UCSD 237C: Project 5 BNN

Steven Daniels

sdaniels@ucsd.edu

Student ID# A53328625

December 9, 2023

1. Introduction

This report details the optimizations performed on a HLS hardware implementation of a binary neural network (BNN) used to classify digits from the MNIST database. One optimal design was explored:

1. BNN using dataflow and AXI4-burst.

2. FFT Design

2.1. Throughput: 204 kHZ

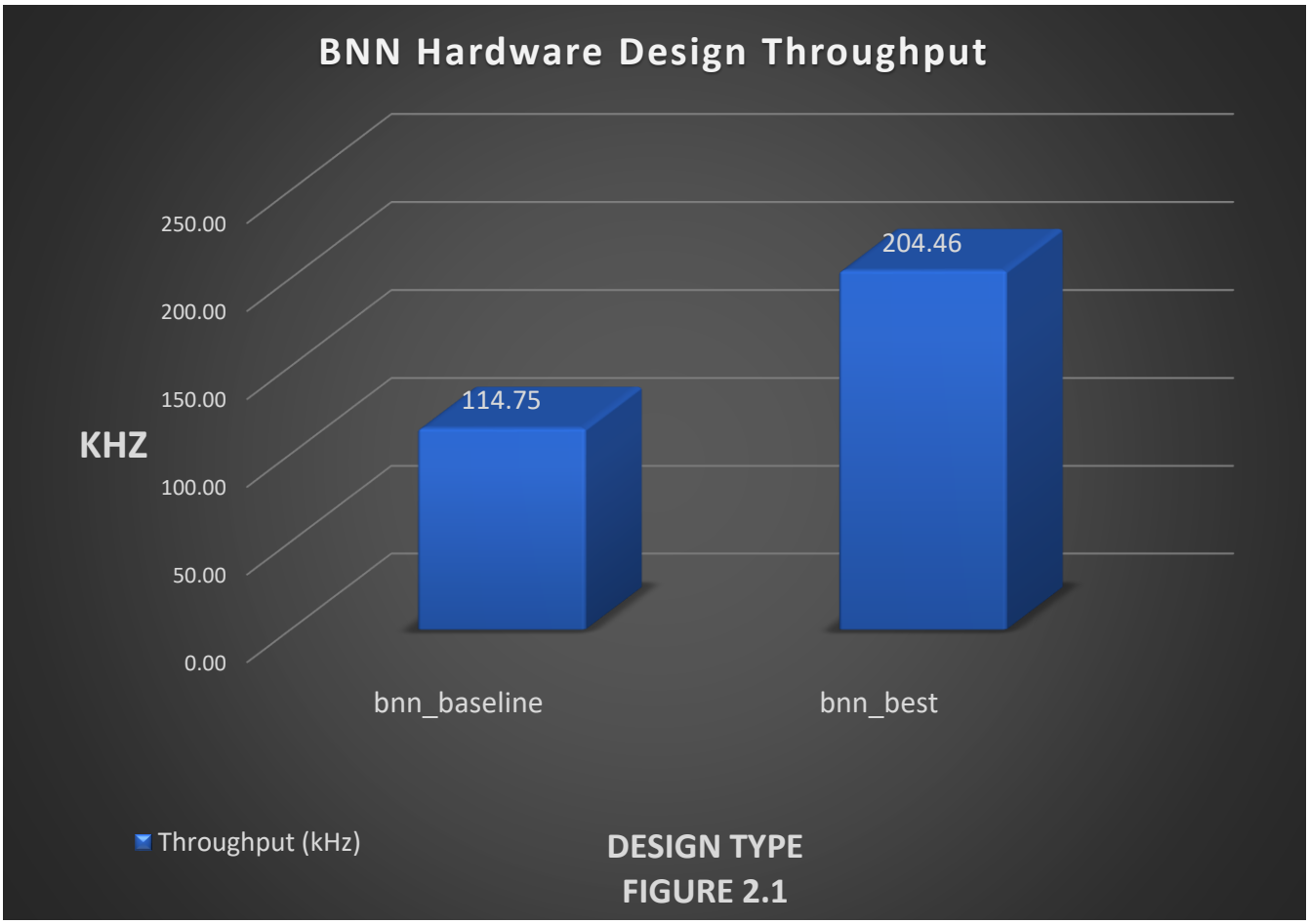

FIGURE 2.1

2.2. Implementation

```
#pragma HLS DATAFLOW
    ITYPE LAYER1_TEMP[LAYER1_SIZE];
    ITYPE LAYER2_TEMP[LAYER2_SIZE];
    ITYPE LAYER3_TEMP[LAYER3_SIZE];

    unsigned int LAYER1_OUT[LAYER1_OUT_SIZE];
    unsigned int LAYER2_OUT[LAYER2_OUT_SIZE];
    unsigned int LAYER3_OUT[LAYER3_SIZE];

    unsigned int IN_TEMP[SIZE];
    memcpy(IN_TEMP,(DTYPE*)IN,SIZE*sizeof(DTYPE));

    matrix_mult(IN_TEMP,
                w1,
                LAYER1_TEMP,
                INPUT_SIZE,
                LAYER1_SIZE);

    layer1_preprocessing(LAYER1_TEMP,
                         LAYER1_OUT,
                         784,
                         16);

    matrix_mult(LAYER1_OUT,
                w2,
                LAYER2_TEMP,
```

```
                LAYER1_SIZE,
                LAYER2_SIZE);

    layer2_preprocessing(LAYER2_TEMP,
                         LAYER2_OUT,
                         128,
                         0);

    matrix_mult(LAYER2_OUT,
                w3,
                LAYER3_TEMP,
                LAYER2_SIZE,
                LAYER3_SIZE);

    ITYPE TEMP3[LAYER3_SIZE];
    for(unsigned int i = 0; i < LAYER3_SIZE; i++)
    {
        TEMP3[i] = 2*(LAYER3_TEMP[i]) - 64;
    }

    memcpy(OUT,(DTYPE*)TEMP3,OUT_SIZE*sizeof(DTYPE));
```

2.3. Resources

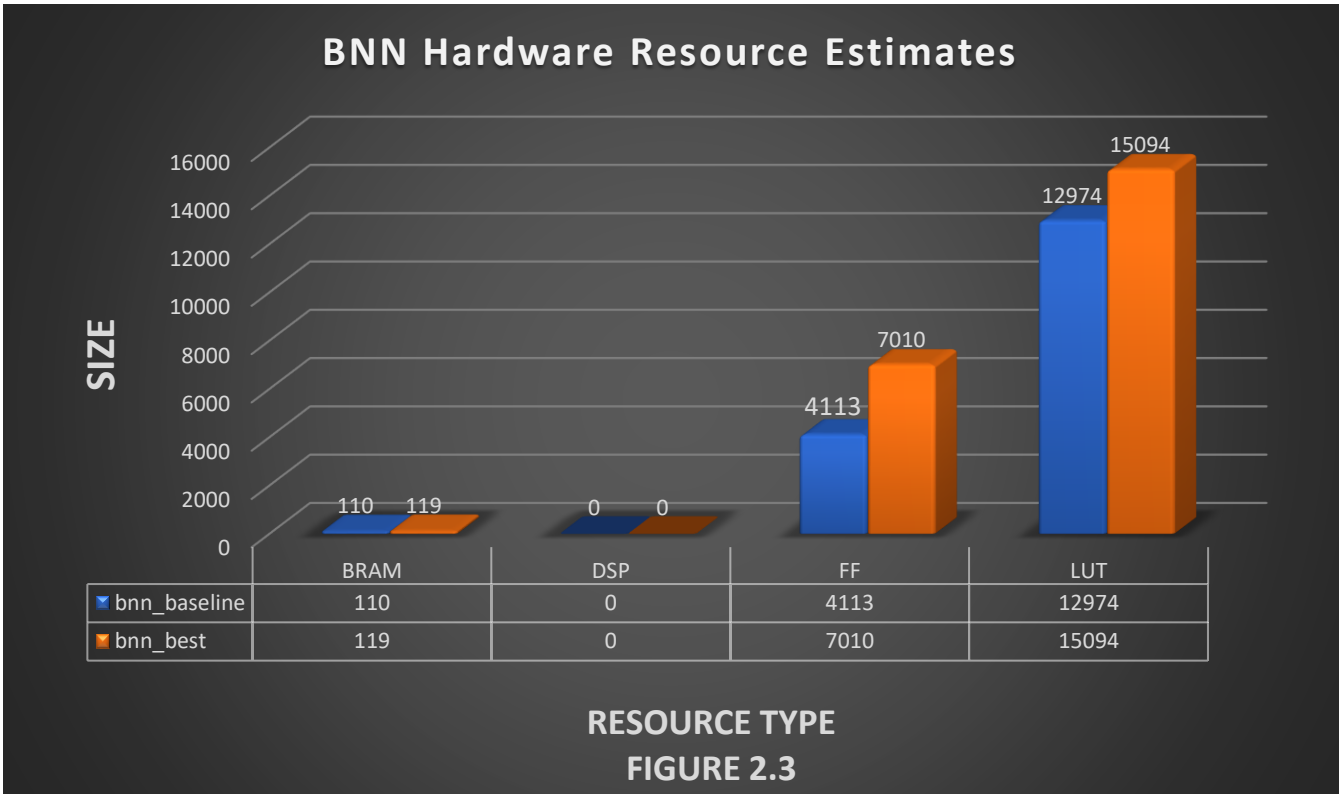

FIGURE 2.3

2.4. Optimizations

- Added dataflow to top level BNN function.

2.5. Analysis

This design provides an optimized version of a BNN. The optimized design that provided the best throughput utilized the dataflow pragma to enable task level pipelining. This was done because BNN are structured in a way that allows the next “layer” in the graph to begin it’s processing before the previous layer finishes its processing. This design provides a moderate increase in throughput over the baseline software implementation with no optimizations as seen in Figure 2.1. This design also produced a moderate amount of increase in resource usage

when compared to the baseline as seen in Figure 2.3. Overall, this design provides a faster way to perform predictions of labeled data from the MNIST database using a binary neural network over the baseline with the tradeoff being high LUT and FF usage.