ON-CHIP FPGA DEBUG INSTRUMENTATION FOR MACHINE LEARNING APPLICATIONS

Daniel Holanda Noronha [1] **Ruizhe Zhao** [2] **Jeff Goeders** [3] **Wayne Luk** [2] **Steven J.E. Wilton** [1]

ABSTRACT

FPGAs show promise as machine learning accelerators for both training and inference. Designing these circuits on reconfigurable technology is challenging, especially due to bugs that only manifest on-chip when the circuit is running at speed. In this demonstration we present instrumentation that accelerates the process of debugging machine learning circuits on-chip by recording domain-specific characteristics of the circuit instead of the raw values of the variables. As a result, the proposed instrumentation combined can achieve 21.8−24.1x longer visibility, while still providing useful information for the designer.

1 INTRODUCTION

The need for a high performance per watt combined with the rapid changing requirements of machine learning algorithms is leading companies to adopt FPGAs as accelerators in data centers (Putnam et al., 2014). Despite these advantages, mapping circuits to those devices is difficult, partially due to the low on-chip observability that hinders debugging. Differently from simulation, which provides full visibility, only on-chip signals that interact with the external world can be observed using external equipment. Since only a limited number of I/O pins is available, a common solution is to use on-chip trace buffers to record the behaviour of the circuit into on-chip memory for subsequent analysis.

Although general purpose trace buffers can be helpful, debugging a machine learning circuit mapped into an FPGA is still very challenging for three main reasons. First, those applications generally require very long run-times (multiple training/inference samples) before overall behaviour can be understood. Second, the correctness of an ML circuit can often not be determined by looking at individual signals/variables. The weight values of the matrix, for example, can only be said to be "correct" when properly acting in concert with the other parts of the system. Third, ML circuits are often designed by using high level frameworks (e.g. Tensorflow or Caffe), which means that visibility in the context of the hardware design may not be as useful.

[1]Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, Canada [2]Department of Computing, Imperial College London, London, UK [3]Department of Electrical and Computer Engineering, Brigham Young University, USA. Correspondence to: Daniel Holanda Noronha <danielhn@ece.ubc.ca>.

Proceedings of the 2nd SysML Conference, Palo Alto, CA, USA, 2019.

In this demonstration, we are showing infrastructure that accelerates the debug of machine learning applications on FPGAs. The heart of our system is instrumentation optimized specifically for these types of applications. Similar to previous work, our instrumentation records data as the circuit runs at speed for later off-line interrogation. Unlike existing debug flows we use domain-specific characteristics of ML circuits to create instruments that maximize trace buffer utilization, while providing information that is meaningful to an engineer.

2 DOMAIN-SPECIFIC INSTRUMENTATION

Many machine learning applications consist of large arrays (eg. activations or weights). Our instruments were designed especially to track those large arrays over time. Instead of tracking the raw information at every cycle our instrumentation continuously gathers information and combines it at every frame. In a CNN, for example, a frame may represent all calculations corresponding to a single input image.

We have evaluated three main instruments. The distribution instrument generates one histogram per frame, which allows the designer to easily detect outliers or errors causing activa-

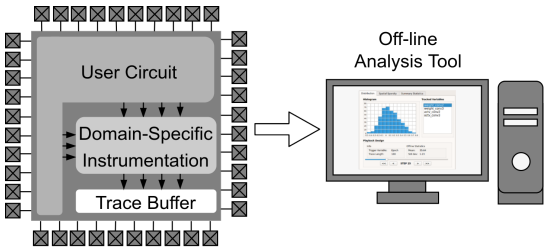

Figure 1. Domain-Specific Debug Instrumentation

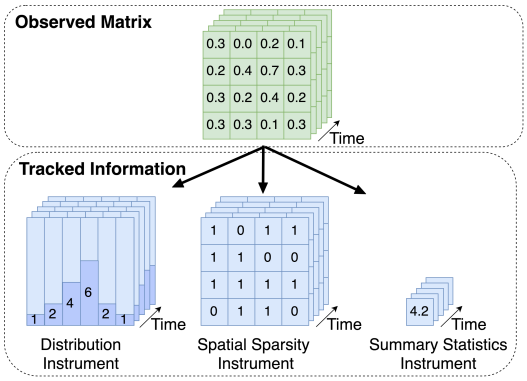

Figure 2. Instruments Overview

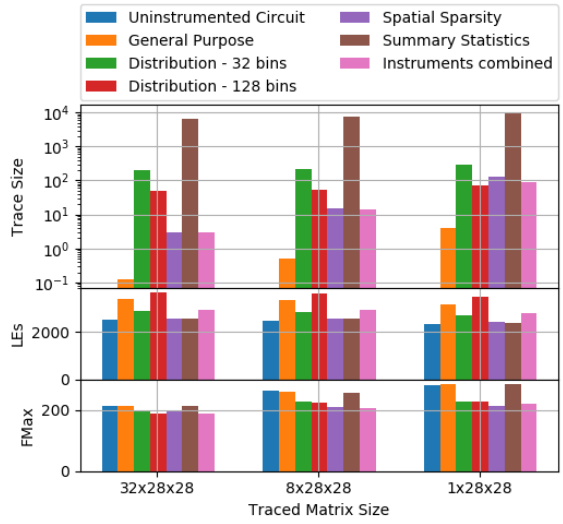

Figure 3. Resources and trace length when compared to general purpose debug.

tions to "clamp" at minimum/maximum values. The spatial sparsity instrument stores the sparsity of the observed matrix while preserving the spatial location of each element, providing a graphical visualization of a potential bug. Finally, the summary statistics, such as measures of tendency, dispersion and sparsity can also be traced at every frame.

3 Evaluation

We compare our technique to the HLS-oriented debug flow presented in (Goeders & Wilton, 2017). The user circuits used for this comparison are composed of kernels that are part of Convolutional Neural Networks (CNNS) generated using (Zhao et al., 2018). Five different configurations are tested for each of the three kernels.

We refer to the number of times information about the entire frame can be tracked as the *trace size*. As shown in Figure 3, the general purpose debug instrumentation is only able to trace the kernels for a few frames, even assuming that we are

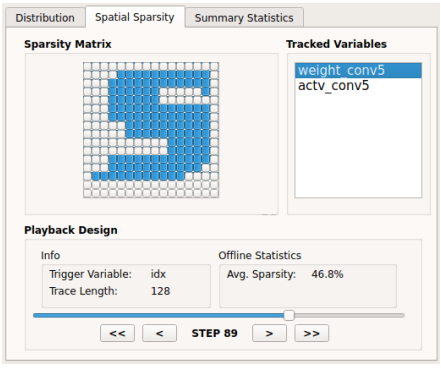

Figure 4. User interface - Spatial sparsity instrument tab

perfectly packing the information. In contrast, the proposed instrumentation can achieve significantly longer trace size at a low area and latency cost. Even if all proposed instruments are combined, information about the frames can be stored for 21.8−24.1x longer.

4 Demonstration Platform

In this demonstration, the proposed instrumentation will be presented through the user interface shown in Figure 4. The graphical interface will be used by the audience for off-line analysis after the data has been traced and part of the audience will be challenged to find artificially added bugs using our GUI. Such interface allows the user to step through multiple frames of different CNN circuits and provides a concrete example of how the proposed instruments can be visualized and used to accelerate the process of debugging real machine learning applications. No special equipment is required at the place of the demo.

We will also make live demonstrations of how to easily instrument HLS-generated circuits and discuss how our application-specific debug instrumentation can be integrated into existing commercial debugging tools.

References

Goeders, J. and Wilton, S. Signal-tracing techniques for in-system FPGA debugging of high-level synthesis circuits. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 36(1):83–96, Jan 2017.

Putnam, A. et al. A reconfigurable fabric for accelerating large-scale datacenter services. In *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*, pp. 13–24, June 2014. doi: 10.1109/ISCA.2014.6853195.

Zhao, R., Ng, H., Luk, W., and Niu, X. Towards efficient convolutional neural network for domain-specific applications on fpga. *arXiv:1809.03318*, Sept 2018.