FR. FPGA Requirements Report Chapter

FR.1 CNN Simulation Data

For our model without Tensor Contraction Layer (TCL) to train to 80% accuracy, the required resources are:

- Number of layers in CNN: 3.
    - 1 Convolution Layer.
    - 1 Max Pooling Layer.
    - 1 Fully Connected Layer (FCL).
- Counts of each mathematical operation during simulation:
    - 60830 Additions.
    - 60164 Additions.
- Memory used: 50MB.

For our model with Tensor Contraction Layer (TCL) to train to 80% accuracy, the required resources are:

- Number of layers in CNN: 4.
    - 1 Convolution Layer.
    - 1 Max Pooling Layer.
    - 1 Tensor Contraction Layer (TCL).
    - 1 Fully Connected Layer (FCL).
- Counts of each mathematical operation during simulation:
    - 177070 Additions.
    - 205764 Multiplications.
- Memory used: 279MB.

FR.2 FPGA Recommendations

Below are three recommended FPGAs that are able to run an image classification CNN.

While not recommended because there are no sources that have implemented a CNN on one, the MAX 10 10M50DAF484C7G FPGA is used by Oregon State University classes and may be an affordable and familiar option for OSU students if it is determined to be sufficient [9]. A quick review of its specifications shows that it seems to have significantly less resources than the FPGAs recommended.

FR.2.1 1st Recommended FPGA: Virtex 7 VC709

The Virtex 7 VC709, formerly known as the Xilinx VC709, was used by paper [1]. Specifically, the FPGA alone is called the Virtex 7 XC7VX690T-2FFG1761C. It performed well for memory efficiency and, for another paper mentioned, performed well for power efficiency compared to other FPGAs. Paper [1] was able to implement an image classification CNN, so this proves it is able to run one.

Virtex 7 VC709 general resources [4]:

- Number of Logic Cells: 693,120
- Number of DSP Slices: 3,600
- Amount of Memory: 52,920 KB
- Number of I/O Pins: 1,000

Virtex 7 VC709 resources needed to compare software vs FPGA implementation:

- Clock Frequency: 710 MHz
- Digital Clock Manager Module: None
- Number of Cores: 0

The Digital Clock Manager Module (DCM) is something only available in specific Xilinx FPGAs [10], this is not one of those Xilinx FPGAs.

The Number of Cores were not able to be found anywhere in the datasheets or elsewhere. It may have a Zynq 7000 SoC integrated on to the same die, but it is unclear if it actually does. AMD mentions that “Zynq 7000 devices are equipped with dual-core ARM Cortex-A9 processors…”

FR.2.2 2nd Recommended FPGA: Altera Stratix V GSD8

The Altera Stratix V GSD8 was used by paper [2] and mentioned in paper [1] to have been used by paper [5]. Paper [2] used eight and paper [5] used two Altera Stratix V-GSD8. By scaling the models down, it should be possible to follow their implementation, or also possible to implement a different, smaller model. This model was trained on the MNIST dataset for handwritten digits, which has a training set of 60000 examples and a test set of 10000 examples.

Altera Stratix V GSD8 general resources [6]:

- Number of Logic Cells: 695,000
- Number of DSP Slices: 1,963
- Amount of Memory: 50,000 KB
- Number of I/O Pins: 696

Altera Stratix V GSD8 resources needed to compare software vs FPGA implementation:

- Clock Frequency: 800 MHz
- Digital Clock Manager Module: None
- Number of Cores: 0

The Digital Clock Manager Module (DCM) is something only available in Xilinx FPGAs [10], this is not a Xilinx FPGA.

The Altera Stratix V FPGAs do not appear to have cores with them. However, the Stratix 10 FPGAs do appear to have a Quad-core ARM Cortex*–A53 MPCore processor [12].

FR.2.3 3rd Recommended FPGA: Altera Cyclone V 5CEA9

There are other Altera Cyclone V options, but the 5CEA9 appears to be the best of the general purpose options. The Altera Cyclone V was suggested by Intel [3]. Intel also has a demo [7] that was posted a month prior [7], but the FPGA used was only an engineering sample, though it is presumably similar. It is difficult to find consistent pricing for all FPGAs, but this one appears to be cheaper compared to the others, which may be an important consideration.

Altera Cyclone V 5CEA9 general resources [8]:

- Number of Logic Cells: 301,000
- Number of DSP Slices: 342
- Amount of Memory: 13,917 KB
- Number of I/O Pins: 480

Altera Cyclone V 5CEA9 resources needed to compare software vs FPGA implementation:

- Clock Frequency: 12.5 MHz
- Digital Clock Manager Module: None
- Number of Cores: 2

Clock frequency depends on clock source and can exceed 12.5 MHz, but that is the frequency of the internal oscillator. See datasheet page 77 [11].

The Digital Clock Manager Module (DCM) is something only available in Xilinx FPGAs [10], this is not a Xilinx FPGA.

The Cyclone V SoC FPGA devices have a dual-core ARM* Cortex-A9 MPCore processor.

FR.3 FPGA Required Resources

FPGA required resources:

- Number of Logic Cells:
    - 301,000 to 650,000
- Number of DSP Slices:
    - 342 to 3,600
- Amount of Memory:
    - 14 to 53 MB
- Number of I/O Pins:
    - 480 to 1,000
- Types of Peripherals:
    - PCIe x8 Interface: Common for interfacing with CPUs.
    - External DDR3 Memory Interface: Useful for larger models or deep learning, does not seem necessary for all CNNs. Other DDRs possibly could be used.

There are many ways to implement CNNs, creating our own is beyond the scope of a senior design project and more on the level of Masters and PhD research. We discussed with our course instructor that this project being research oriented would be more productive, so reviewing others' work is more appropriate for this project. Therefore, the FPGA requirements are based on different implementations and FPGAs, thus it is difficult to provide an exact number for each requirement and so a range is given based on the recommended FPGAs from the reviewed papers.

To get a better estimation of the resources required for a different CNN than the ones reviewed, paper [2] provides a discussion and equations of how to estimate resource usage. The paper focuses on estimating the following resources: LUT, FF, Block-RAM (BRAM), and DSP. A summary will be provided below, but it is encouraged that the original paper [2] be reviewed.

The paper [2] states that if one were to use the stochastic gradient descent (SGD) optimization method for a CNN, then the input data of a layer in CNN can be stored as a 4-dimension tensor, where the meaning of 4 dimensions are:

- N = number of feature images in 1 training sample
- R = number of rows in one image
- C = number of columns in one image
- B_s = the size of a minibatch (subset of training data)

For example, if one were to use the MNIST database for training, the images would be 28x28 pixels, so R = C = 28. Paper [2] used 150 minibatches, thus B_s = 400.

Below are the final equations for estimating resource usage for modules. Multiple modules would be needed, so these values should be multiplied by the number of each module in the CNN.

The total resource usage of a convolutional module can be estimated as:

$$\begin{cases} LUT &= N_k \times K^2 \times LUT_{ma} + LUT_{io} \\ FF &= N_k \times K^2 \times FF_{ma} + FF_{io} \\ BRAM &= N_k \times K^2 \times (BRAM_{ma} + 1) \\ & \quad + BRAM_{io} + BRAM_{buff} \\ DSP &= N_k \times K^2 \times DSP_{ma} + DSP_{io} \end{cases} \tag{12}$$

The total resource usage of a backward convolutional module can be estimated as:

$$
\begin{cases}
Perf & = & 2 \times N_k \times F \times K^2 \\
BW_{pcie} & = & 2 \times DataLen \times F \\
BW_{dram} & = & (2+S) \times N_k \times DataLen \times F \\
LUT & = & N_k \times K^2 \times LUT_{ma} + LUT_{io} \\
FF & = & N_k \times K^2 \times FF_{ma} + FF_{io} \\
BRAM & = & N_k \times K^2 \times (BRAM_{ma} + 1) \\
 & & +2 \times BRAM_{buff} + BRAM_{io} \\
DSP_{conv} & = & \times N_k \times K^2 \times DSP_{ma} + DSP_{io}
\end{cases}
\tag{13}
$$

Where other variables and subscripts besides the common FPGA resources LUT, FF, BRAM, and DSP mean:

- Perf = Theoretical Performance in FLOPS
- N_k = Number of Kernels
- F = Frequency
- K^2 = Filter Size
- ma = Multiplication Addition Operation
- io = Input Output
- buff = Input Buffer

FR.4 FPGA Training Time and Accuracy

To fully train an FPGA using the implementation described in paper [2], it would take 1 hour, 11 minutes, and 12 seconds to get an accuracy of 62.06%.

Training time and accuracy depends on the type of implementation. We will focus on paper [2]'s implementation of LeNet-5 because it provides the most information and also uses the MNIST handwritten digits dataset, just as we did with our C++ simulations.

The MNIST dataset has a training set of 60000 examples and a test set of 10000 samples. They divided the training set into 150 minibatches, each 400 in size, and ran the training process for 200 iterations. The table below shows the training time per layer for 1 iteration.

Table FR.1-1: Average execution time (in seconds) per iteration of LeNet-5 on F-CNN, CPU, and GPU.

Layers	N_k		F-CNN		Caffe (CPU)		Caffe (GPU)	
	Forward	Backward	Forward	Backward	Forward	Backward	Forward	Backward
L1(Conv)	12	12	0.59	1.21	6.84	7.70	2.55	7.19
L2(Pool)	24	24	0.53	0.57	1.85	0	0.04	0
L3(Conv)	12	12	4.67	10.32	36.37	33.11	5.33	3.38
L4 (Pool)	24	24	0.17	0.18	1.05	0	0.01	0
L5 (MLP)	24	24	0.92	1.82	2.04	2.02	0.05	0.4
L6 (MLP)	24	24	0.18	0.20	0.08	0.07	0.03	0.02
Total (Speedup)			21.4(4.3×)		91.1 (1×)		18.7 (4.9×)	
Power (Energy Efficiency)			27.3Watt (7.5×)				235Watt (1×)	

For an FPGA (F-CNN column), the sum of the forward propagations is 7.06 seconds and the backward propagations is 14.30 seconds. The total sum for 1 iteration is 21.36 seconds. Again, the training process had 200 iterations, which would be a training time of 4273 seconds or 1 hour, 11 minutes, and 12 seconds.

Paper [2] estimated that their FPGA CNN should be able to reach an overall accuracy of 62.06, as seen in the table below.

Table FR.1-2: Analysis and implementation results of AlexNet (32-bit float, Bs=96, F=150MHz, time in ms, performance in GFLOPS).

Layers	Analysis			FPGA Implementation			CPU Implementation (Caffe)		FPGA2015 [4]
	N_k	Performance	Resource bound	N_k	Time	Performance	Time	Performance	Performance
L1	2.4	81.76	BRAM	3	113.6	89.04	1082.1	9.34	27.50
L2	13.9	79.08	BRAM	12	321.9	66.79	2267.3	9.48	83.79
L3	21.3	43.36	BW_{DRAM}	36	262.8	54.62	1801.1	7.97	78.81
L4	21.3	43.36	BW_{DRAM}	36	198.6	54.20	1415.1	7.61	77.94
L5	21.3	43.36	BW_{DRAM}	36	133.0	53.98	986.5	7.27	77.61
Overall					1029.9	62.06	7552.1	8.46	61.62

So, to fully train an FPGA using the implementation described, it would take 1 hour, 11 minutes, and 12 seconds to get an accuracy of 62.06%. To put this into some perspective, training our C++ model on a typical laptop CPU only needed 1/12 of the same MNIST dataset (5000 images) and about a minute to reach 80% accuracy. Paper [2] does acknowledge that FPGAs have limitations for training CNNs, saying that “Specifically, the computation resources on modern FPGAs are too limited to implement the whole training process…” The purpose of their paper was to address such problems and present an FPGA-based CNN training framework. However, FPGAs may still be limited compared to a CPU’s ability to train CNNs, so that could explain the difference in time and accuracy between the FPGA CNN and our C++ CPU CNN.

FR.5 References and File Links

FR.5.1 References

[1] Huang, Chao, et al. “A layer-based structured design of CNN on FPGA.” 2017 IEEE 12th International Conference on ASIC (ASICON), 2017 October 28, https://doi.org/10.1109/ASICON.2017.8252656. Accessed 2024 February 05.

[2] Zhao, Wenlai, et al. “F-CNN: An FPGA-based framework for training convolutional Neural Networks.” 2016 IEEE 27th International Conference on Application-Specific Systems, Architectures and Processors (ASAP), 2016 July 8, https://doi.org/10.1109/asap.2016.7760779. Accessed 2024 February 05.

[3] Intel FPGA. “Machine Learning on FPGAs: Neural Networks.” YouTube, 2016 July 14, https://www.youtube.com/watch?v=3iCifD8gZ0Q. Accessed 2024 January 31.

[4] “AMD Virtex 7 FPGA VC709 Connectivity Kit.” AMD, https://www.xilinx.com/products/boards-and-kits/dk-v7-vc709-g.html. Accessed 2024 February 06.

[5] Suda, Naveen, et al. “Throughput-Optimized OpenCL-based FPGA Accelerator for Large-Scale Convolutional Neural Networks.” ACM/SIGDA International Symposium on FPGAs, 2016 February 16, https://dl.acm.org/doi/10.1145/2847263.2847276. Accessed 2024 February 06.

[6] “Altera Product Catalog.” Altera, https://www.intel.com/content/www/us/en/content-details/714194/stratix-v-fpga-family-overview-product-table.html. Accessed 2024 February 06.

[7] Intel FPGA. “Convolution Neural Network CNN Implementation on Altera FPGA using OpenCL.” YouTube, 2016 June 20, https://www.youtube.com/watch?v=78Qd5t-Mn0s. Accessed 2024 January 31.

[8] “Cyclone® V FPGA and SoC FPGA.” Intel, https://www.intel.com/content/www/us/en/products/details/fpga/cyclone/v/products.html. Accessed 2024 February 06.

[9] "DE10-Lite Board.”, Terasic, https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=218&No=1021&PartNo=2#contents. Accessed 2024 February 06.

[10] “Digital Clock Manager (DCM) Module.” AMD, https://www.xilinx.com/products/intellectual-property/dcm_module.html. Accessed 2024 February 06.

[11] “Cyclone V Device Datasheet.” Intel, https://www.intel.com/content/www/us/en/docs/programmable/683801/current/cyclone-v-device-datasheet.html. Accessed 2024 February 06.

[12] “Intel® Stratix® 10 FPGA and SoC FPGA.” Intel, https://www.intel.com/content/www/us/en/products/details/fpga/stratix/10.html. Accessed 2024 February 06.