

《智能计算系统》课程实验报告

实验内容：深度学习处理器运算器设计

姓名：梁宇龙

学号：202228013329016

一、运行环境说明

环境	版本
macOS	12.6.2
icarus-verilog	11.0 (stable)
GTKWave	v3.3.107

表 1 实验环境说明

在源码附件中均保存运行结果可执行程序 and 波形图 VCD 文件，可在 Linux 平台下验证。

二、串行内积运算器

1、实验代码补全

乘法器需计算神经元与权重乘积。

```
wire signed [31:0] mult_res = neuron * weight;
```

加法器对暂存结果与刚刚完成运算的乘法结果进行累加。

```
wire [31:0] psum_d = psum_r + mult_res;
```

对输出信号置高电平需满足已经完成最后一个输入数据的运算，此时 ctl[1]为 1。

```
else if(ctl[1]) begin
    vld_o <= 1'b1;
end
```

2、实验运行结果

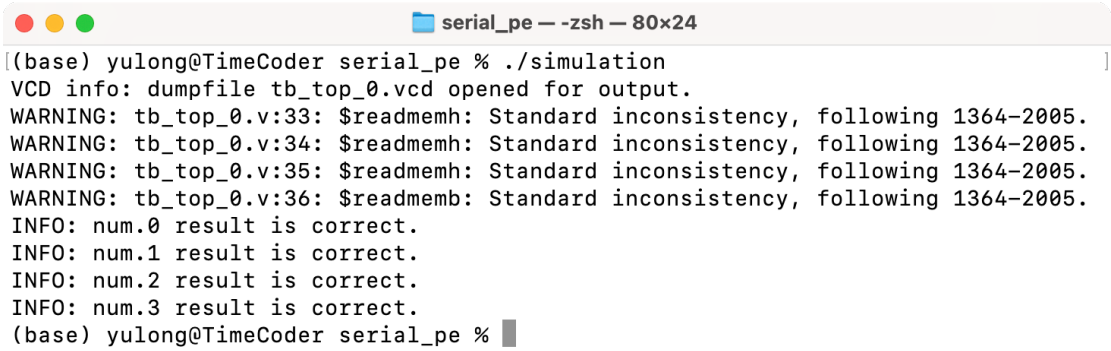


图 1 串行运算器执行结果

3、波形图表示

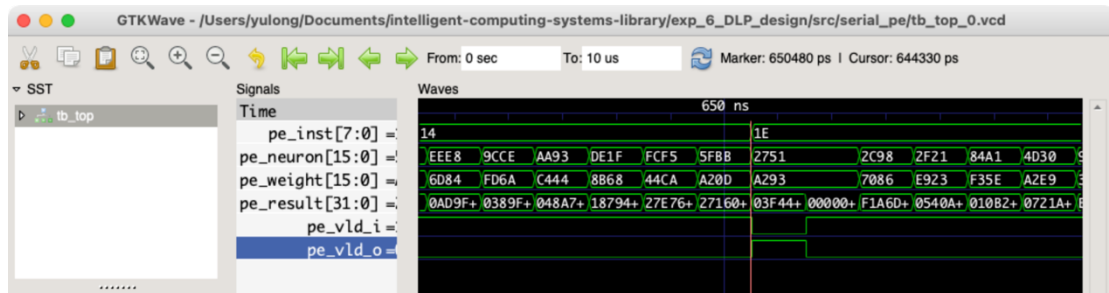


图 2 串行运算器波形图表示

三、并行内积运算器

1、乘法器 pe_mult.v 代码补全

并行运算器输入端为 512 位，即存储了 32 个神经元数值，故对每一次参数处理要进行 32 次运算，并得出 32 个乘法运算结果。

```
for(i=0; i<32; i=i+1)
begin:int16_mult                      /* TODO */
    assign int16_neuron[i] = mult_neuron[(i*16+15):(i*16)];
    assign int16_weight[i] = mult_weight[(i*16+15):(i*16)];
    assign int16_mult_result[i] = int16_neuron[i] *
int16_weight[i];
end
```

在完成并行运算后，需要将 32 个结果拼接传入到加法器中。

```
assign mult_result =
{int16_mult_result[31],int16_mult_result[30], ... ,int16_mult_res
ult[0]};
```

2、加法器 pe_acc.v 代码补全

加法器将乘法器的拼接结果拆开并进行加法运算，运算方式是每两个数进行相加，逐次减半，最终加至一个数。由于乘法器共输入 32 个乘法结果，故第一次加法运算结束后为 16 个结果，以此类推。

```
for(j=0; j<(32/(2**i)); j=j+1)
begin:int16_adder
    if(i==0) begin /* TODO */
        assign int16_result[0][j] = mult_result[(j*32+31):(j*32)];
    end else begin /* TODO */
        assign int16_result[i][j] = int16_result[i-1][j*2] +
int16_result[i-1][j*2+1];
    end
end
```

3、程序主入口代码补全

程序主入口对与串行运算器方法相同，此处不再展示。

4、实验运行结果

```
parallel_pe --zsh -- 80x24
(base) yulong@TimeCoder parallel_pe % ./simulation
VCD info: dumpfile tb_top_1.vcd opened for output.
WARNING: tb_top_1.v:33: $readmemh: Standard inconsistency, following 1364-2005.
WARNING: tb_top_1.v:34: $readmemh: Standard inconsistency, following 1364-2005.
WARNING: tb_top_1.v:35: $readmemh: Standard inconsistency, following 1364-2005.
WARNING: tb_top_1.v:36: $readmemb: Standard inconsistency, following 1364-2005.
INFO: num.0 result is correct.
INFO: num.1 result is correct.
INFO: num.2 result is correct.
INFO: num.3 result is correct.
(base) yulong@TimeCoder parallel_pe %
```

图3 并行运算器执行结果

5、波形图表示

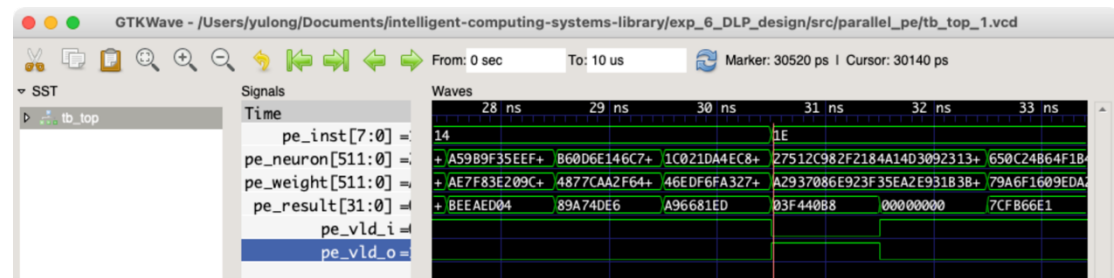


图4 并行运算器波形图表示

四、矩阵运算单元

由于矩阵运算单元在内部实现上使用并行运算器，因此本部分不重复展示并行运算器代码。

1、代码补全

对输入规模的存储，复位时至 0，处理器运行时将规模输入参数与信号进行存储。

```
/* TODO: inst_vld & inst */
if (!rst_n) begin
    inst_vld <= 0;
    inst <= 0;
end else begin
    inst_vld <= ib_ctl_uop_valid;
    inst <= ib_ctl_uop;
end
```

对迭代器设置。迭代器负责对并行运算器进行参数更新，在正常运算（pe_vld_i 置 1）的情况下迭代器进行迭代，当运算完成（pe_ctl[1] 置 1）时，迭代器复位。

```
/* TODO: iter */
if(!rst_n) begin
```

```

    iter <= 8'b0;
end else if(pe_ctl[1]) begin
    iter <= 8'b0;
end else if(pe_vld_i) begin
    iter <= iter + 1'b1;
end

```

在此处对 pe_vld_i 进行调整，除了输入的神经元数据、权值数据和控制信号都有效外，保证运算完成控制信号 ib_ctl_uop_ready 置 1（单元要向外层单元传递运算结果）时单元停止运算。

```

wire pe_vld_i = ib_ctl_uop_valid && wram_mpe_weight_valid &&
nram_mpe_neuron_valid && !ib_ctl_uop_ready;

```

对运算完成控制信号处理，当运算完成（pe_ctl[1]置 1）时将控制信号置 1，使上层单元获取运算结果，并将神经元和权值控制信号置 1，此刻不再接收新的数据。

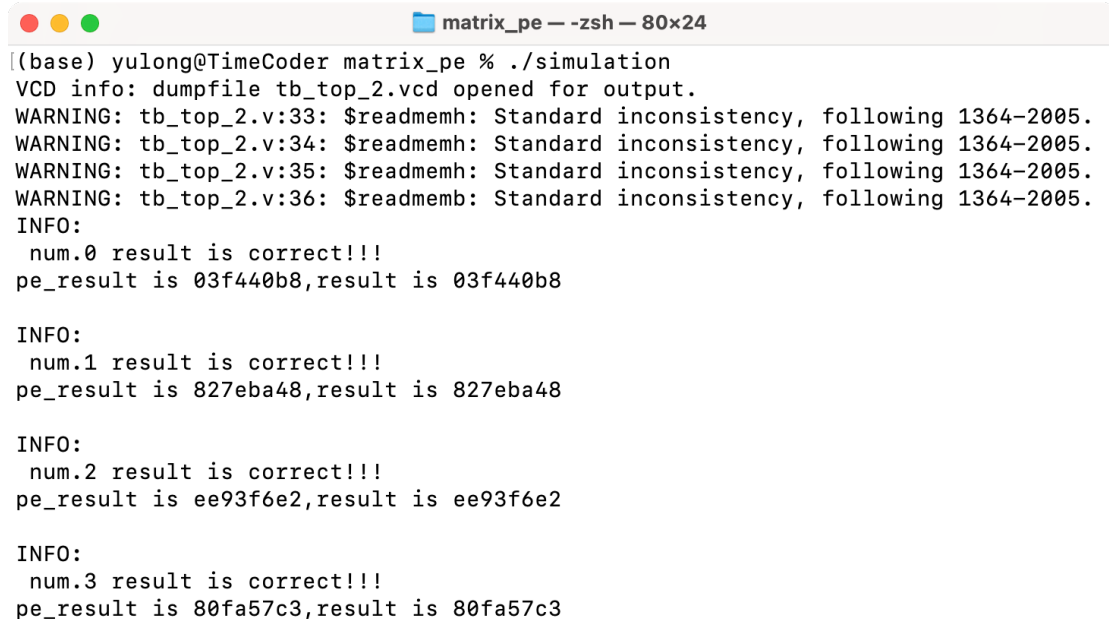
```

if (!rst_n) begin
    ib_ctl_uop_ready <= 0;
end else if (pe_ctl[1]) begin
    ib_ctl_uop_ready <= 1;
end else begin
    ib_ctl_uop_ready <= 0;
end

assign nram_mpe_neuron_ready = ib_ctl_uop_ready;
assign wram_mpe_weight_ready = ib_ctl_uop_ready;

```

2、实验运行结果



```

(base) yulong@TimeCoder matrix_pe % ./simulation
VCD info: dumpfile tb_top_2.vcd opened for output.
WARNING: tb_top_2.v:33: $readmemh: Standard inconsistency, following 1364-2005.
WARNING: tb_top_2.v:34: $readmemh: Standard inconsistency, following 1364-2005.
WARNING: tb_top_2.v:35: $readmemh: Standard inconsistency, following 1364-2005.
WARNING: tb_top_2.v:36: $readmemb: Standard inconsistency, following 1364-2005.
INFO:
num.0 result is correct!!!
pe_result is 03f440b8,result is 03f440b8

INFO:
num.1 result is correct!!!
pe_result is 827eba48,result is 827eba48

INFO:
num.2 result is correct!!!
pe_result is ee93f6e2,result is ee93f6e2

INFO:
num.3 result is correct!!!
pe_result is 80fa57c3,result is 80fa57c3

```

图 5 矩阵运算单元执行结果

3、波形图表示

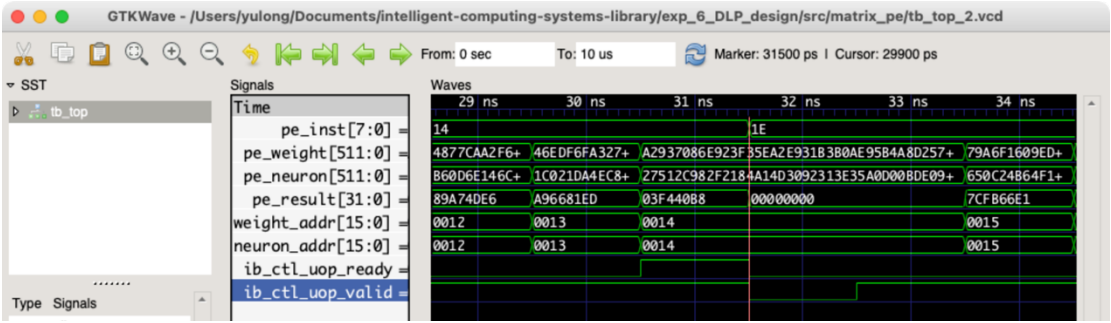


图 6 矩阵运算单元波形图表示