late stage in the project.
   - *Solution:* We learned to be extremely specific in our prompts and to perform rigorous code reviews (diff checks) after every AI-generated change. We also realized the importance of having comprehensive regression tests that check core system parameters (like clock rates) to catch such accidental deletions early.

2. **Context Limitations and Integration Overhead:**

   - *Issue:* As the project scale increased, the AI struggled to maintain a holistic view of the entire project. It would sometimes generate code that was locally correct but inconsistent with global definitions (e.g., in `project_pkg.sv`) or interface protocols defined in other modules, simply because it couldn't "see" all files simultaneously.
   - *Solution:* We maintained three key documents (`DOCUMENT.md`, `README_INSTRUCTION.md`, `README_REQUIREMENTS.md`) that served as the project's "knowledge base." At the start of each new AI session, we fed these documents to the model. This allowed the AI to quickly "go through" and grasp the project's architecture, requirements, and current status, significantly reducing context errors.

5.4 Prompt Engineering Showcase

Case 1. Using documents for AI to go through the project and summarize the progress

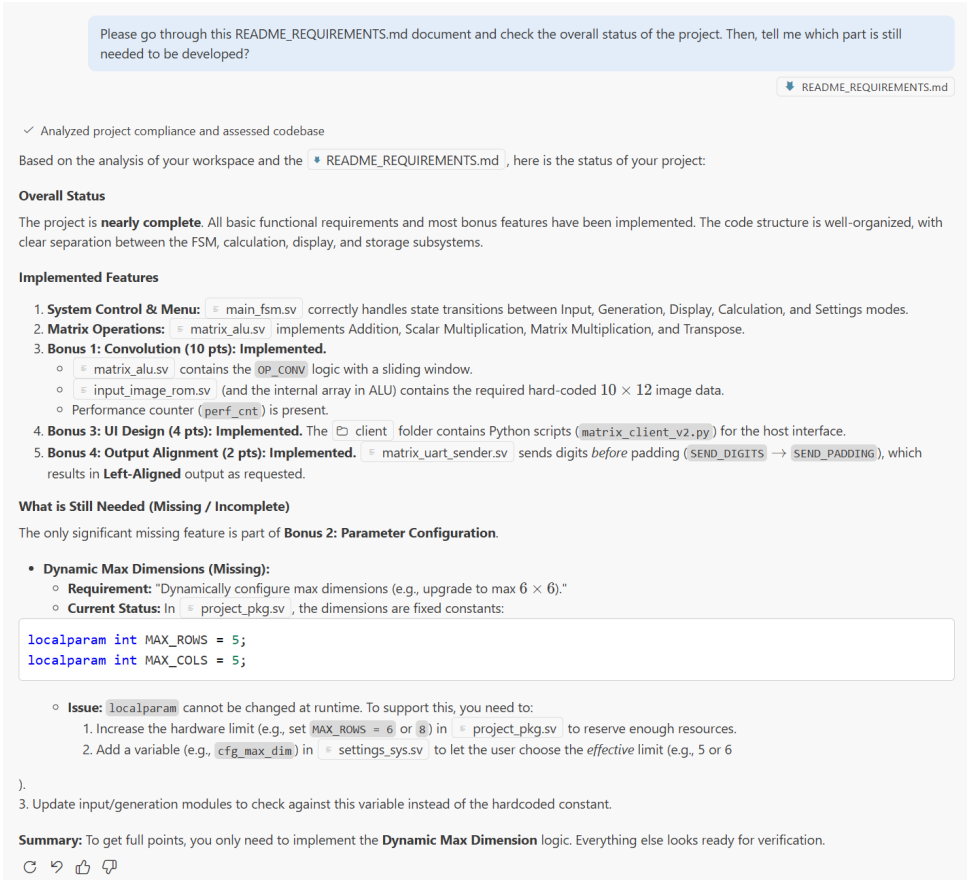

图 2: Prompt Example 1

Strategy: Context-Aware Project Management (RAG).

Description: We utilized the AI's ability to process large contexts by feeding it the `README_REQUIREMENTS.md` and the project file structure. Instead of asking for specific code snippets, we prompted the AI to perform a holistic audit of the project status.

Outcome: The AI successfully cross-referenced the requirements with the existing codebase, identifying that while the core logic (FSM, ALU, Convolution) was complete, the "Dynamic Parameter Configuration" feature was missing. This served as an effective automated quality assurance step.

Case 2. AI-Assisted Debugging (Synthesis Error Analysis)

图 3: Prompt Example 2

Strategy: Error Log Analysis & Concept Explanation.

Description: When encountering cryptic Vivado synthesis errors (specifically `[Synth 8-659] type mismatch` in `matrix_alu.sv`), we pasted the error log directly into the chat. The AI identified the root cause: a subtle mismatch between SystemVerilog **Packed Arrays** and **Unpacked Arrays**.

Outcome: The AI clearly explained the memory layout differences between the two types (contiguous vs. scattered) and provided the correct syntax to fix the assignment. This turned a potentially hours-long debugging session into a 5-minute fix.

5.5 Reflection and Conclusion

This project served as a comprehensive experiment in integrating Large Language Models (LLMs) into the traditional FPGA development workflow. The experience demonstrated that AI is a powerful force multiplier, transforming the developer's role from a "coder" to a "system architect" and "verifier."

Key Takeaways:

1. **The Importance of Context:** The most significant lesson learned was that AI is only as good as the context it is given. Our strategy of maintaining a "Knowledge Base" (the three README files) and feeding it to the model at the start of sessions proved to be a game-changer. It effectively solved the "amnesia" problem common in long conversations and ensured that generated code adhered to global project constraints.

2. **Verification is Non-Negotiable:** While AI accelerated the *writing* phase, it did not eliminate the need for the *verification* phase. In fact, it made verification even more critical. The incident where a clock divider logic was silently removed highlights that AI can introduce subtle, "silent" bugs. A robust testbench and strict code review process are essential safety nets when working with AI-generated code.

3. **Prompt Engineering as a Skill:** We found that "Prompt Engineering" is not just a buzzword but a necessary technical skill. Writing clear, unambiguous instructions and breaking down complex tasks into "Chain-of-Thought" steps (as seen in the Convolution implementation) yielded significantly better results than vague, high-level requests.

Conclusion: Ultimately, the project was completed successfully with all basic requirements and significant bonus features implemented. The use of AI allowed us to tackle more advanced features (like the Convolution accelerator and Python client) within the limited timeframe. We believe that the future of digital design lies in this hybrid approach: combining human creativity and architectural oversight with AI's speed and pattern-matching capabilities.

6 Suggestions for Project and Course Improvement

6.1 Project Design in the AI Era: From Coders to Managers

In the age of Large Language Models, the role of a developer is fundamentally shifting. AI acts as a tireless, high-efficiency "coder," pushing humans towards the roles of **Project Managers** and **Product Managers**. Course projects should reflect this evolution:

- **Scale and Complexity (The Project Manager View):** Projects should be designed with enough scale to challenge students' management capabilities. The current Matrix Calculator project is a prime example: its complex, interdependent subsystems (Input, Calculation, Display) force students to focus on system architecture, interface definitions, and progress tracking rather than just syntax. Future projects should maintain or exceed this level of complexity to train students in managing large-scale AI-assisted workflows.

- **Innovation-Driven (The Product Manager View):** Since AI significantly lowers the barrier to implementation, the core value of a project shifts from "can you build it?" to "what can you build?". Projects should encourage students to act as Product Managers, focusing on innovative ideas and user experience. The assessment should reward students who leverage AI to realize creative features that would have been impossible to complete manually within the semester.

6.2 Assessment Reform: Evaluating Creativity and Workflow

Traditional assessment methods based solely on code correctness are becoming less relevant. To evaluate a student's true capability in the AI era, we suggest the following adjustments:

- **Focus on "Creative Capability":** We are no longer traditional "code monkeys." The most critical skill now is the ability to combine domain knowledge with cutting-edge AI workflows to create better products faster. Assessment should focus on the *outcome* and the *process of creation* rather than just the lines of code.
- **Hackathon-Style Evaluation:** We recommend incorporating "Hackathon" or "Vibe-Coding" style assessments. These formats test a student's ability to use AI tools to perform creative work under time constraints. This mirrors the real-world demand for rapid prototyping and innovation, effectively measuring who can best leverage the "AI lever" to solve problems.

Appendix: Git History

```
a2d28b8 - DraTelligence, 2025-12-26 : chore: 更新了项目文档
01c4ebb - rickysun2006, 2025-12-25 : feat: 完成了项目文档中有关ai usage的部分（open source还没
↪  写!）
583c3c5 - DraTelligence, 2025-12-24 : feat: 完成了项目文档中有关结构设计的部分，并注明了我个人的贡
↪  献
cd4f67d - Ricky Sun, 2025-12-24 : feat: 添加项目文档，包含开发者信息、贡献和项目架构设计描述
d31ace1 - Ricky Sun, 2025-12-23 : feat: 添加了文档文件以支持项目说明
01ae08e - DraTelligence, 2025-12-17 : feat: 更新README.md，添加项目概述、功能、运行指南及项目结构
ad3c8df - DraTelligence, 2025-12-17 : fix: 真的真的修了refresh功能!
53e83bb - DraTelligence, 2025-12-17 : feat: 为v1客户端也添加了v2的一些很cooooooool的改进
65f58d0 - DraTelligence, 2025-12-17 : feat:  优化了calc模式的ui
5cf536b - DraTelligence, 2025-12-17 : fix: 修复了不能正常自动识别卷积核的问题
688eec4 - DraTelligence, 2025-12-17 : Merge branch 'main' of
↪  https://github.com/rickysun2006/cs207-digital-logic-project
93ebf6a - DraTelligence, 2025-12-17 : feat:基本完成了算术逻辑
fa8a1cf - DraTelligence, 2025-12-17 : feat: 完成了算术模式下的复杂交互逻辑
90d4ade - DraTelligence, 2025-12-17 : feat: 更新了ui，并增加了输入数值检测
27b0717 - DraTelligence, 2025-12-17 : feat: 现在打开客户端时会自动连接服务器了
1fcb565 - DraTelligence, 2025-12-17 : fix: 修复了一系列错误
58fb187 - DraTelligence, 2025-12-17 : feat: 修改了客户端的ui，使得功能界面更加美观
d892418 - DraTelligence, 2025-12-17 : fix: 现在进入display mod时不会堆叠重复的统计信息了
a197302 - DraTelligence, 2025-12-17 : feat: 将premium客户端进入各个模式的提示词与最新版同步
d792370 - DraTelligence, 2025-12-16 : feat: 添加了进入模式的消息提示，用于帮助客户端自动化
1decbeb - DraTelligence, 2025-12-16 : feat: 尝试制作了一个功能更丰富的客户端
3bdc748 - DraTelligence, 2025-12-16 : feat: 更新了操作文档
196d37e - DraTelligence, 2025-12-16 : feat: 实现了系统自动选取运算数的逻辑。这是最后一个功能点了!
6aa7798 - DraTelligence, 2025-12-16 : fix: 修复了杂七杂八的bug
de5038e - DraTelligence, 2025-12-16 : feat: 实现了配置功能
5d41078 - DraTelligence, 2025-12-16 : feat: 添加了err状态及倒计时功能
c9dfd5c - DraTelligence, 2025-12-16 : feat: 修改了calc系统的交互逻辑
4eba0d8 - DraTelligence, 2025-12-16 : feat: 直接将所有矩阵运算的结果输出，并删除了所有预留的写回内
↪  存的逻辑接口
```

37f9ac5 - DraTelligence, 2025-12-16 : feat: 统一调整所有运算的输出逻辑为直接打印到终端，避免了数据溢出
3edecd0 - DraTelligence, 2025-12-16 : 调整了卷积输出逻辑，现在可以正常输出127范围以外的数字了
00e51fd - DraTelligence, 2025-12-16 : feat: 将python cache文件加入了gitignore
9795e81 - DraTelligence, 2025-12-16 : feat: 添加了一个模拟uart，用于调试的客户端入口
76baea4 - DraTelligence, 2025-12-16 : feat: 在原本客户端的基础上尝试完成一套更好的逻辑
fe82ff0 - DraTelligence, 2025-12-16 : fix: 修正了system log偶现一条输出被拆分的问题，并移除了无用的confirm。
b89d062 - DraTelligence, 2025-12-16 : feat: 为所有类型的运算添加了显示时钟周期数的功能
e95ec2d - DraTelligence, 2025-12-16 : feat: 修复了数码管显示不符合预期的问题
72037cc - DraTelligence, 2025-12-16 : fix: 修复了进入calc模式时偶现自动进入加法模式的问题
b6104bf - DraTelligence, 2025-12-16 : feat: 修复了calc模式下选择矩阵时格式化输出不正确的问题
38384c4 - DraTelligence, 2025-12-16 : feat: 修复了随机数只能生成0和1的bug
8c7ff2d - DraTelligence, 2025-12-16 : feat: 彻彻底底解决了时序违例
9dcab5b - DraTelligence, 2025-12-16 : feat: 更新了gitignore以忽略报告材料目录
48aa072 - DraTelligence, 2025-12-15 : fix: 修复了pkg中注释和实际不匹配的小问题
16caf92 - DraTelligence, 2025-12-15 : feat: 修改了全部模块与uart的交互逻辑
2f1ebdf - DraTelligence, 2025-12-15 : feat: 更改了卷积结果的传输逻辑
8f4cb03 - rickysun2006, 2025-12-15 : feat: alu卷积经过仿真验证
18cfc86 - rickysun2006, 2025-12-15 : Merge branch 'main' of https://github.com/rickysun2006/cs207-digital-logic-project
30737a0 - rickysun2006, 2025-12-15 : feat: 卷积功能实现，增加矩阵行列的最大维度至10x12，已经过仿真验证
4cd365a - DraTelligence, 2025-12-15 : feat: 将卷积运算也改为串行，以尝试解决时序违例问题，未通过仿真
678141c - DraTelligence, 2025-12-15 : feat: 实现了calc选择时用户查看已有矩阵的功能
df0574f - Ricky Sun, 2025-12-15 : feat: 大幅优化客户端ui
b8b2ed4 - Ricky Sun, 2025-12-15 : feat: 添加用户使用说明书，涵盖硬件设置、软件安装及操作指南
19d394c - Ricky Sun, 2025-12-15 : feat: 添加存储控制组件以支持矩阵数据的刷新和获取
2e0bbcb - DraTelligence, 2025-12-15 : feat: 为计算模块添加了基本的数码管显示
94df554 - Ricky Sun, 2025-12-15 : feat: 限制矩阵输入最大维度为5x5并优化数据处理逻辑
36307e8 - Ricky Sun, 2025-12-15 : feat: 修改解码逻辑以支持完整的8位二进制范围
2264336 - Ricky Sun, 2025-12-15 : feat: 恢复客户端
2bbb59e - DraTelligence, 2025-12-15 : feat: 对部分ui进行了修改
b4650b9 - DraTelligence, 2025-12-14 : feat: 为input模式添加了输入回显功能
f830d2f - DraTelligence, 2025-12-14 : fix: 再次修复了数码管和led灯显示不符合预期的问题
09c16b4 - DraTelligence, 2025-12-14 : Merge branch 'main' of https://github.com/rickysun2006/cs207-digital-logic-project
47637c9 - Ricky Sun, 2025-12-13 : feat: 升级客户端ui，加入明亮/黑暗模式切换，增加自适应layout
d60fd98 - DraTelligence, 2025-12-13 : fix: 修复了led不亮的问题
da84105 - DraTelligence, 2025-12-13 : fix: 修改了数码管状态显示与预期不符的问题
debde6e - Ricky Sun, 2025-12-13 : fix: reduce matrix storage limit to save resources; implement clock divider for timing closure
8f0524d - DraTelligence, 2025-12-13 : feat: 将矩阵乘法的计算改为了串行，以尝试解决时序违例问题
542141b - DraTelligence, 2025-12-13 : feat: 完成了计算模块的框架
5ce835d - DraTelligence, 2025-12-13 : fix: 修复了大量问题
944c38d - DraTelligence, 2025-12-13 : feat: 现在gen模式下可以连续输入矩阵，同时点亮一盏数码管指示工作中
41e0ca6 - DraTelligence, 2025-12-13 : chore: 格式化了system_core.sv文件
add1581 - DraTelligence, 2025-12-13 : fix: 修复了一个细小的bug，现在可以连续输入矩阵了

```
b5f06f3 - DraTelligence, 2025-12-13 : feat:  修改了所有controller以适应新的主状态机状态定义
39e01f7 - Ricky Sun, 2025-12-13 : feat: 卷积仿真成功
d517467 - DraTelligence, 2025-12-13 : Merge branch 'main' of
↪  https://github.com/rickysun2006/cs207-digital-logic-project
f8594e9 - DraTelligence, 2025-12-13 : feat: 从主状态机中移除了calc模式的详细子状态。
03caae4 - Ricky Sun, 2025-12-13 : feat: 增加卷积运算功能，添加周期计数器并扩展矩阵尺寸支持，写了卷
↪  积仿真（待验证）
5df1630 - DraTelligence, 2025-12-12 : feat: 完全重构了system_core，引入了更模块化的控制器设计
65aada1 - DraTelligence, 2025-12-12 : feat: 新增了两个controller模块，根据模式分配LED和数码管的显
↪  示内容
0c4f74c - DraTelligence, 2025-12-12 : feat: 增加了输入输出控制器模块，以提升系统的模块化和可维护性
a0d72d8 - DraTelligence, 2025-12-10 : feat: 初步实现了矩阵展示功能
1ef4c87 - DraTelligence, 2025-12-10 : feat: 初步实现了矩阵输入功能
993fb6f - DraTelligence, 2025-12-10 : refactor: 重命名了矩阵生成模块路径，适配项目其余部分索引
544be74 - Ricky Sun, 2025-12-10 : refactor: 客户端 simplify serial data handling and improve
↪  matrix data sending
14aef77 - DraTelligence, 2025-12-10 : feat: 单独分离出了矩阵发送模块
f292b3d - DraTelligence, 2025-12-10 : fix: 修改了uart中时钟频率和项目不匹配的问题
c882f47 - DraTelligence, 2025-12-10 : Merge branch 'main' of
↪  https://github.com/rickysun2006/cs207-digital-logic-project
b7e3911 - Ricky Sun, 2025-12-10 : feat(matrix_alu): 增强性能
f7d3103 - DraTelligence, 2025-12-10 : fix: 修复了默认波特率不符合预期的问题
77c060a - DraTelligence, 2025-12-10 : chore: 修改了变量的命名，以更符合直观
7783121 - DraTelligence, 2025-12-10 : feat: 完成了负责生成随机矩阵的模块，实现了矩阵的创建和数据写
↪  入功能
e6d7a62 - DraTelligence, 2025-12-09 : feat: 完善了随机数生成器，使其直接生成范围内的数
d2422de - DraTelligence, 2025-12-09 : feat: Removed redundant wr_en/wr_id logic
d2ff6f7 - DraTelligence, 2025-12-09 : chore: 将随机数逻辑单独模块化
2d34b6b - DraTelligence, 2025-12-09 : feat: 完成了输入控制器模块的设计与实现S
130c5fb - DraTelligence, 2025-12-09 : fix: 删除了用于测试的多余代码
37896e9 - Ricky Sun, 2025-12-09 : feat: 添加README_REQUIREMENTS，方便和copilot对接需求
3e0a254 - Ricky Sun, 2025-12-09 : feat: 完成uart通信客户端（未上板测试）
251c44c - DraTelligence, 2025-12-09 : Merge branch 'main' of
↪  https://github.com/rickysun2006/cs207-digital-logic-project
1322a86 - DraTelligence, 2025-12-09 : fix: 修复了 input_controller.sv 中选择矩阵 ID B 时的错误变
↪  量赋值
e4519e8 - Ricky Sun, 2025-12-09 : feat: 添加UART传输状态机以处理结果传输（结果回传和半自动交互，为
↪  后续客户端做铺垫）
44ba94b - Ricky Sun, 2025-12-09 : feat: 把system_core给写掉了，但是这一部分没法用iverilog跑仿真，
↪  需要去vivado上跑
9e3f9c2 - DraTelligence, 2025-12-09 : Merge branch 'main' of
↪  https://github.com/rickysun2006/cs207-digital-logic-project
a871c25 - Ricky Sun, 2025-12-09 : feat: 完成了alu算术部分的设计，基本完成alu的tb并跑通，但alu还需
↪  进一步测试
35b28bb - DraTelligence, 2025-12-07 : chore: 删除了误上传的仿真编译文件
5a89525 - DraTelligence, 2025-12-06 : feat: 完成了矩阵存储系统和输入控制器系统的初步实现
4d14340 - DraTelligence, 2025-12-06 : refactor: 将uart代码移至common目录下
b6bad6d - DraTelligence, 2025-12-06 : fix: 回退了错误的对段选逻辑的修改
004e7c1 - DraTelligence, 2025-12-05 : feat: 更新了matrix_unit.sv以符合SystemVerilog编码风格
33fd053 - DraTelligence, 2025-12-03 : docs: 给部分结构体成员添加了更详尽的注释
```

```
e2f7bc0 - Ricky Sun, 2025-12-03 : feat: 添加对.vvp文件的忽略规则，防止仿真时候拉屎
3dbe746 - Ricky Sun, 2025-12-03 : feat: 完成matrix_unit模块的存储逻辑，仿真验证通过
b48a271 - Ricky Sun, 2025-12-03 : feat: 生成.gitignore to include Vivado generated files and
↪  directories
a391094 - DraTelligence, 2025-12-03 : feat: 基本完成了主状态机的设计
2a94978 - DraTelligence, 2025-12-02 : feat: 完成了fsm的基本框架，完成
↪  了STATE_IDLE和STATE_CALC_SELECT         状态的逻辑，并对上层模块做了相应的修改。
c0f2bff - DraTelligence, 2025-12-01 : fix(xdc): 修正数码管引脚映射顺序，使其与物理排列一致
4910de4 - DraTelligence, 2025-12-01 : feat: add code_t for 7 seg display
367ad87 - DraTelligence, 2025-12-01 : feat: add seven segment display driver module
56b7a09 - DraTelligence, 2025-12-01 : feat: 完成了物理顶层模块，在其中实例化了按钮消抖模块，并实例
↪  化了逻辑顶层模块 system_core。
0c59491 - DraTelligence, 2025-12-01 : feat: Add button debounce module
6d85538 - DraTelligence, 2025-12-01 : feat: 添加完整约束文件并更新相关文档
52f2990 - DraTelligence, 2025-11-29 : feat(uart): 引入第三方 UART 驱动包
c3b0f3a - DraTelligence, 2025-11-29 : feat: 新增 SystemVerilog 项目包定义
0015199 - DraTelligence, 2025-11-23 : chore: initialize project structure and scaffolding
6ce310f - rickysun2006, 2025-11-18 : Initial commit
```