Answer to Reviewers

We sincerely thank all the reviewers and editors for providing us so many valuable suggestions to improve the quality of this paper. All the review comments are addressed and answered. Please see the answers below and the corresponding contents in the paper.

The newly added contents and the major revised contents are marked in blue in the paper and in this letter.

The major improvements are listed as follows:

Reviewer 1

The goal of this paper is to provide a survey of neural networks built using FPGAs. The networks are categorized according to several approaches and then some comparison is attempted.

The paper does cover the many techniques that have been applied to FPGA-based designs, but it needs more and better organization. The usual flow is to mention an approach and then list various papers that have applied that technique. More analysis could be provided. Based on the listed works, did any trend appear? Did anyone do it better? If so, why? I realize this may often be hard, but that is the value that you will contribute with this paper.

I do not recommend publication of this paper until the issues described here have been addressed. I think significant work is required.

There are numerous spelling errors that could have easily been corrected with a spell checker. There are many grammatical errors that need to be addressed.

Answer:

Section 2

1. The paper begins with a very brief overview of neural networks and only tries to explain several types of layers. I believe you are just trying to show that there are several types of layers that need to be computed, but you also need to give some idea about the overall system, especially since that is important for later discussions. Figures would be helpful. Consider that many people new to the area would want to start with a survey paper. They need a clear explanation of the structure. As currently written, this section will make sense to someone that has worked in the area, but such a reader does not need this introduction. Someone new to area will not be able to

understand the bigger picture and challenges.

Answer: Thanks for your advice. According to this comment and the following two comments, we rewrite this section. Now, section 2.1 introduces NN and section 2.2 introduces FPGA based system.

For the NN part, we put the equations into the new Figure 1(b) to make them easier to be understood. We shorten the introduction to the layers like ReLU and pooling to make this section more focused on the things related to accelerator design. We also add some quantitive results of NN models to show that FC layers and CONV layers are the bottleneck that should be focused on.

For the FPGA part, we introduce a typical structure of an accelerator and give a comparison between the requirement of NN models and avalable resource on an FPGA chip.

2. You have not defined what ”weights” are when you first mention them. ”The weights of this layer...”

Answer: Thanks for your comments. We add the definition of the weights and activations of each NN layer in this part now.

We refer the parameter of each layer as weights and the input/output of each layer as activations through this paper.

3. You should briefly discuss the difference between inference and training. You also need to discuss accuracy. These are important concepts. Your work focuses on using FPGAs for inference. You don’t seem to consider accuracy in your overall comparisons.

Answer: Thanks to your comments. We now clarifies this point at the beginning of section 2.1.

In this paper, we only focus on the inference of NN, which means using a trained model to predict or classify new data. The training process of NN is not discussed in this paper.

Section 3

4. You begin the section stating that accuracy, throughput, efficiency and flexibility are important. You need to more carefully define what you mean for each of those properties. For example, you give an equation for throughput, but have not defined the terms in the equation. How are each of the terms measured? There is no definition of flexibility.

Answer: Thanks for your comment. We think that we should focus on speed and energy efficiency first and consider the model accuracy as constraint to the design. A discussion on how to improve the accuracy of a NN model is not the task of this paper. For the flexibility, the standard is hard to define. We will discuss it separately from the aspect of design automation in section 7.

We add the definitions of the terms in the equitions in this secction.

Peak performance, measured in operations (multiplication or addition) per second, is achieved when all the computation units work every clock cycle. Utilization denotes the average ratio of working cycles of the computation units. The workload measures the number of operations in the target neural network.

5. You make the following statement, "Different network structures like the ones in [17, 22, 46] surely affect model accuracy, but is out of the discussion of this paper." Why? I don't think you have specifically defined the focus of this paper and you need to say more here why these papers are not relevant to this paper.

Answer: Thanks for your question, The discussion on this part is moved to the beginning of section 4. Changing the structure of NN models, from AlexNet to VGG and ResNet or MobileNet hardly affect the hardware design, because the basic opertions in the network remains. We want to focus on the techniques that specially benefits customized hardware.

A larger NN model usually results in a higher model accuracy. This means it is possible to tradeoff between the model accuracy and the hardware speed or energy cost. Neural network researchers are designing more effcient network models from AlexNet [22] to ResNet [16], SqueezeNet [19] and MobileNet [18]. The main differences between these networks are the size of and the connections between each layer. The basic operations are the same and hardly affect the hardware design. For this reason, we will not focus on these techniques in this paper.

6. "model compression methods" has not been defined when you first use the term.

Answer: Thanks for your comment. We remove this phrase in section 3 and add a brief description in the beginning of section 4.

Other methods try to achieve the tradeoff by compressing existing NN models. They try to reduce the number of weights or reduce the number of bits used for each activation or weight, which help reudce the computation and storage complexity. Corresponding hardware designs can benefit from these NN model compression methods. In this section, we investigate these hardware oriented network model compression methods.

Section 4

7. It's not clear to me why this section includes "Software Design" in its title. These techniques are also relevant to any hardware design.

Answer: Thanks for your question. The title of this section is changed to hardware oriented model compression.

8. Section 4.1.3 - What are BW_weight and BW_neurons? I'm guessing BW means "bit width"? Explain how the comparison is being made. Where is the data coming from?

Answer: Thanks for your question. We add a detailed explanation of the configuration in the caption of Figure 3 and some more description on the data.

Fig. 3. Comparison between different quantization methods from [11,14,23,40,65,66]. The quantizationconfiguration is expressed as (weight bit-width)(activation bit-width). The "(FT)" denotes that the networkis fine-tuned after a linear quantization.

We compare some typical quantization methods from [11,14,23,40,65,66] in Figure 3. All the quantization results are tested on ImageNet data set and the absolute accuracy loss compared with corresponding baseline floating point models is recorded.

9. "... we see that..." How do we see this?

Answer: Thanks for your question. The result of linear quantization shows it clearly that 8-bit is a bound to keep less than 1% accuracy loss. For non-linear quantization, current results are good with 2-bit weights. Further reduce the bit-width to 1-bit directly is equally a linear quantization. We have not seen more results on non-linear quantization with less bits for activations. So we do not give more comments on non-linear quantization here.

For linear quantization, 8-bit is a clear bound to ensure negligible accuracy loss. With 6 or less bits, using fine-tune or even training each weight from the beginning, will cause obvious accuracy degradation. If we require that 1% accuracy loss is within the acceptable range, linear quantization with at least 8×8 configuration and the listed non-linear quantization are available. We will further discuss the performance gain of quantization in section 5.

10. Section 4.2 - What is the "lasso object function"?

Answer: Thanks for your question. The lasso object function refers to a L1 normalization to the weights during training to directly get a sparse model. The phrase "lasso object function" is wrong and we change it to "lasso method" now.

Section 5

11. Figure 2 - Explain more what you are doing here. What is the point of this comparison? Why not do comparisons with and without DSPs? It looks like you have synthesized multipliers and adders in different ways to get their resource usage. You need to provide more details on how you did this. There's not enough information for me to determine whether the comparisons are fair. How did you describe the functional units? Did you just use A + B in Verilog with appropriate signal bit widths and types? For

floating point, did you instantiate the cores from the library? What if you were to use an Altera part with the hardened FP in the DSP?

Answer: Thanks for your questions. The experiment here is to show how the FPGA accelerator can benefit from quantization. We agree that simply using the logic resource consumption on FPGA is not enough. So we add more results on the multiply-and-add function with DSP units for both Xilinx and Altera FPGA. Detailed experiment setup is also included.

The size of computation units of different bit-widths is compared in Table 1. Three kinds of implementations are tested: separate multiplier and adder with only logic resource on Xilinx FPGA, multiply-add function with DSP units on Xilinx FPGA, and multiply-add function with DSP units on Altera FPGA The resouce consumption is the synthesis result by Vivado 2018.1 targeting Xilinx XCKU060 FPGA and Quartus Prime 16.0 targeting Altera Arria 10 GX1150 FPGA. The pure logic modules and the floating point multiply and add modules are generated with IP core. The fixed point multiply and add modules are implemented with $A*B+C$ in verilog and automatically mapped to DSP by Vivado/Quartus.

We first give an overview of the size of the computation units by logic-only implementations. By compressing the weights and activations from 32-bit floating point number to 8-bit fixed point number, the multiplier and the adder are scaled down to about 1/10 and 1/50 respectively. Using 4-bit or smaller operations can bring further advantage but also incur significant accuracy loss as introduced in section 4.1.

Recent FPGAs consist of large number of DSP units, each of which implements hard multiplier, pre-adder and accumulator core. The basic pattern of NN computation, multiplication and sum, also fits into this design. So we also test the multiply and add function implemented with DSP units. Because of the different DSP architectures, we test on both Xilinx and Altera platforms. Compared with the 32-bit floating point function, fixed point functions with narrow bit-width still shows advantage. But for Altera FPGA, this advantage is not obvious as where the DSP units natively supports floating point operations.

Tab. 1: FPGA resource consumption comparison for multiplier and adder with different types of data.

	Xilinx Logic				Xilinx DSP			Altera DSP	
	multiplier		adder		multiply & add			multiply & add	
	LUT	FF	LUT	FF	LUT	FF	DSP	ALM	DSP
fp32	708	858	430	749	800	1284	2	1	1
fp16	221	303	211	337	451	686	1	213	1
fixed32	1112	1143	32	32	111	64	4	64	3
fixed16	289	301	16	16	0	0	1	0	1
fixed8	75	80	8	8	0	0	1	0	1
fixed4	17	20	4	4	0	0	1	0	1

12. In section 5.1.1, "Operations with 32-bit fixed point data consumes similar resource as 32-bit floating point operations." This does not make sense to me. Needs more explanation.

Answer: Thanks for your question. The number of multiplication and addition for a NN model is approximately the same, so a resonable estimation of the hardware is 1:1 multiplier and adder. Adding the resource for 1 multiplier and 1 adder, fixed32 operations is similar to fp32. We do not claim this point in the paper now.

Section 6

13. The challenge for this paper is that there are so many different FPGA platforms used combined with many different networks that have been implemented. It becomes very hard to make any comparisons. While it is good that you are trying to gain some overall understanding based on the results of your survey, what you have is not well-justified and insufficient. I think you should begin by first discussing what are important trends that you want to observe and then explain how you will gather the data and show the results. You have chosen to look at energy efficiency, but what about accuracy? What features affect that? For that matter, your energy efficiency comparison has no consideration for accuracy. In the extreme case I could build a circuit that has 1% accuracy and is very energy efficient, but is of no practical use so I don't think your current comparison is meaningful without more context.

Answer: Thanks for your comments. We agree that comparing the design without considering the energy efficieny is not fair. We rewrite the bit-width reduction part and discuss the designs with 1/2 bit separately. The results show that 1/2 bit designs are faced with more challenges on accuracy rather than performance. For the rest of the designs, the accuracy of inference can be kept within a fair range(1%)

Bit-width Reduction

Among all the designs, 1-2 bit based designs[21][33][35] show outstanding speed and energy efficiency. This shows that extremely low bitwidth is a promising solution for high performance design. As introduced in section 4.1, linear quantized 1-2 bit network models suffer from great accuracy loss. Further developing related accelerator will be of little use. More efforts should be put on the models. Even trading speed with accuracy can be acceptable consider the current hardware performance.

Besides the 1/2bit designs, the rest of the designs adopts 32-bit floating point data or linear quantiztion with 8 or more bits. According to the results in section 4.1, within 1% accuracy loss can be achieved. So we think the comparison between these designs are fair in accuracy. INT16/8 and INT16 are commonly adopted. But the difference between these designs are not obvious. This is because the under-utilization of DSPs discussed in section 5.1.1. The advantage of INT16 over FP32 is obvious except for [63], where the hard-core floating point DSPs are utilized. To a certain extent, this shows the importance of fully utilizing the DSPs on-chip.

14. Why did you pick those particular designs for Table 1?

Answer: Thanks for your question. We mainly review the published work in proceedings of the conference in FPGA, EDA and architecture from 2015. Among them, we select those with a whole system design and evaluation on real NN models with speed and power reported, so that we can give a fair evaluation and a relatively complete overview of state-of-the-art designs.

We mainly reviewed the FPGA based designs published in the top FPGA conferences (FPGA, FCCM, FPL, FPT), EDA conferences (DAC, ASPDAC, DATE, ICCAD), architecture conferences (MICRO, HPCA, ISCA, ASPLOS) since 2015. Because of the diversity in the adopted techniques, target FPGA chips, and experiments, we need to tradeoff between the fairness of comparison and the number of designs we can use. In this paper, we pick the designs with: 1) whole system implementation; 2) experiments on real NN models with reported speed, power, and energy efficiency.

15. In the text and Table 1 you refer to the various designs by their reference number, but then in Fig. 6 you use author names. This makes it very hard to figure out what the text is referring to in the graph. A similar issue is a statement like, ”1-2 bit based designs show...”. Which points are those on the figure?

Answer: Thanks for your comment. Now we use the reference number as the uniform format to refer to different designs in section 6.

16. Fig. 6: Y label should be GOP/s, not GOP.

Answer: Thanks to your comment. We have corrected it.

17. Section 6.0.4 - estimate of achievable performance of an ideal design. Needs more discussion. Why is this a valid estimate? Why are we not anywhere close to this?

Answer: Thanks for your question. The estimation here is too ideal. We change the baseline to [28] and use the reported speed up in [13] to estimate the gain from sparsity. We also add a new paragraph discussing the possible challenges in achieving the goal. Existing techniques are mostly evaluated independently. Integration of the techniques may bring new problems. Detailed discussion is as below:

Here we estimate the achievable speed of an ideal design. We use the 16-bit fixed point design in [28] as a baseline, which is the best 16-bit fixed point design with both the highest speed and energy efficiency. 8-bit linear quantization can be adopted according to the analysis in section 4.1, which achieves another 2× energy efficiency and speedup by utilizing 1 DSP as 2 multipliers. Fast convolution and frequency optimization further improves the system by 4× and 2× respectively. Consider a sparse model which is similar to the one in [13] with 10% non-zero values. We can estimate a similar 6× improvement as [13]. In general about 12× speedup and better energy efficiency can be achieved, which means 36TOP/s speed with about 25W. This shows that it is possible to achieve over 1 magnitude higher energy efficiency on FPGA over 32-bit floating point process on GPU.

The left problem is: does all the techniques: reuse of DSP, sparsification, quantization, fast convolution work well together? Pruning a single element in a 2D convlution kernel is of no use for fast convolution because the 2D kernel is always processed as a whole. Directly pruning 2D kernels as a whole may help. But the reported accuracy of this method is relatively lower [31], which means the compression space is less. The irregular data access pattern for processing sparse network and the increase in parallelism also brings challenge to the design of memory system and scheduling strategy.

Reviewer 2

This paper is well written and presents the great effort of the authors with surveying the existing FPGA-based neural network accelerator designs. There are several minor issues that require a minor revision.

1. The last paragraph of Section 1, Section 3 is missing in the paper structure introduction.

Answer: Thanks for your comments. We add the description for section 3 in this version.

2. Section 4.1.3, for the comparison of linear and non-linear quantization methods, I suggest either more examples should be collected for non-linear quantization method or the explanation of the observation need to be revised, e.g. the bit-width of neurons are bounded with 32, why?

Answer: Thanks for your question. We agree that the conclusion here is not correct. To the best of our knowledge, no work applies quantization on activations with the weights already non-linear quantized. On the other hand, few FPGA based hardware design targets at non-linear weight quantization. So we do not claim this point in the paper now.

3. Section 5.1.1 paragraph 4, for the declaration "All the IPs are .... actual hardware cost". If the IPs are required to avoid using DSPs, this comparison is only providing an ideal hardware cost in terms of logic resources. This need to be modified.

Answer: Thanks for your comments. The pure logic result is used to show the scale differece between different computation units. We add the synthesis result with DSP in this version.

4. Section 7.1, DnnWeaver should be either a mixed method or instruction based.

Answer: Thanks for your comment. The phrases "HDL model based method" and "instruction based method" may have caused confusion. We do the classification by judging whether the hardware changes with the target network. With our understanding of DnnWeaver[43], it is a HDL model based method. Even though DnnWeaver designs an instruction set, ”The compiler statically translates these instructions to microcodes and state machines”. According to the flow graph of DnnWeaver, i.e. Figure 3 in [43], the instructions are used to generate the hardware. So we classify DnnWeaver as HDL model based method here.

To address this issue, we refer to the two kinds of methods as ”hardware design automation” and ”software design automation” respectively in this version. We also add a more detailed description of our classification criteria at the beginning of section 7.

In certain application senarios, various NN models are to be supported with the FPGA accelerator. Whether the accelerator can response to the change in network model promptly and keeps high performance becomes a key feature. To address this problem, various researches have been carried out to automatically map a NN model to FPGA. Mainly two kinds of methods are used: hardware design automation and software design automation. Hardware design automation generates different hardware designs according to different NN models. Software design automation keeps a same accelerator and generates different inputs to the accelerator.

Several minor typos: 1. Section 4.4, paragraph 2, ”The left weights are then fined-tuned are....” should be ”The left weights are then fined-tuned with”. 2. Section 5.1, paragraph 1, ”then the this also A”, seems a sentence is missing here. 3. Section 5.2, paragraph 1, ”and can further accelerated in frequency domain” should be ”can be ...”. 4. The figures in this paper need further arrangement to make it more clear and beautiful.

Reviewer 3

This paper presents a survey of FPGA-based neural network accelerators. The survey is of limited value though because it focuses only on CNN. Moreover, the survey does not provide a high-level view of the different architectural choices in designing FPGA-based CNNs. Instead, it looks into low-level design optimizations. However, the evaluation section is well done and provides a comprehensive summary of the quantitative performance of the different proposals.

1. The design methodology considers model accuracy, throughput, energy-efficiency but does not consider latency as an important optimization function (specially in non-batched mode).

Answer: Thanks for the comment. Latency is certainly an important optimization target. We add corresponding description in section 3. But we will not discuss this point in detail because of the following reasons:

- Most of the NN accelerators does not process inputs concurrently.
- The realization of concurrent process is simple because the computation graph of a certain NN model is fixed. Usually batch process[28] or layer pipeline[24] is used to concurrently process different inputs. The concurrency equals to the batch size or the number of pipeline stages.

We also give some explanation on this point in the paper:

Most of the FPGA based NN accelerators compute different inputs one at a time, which means the latency of the system is the reciprocal of throughput. The concurrent process of different inputs is usually realized with batch[28]] or layer pipeline[24]. In these cases, the concurrency equals to the batch size or the number of pipeline stages. The latency of the accelerator can be described as equation 1.

$$latency = \frac{concurrency}{throughput} \tag{1}$$

So in this paper, we focus more on optimizing the throughput. As different accelerators may be evaluated on different NN models, a common criterion of speed is the $actual\_performance$, which eliminates the effect of different workload to some extent.

2. Please explain Equation 3 on throughput. Why is utilization included in the equation?

Answer: Thanks for your question. The notations used here needs more explanation and we have added them. In equation 1, we express the actual performance as the product of peak performance and the utilization ratio of the computation units. This is because these two parts can be optimized with different techniques, which guides the discussion in the following sections. This part is modified to be clearer.

Speed. The throughput of an NN accelerator can be expressed by equation 2. Peak performance, mTo parallelize the execution of the loops, we unroll the loops and parallelize the process of a certain number of iterations on hardware. The number of the parallelized iteraions on hardware is called the unroll parameter. Inappropriate unroll parameter selection may lead to serious hardware underutilization. Take a single loop as an example. Suppose the trip count of the loop is M and the parallelism is m. The utilization ratio of the hardware is limited by $m/M\lceil M/m\rceil$. If M is not divisible by m, then the utilization ratio is less than 1. Furthermore, in the case of NN process, the total utilization ratio will be the product of the utilization ratio along each of the loops.easured in operations (multiplication or addition) per second, is achieved when all the computation units work every clock cycle. Utilization denotes the average ratio of working cycles of the computation units. The workload measures the number of operations in the target neural network. With a certain FPGA chip, the on-chip resource is limited. We can increase the peak performance by reducing the size of each computation unit and increasing the working frequency. Reducing the size of computation units can be achieved by simplifying the basic operations in neural network model, which may hurt the model accuracy and requires hardware-software co-design. Increasing working frequency, on the other hand is pure hardware design work. A high utilization ratio is kept by reasonable parallelism implementation and efficient memory system. Most of this part is affected by hardware design. But optimization on NN models can also reduce the storage requirment of a neural network model and benefits the memory system.

$$throughput = \frac{actual\_performance}{workload} = \frac{peak\_performance \times utilization}{workload} \tag{2}$$

3. Energy-total in Equation 4 is not exactly energy efficiency. It is simply total energy. I would have liked to see performance per watt as a metric instead.

Answer: Thanks for your comment. We try to improve this part by showing the measurement at the very beginning and give another equation to explain the relation between energy efficiency and the energy cost to compute each input

Energy Efficiency. Energy efficiency is the actual performance within a certain power cost, i.e. the average number of operations can be executed within certain energy budget. This is usually measured in (OP/s/W) or (OP/J). Given a certain network model, the workload is fixed. Increasing the energy efficiency of a neural network accelerator means to reduce the total energy cost, E_{total} to process each input as shown in equation 3. This energy cost comes from 2 parts: computation and memory access, which is shown in equation 4.

$$energy\_efficiency = \frac{E_{total}}{workload} \tag{3}$$

$$E_{total} \approx N_{op} \times E_{op} + N_{SRAM\_access} \times E_{SRAM\_access} + N_{DRAM\_access} \times E_{DRAM\_access} + E_{static} \tag{4}$$

4. The hardware design section can benefit from a high-level classification of the architectures used in FPGA implementation and their summary: for example systolic array, daisy-chain like Intel-DLA architecture etc. Currently the focus is on simple low-level optimizations.

Answer: Thanks for your suggestion. With our understanding of systolic array, the main purpose is to transfer data between adjacent PEs to avoid the large fan-out in broadcasting the data to the PEs. This is also claimed in [55]. The daisy-chain archictecture in [3] works similarly. For this reason, we put this technique in section 5.1.3.

On the other hand, using systolic array requires certain data reuse between adjacent cycles to cover the warming up latency of the systolic pipeline. For this reason, using systolic array brings limitation to the loop execution order and should be a higher level technique. In this paper, we classify the hardware design techniques following a inside to outside mannar into three aspects:

- Computation unit level, which only involves the implementation of the basic computation units.

- Loop unrolling level, which involves how the data are fetched from the on-chip buffer and how the workload is parallelized.

- System level, which involves the data transfer between on-chip buffer and off-chip DRAM, resolving the bandwidth limitation and increasing hardware utilization ratio.

So we think that the systolic array design should be a loop unrolling level technique and have reorganized corresponding sections.

5. The description of Figure 3 should be improved. I read this part many times and still could not understand what Figure 3(a) and 3(b) are representing.

Answer: Thanks for your comment. We remove the figure and use a simpler example of a single loop to explain the concept of loop unrolling and hardware underutilization. The explanation is shown as below:

To parallelize the execution of the loops, we unroll the loops and parallelize the process of a certain number of iterations on hardware. The number of the parallelized iteraions on hardware is called the unroll parameter. Inappropriate unroll parameter selection may lead to serious hardware underutilization. Take a single loop as an example. Suppose the trip count of the loop is M and the parallelism is m. The utilization ratio of the hardware is limited by $m/M\lceil M/m \rceil$. If M is not divisible by m, then the utilization ratio is less than 1. Furthermore, in the case of NN process, the total utilization ratio will be the product of the utilization ratio along each of the loops.

6. The approaches to efficiently utilize on-chip memory is missing from the survey, even though it is an important consideration in most designs.