手写数字识别项目的二值化硬件实现

# 【项目背景】

现实中很多复杂工作的实现需要在现场利用FPGA或ASIC等开发板进行调度，这种实现方式有很显著的灵活性，不必依赖于后台的数据传输过程，是一种极有价值的独立处理单元。另外这样的硬件工具相比于传统的计算机而言，有着非常强大的并行能力，有利于数据的快速计算，得到实时结果。如果将特定的分析逻辑移植在开发板中，它将可以以最低的成本和最快的速度来实现特定的目标工程。

另一方面，FPGA等开发板在处理不同类型的数据有很大的差异。由于其灵活的门级结构，FPGA可以对二值化数据进行灵活的调用，而对于浮点数而言，其会消耗极大的逻辑资源（包括查找表和寄存器等）以及布线资源，很大程度上增加了电路的复杂度。于是可以寻求在尽量维持数据和计算精度的同时将数值进行二值化，以最大化利用开发板的工程优势。

考虑到当下如计算机视觉等数据处理工作体现出了越来越实时化的特征，开发板的特点可以在此应用。我们尝试将神经网络进行二值化的训练并将其部署到具备实时性的设备上，进而以此对大量的数据进行处理。在此项目中，我们使用FPGA作为开发工具，建立二值化的全连接神经网络实现手写数字的图像识别（利用MNIST数据库, Mixed National Institute of Stantards and Technology database）。

# 【设计实现】

整体的设计思路分为两个大模块，分别在python的环境下对神经网络进行二值训练，以及运用硬件描述语言verilog实现在FPGA上的实际部署。

## 二值化神经网络构建

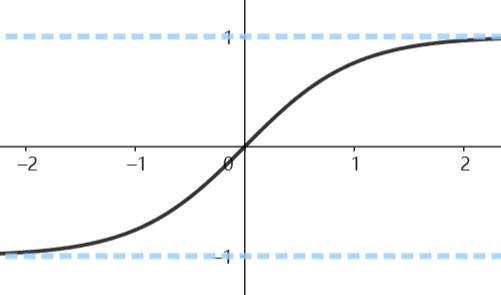
1. **数据处理**

MNIST数据集提供了70000张大小为28\*28的灰度数字图（其中60000张为训练集，10000张为测试集）。为满足全二值化的需求，数据集的二值化是数据处理的关键。

数据集先进行标准化，将均值和标准差都控制为0.5，再将小于0.5的数据全部映射到-1，大于0.5的数据全部映射到1。接着在进行网络后的第一步是把单个的图片数据经过flatten函数扁平化处理，一张图片对应784bit的张量数据。

1. **网络搭建**

在神经网络的训练过程中并不能直接让网络在二值的环境中进行，否则网络中的权重始终是二值会使得训练的梯度难以计算。考虑梯度的因素，在本二值化神经网络中利用tanh函数的特性，将神经网络在全精度的环境下进行训练，并使得训练过程中需二值化权重不断将二值的目标趋近。



图表 1 tanh函数形式

另外关键的一点是对乘法操作的摒弃。在对于神经网络的训练过程中不可避免使用到神经元与权重之间的乘法操作，但是在硬件中数值以二进制的形式存储与运算。考虑到两者的区别，构造了二进制下异或操作与-1/1的乘法计算相联系。在全二值化的网络中，有如下对应（在权重从训练网络中调出时，-1映射为1，1映射为0）：

|  |  |
| --- | --- |
| (-1) \* (-1) = 1 | 1 ⊕ 1 = 0 |
| (-1) \* 1 = -1 | 1 ⊕ 0 = 1 |
| 1 \* 1 = 1 | 0 ⊕ 0 = 0 |

本项目手写数字识别利用的是双层的全连接层进行判别。对于网络整体而言逻辑并不复杂，数据进行神经网络后经过flatten层后通过二值化的全连接层binary\_dense1，随后跟随二值化的激活层binary\_activation以增加网络的非线性因素以及dropout随机丢弃，最后再经过第二个二值化的全连接层binary\_dense2。

综上来看，训练该全二值化的神经网络关键是对BinaryDense和BinaryActivation两个类定义进行设计。

1. BinaryDense

二值化的全连接层思路源于全精度的全连接层。输入参数分别为输入与输出的维度（神经元数量）及一个是否使用偏置的布尔参数。由于第二层全连接层直接输出结果，对于平移的需求不大，故考虑复杂程度可以省去第二层全连接层的偏置项。

训练包含可学习的参数：kk, nmk, kernel, bias（如若use\_bias = 1）。为了达到实现参数二值化的目的，kk在训练的过程中会不断增大（直至1000），而对于kernel\*kk而言其会虽kk的增大而显著变大，对其使用tanh函数后可见随着kk的不断变大tanh (kernel\*kk)会趋于-1或1，也即是趋于sign (kernel) 的大小。由此训练便可将kernel的值成为-1或1。

对第一层全连接层来说，输出out的计算为out = nmk \* matmul(inputs, tanh(kernel \* kk)) + bias。而当kk>1000后tanh(kernel\*kk)会直接改为sign(kernel)得到二值化结果。

1. BinaryActivation

为了配合BinaryDense的设计，在全连接层后的激活层中并没有使用传统的relu或sigmoid等激活函数，它们虽然引入了非线性因素提高模型表达，但是会使得隐藏层的神经元数值大小成为全精度数据，不符合项目的构想，在本激活函数的设计中还是同样利用tanh的相关特性：在BinaryActivation中仅仅只有kk一个可学习参数，其与上述的设计思路相同，利用tanh(inputs\*kk)进行训练，由于kk的不断增大，输出将趋于sign(inputs)直到kk足够大大过1000时直接改tanh为sign函数。

BinaryActivation的输入就是BinaryDense的输出，也即通过该激活层，全连接层的输出也将被二值化并以二值的形式进入第二层全连接层，大大简化了层层之间的计算量。于是，这个神经网络的输入层神经元、隐藏层神经元、全连接层权重均为二值化数值，满足设计的预期。

1. **二值化训练**

在本网络的训练中，引入wait参数以控制kk的大小。开始训练时参数kk会被初始化至10，每当一轮训练后的正确率小于最优情况正确率，则wait加1，设置中给定patience大小为3也即每当三轮训练总未有正确率的提升，kk的值会乘2（系数可调，取决于具体网络特点），直到kk不断累乘到大于1000，此时网络内需二值化的参数均已经符合要求，并终止训练。

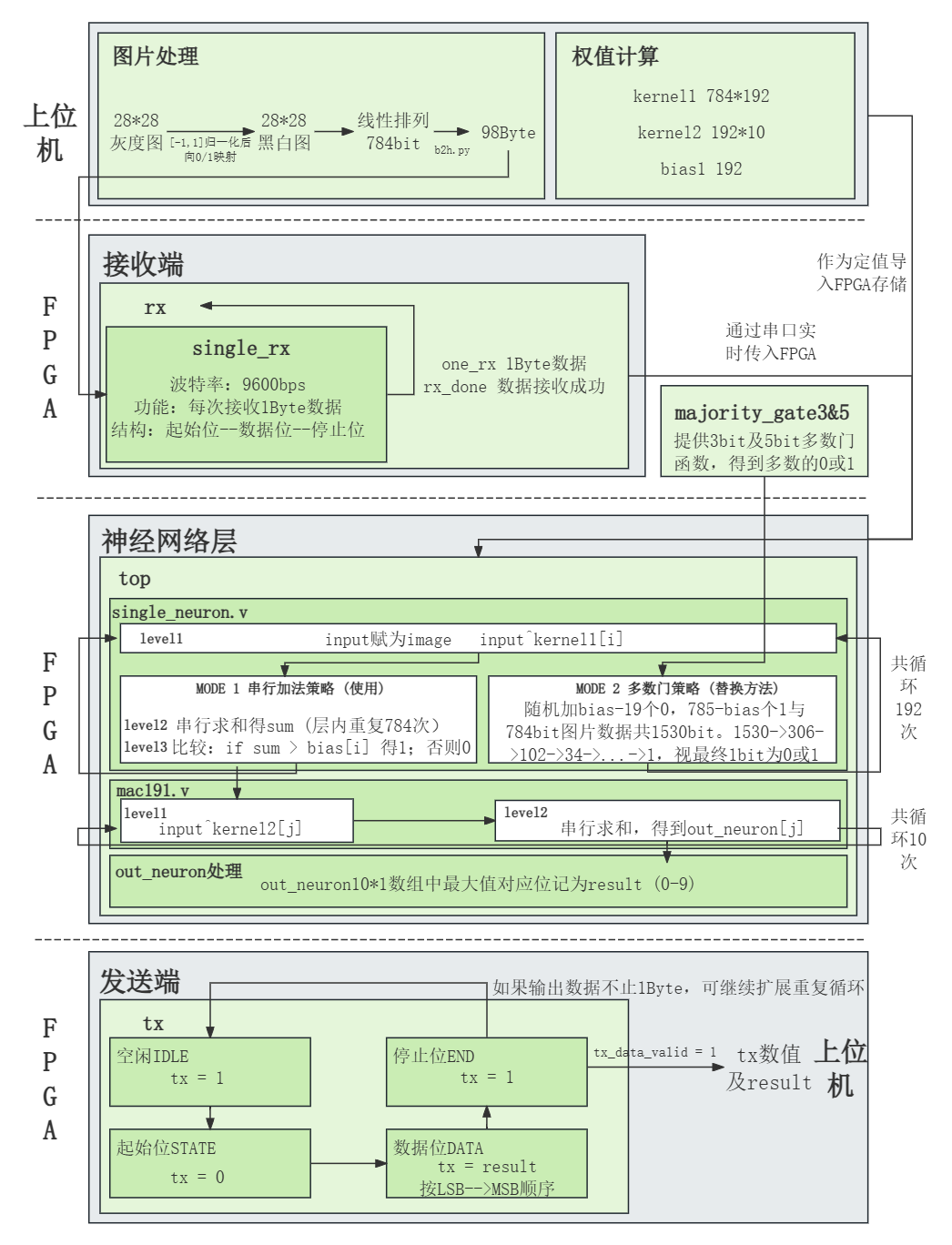
另外，由于手写数字识别项目中训练次数较少，为综合欠拟合与过拟合而实时调整学习率必要性不大（对于精度的提升并不多），故在手写数字识别中保持学习率为0.001并不作过多调整。

二值化导出方面，有如下的对应：

|  |  |  |  |
| --- | --- | --- | --- |
| 层数 | 参数 | 训练过程 | 二值化计算过程 |
| BinaryDense1 | Input (image) | / -1 / 1 | / 0 / 1 （利用sign函数） |
| Kernel1 | / -1 / 1 | / 1 / 0 （ker = ~(ker == 1)) |
| Bias | 全精度浮点数 | 整型数 （floor((784-bias)/2) |
| 计算过程 | | matmul(inputs,ker)+bias | sum(inputs^ker) > bias ?1 :0 |
| 隐藏层 | neuron | / -1 / 1 | / 0 / 1 |
| BinaryDense2 | Input (neuron) | / -1 / 1 | / 0 / 1 |
| Kernel2 | / -1 / 1 | / 1 / 0 （ker = ~(ker == 1)) |
| 计算过程 | matmul(inputs,ker) | sum(inputs^ker) |
| 预测结果 | | 最大值对应序号 | 最大值对应序号 |

## 二值化网络移植FPGA开发板

在将二值化的神经网络移植到开发板的同时，需要完成数据传入传出模块的设计，FPGA模块的设计架构如下图所示：



图表 2手写数字识别项目FPGA实现设计层次

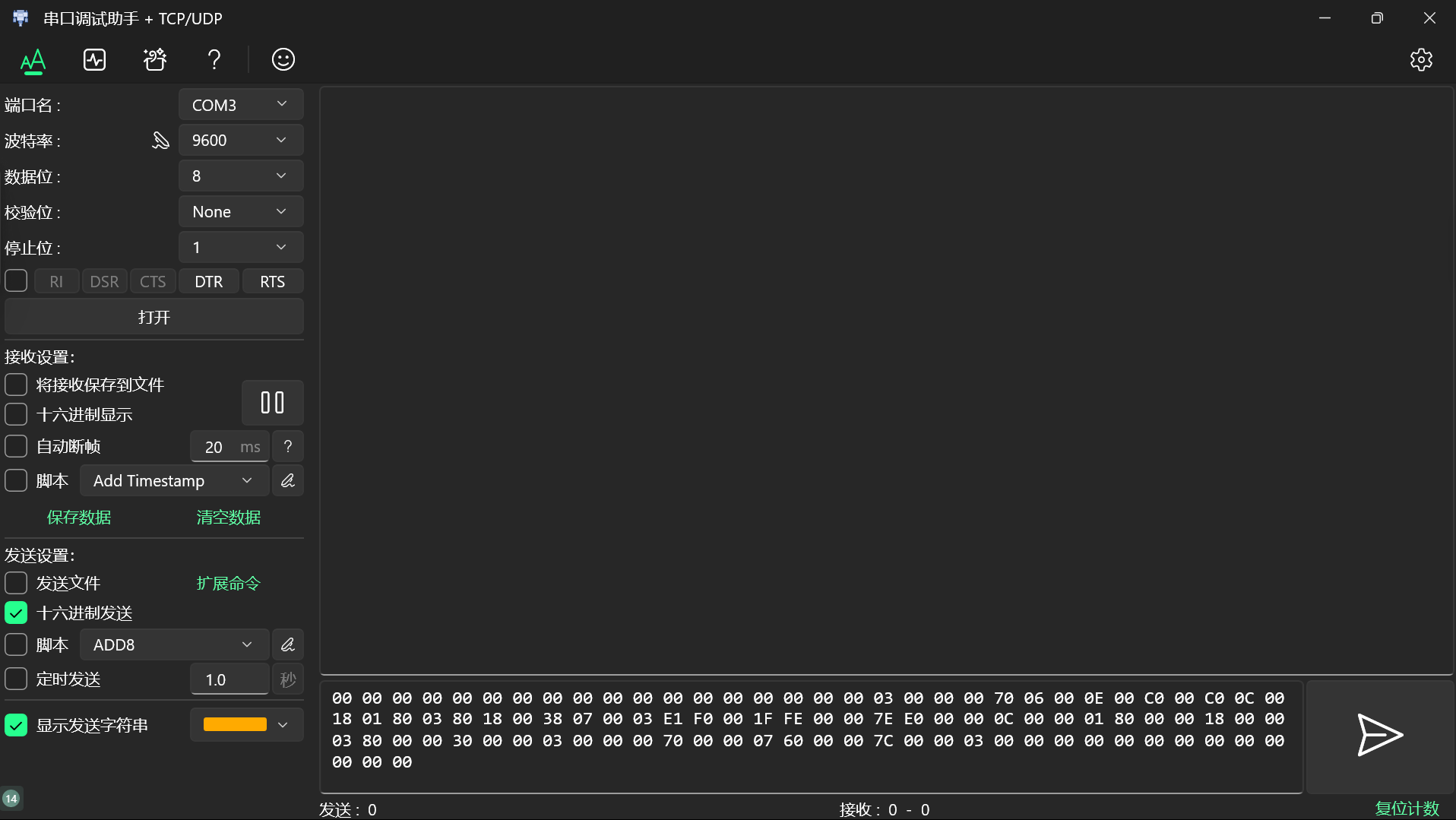
1. **数据输入**

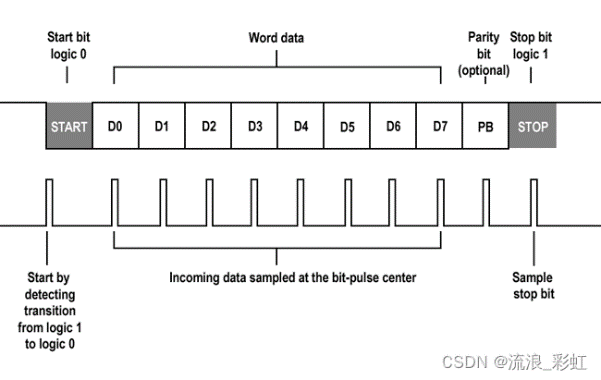
在手写数字识别工程中，需要传入两部分数据。一部分是预训练好的神经网络权重数据，包括了两级全连接层的weight及bias；另一部分是实时的图片信息。在本项目中，输入的图片像素为28\*28的灰度图，为方便数据的输入，在输入前先进行了图片数据的二值化（归一化后再经过sign函数得到），在此基础上将二维数据展平到一维上，得到大小为784的一维张量。

上述两类数据通过不同的方式输入并存储于FPGA中：

神经网络内的权重数据（ker1.txt：第一层全连接层的权重大小，bias.txt：第一层全连接层的偏置大小，ker2.txt：第二层全连接层的权重大小）通过readmemb将二进制数据存在模拟器中的虚拟存储（vivado等仿真工具会优化后决定将其存入FPGA的BRAM、DRAM或其它存储单元内），方便硬件直接调用数据。

在设计中图片数据由串口通过UART通信协议输入，该通信协议为串行异步全双工协议。在FPGA端构造了一个数据接收端rx（内含子模块single\_rx），在实验的电脑端则由工具串口调试助手实现数据的输入。本例中UART数据传输协议配置为9600bps的波特率，一个传输单元由1bit的高电平的起始位、8bit的数据位和1bit的停止位组成。



图表 3串口调试助手页面

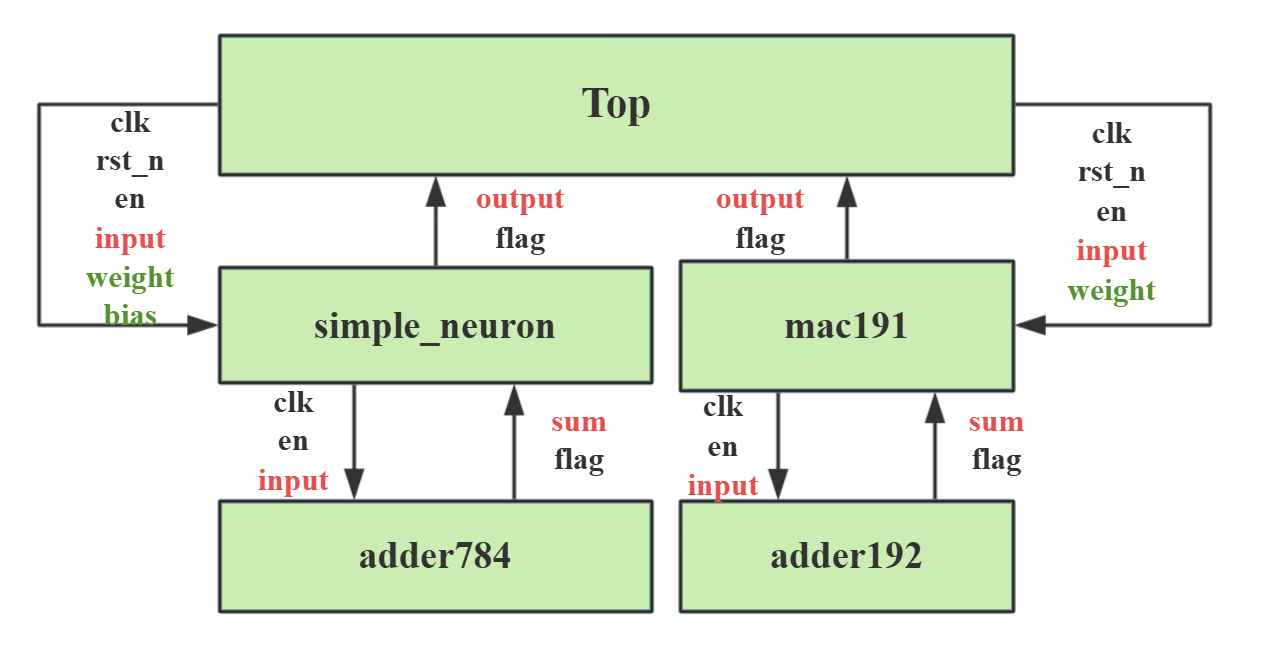
图表 4 UART通信协议

每份图片包含784bit (98Byte)数据，在本UART配置下共需98个数据帧发送完成，也即理想状态下仅98/960 ~= 0.10s 即可完成一份图片的输入。单个Byte的数据接收由single\_rx模块完成。由于FPGA系统时钟周期（20ns）远远小于设置的波特率下1bit数据的传输时间（104166ns），可以保证在接收端精准完成数据接收，在数据传输伊始先通过三次的打拍实现时钟的对齐，在此基础上后续的数据位等的传输在计数器基础上实现，当系统时钟周期在每个比特数据进入后的第2604个系统时钟周期（即该比特数据传输正中间的时刻）接收比特的电平信息并寄存在数组中。

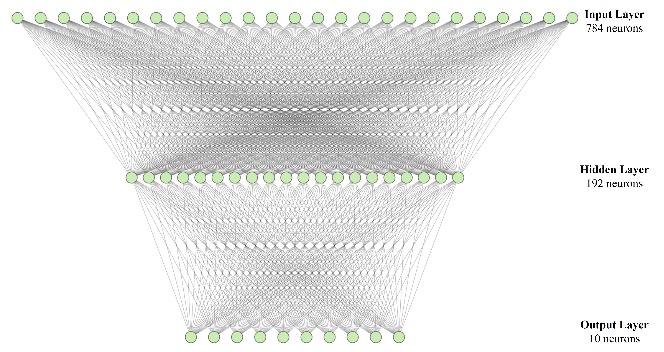
当计数器来到停止位时，rx\_done标记信号从低电平变为高电平，并等待下一个字节数据的输入。相同的原理下，在rx模块同样也设置针对字节的计数器，当其计入98时，单个图片数据即以rx\_data完整存储于RAM中。

1. **网络搭建**

此项目选取两层的全连接层作为训练的网络，可以简便实现全网络的二值化实现。在本项目中，硬件设计层面为三级的架构：顶层控制模块为top，其中例化simple\_neuron及mac191两个二级模块，分别构造第一、二层全连接层，这两个模块下分别包含了adder784和adder192这两个底层模块。



图表 5 全连接层硬件设计



图表 6 本项目全连接层神经网络图示

**第一级模块**：Top级主要起到关键的控制作用，在处理单个图片时经过不同的层级有时间上的先后关系，需要在此通过串行形式将两个全连接层相连。由于两层神经元数目较多，故采用各层输出端单个神经元为一模块（即simple\_neuron和mac191），以串行的形式输出各层的不同神经元。一方面串行的构造方式可以充分利用到模块的相同结构而避免过多占用硬件资源，另一方面也提高了结构的可复用性。另外，本例中FPGA硬件的系统时钟周期为20ns，综合后对时间进行测算可知神经网络计算部分时间占比很小，串行对时间的影响可以忽略。

对于两层全连接层共构造两个计数器完成对simple\_neuron和mac191的控制，前者当计数器192时，说明隐藏层192个神经元数据皆已计算获得；后者计数器到10时，得到输出层10个数据，其大小表示了图片数字为 0~9共10个数字的可能性。

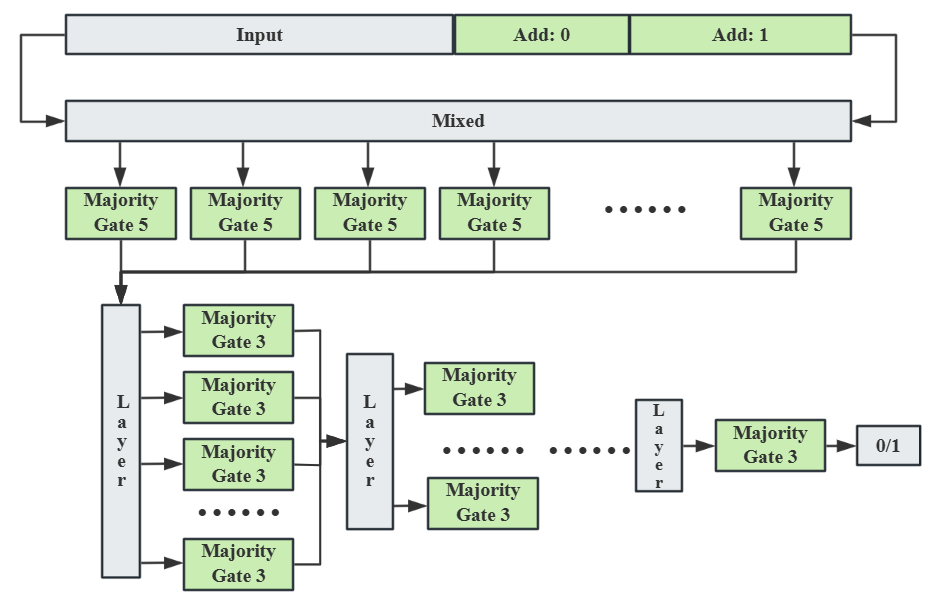
**第二级模块**：simple\_neuron从Top级中获取全连接层的权重weight、偏置bias以及上一层输入层的784个二值神经元数据。计算一个神经元需要所有的输入层的神经元与weight作异或操作后再对784个异或结果（0或1）累加，累加后与对应的bias进行比较大小再得到二值化的隐藏层神经元。

第二层全连接层作为最后一层，没有使用偏置而是直接得到10个整型的数据，再通过比较大小得到最大值对应的序号。其余的构造与第一层全连接层一致。

**第三级模块**：在硬件处理中对于并行工作有一定的资源限制，并行累加在具体实现上有困难，于是在第二级模块的基础上例化出累加器模块作为其子模块独立研究。在并行层面上构造对异或结果（第一层全连接层784个二值数据，第二层全连接层192个二值数据）的累加器即是一个大型的for循环结构，硬件上这样迭代多次而相同的电路结构会被重复多次，每次迭代都占用独立的LUT等资源。为解决这样的资源问题，有效的方式是对循环进行拆分，即完整的循环拆解成多个小循环并使每个小循环在一个或多个时钟周期内完成。在这个神经网络中数据量不大，串行计算次数的阶数为3阶。如果只用最小的硬件资源——设置累加器为单个加法的串行，即经过784个时钟周期完成累加，则串行重复的数量级为10^5，考虑到默认时钟周期为20ns，所用时间的数量级仍为10^(-2)~ 10^(-3)s之间，几乎可以忽略。

|  |  |
| --- | --- |
| 第一阶 | 2层全连接层 |
| 第二阶 | （最大重复单元）第一层全连接层的**192**个输出元 |
| 第三阶 | （最大重复单元）第一层全连接层中**784**个二值累加 |

在处理更为复杂的网络时，为综合累加计算中的硬件资源及时间消耗，可以引入小型的并行单元，另一方面要优化结构尽量减少串行的阶数。同时也可以引入不同的算法结构如多数门策略（需要更多的输入数据）。



图表 7 多数门策略结构图

多数门策略的总体思路是把求和大小与偏置大小作比较进行改变，改为构造包含输入数据和偏置数据的0/1二值化数组并比较其中的0/1数量多少。这样的构造对于减少循环次数有很好的效果。图示以5位多数门和3位多数门为例，补充的1的数目要比补充的0的数目多出size(Input)-2\*bias，随机打乱后再经过数个多数门得到最终的一位二值数据，在数据量大的情况下会趋于准确解。

3位及5位多数门在本质上即是简单的与门及或门的二级逻辑结构，构造简单。

由于数据分布并不能实现完全的随机，在数据量较小的时候误差会体现较为明显，以784个输入数据为例，实验测得准确率稍高于90%，在这个情境下不能替代精准的累加计算。

1. **数据输出**

数据输出与输入部分类似，采用UART通信协议从FPGA将结果发送回上位机。上位机端在实验过程中依旧是采用电脑的串口调试助手实现，在FPGA端通过编写tx模块实现数据的发送功能。

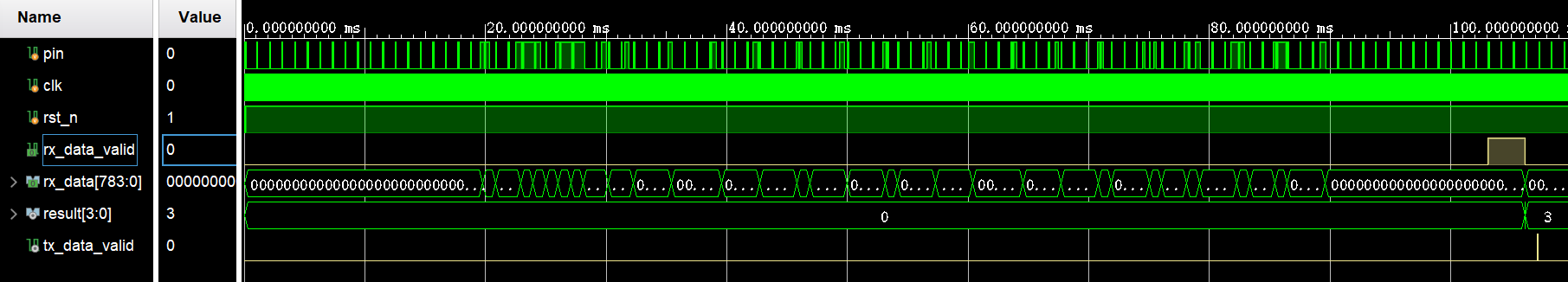
tx模块中构造一个状态机，包括四个状态：STATE\_IDLE（空闲）, STATE\_START（起始位）, STATE\_DATA（数据位）, STATE\_END（停止位）。和数据输入部分一致，此处亦采用1Byte为一数据帧，由于本例中传回数据仅为一数字，故一共只需1Byte便可实现目标。

## 测试结果

实验结果包括仿真工具及FPGA上板测试均符合预期。

这里采用vivado进行仿真测试，经过仿真后主要输出的波形如下图所示。选取的波形信号为：

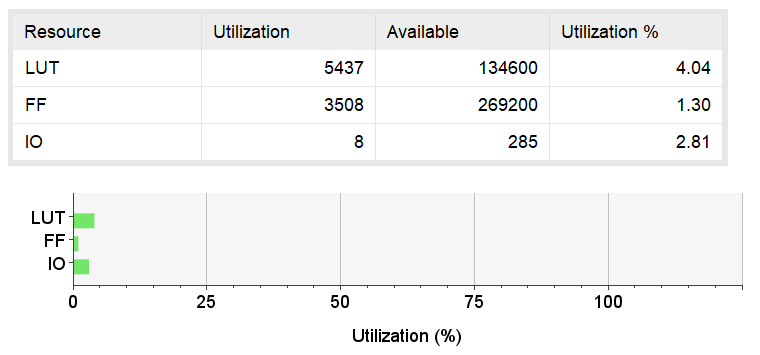
**pin**：uart协议输入信号，**clk**：系统时钟，**rst\_n**：复位信号，**rx\_data\_valid**：图片完全接收标志，**rx\_data**：图片寄存张量，**result**：预测数字，**tx\_data\_valid**：uart输出成功标志。



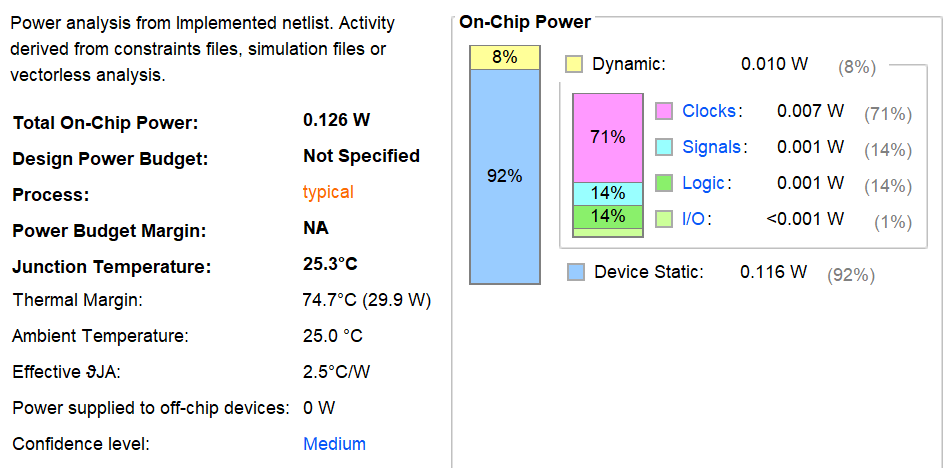
图表 8 仿真波形图

在自定义的testbench文件下仿真，电路可以经由uart输入、神经网络、uart输出得到最终结果。时间上看，项目完成的时间约为0.1s，其中绝大部分时间为导入图片数据，而神经网络判断的时间约为4ms，可见二值化网络后对预测的速度提升明显。而图片数据导入的时间的控制仅需要提高波特率即可，本例中选择的波特率为9600bps，传输速率仅为1.2KB/s，实际应用中串行的通信模式可以有很大的波特率（一般都大于115200bps），主要取决于高速通信端口（如usb接口）和接入FPGA的另一端外设的系统频率。

利用vivado工具对电路进行综合，可以看到硬件的资源配置等占用情况。



图表 9 硬件资源占用情况

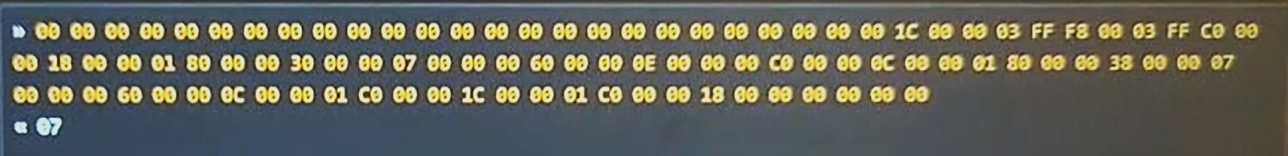


图表 10 硬件功率情况

最终生成比特流文件，将其成功烧录至FPGA板（核心板型：XC7A200T）。使用USB线将其与电脑相连，将图片数据二进制数据转换为十六进制数据并利用串口调试助手传输，配置为波特率9600bps，数据位：8，校验位：None（本项目数据传输数据量小，无设置必要），停止位：1。发送数据，在串口调试助手界面可见预测结果传回界面，为7；同时，FPGA板上LED灯也显示7（共四个LED灯从左至右为LED0(LSB)~LED3(MSB)，以二进制形式表示即0111，十进制对应7）。



图表 11 FPGA板LED显示结果



图表 12 串口调试助手数据发送及接收