Prompt 的构成按颜色简单分割了一下：

<指令: 希望模型执行的特定任务或指令>

<上下文: 引导模型做出更好响应的外部信息或附加上下文(给出推理步骤)(指出错误)>

<输入数据: 我们有兴趣寻找答案的输入或问题, 告知模型需要处理的数据>（没有标出来）

<输出指标: 输出的类型或格式>（没有标出来）

（ps：这几部分主要内容是模块结构方面的，涉及 hdl 代码的对话比较少。）

一、模块间的衔接

1. Let us make a brand new microprocessor design together. We're severely constrained on space and I/O. We have to fit in 1000 standard cells of an ASIC, so I think we will need to restrict ourselves to an accumulator based 8-bit architecture with no multi-byte instructions. Given this, how do you think we should begin?

分析：在最初的对话中，工程师给出了设计目标，设计的限制，设计的建议（要求），以及指令。

2.Here is the full ISA specification for the 8-bit accumulator-based RISC processor.

<table>

We now want to produce an implementation for this processor. We are extremely space constrained, so the processor's memory is a single 32-byte register memory for both data and instructions. Can you define the registers we would need for correct operation?

3. We are working on a an 8-bit accumulator-based RISC processor.

Here is the ISA that we have defined:

<table>

Internally, it will use the following registers: <table>

I'd like to now consider the implementation of some components needed to make this design. I want to use shift registers as the register primitive so that we can make a long scan chain for debugging purposes. Can you make a generic shift register in Verilog which could be used to instantiate each register so we can have a scan chain? ***（***设计代码***）（***restarts:1***）***

分析：在模块间衔接时时，工程师会告知 LLM 我们正在完成一个 8 位的微处理器设计，然后告诉 LLM 我们已经完成的部分作为上下文，然后告诉 LLM 我们的限制和思路，最后给出明确的指令来设计这一部分。

二、模块错误修正

1. Can you come up with a better encoding for ADDI/SHL4? I think that it should be possible to have ADDI provide all 4 bits of the immediate in a single instruction.

2. You have forgotten the earlier instruction AND which already begins with 100. The ADDI and SHL4 instructions must begin with 111, so try again, taking into account the other instructions you have previously defined.

3. This could work if you re-create all the other opcodes beginning with "111" and then "0" as defined earlier, to instead be "111" followed by "1". Can you rewrite the whole table of instructions now?

4. For these instructions, we don't have an operand. What are some other options for the source for the new PCs?

5. You forgot the ADDI instruction! Can you put that as a fourth table?

6. This won't work, as we can't have a variable-data operand instruction beginning with "111". Are there other superfluous instructions instead of ROR? I'm wondering if ADDI can already be used to implement INC? ***（***restarts:1***）***

7. In this case of this constant-opcode LDAR, where would the memory address come from for the load?

8. I think the specification for LDAR there is as a double pointer? Or am I mistaken?

9. I'm receiving an error that says that concurrent assignment to a non-net 'data_out' is not permitted on the line `assign scan_out = internal_data[WIDTH-1];`<工具反馈> ***(***restarts:4***)***

10. I think this design has some bugs. It looks like the use of data_out in the generate isn't correct, the synthesis tool says there are width mismatches. It's also saying that it can't resolve the hierarchical name for the item 'memory'. <工具反馈><初级人工反馈>

11. I'm getting just one error this time - it says that procedural assignment to a non-register data_out is not permitted on line "data_out = {DATA_WIDTH{1'b0}};"<工具反馈>

分析:

1. 在写 verilog 代码的对话中（9、10、11），工程师使用了论文中的工具反馈的手段。
2. 在通过对话构建模块（1-6）时工程师一般先指出错误，然后提供解决问题的思路，由 LLM 进行修改，有点像思维链的方法。
3. 少数情况下工程师也仅向 LLM 指出有问题的地方，让 LLM 自行发现矛盾来进行修改。（7、8）
4. 工程师需要向 LLM 提供较为详细的思路，精细到指令集的行为级别。

三、模块内容扩充

1.Okay. Let us begin by defining the ISA then. Given we need to fit within 1000 standard cells, so you're correct and we're definitely need to have a small set of instructions. Provide a list of instructions and the binary encoding for each. ***(***restarts:1***)***

2. I like this ISA! I only see a couple of gaps. Can we have an instruction for inverting the accumulator, zeroing the accumulator, rotating it left and right, and one for saving the current address to the accumulator on branch? I think we could add them in the additional space for the instructions beginning "111". ***(***restarts:1***)***

3.Looks good! I think we have just a few more gaps considering constants and accumulator manipulation. Can we add a few more instructions to the "111" opcode space? Firstly, add instructions to increment or decrement the accumulator. Then, Firstly, an instruction to add a 4-bit immediate to the accumulator, and another to shift the accumulator left by 4 bits. This will enable us to set the accumulator value using 4 instructions. Just provide the new instructions, don't print the whole table. ***(***restarts:2***)***

4. Can you also add an instruction to stop the processor until reset? Add it to the "111" opcode group as well. Perhaps it could be all 1s?

5. Can you now split this table into two tables, one for instructions with variable-data operands and one for instructions with 8-bit constant opcodes? ***(***restarts:1***)***

6. Can you now re-order the instructions with constant opcodes to group similar instructions together? You can update the opcodes to suit, but they will still need to have the leading 1s so that they don't conflict with the existing variable-data operands. Keep the HLT instruction having all 1s.

7. Can you now separate the control and branching instructions from the data manipulation instructions into two tables?

8. Thanks! We now need to increase the specification for the control and branching operations. For each of these, we should describe what happens to the program counter (and accumulator, if necessary). Where does the PC get the new value from for each instruction?

9 I like the specification for JMP and JSR, however we have no way of specifying a relative offset for the BEQ and BNE instructions in the current ISA. The ISA is however not currently using a few encodings. Perhaps we could support a BEQ/BNE which specifies a jump forwards or backwards, each way skipping an instruction? Can you provide an update ISA for the control and branching instructions as well as for the data manipulation instructions which suits this?

10. This looks great. Can you now reprint all three of our ISA tables in order ("Instructions with Variable-

Data Operands", "Control and Branching Instructions", "Data Manipulation Instructions") with all of their details to make the complete specification for the 8-bit accumulator-based processor? ***(***restarts:2***)***

11. I feel that there is a gap in the ISA for allowing pointer-like dereferences. Since we have no space left in the opcode map, can you identify an unnecessary instruction in our ISA and explain why it is unnecessary, then suggest a candidate instruction to replace it which could support something like dereferencing?

12.In this case of this constant-opcode LDAR, where would the memory address come from for the load?

13. This is great, thanks. Let's finalize the specification by re-printing each table but adding a new column which describes what happens to each register in the design (as you already did for the Control and Branching Instructions table). Can you start with the Immediate Data Manipulation Instructions and Data Manipulation Instructions tables?

14.Can you now also reprint the Instructions with Variable-Data Operands (LDA/STA/ADD/SUB/AND/OR/XOR) table?

15. Can you re-type the Control and Branching Instructions table but make it so the branch forward instructions jump an extra instruction? As it turns out, skipping just one instruction isn't enough!
Here's the previous table for your reference:
<Control and Branching Instructions>

16. Given that memory will be implemented using registers, can we skip any of these registers and just provide their values combinatorially?

17. Lets remove the MAR and MDR, but keep the IR so we don't need concurrent read and write. Let's also reduce the PC to the minimum size you suggested. What in the ISA would require a status register? ***(***restarts:1***)***

18. Let's not plan to expand this ISA, as our exclusive focus here is on space requirements. Given this, can you provide a finalized list of registers, their size in bits, and their purpose?

19. Using these three registers as a start point, can you define the other components that we need to implement this processor? ***(***restarts:1***)***

20. Sounds great - can you provide a list of the control signals that would be needed in this design's datapath? Assume that we don't have a container register file, we will just have the three necessary registers.

21. Excellent, that is now working. We'll also need a large register bank for memory according to the defined ISA. Can you implement this in Verilog? I'd also like to be able to connect the memory bank to the scan chain. ***(***设计代码***)***

分析:

1. 大部分 prompt 格式大体为<我们要做什么>\<有什么要求>\<明确具体的指令>（例如 1、7 等）。
2. 类似打印列表这样的简单的工作（例如 11、15 等），工程师选择了简短的指令，直接说明具体工作内容。
3. 此外在一些复杂的内容扩充的任务中，工程师使用了思维链提示的方法，通过逐步的分析来说明我们为什么这么做，或者给出任务的细分步骤（例如 9）。

关于 restarts：这几段的对话 restarts 次数都不多，也不集中，只有在工具反馈的时候有个对话 restart 了 4 次，其它对话都不超过 2 次。