

芯动力——硬件加速设计方法

第七章 基于平头哥E902处理器的SoC设计

——(7) 案例：无剑100SoC与softmax硬件加速IP的集成

邸志雄@西南交通大学

zxdi@home.swjtu.edu.cn

slides与源代码网址 <http://www.dizhixiong.cn/class5/>

案例：无剑100 SoC与Softmax
硬件加速IP的集成

通过该案例来展示集成方法与验证方法

主要主要基于PYNQ板卡作品的主要区别

主要区别

- 计算单元通过AHB总线挂载到处理器上
- 源数据、计算中间数据和结果保存到RAM中而非DDR
- 需要PC端完成计算结果和标准结果的对比

通过本节内容掌握无剑100 SoC集成硬件IP的方法

SoftMax激活函数简介

含义:

SoftMax 函数是神经网络中的一种输出层函数，计算输出层的值，主要用于神经网络最后一层。

- 假设我们有一个数组 x ， x^i 表示 x 中的第 i 个元素，那么这个元素的SoftMax值就是：

$$S_i = \frac{e^{x_i}}{\sum_{j=1}^N e^{x_j}}$$

也就是该元素的指数与所有元素指数和的比值。

设计方案

1、计算 $e^{x-x_{\max}}$ 的值，简化设计过程值。

- 输入数据范围在 $[-10, 10]$ 时， x 符号不确定、 e^x 范围为 $[0.0000454, 22026.4657948]$ 这样设计和计算过程将会十分复杂。

$$S_i = \frac{e^{x_i}}{\sum_{j=1}^N e^{x_j}} = \frac{e^{x_i - x_{\max}}}{\sum_{j=1}^N e^{x_j - x_{\max}}}$$

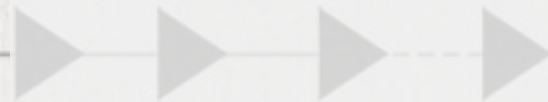
- 只需要计算 $e^{x-x_{\max}}$ 即可。
- $(x - x_{\max}) \in [-20, 0]$, 符号全为负，可以省略符号位 简化设计过程。
- $e^{x-x_{\max}} \in [0, 1]$, 在数据宽度相同的条件下提高了数据精度。



设计方案

采用创新的基底查找法

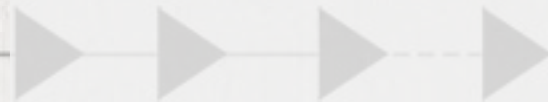
2、LUT直接存放 e^i 的值，计算精度损失很小值。

- 传统的解决方案是通过Taylor级数展开，将转化后的多项式系数保存在ROM表。但是Taylor级数展开，存在着不可避免的精度损失。
 - 在ROM表中直接存放 e^i 的值，没有计算精度损失。
- 



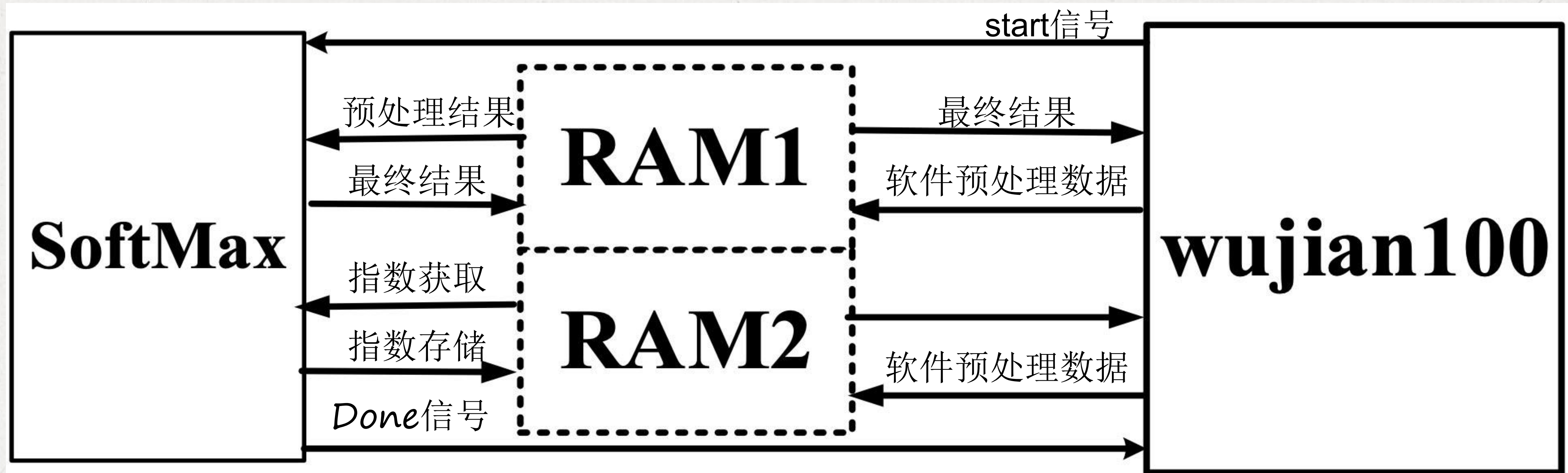
设计方案

3、使用特殊的定点化方式，配合移位操作进行基底划分值。

- 16bit定点数中附带小数点位置信息，其中高三位用于表示小数点位置，剩下的是数据位，用于记录数据。
 - 这样使数据更清晰易操作，丰富信息量。
 - 尤其在输入范围较大的情况下，优势非常明显。
- 

整体硬件结构

- SoftMax 函数计算涉及复杂的指数和除法计算，在运算中需要计算所有指数值后才可计算每一个函数值，设计中采用了乒乓RAM操作。



整体硬件结构

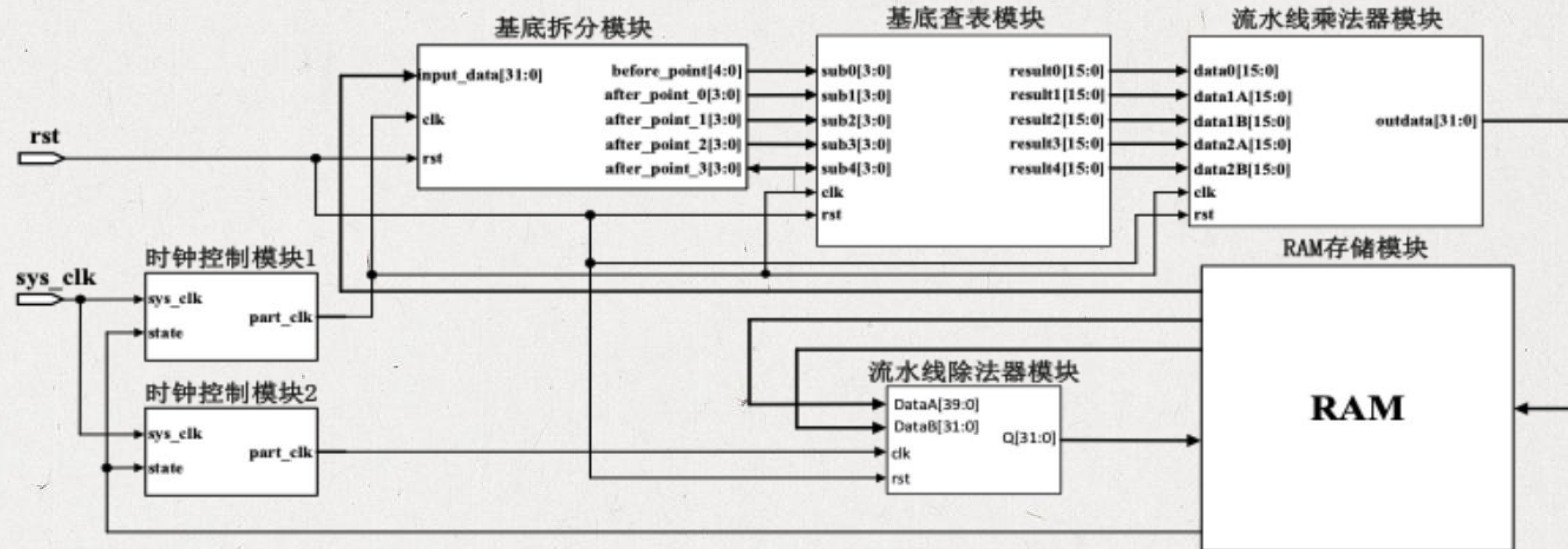
整体硬件结构

设计中的硬件部分整体框架结构图如下：

指数计算

除法计算

两条流水线



- 1. 硬件部分获得CPU端处理后数据并进行基底拆分，经查找表后获得各个部分数据并送入乘法器，计算结果送入RAM中。

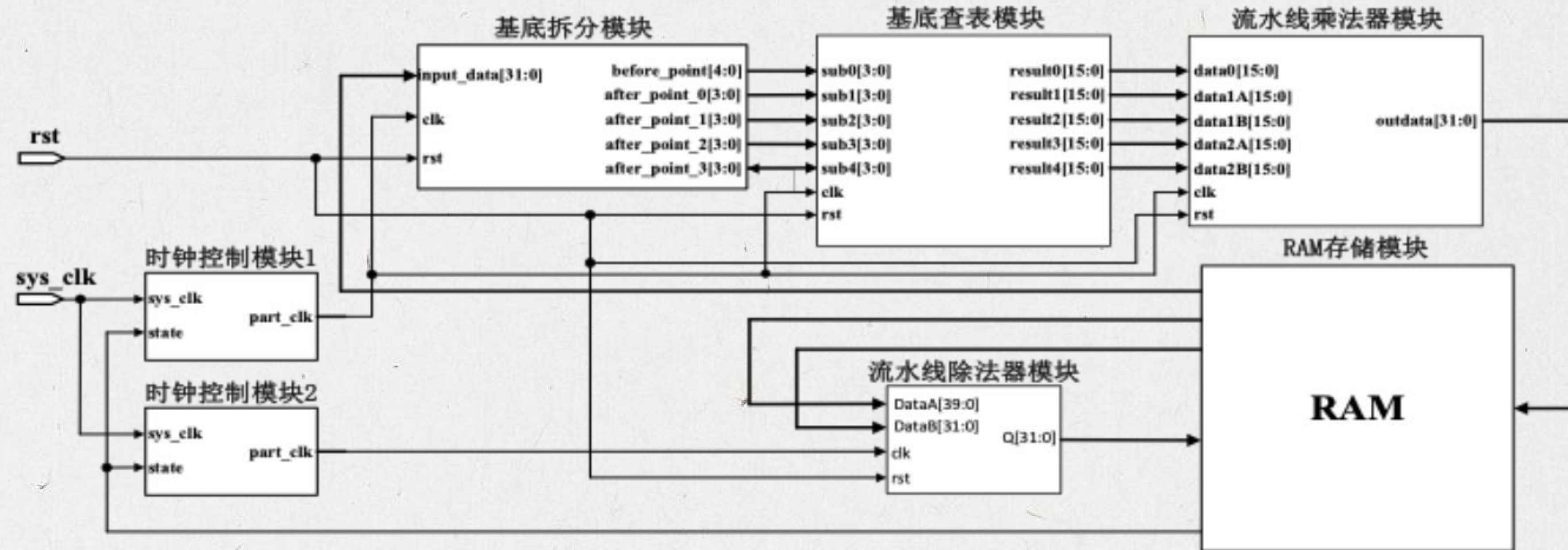
整体硬件结构

设计中的硬件部分整体框架结构图如下：

指数计算

除法计算

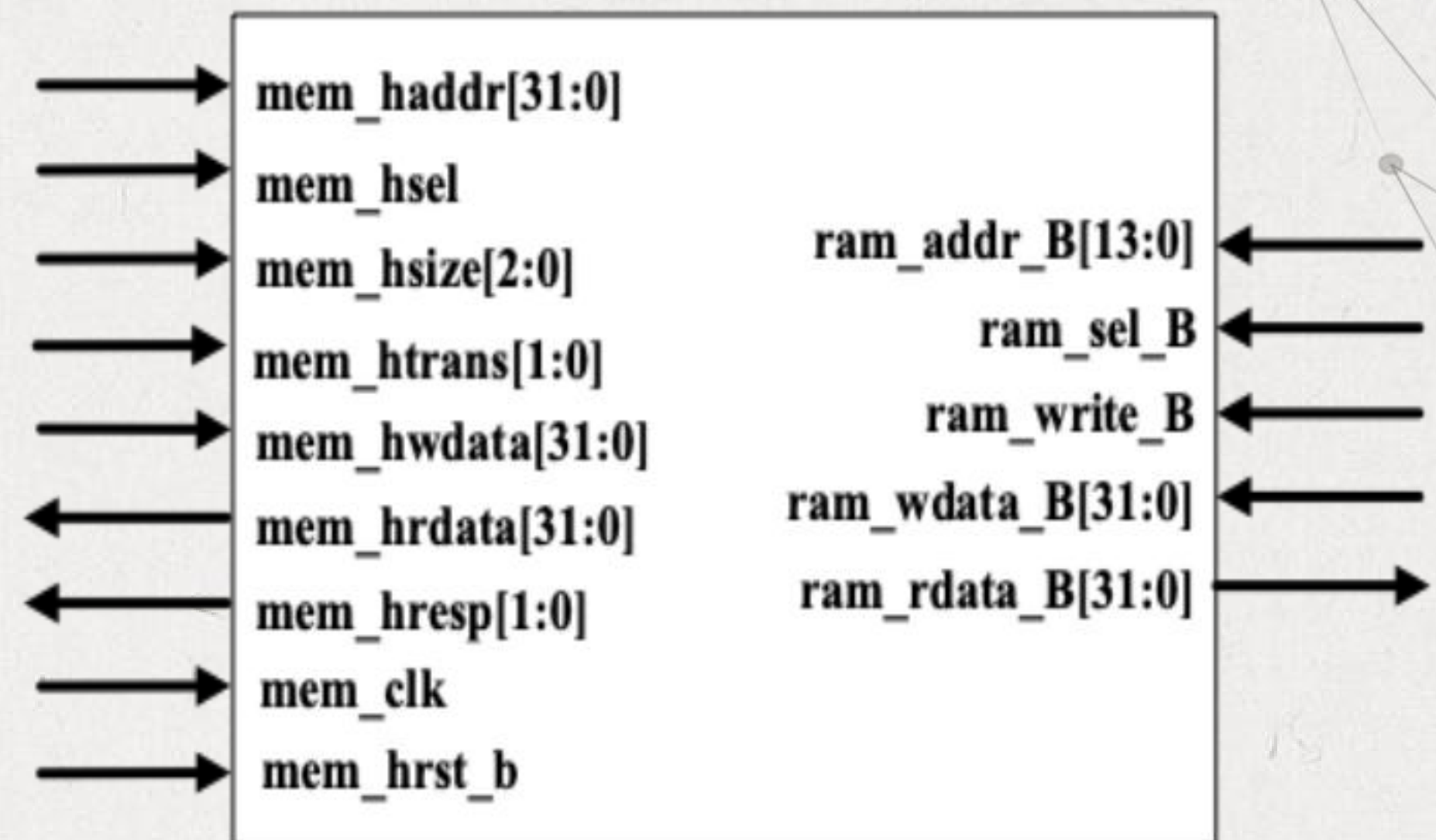
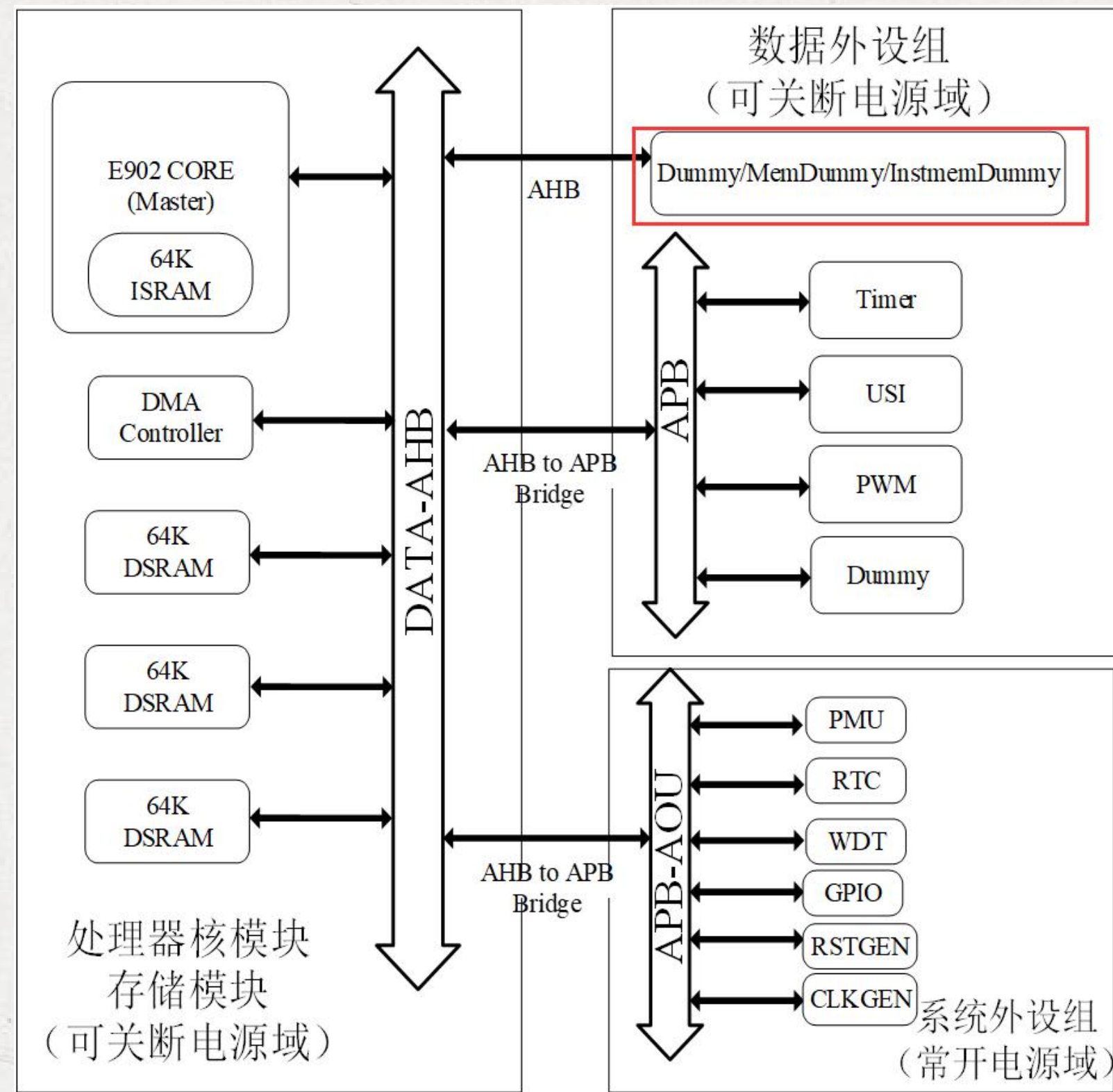
两条流水线



- 2. 硬件部分从RAM中获得指数计算中间结果并送入除法器，计算出最终结果后送入RAM，供CPU读取。

RAM存储模块

- 该模块包含了**两个双端口RAM模块**，用于构成乒乓RAM操作。其中RAM的双端口中一个端口为AHB读写接口，用于集成到wujian100 SoC中，另外一个端口用于SoftMax计算模块存放指数中间运算和最终运算结果。



RAM存储模块

- 两个双端口RAM模块通过wujian100 SoC预留的Dummy模块集成到系统中，分别为ahb总线上的从机s7和s8。

0x4001_0000~0x4001_FFFF	Dummy	64K	S7	main_dummy_top0
0x4002_0000~0x4002_FFFF	Dummy	64K	S8	main_dummy_top1

将输入原始数据送入s7并完成计算的配置



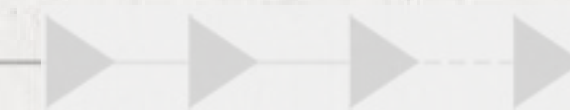
SoftMax硬件计算单元主动从s7中读取数据并将计算中间结果存放入s8中



求和单元会把中间结果进行累加



计算单元会完成求和与除法运算，结果存入s7

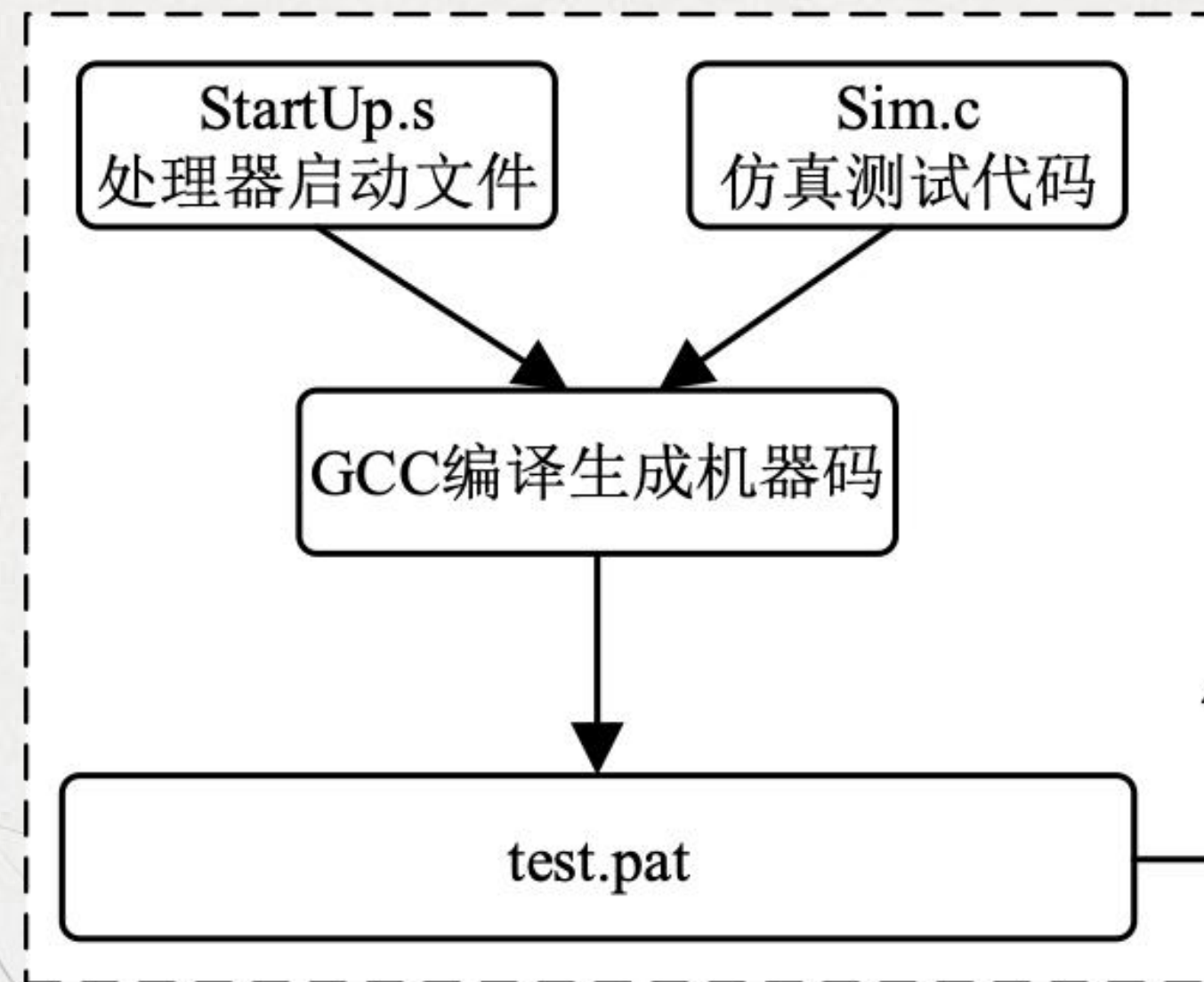


仿真原理

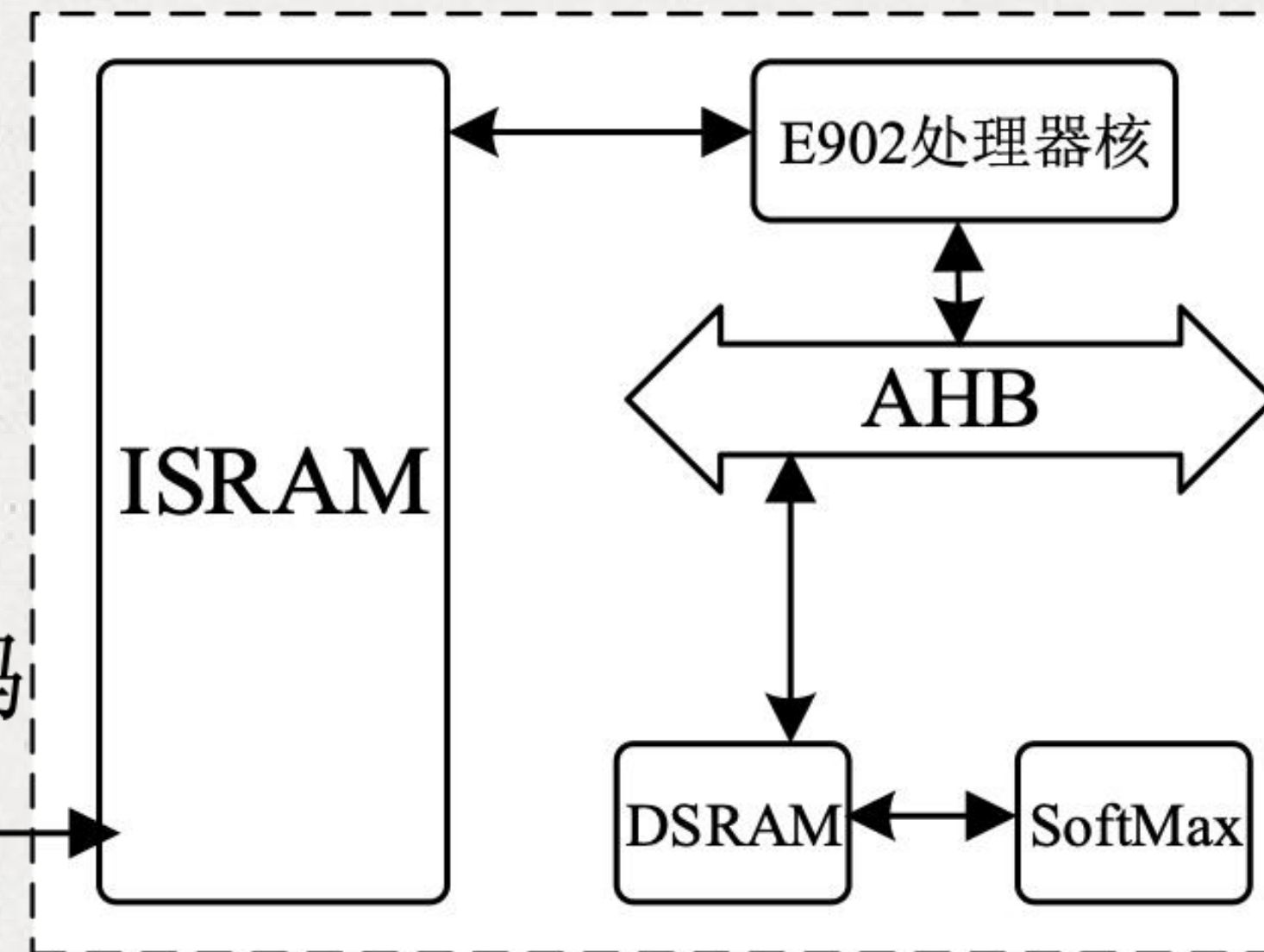
下图是一个仿真流程总图，包括了最简单的编译流程以及系统仿真框架。

- 通过RISC-V GCC工具链将编写的C代码仿真程序变成二进制的机器码test.pat。
- Testbench中在仿真初始前将机器码按地址存储至**ISRAM**中，系统复位后处理器软硬件协同工作。
- 仿真主要使用平头哥官方提供的**RISC-V工具链**和**仿真脚本**。

编译流程



CPU执行(Vivado仿真)



搭建仿真环境

- wujian100开源平台提供了一套完整的仿真流程，用户只需搭建好仿真环境后就可以通过调用提供的仿真脚本完成仿真，但在仿真前需要预先安装好RISC-V工具链。E902处理器的设计中含有用户自定义指令，需要对标准RISC-V编译器进行修改，因此需要使用平头哥官方提供的工具链。

```
Directory Structure
|--Project          //open source project work directory
|--riscv_toolchain //tool chain install directory download from t-head.cn
|--wujian100_open  //wujian100_open project get from github
|--case            //test case example for simulation
|--doc             //wujian100_open user guide
|--fpga            //FPGA script
|--lib             //compile script for simulation
|--regress         //regression result
|--sdk             //software design kit
|--soc             //Soc RTL source code
|--tb              //test bench
|--tools           //simulation script and setup file
|--workdir         //simulation directory
|--LICENSE
|--README.md
```

存放在同一目录

外设的仿真案例

存放的仿真结果

测试仿真文件

测试仿真文件
与配置文件



搭建仿真环境

- 由于仿真过程主要利用提供的脚本，而脚本中会通过相对路径调用RISC-V工具链，因此这里需要按照说明将RISC-V工具链和wujian100_open开源项目放入同一目录下。

```
Directory Structure
|--Project          //open source project work directory
|--riscv_toolchain //tool chain install directory download from t-head.cn
|--wujian100_open  //wujian100_open project get from github
|--case            //test case example for simulation
|--doc             //wujian100_open user guide
|--fpga            //FPGA script
|--lib             //compile script for simulation
|--regress         //regression result
|--sdk             //software design kit
|--soc             //Soc RTL source code
|--tb              //test bench
|--tools           //simulation script and setup file
|--workdir         //simulation directory
|--LICENSE
|--README.md
```

存放在同一目录

外设的仿真案例

存放的仿真结果

测试仿真文件

测试仿真文件
与配置文件

搭建仿真环境

- 开源工程提供的外设仿真源文件位于 case 目录下，该目录下包含处理器运行的 c 代码和用于仿真强制激励的 verilog 源码。
- 通过调用仿真脚本会将测试的 c 代码生成二进制可执行 test.pat 文件存放如 workdir 目录下，仿真的结果会存放在 regress 目录。

tools 目录下存放有仿真脚本和配置文件

存放在同一目录

Directory Structure

```
--Project          //open source project work directory
|--riscv_toolchain //tool chain install directory download from t-head.cn
|--wujian100_open  //wujian100_open project get from github
|--case            //test case example for simulation
|--doc             //wujian100_open user guide
|--fpga            //FPGA script
|--lib             //compile script for simulation
|--regress         //regression result
|--sdk             //software design kit
|--soc             //Soc RTL source code
|--tb              //test bench
|--tools           //simulation script and setup file
|--workdir         //simulation directory
|--LICENSE
|--README.md
```

外设的仿真案例

存放的仿真结果

测试仿真文件

测试仿真文件
与配置文件

调用仿真脚本编译程序

- wujian100开源项目提供了vcs、iverilog两种仿真工具的运行脚本，在仿真前需要先将仿真工具的路径放入setup.csh脚本中，如果采用vcs仿真工具需要额外将其license路径放入setup.csh文件中。在仿真前需要先运行setup.csh脚本，然后再进入相关路径调用②中所示脚本。

Get ready for simulation

```
1. cd wujian100_open/tools
2. vim setup.csh then add the vcs path and license
① 3. source setup.csh //if not success you can touch a new file named setup.csh and copy the content
to the new file. then source the new file
4. cd wujian100_open/workdir
5. if you want to use iverilog as simulation tool please execute the command './tools/run_case -sim_tool
iverilog ../case/timer/timer_test.c' or if you want to use vcs as simulation tool please execute the command
② ./tools/run_case -sim_tool vcs ../case/timer/timer_test.c
```

- 执行②中的脚本会首先编译测试程序，然后在workdir目录下生成test.pat二进制文件，根据调用脚本传入仿真工具的参数选择用vcs/iverilog工具进行仿真。这里我们使用vivado进行仿真时只需调用脚本生成test.pat文件后，再通过测试文件中将其加载入指令RAM中即可。

wujian100 SoC仿真:②代码加载至指令RAM

tb.v测试激励文件中完成将代码加载入指令RAM, 使用Vivado工具进行仿真时需要详细写出test.pat文件的路径。

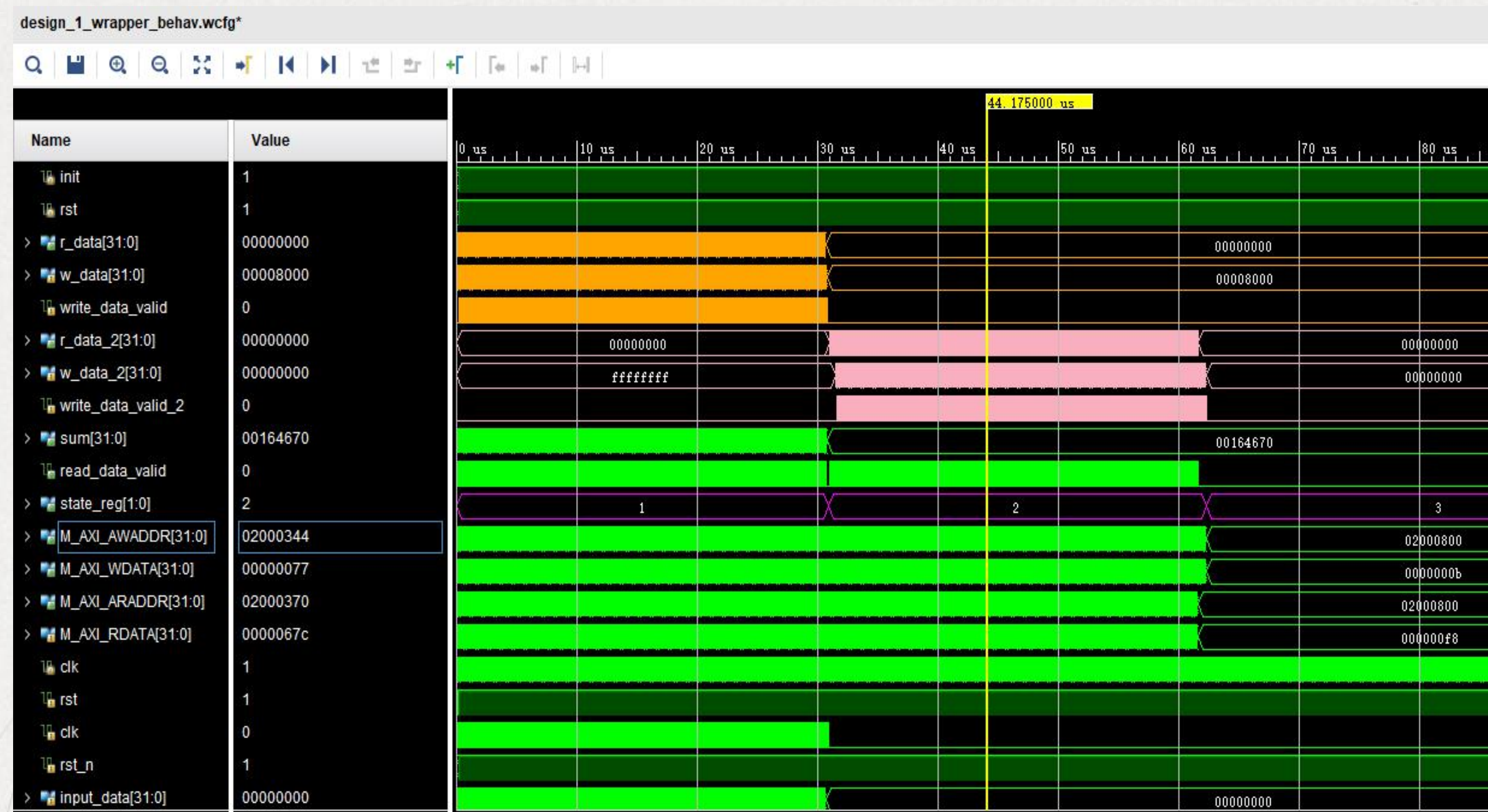
```
////////////////////////////////program download////////////////////////////////
initial
begin : load_program
integer j;
integer k;
  reg [31:0] one_word;
`ifdef iverilog
  reg [31:0] temp_mem[16384];
`else
  reg [31:0] temp_mem[integer];
`endif
  $readmemh("test.pat", temp_mem); ; 详细列出test.pat文件路径
  @(posedge PI_SOC_RST_B);
  for(k=0; k<32'h4000; k=k+1)
  begin
    one_word[31:0] = temp_mem[k];
    wujian100_open_tb.x_wujian100_open_top.x_retu_top.x_smu_top.x_sms_top.x_isram_top.x_sms_sram.x_fpga_spram.x_fpga_byte3_spram.mem[k][7:0] = one_word[7:0];
    wujian100_open_tb.x_wujian100_open_top.x_retu_top.x_smu_top.x_sms_top.x_isram_top.x_sms_sram.x_fpga_spram.x_fpga_byte2_spram.mem[k][7:0] = one_word[15:8];
    wujian100_open_tb.x_wujian100_open_top.x_retu_top.x_smu_top.x_sms_top.x_isram_top.x_sms_sram.x_fpga_spram.x_fpga_byte1_spram.mem[k][7:0] = one_word[23:16];
    wujian100_open_tb.x_wujian100_open_top.x_retu_top.x_smu_top.x_sms_top.x_isram_top.x_sms_sram.x_fpga_spram.x_fpga_byte0_spram.mem[k][7:0] = one_word[31:24];
  end
end
```

这段代码将test.pat文件预先存入temp_mem数组中, 再将其中的内容加载进指令RAM中。

```
initial
begin : load_data
integer j;
  @(posedge PI_SOC_RST_B);
  $display("\t*****START TO LOAD PROGRAM*****\n");
  for(j=0; j<32'h4000; j=j+1)
  begin
    wujian100_open_tb.x_wujian100_open_top.x_retu_top.x_smu_top.x_sms_top.x_sms0_top.x_sms_sram.x_fpga_spram.x_fpga_byte3_spram.mem[j][7:0] = 8'h0;
    wujian100_open_tb.x_wujian100_open_top.x_retu_top.x_smu_top.x_sms_top.x_sms0_top.x_sms_sram.x_fpga_spram.x_fpga_byte2_spram.mem[j][7:0] = 8'h0;
    wujian100_open_tb.x_wujian100_open_top.x_retu_top.x_smu_top.x_sms_top.x_sms0_top.x_sms_sram.x_fpga_spram.x_fpga_byte1_spram.mem[j][7:0] = 8'h0;
    wujian100_open_tb.x_wujian100_open_top.x_retu_top.x_smu_top.x_sms_top.x_sms0_top.x_sms_sram.x_fpga_spram.x_fpga_byte0_spram.mem[j][7:0] = 8'h0;
  end
end
```

仿真运行结果

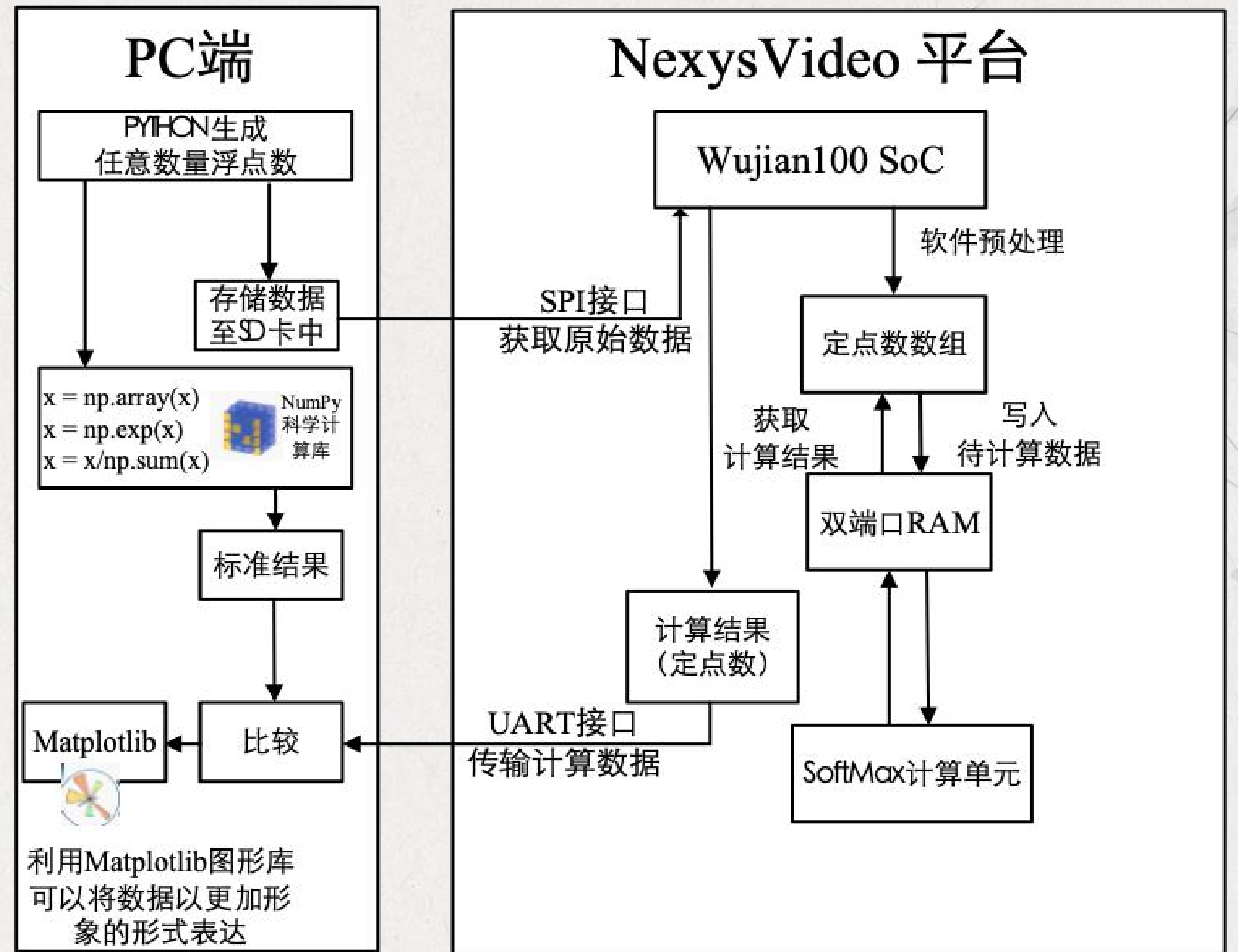
- 为了便于观察结果的正确性，仿真测试时输入的数据为512个32'h00000000。
- CPU输入数据0转换为浮点数是0，SoftMax硬件 计算结果为0x00800000转换为浮点数为0.001953125, $0.001953125 \times 512 = 1$ 。可验证FPGA计算结果正确。



搭建仿真环境

- PC端主要负责**随机数据**和**标准结果**的产生，并与FPGA端计算结果对比并显示误差。
- FPGA端主要分为处理器和SoftMax计算单元。处理器获取SD卡中预先存入的原始数据，经预处理后送入SoftMax计算单元，计算完成后将结果发送至上位机进行对比。

FPGA平台的验证框架



软硬件协同工作

处理器端进行一次计算时的操作

SD卡中获取原始数据进行预处理

送入SoftMax硬件计算单元进行配置

```
f_mount(&FatFs, "", 0); /* Give a work area to the default drive */
fr = f_open(&Fil, "data.txt", FA_READ);
if(fr == FR_OK)
{
    f_gets(buf, 10, &Fil);
    num = atoi(buf);
    // printf("the data number is %d\r\n", num);
    float_data = (float*)malloc(sizeof(float)*num);
    fix_data = (int*)malloc(sizeof(int)*num);
    //get the float data
    for(i=0; i<num; i++)
    {
        f_gets(buf, 20, &Fil);
        float_data[i] = atof(buf);
        // printf("%f\n", float_data[i]);
    }
    max_deal(float_data, num); //max - data[i]
    float2fix16(float_data, fix_data, num); //float to 16 fix_point
    for(i=0; i<num; i++) //write fix_point data to softmax hardware
    {
        write_data_to_softmax(0, fix_data, num);
    }
    softmax_num_config(num);
    softmax_cal_start(); //hardware enable
    printf("hardware starts running\r\n");
    while(get_softmax_status()); //wait until hardware finished
    printf("hardware finished\r\n");
    read_result_from_softmax(0, float_data, num); //get the sub_results from fpga
    read_result_from_softmax(0, fix_data, num); //get the sub_results from fpga
}
```

软硬件协同工作

①从SD卡文件中读取本次进行SoftMax函数计算的数据个数，并从SD卡中加载原始计算的浮点数据至RAM中。

②遍历原始浮点数据获得最大值max，并将每个数据data[i]更新为max-data[i]

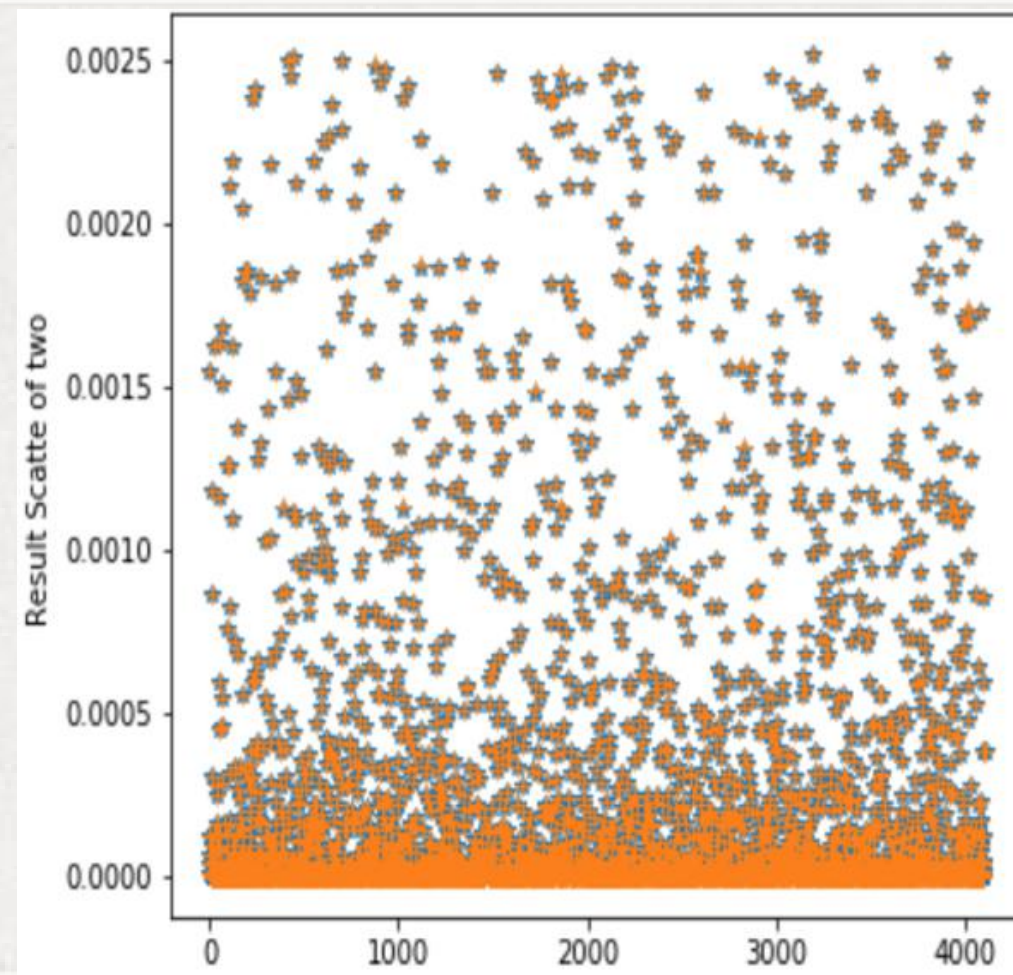
③将更新后的数据做定点化处理。

④从SD卡文件中读取本次进行SoftMax函数计算的数据个数，并从SD卡中加载原始计算的浮点数据至RAM中。

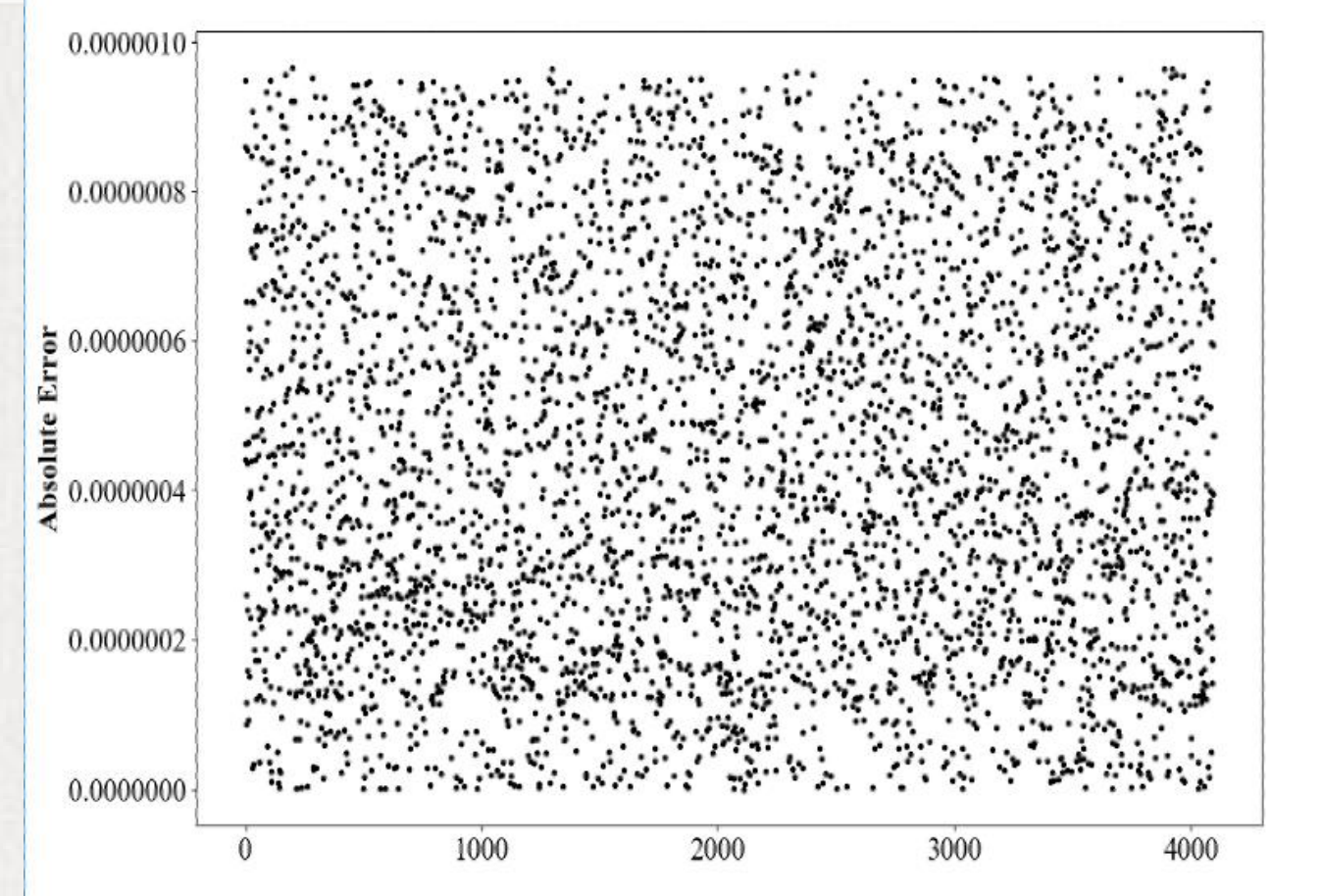
⑤当SoftMax硬件单元计算完成后获取计算的结果。

```
f_mount(&FatFs, "", 0); /* Give a work area to the default drive */
fr = f_open(&Fil, "data.txt", FA_READ);
if(fr == FR_OK)
{
    f_gets(buf, 10, &Fil);
    num = atoi(buf);
    // printf("the data number is %d\r\n", num);
    float_data = (float*)malloc(sizeof(float)*num);
    fix_data = (int*)malloc(sizeof(int)*num);
    //get the float data
    for(i=0; i<num; i++)
    {
        f_gets(buf, 20, &Fil);
        float_data[i] = atof(buf);
        // printf("%f\n", float_data[i]);
    }
    max_deal(float_data, num); //max - data[i]
    float2fix16(float_data, fix_data, num); //float to 16 fix point
    for(i=0; i<num; i++) //write fix_point data to softmax hardware
    {
        write_data_to_softmax(0, fix_data, num);
    }
    softmax_num_config(num);
    softmax_cal_start(); //hardware enable
    printf("hardware starts running\r\n");
    while(get_softmax_status()); //wait until hardware finished
    printf("hardware finished\r\n");
    read_result_from_softmax(0, float_data, num); //get the sub_results from fpga
    read_result_from_softmax(0, fix_data, num); //get the sub_results from fpga
}
```

FPGA平台验证



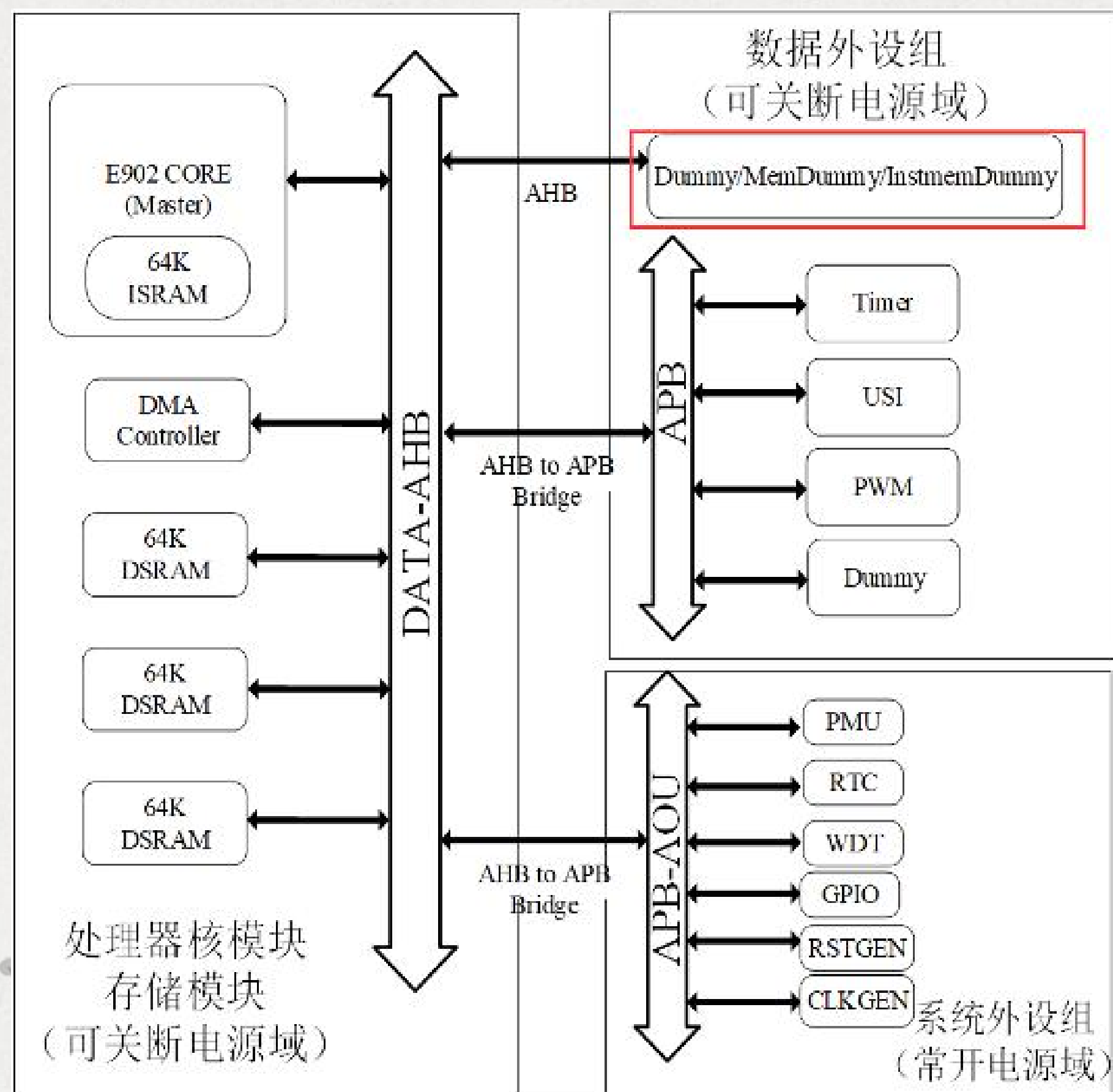
FPGA端结果和标准结果比
对



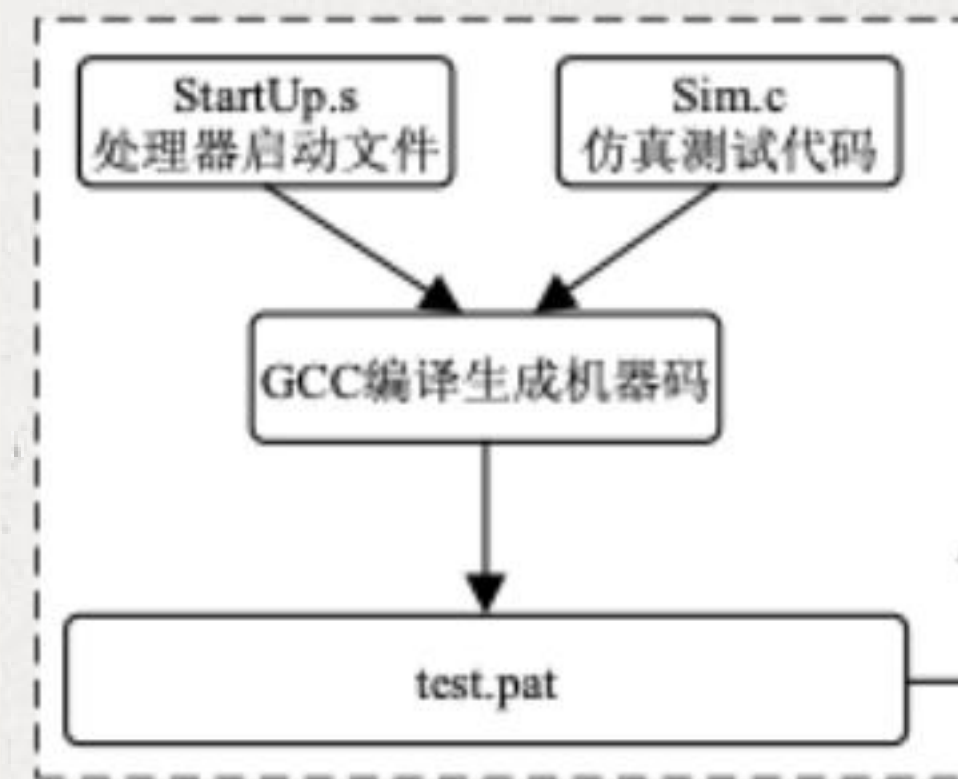
误差散点图

- 随机生成4096个 $[-10,10]$ 内随机浮点数据，上面两张图分别显示了FPGA端计算结果与标准结果的比对和误差显示。
- 绝对误差小于 10^{-6} ，可以满足较高的精度要求。

小节



编译流程



CPU执行(Vivado仿真)

