Squeezing the last MHz for CNN acceleration on FPGAs

Li Li*, Dawen Xu*†, Kouzi Xing*, Cheng Liu†[1], Ying Wang†, Huawei Li† and Xiaowei Li†
*Hefei University of Technology, Hefei, China
†Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
Email: xudawen@gmail.com {liucheng, wangying2009, lihuawei, lxw}@ict.ac.cn

***Abstract*—Neural networks especially the convolution neural networks (CNN) have become prevalent and numerous CNN accelerators have been developed to achieve higher performance. While clock frequency determines the operation speed and has direct influence on the performance of the accelerators, we propose to apply overclocking, a circuit optimization approach that enables higher clock frequency, on general CNN accelerators. This technique brings significant performance improvement, but it leads to moderate timing errors, wrong computing results and low prediction accuracy.**

By taking advantage of the inherent fault tolerance of neural networks, we opt to learn the computing errors together with the application data with additional on-accelerator training. In this case, the resulting models can be resilient to the errors and do not necessarily suffer considerable prediction accuracy loss. In addition, we also take the worst case of overclocking into consideration with a series of approaches ranging from fault detection to fault recovery in case of hardware crash. Finally, we demonstrate the use of overclocking on a CNN accelerator implemented on Xilinx KCU1500 with comprehensive experiments. The experiments show that overclocking in combination with the on-accelerator neural network training improves both the neural network performance and energy efficiency with small prediction accuracy loss.

I. Introduction

Recent years have seen great success of convolutional neural networks (CNNs) on massive fields. The widespread adoption of CNNs motivates the continuous optimization of CNN accelerators for higher performance and energy efficiency. Numerous approaches from various angles have been proposed and explored [1] [2]. Particularly, many of them such as low-precision quantization [3] and approximate computing [4] have been demonstrated to be able to improve the performance or lower the power consumption of the CNN accelerators significantly. In spite of the computing errors caused by these approaches, the prediction accuracy loss of the neural networks is negligible according to the prior works [5]. One of the key reasons can be attributed to the inherent fault-tolerance of the neural networks, which means that minor variations on the computing of neural networks typically will not lead to considerable difference in terms of prediction accuracy. This feature essentially prevents the propagation of the minor computing errors to the neural networks prediction errors. Therefore, the corresponding approaches that allow design trade-off between computing accuracy and performance can be applied to CNN accelerator optimization.

Inspired by this observation, we notice that overclocking, which allows the circuit running at higher clock frequency with minor timing violations, can also compromise between performance and computing accuracy. When applying overclocking to CNN accelerators, it can potentially improve the performance and energy efficiency with minor prediction accuracy loss. Compared to prior optimization approaches, overclocking brings multi-folded unique advantages especially to the FPGA based CNN accelerators.

As demonstrated in [6], convolution is usually computing-bound and it is the most time-consuming part of the CNNs, while full connection layers are mostly memory-bound. Particularly, it can be found that newer neural networks such as ResNet and DarkNet involve more convolution layers and less full connection layers. Thus, convolution neural networks are mostly constrained by the computing capability of the accelerators. It is known that the frequency of the CNN accelerators determines the operation speed directly. However, the clock on FPGAs usually ranges from 150 MHz to 300 MHz and remains much lower compared to the ASIC designs. The bandwidth of DRAM on FPGAs and ASICs does not have any clear gap [7]. Thus, computing of the FPGA based CNN accelerators is more stringent and becomes the major performance bottleneck. Overclocking can greatly alleviate this problem without redesigning the CNN accelerators and is mostly orthogonal to the other optimizations. Another advantage brought by overclocking is the higher energy efficiency [8]. Typically the overall power of the FPGA based CNN acceleration system consists of kernel accelerator power and background power. The kernel accelerator power roughly scales with the clock frequency and performance, so overclocking will not incur more energy consumption. The background power changes little with clock frequency. Taking both parts into consideration, overclocking improves the energy efficiency.

To deploy overclocking on FPGA based CNN accelerators, there are also several challenges that need to be addressed. Timing errors caused by overclocking may probably located on the critical paths which may spread across the design and change with the data, while the critical path can be affected by the work environment such as the temperature and input data, it may deteriorate or vary at runtime [8]. As a result, the influence of overclocking timing errors on the computing

[1]Corresponding author is Cheng Liu.

978-1-7281-4718-5/19/$31.00 ©2019 IEEE
DOI 10.1109/ITC-Asia.2019.00039

of CNNs is rather complex. It may lead to computing errors when the computing data paths are affected or system stall when some critical control signals get affected. To ensure robust computing on the overclocked CNN accelerators, all the errors must be detected and handled appropriately. In this case, different fault detection strategies and fault recovery strategies should be adapted accordingly.

Overclocking essentially focuses on relaxing the design constraints and the safety margins. A number of works [9] [10] adopted better than worst case design to allow the system to work aggressively and recover from timing errors with additional error checkers. They demonstrated that the benefits brought by removing the safe margin outweigh the cost of monitoring and recovering from errors. The authors in [11] explored the design trade-off between performance and accuracy on FPGAs through overclocking. This work demonstrated the potential of applying the trade-off on some image processing applications. Recently, Wang etc. in [12] investigated the use of overclocking on CNN accelerators, but they mainly targeted at the clock frequency tuning with complex timing models and explored only small neural networks due to the time-consuming simulation.

In this work, we explored the use of overclocking on FPGA based CNN accelerators. By taking advantage of the inherent fault-tolerance of neural networks as well as the FPGA reconfiguration capability, we aim to boost the CNN accelerator clock and mitigate the negative influence of the timing errors with negligible cost. The contributions of this work is concluded as follows:

- We proposed to apply overclocking on FPGA based CNN accelerators to alleviate the stringent computing bottleneck and improve the overall energy efficiency.
- For all the possible negative influence of overclocking, we proposed a series of error detection and recovery mechanisms to ensure resilient neural network execution.
- With comprehensive experiments, we demonstrated that the use of overclocking can improve the accelerator performance and energy efficiency significantly with negligible prediction accuracy loss.

The paper is organized as follows. Section II motivates the use of overclocking. Section III presents the use of overclocking on FPGA based CNN accelerators with additional error detection and recovery. Section IV performs comprehensive experiments and demonstrates the benefits of using overclocking. Section V concludes this work.

II. Motivation

Clock frequency determines the accelerator operation speed and directly affects the performance. Accordingly, it also has influence on the neural network runtime and energy efficiency. In this section, we take an open-sourced CNN accelerator named PipeCNN [13] as an example and analyze the influence of clock frequency on the neural network performance and energy efficiency.

The accelerator is implemented on KCU1500 and attached to a desktop computer with Intel i7-6700@3.40GHz. The basic

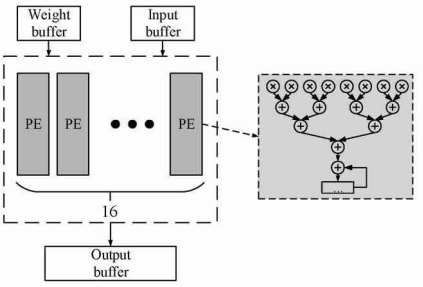

Fig. 1. Baseline CNN accelerator architecture.

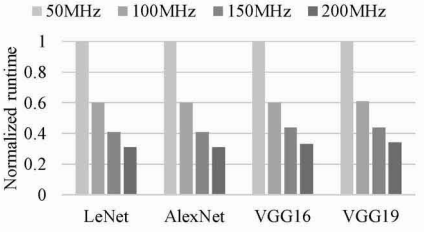

Fig. 2. Normalized runtime of neural networks executed on CNN accelerators with different clock frequency.

convolution structure is shown in Fig 1. The computing array consists of 16 dot production units which is also named as processing elements (PE). Each PE allows a parallel dot production of two 4-data vectors. The optimized clock frequency according to the SDAccel compilation is 200 MHz.

In order to evaluate the influence of clock frequency, we set the clock to 50 MHz, 100 MHz, and 150 MHz respectively. A set of neural networks including LeNet, AlexNet, VGG-16 and VGG-19 are used as the benchmark. Both the runtime and energy efficiency are normalized to the 50 MHz implementation. Normalized runtime of the neural network benchmark executed on the accelerators are shown in Fig 2. It shows that the overall performance of the neural network benchmark almost increases proportional to the clock frequency. It can be predicted that the processing remains computing-bound and high-frequency CNN accelerator design will be beneficial.

To analyze the energy consumption, we obtain the power consumption from SDAccel 2017.1. Given the power consumption and the runtime, we calculated the energy delay product (EDP) which can be used as an energy efficiency metric. The experiment result is presented in Fig 3. It shows that higher clock frequency enhances both the neural network performance and energy efficiency. This motivates the use of overclocking that enables even higher clock frequency.

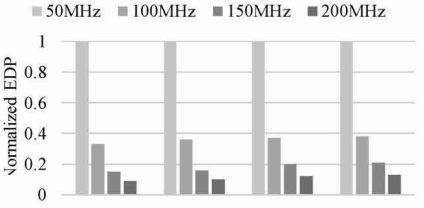

Fig. 3. Normalized EDP of neural networks executed on CNN accelerators with different clock frequency.

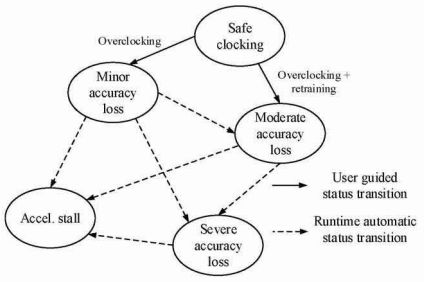

Fig. 4. Overclocked CNN accelerator status transition graph.

III. CNN ACCELERATOR OVERCLOCKING

Overclocking can boost the clock frequency of CNN accelerators and is potentially beneficial to both the performance and energy efficiency. However, the timing violation may lead to distinct errors which may degrade the neural network prediction accuracy or even affect the functionality of the accelerators. In this section, we will explore the use of overclocking on FPGA based CNN accelerators to benefit from overclocking and minimize its negative influence.

A. Overclocking overview

Based on the intensity of overclocking and timing violation, we divide the overclocking incurred errors into different categories so that corresponding design methods can be used to alleviate the resulting problems efficiently. While it is difficult to precisely quantize the timing violations directly at runtime, we use the neural network prediction accuracy loss as the main classification metric. When there is only minor prediction accuracy loss which is less than 1%, the penalty is usually acceptable and nothing needs to be done.

When the prediction accuracy loss ranges from 1% to 10%, the accelerator still functions and the moderate accuracy loss can be salvaged. As the neural network models are usually obtained from training on general purposed processors (GPPs) which assumes the neural networks to be executed on an equivalent computing device, the computing on overclocked CNN accelerator varies and does not match with the assumption. This can be alleviated with neural network retraining.

When there are severe timing violations and the accuracy loss is over 10%, there is usually little chance to recover without improving the timing and the status of the accelerator may not be steady. It may result in considerable computing errors and dramatic prediction accuracy loss or even system stall due to critical control signal errors.

In addition, the timing errors incurred by overclocking are difficult to predict and there is no guarantee that the accelerator can function properly with just slightly overclocking. Typically, we expect that the lower overclocking setup may also lead to severe computing errors with lower probability. Fig 4 shows the possible accuracy loss status transition graph. It indicates that we should take the worst case into consideration even when the overclocking setup is relatively conservative.

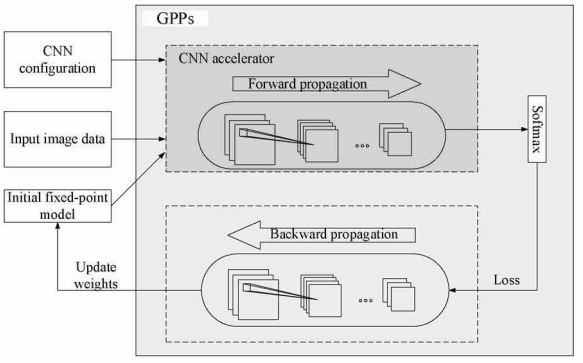

Fig. 5. On-accelerator neural network retraining framework.

B. Techniques for mitigating overclocking errors

As discussed in the above section, there are different overclocking options which have distinct negative influence on the neural network execution. Corresponding approaches need to be applied to mitigate all the possible negative influence.

1) Accuracy loss estimation: Before we start to address the different overclocking problems, we need to determine the status of the overclocked CNN accelerator at runtime. Assume that the CNN accelerator performs inference of a stream of input data continuously. To estimate accuracy loss at runtime, we insert a set of reference data to the input data stream periodically. The actual prediction accuracy of the reference data can be computed in advance. When the reference data are processed on the overclocked CNN accelerator, the prediction accuracy of the reference data can be obtained. By comparing the golden accuracy and the measured accuracy, we can obtain the accuracy loss and determine the status of the CNN accelerators accordingly.

2) Overclocked accelerator with moderate accuracy loss: When there is only minor accuracy loss, nothing needs to be done by default. When there is moderate prediction accuracy loss due to the overclocking incurred errors, we try to retrain the neural network model on top of the CNN accelerator. The basic idea is to have both the application data and the computing error patterns learned by the neural network models such that the networks can be less sensitive to the computing variations.

Fig 5 shows the retraining framework built on top of Caffe. The forward propagation that is usually performed on GPPs is transferred to the FPGA based CNN accelerator while the backward propagation remains on host CPUs. As the computing on the CNN accelerator is fixed point, the updated weight must be converted to fixed point before they are moved to FPGA for forward propagation. Accordingly, the result of the forward propagation needs to be converted to float point when it is sent to the host for backward propagation.

In this work, we utilized an OpenCL based CNN accelerator. The communication between host and the accelerator can be conveniently achieved with the OpenCL API. Many RTL based CNN accelerators can also be wrapped up with OpenCL API and reuse the retraining framework with minor modification.

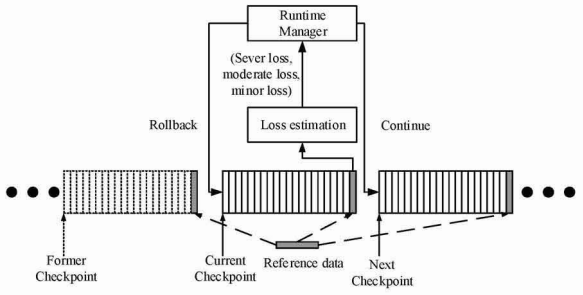

Fig. 6. Runtime checkpoint strategy.

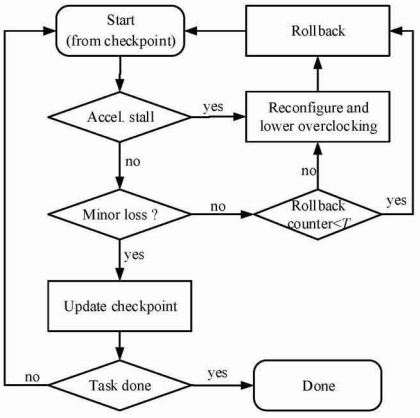

Fig. 7. Runtime overclocking management.

Similar platform can also be found in [6] and [14].

3) Overclocked CNN accelerator with severe accuracy loss: When the accuracy loss gets too large, it indicates low-probability severe timing violations occur on the CNN accelerators. The basic idea is to repeat the execution. However, when the errors are detected, previous computing results may already suffer the errors and the results can already be wrong. To address this problem, we opt to utilize the checkpoint strategy to recover from previous checkpoint. Fig 6 shows the basic control flow of the checkpoint strategy. Basically, we divide the input data stream into blocks and a set of golden reference data will be added to the end of each block. With the reference data, neural network accuracy can be measured and used for accuracy loss estimation. Then the runtime manager can decide the processing with the estimation. When the accuracy loss is acceptable, the processing will continue. When the accuracy loss is too large, the processing will roll back to the checkpoint. The runtime manger will be detailed in the next subsection.

4) Runtime overclocking management: As discussed in the above section, the accelerator status may transit at runtime, while corresponding remedies will be applied in different situations. To bridge the gap, we propose a runtime overclocking management strategy as shown in Fig 7 to handle all the possible overclocking problems at runtime automatically.

When the overclocked CNN accelerator is applied for inference, a watchdog must be enabled to detect if the accelerator is stalled at runtime. When the accelerator is stalled, it indicates that critical signals are corrupted. The manger will invoke the reconfiguration and reload an implementation with lower overclocking. Meanwhile, it also rolls back the processing using the checkpoint strategy. When there is no system stall, accuracy loss will be evaluated with the granularity of a block or checkpoint. When there is just minor accuracy loss, the block execution is considered to be valid and the checkpoint will be updated. When the accuracy loss is non-trivial and there have been multiple rollbacks on this checkpoint, we consider the accelerator overclocking to be too aggressive. Then we reconfigure the FPGA with a more conservative implementation and roll back for another execution on current checkpoint. Note that we set a rollback counter and compare it with a threshold T to determine if we should choose the reconfiguration route or the simple rollback route. This scheme is used to avoid frequent reconfiguration.

IV. Experiments

In this section, we analyze the benefits of overclocking on a set of representative convolution neural networks. and estimate the trade-offs of the strategies used to mitigate the overclocking incurred errors.

A. Experiment setup

We used PipeCNN as the baseline CNN accelerator and had it implemented on KCU1500 FPGA board. Four representative neural networks including LeNet, AlexNet, VGG-16 and VGG-19 are used to benchmark the performance and energy efficiency of the CNN accelerator. The implementation clock frequency of LeNet and AlexNet is 210 MHz while the clock frequency for VGG-16 and VGG-19 is 190 MHz. Then we gradually overclock the implementation with 10 MHz step until the accelerator gets stuck or crashes frequently. For the 210 MHz implementation, the highest overclocking setup is 260MHz. For the 190 MHz implementation, the highest overclocking setup is 240 MHz. Finally, we evaluate the performance and energy efficiency of the accelerators using all the available overclocking configurations.

B. Accuracy, performance and energy efficiency

The prediction accuracy of the benchmark neural networks on the CNN accelerators with overclocking is presented in Fig 8. It can be found that the prediction accuracy regardless of top1 or top5 typically remains the same under moderate overclocking. However, when the clock continues to rise, it may reach to a tipping point where the prediction accuracy drops clearly. Fortunately, the prediction accuracy can be improved with just on-accelerator retraining. When the clock further increases, the prediction accuracy drops dramatically. In spite of the retraining, the prediction accuracy is still not acceptable in practice. Part of the reasons can be that the number of timing violations increases rapidly with higher clock frequency.

We further evaluate the normalized performance over the original CNN accelerators. As given in Fig 9, the performance with the extreme accelerator overclocking achieves 1.25X

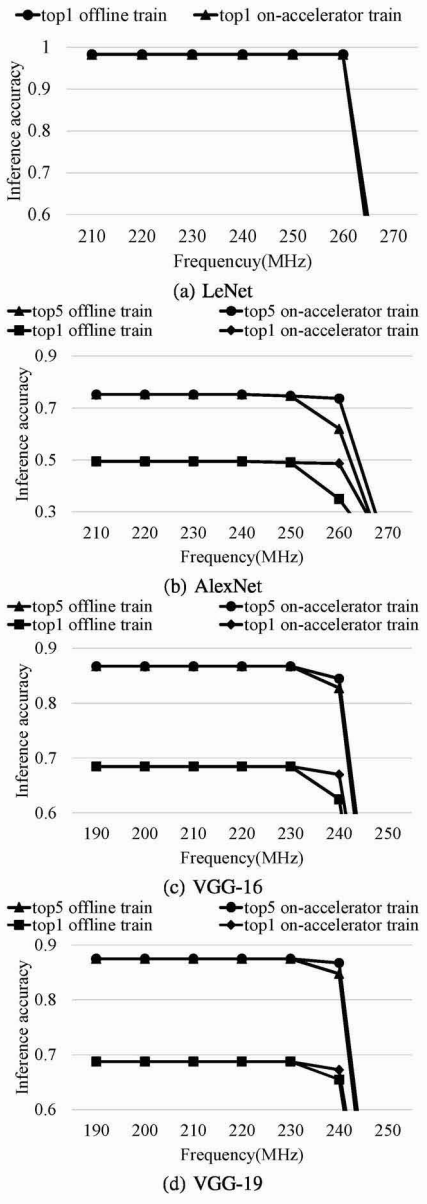

Fig. 8. The prediction accuracy of the benchmark neural networks on accelerators with different overclocking

speedup on average compared to the baseline design. We use EDP as the energy efficiency metric and compare the different overclocking configurations as exhibited in Fig 10. According to the figure, overclocking on FPGA based CNN accelerators achieves more significant energy efficiency improvement when compared to the performance improvement. A key reason is that the power consumption does not scale much with the clock frequency due to the relatively higher background power consumption. On the contrast, the performance which contributes more to the EDP metric scales much better.

In order to quantize the exact computing errors caused by overclocking, we particularly analyze the last layer output of the neural networks which is typically a vector. We compare it

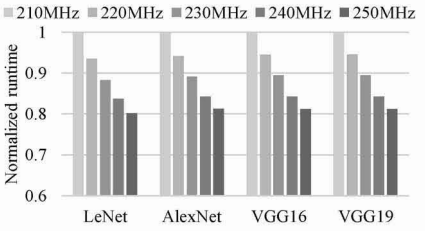

Fig. 9. Normalized performance of neural networks executed on CNN accelerators with different overclocking.

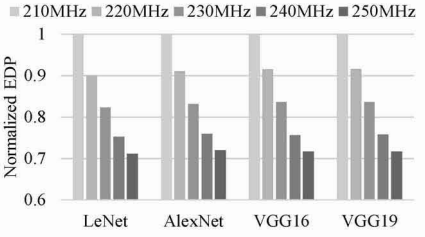

Fig. 10. Normalized EDP of neural networks executed on CNN accelerators with different overclocking.

with the output without overclocking. And we use the percentage of changed output data and the Euclidean distance of the output vectors to characterize the difference. The comparison is presented in Table I. It can be observed that computing errors can be completely hidden by the neural networks when the clock frequency is less than 240 MHz in the experiment. Moreover, the small Euclidean indicates that the computing error amplitude is relatively small though the percentage of the affected results is high when the clock is higher than 240 MHz.

C. Optimization trade-offs

As discussed in this paper, the timing error can be affected by many factors and it may change at runtime. There is no guarantee that the behavior of the CNN accelerator can keep stable even overclocking is just slightly higher than the original clock. We use the accelerator overclocked at the tipping point to perform the neural network computing. Then we keep measuring its prediction accuracy. As shown in Fig 11, we find that the accuracy is rather stable. Although we still can not ensure the stability of the overclocked CNN accelerator, we can be sure that the probability of the severe errors such as accelerator hangup or considerable accuracy loss is rather low. As we did not observe the cases after processing 100000 pictures, we assume the probability of an severe error when processing an input picture is lower than 1/100000.

With the severe errors, we need to invoke the checkpoint-based error recovery strategy. However, this strategy incurs

TABLE I
FAULT RATE AND EUCLIDEAN DISTANCE ON THE LAST OUTPUT LAYER OF THE NEURAL NETWORK

Frequency(MHz)	210	220	230	240	250	260
Euclidean distance	0	0	0	0	35.2	104.8
Fault rate(%)	0	0	0	0	56.6	72.1

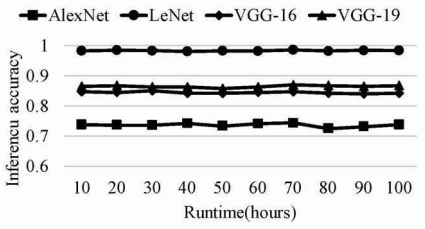

Fig. 11. Overclocking stability analysis

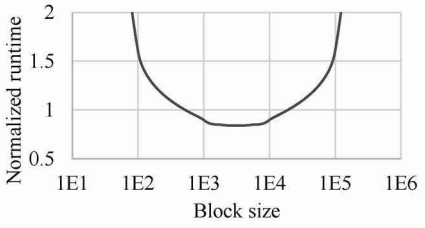

Fig. 12. The relative overclocking runtime with different block size.

cost including error detection and error recovery. We evaluate the overhead of the checkpoint-based strategy. A few parameters are critical to the overhead. One of them is the block size, which decides the granularity of checkpoint. One of them is the critical fault rate. It refers to the probability that CNN accelerator got severe errors when processing a single input figure. Another one is the reference data size.

Suppose reference data size is 100 and fault rate under 250 MHz is 1E-5. The relative runtime of overclocking over the normal design without overclocking is shown in Fig 12. When the block size is too big, the roll back cost will be too large and incurs additional runtime. When the block size is too small, the inserted reference data becomes non-trivial and the overall runtime also gets deteoriated. There is an optimal block size given specified fault rate and reference data size. We also analyze the trend of the relative runtime under different fault rate. As shown in Fig 13, the relative runtime increases slightly at lower relative fault rate range but grows rapidly when the fault rate is larger than 1E-5.

V. Conclusion

Neural networks executed on FPGA based CNN accelerators are typically bounded by the computing. While overclocking that boosts the clock frequency directly affects the accelerator operation speed and performance, it also leads to some unpredictable timing violations. By taking advantage of the inherent neural network fault-tolerance and FPGA reconfiguration capability, we propose a series of error detection and recovery scheme to alleviate the accuracy loss. According to the experiments on a set of neural network benchmark, we can obtain both the performance and energy efficiency significantly with negligible accuracy loss.

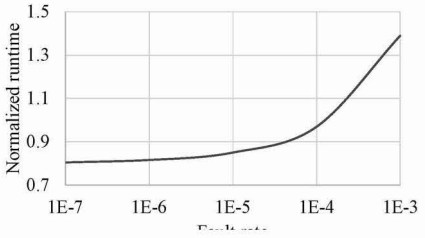

Fig. 13. The relative overclocking runtime with different fault rate.

Acknowledgement

This work is supported in part by the National Natural Science Foundation of China (No. 61874124), Chinese Academy of Sciences STS (No. KFJ-STS-SCYD-226).

References

[1] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 243–254.

[2] Y. Wang, J. Xu, Y. Han, H. Li, and X. Li, "Deepburning: Automatic generation of fpga-based learning accelerators for the neural network family," in *Proceedings of the 53rd Annual Design Automation Conference*, ser. DAC '16. New York, NY, USA: ACM, 2016, pp. 110:1–110:6.

[3] K. Hwang and W. Sung, "Fixed-point feedforward deep neural network design using weights +1, 0, and -1," in *2014 IEEE Workshop on Signal Processing Systems (SiPS)*, Oct 2014, pp. 1–6.

[4] S. Hashemi, R. I. Bahar, and S. Reda, "Drum: A dynamic range unbiased multiplier for approximate applications," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD '15. Piscataway, NJ, USA: IEEE Press, 2015, pp. 418–425.

[5] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *CoRR*, vol. abs/1510.00149, 2015.

[6] C. Zhang, G. Sun, Z. Fang, P. Zhou, P. Pan, and J. Cong, "Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2018.

[7] P. Dorsey, "Xilinx stacked silicon interconnect technology delivers breakthrough fpga capacity, bandwidth, and power efficiency," 2010.

[8] B. Greskamp and J. Torrellas, "Paceline: Improving single-thread performance in nanoscale cmps through core overclocking," in *Proceedings of the 16th International Conference on Parallel Architecture and Compilation Techniques*, ser. PACT '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 213–224.

[9] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A low-power pipeline based on circuit-level timing speculation," in *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 36. Washington, DC, USA: IEEE Computer Society, 2003.

[10] A. K. Uht, "Going beyond worst-case specs with teatime," *Computer*, vol. 37, no. 3, pp. 51–56, 2004.

[11] K. Shi, D. Boland, and G. A. Constantinides, "Accuracy-performance tradeoffs on an fpga through overclocking," in *2013 IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines*, April 2013, pp. 29–36.

[12] Y. Wang, J. Deng, Y. Fang, H. Li, and X. Li, "Resilience-aware frequency tuning for neural-network-based approximate computing chips," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2736–2748, 2017.

[13] D. Wang, K. Xu, and D. Jiang, "Pipecnn: An opencl-based open-source fpga accelerator for convolution neural networks," in *2017 International Conference on Field Programmable Technology (ICFPT)*, Dec 2017, pp. 279–282.

[14] R. DiCecco, G. Lacey, J. Vasiljevic, P. Chow, G. Taylor, and S. Areibi, "Caffeinated fpgas: Fpga framework for convolutional neural networks," in *2016 International Conference on Field-Programmable Technology (FPT)*, Dec 2016, pp. 265–268.