### 数据库方案思路

目前数据库的主要工作在对一个完整设计的代码的处理上，对处理后的代码的要求包括数据库能完整的实现所需设计，做到不缺少相应模块；模块间的相似度要做低，做到不重，避免提问时从数据库中找到多个相似答案造成大模型回答混乱。此外，处理后代码文本需要按模块分段以存入数据库中。因此，目前的数据库方案主要需要完成的工作是，对代码处理降低代码相似度，使不同模块的特征区分开来；对代码按模块进行分段，存入数据库中。最后依据一个小模块为例对数据库方案进行评估。

### 代码相似度降低方案

#### 模块前添加注释

通过在每一个模块前添加注释，比如添加模块实现的功能，该模块如何实现的等信息，降低不同模块间的重合度，使模块的特征更加凸显。

#### 模块名称命名规范

规范模块名称命名，降低模块间重合度，突出模块特征。

#### 形式化注释

对模块内的关键信号添加形式化注释，也对模块内某一重要的always块添加注释，比如一个状态机电路，他的状态转移情况对于不同的状态机电路总是不大一样的，此时添加一些状态转移的描述也可以降低代码间的相似度。

### 代码分段方案

#### 按模块分段

将完整的设计代码按.v模块分为多个txt文本，以实现分段。

#### 对于大模块的处理

可能存在一些.v文件虽然只有一个模块，但有很高的代码量，这个时候有两种处理思路，一种是先尝试让大模型全吃进去，如果可以正常处理学习并有较好的回答那便没有问题。另一种思路是将这个大模块分割成连续的两个或多个部分，先生成第一个部分，在第二个代码文件开头添加注释，比如“第二部分”“继续回答”，当第一个txt输出生成第一个回答后，按照提问者要求“继续回答”，把第二个txt以一个新的回答来告诉提问者。

### 数据库方案评估

以CTRL.v模块为例：

|  |
| --- |
| //The CTRL module is the controller in the CPU core, which outputs the control signals of other modules in the CPU, is responsible for controlling the program running process, and realizes the classic CPU workflow of fingering, decoding, execution, and writing back.  module CTRL(  input clk,  input rst\_n,  input [3:0]ins,  output reg CTRL\_REG\_write,  output reg CTRL\_REG\_read,  output reg CTRL\_ROM\_en,  output reg CTRL\_ROM\_read,  output reg CTRL\_RAM\_en,  output reg CTRL\_RAM\_write,  output reg CTRL\_RAM\_read,  output reg CTRL\_PC\_en,  output reg CTRL\_PC\_chg\_en,  output reg CTRL\_ACC\_en,  output reg [1:0]fetch,  output reg ADDR\_sel  ); |

红字为添加修改的地方，目的是减小代码重合。

|  |
| --- |
| //定义指令编码  localparam NOP = 4'b0000;//空指令 short  localparam LDO = 4'b0001;//从ROM取数据 long  localparam LDA = 4'b0010;//从RAM取数据 long  localparam LDR = 4'b0011;//从ACC取数据 short  localparam PRE = 4'b0100;//从REG取数据 short  localparam STO = 4'b0101;//向RAM写数据 long  localparam ADD = 4'b0110;//操作数相加 short  localparam SHL = 4'b0111;//逻辑左移 short  localparam SHR = 4'b1000;//逻辑右移 short  localparam SAR = 4'b1001;//算数右移 short  localparam INV = 4'b1010;//按位取反 short  localparam AND = 4'b1011;//按位与 short  localparam OR = 4'b1100;//按位或 short  localparam XOR = 4'b1101;//按位异或 short  localparam JMP = 4'b1110;//跳转 long  localparam HLT = 4'b1111;//停机指令 short  //定义状态机编码  localparam IDLE = 4'hf;//初始状态  localparam S0 = 4'h0;//取指0  localparam S1 = 4'h1;//译码 PC+1  localparam S2 = 4'h2;//HLT停机  localparam S3 = 4'h3;//Long0 取指1 RAM/ROM寻址地址  localparam S4 = 4'h4;//Long1 PC+1  localparam S5 = 4'h5;//Long2 LDO/LDA访存取数  localparam S6 = 4'h6;//Long2 STO\_0读REG  localparam S7 = 4'h7;//Long3 STO\_1写RAM  localparam S8 = 4'h8;//Short0 PRE/ADD读REG  localparam S9 = 4'h9;//Short0 LDR\_0 ACC写REG  localparam S10 = 4'ha; |

这一部分是CTRL模块中定义参数的一段代码，实际上可以单拿出来创一个新的define.v文件，在CTRL.v模块上引入头文件define即可，通过这样的代码分段减少一次回答的文本量，减轻大模型工作负担。

|  |
| --- |
| //定义CTRL状态寄存器  reg [3:0] CTRL\_cstate = IDLE;  reg [3:0] CTRL\_nstate = IDLE;  //CTRL状态转移同步逻辑  always @(posedge clk) begin  if(!rst\_n)  CTRL\_cstate <= IDLE;  else  CTRL\_cstate <= CTRL\_nstate;  end |

对于大多数的状态机电路，这一段代码重复度较高，通过修改信号命名以及一些注释以降低代码相似度。