AI Benchmarking on Achronix Speedster®7t FPGAs

Aiken Cairncross, Basile Henry, Chris Chalmers, Douglas Reid,
Jonny Shipton, Jon Fowler, Liz Corrigan, Mike Ashby
Myrtle.ai

***Abstract*—Deployments of machine learning networks with auto-regressive critical paths, or recurrence, often poorly utilize AI accelerator hardware. Such networks, like those used in Automatic Speech Recognition, must run with low latency and deterministic tail-latency for at-scale real time applications. In this paper we present an overlay architecture for an inference engine which is then implemented on a Speedster7t FPGA. The Speedster7t is an AI optimized device from the Achronix Semiconductor Corporation. We demonstrate potential high utilization rates for the type of network considered. Specifically, we describe a double clocking method that achieves a clock frequency at 74.7% of the rated frequency of the Machine Learning Processor blocks in the Speedster device. We show that the device can achieve 36.4 TOPS on a standard set of AI benchmarks and show that it can achieve circa 60% of the device headline efficiency in a range of scenarios. We then highlight the benefit of this architecture for low latency real time applications such as Automatic Speech Recognition.**

I. Introduction

Since 2019, a new generation of AI optimised FPGA devices have been released by major FPGA device manufacturers. The Versal family from Xilinx and the Intel Stratix 10 NX device both contain hardened compute engines specifically targeted to AI workloads. Most recently Achronix Semiconductor Corporation have launched their AI optimised Speedster7t family, also containing hardened compute engines that specifically target AI workloads. As AI becomes prevalent in all domains, the need to deploy AI on FPGAs has led to architectural innovation, focusing on adding sufficient compute to support the core matrix multiplication operations at the heart of all deep neural network processing, whilst enabling flexible use of FPGA fabric to enable the wide range of other operations required in AI processing. FPGAs have historically been used in embedded devices, consumer products, automotive and telecommunications equipment. Now, FPGAs are also being used in the data center for AI inference acceleration at scale, as demonstrated by Microsoft's Project Brainwave [1], SK Telecom [2] and Kuaishou [3]. These deployments are being used for low latency real time deployments, where the FPGA architecture supports efficient processing at low batch sizes, enabling AI to be used where latency affects user experience, such as conversational AI services. Using FPGAs for these deployments provides a cost effective alternative to AI inference ASICs, while maintaining advantages of flexible software programming that better tracks fast moving AI development.

In this paper we discuss the features of the latest AI optimised FPGA device from Achronix, the Speedster7t, and how this can be used effectively for real time applications such as Automatic Speech Recognitin (ASR). We describe an overlay architecture that can achieve highly efficient use of the FPGA and we provide a set of benchmark results for core compute metrics GEMV and MLP AI operations. Finally we compare achieved TOPS and memory bandwidth figures to the headline performance figures of the device to highlight achievable efficiency when deployed for AI workloads.

II. Speedster7t Device Features for AI

When considering an FPGA device for AI inference, key features of relevance are the numerical precision supported by the device, the amount of available compute and the specification of high speed memory interfaces. These factors combined affect the overall quality and performance that can be achieved for networks of interest. In this section we describe the features of the Speedster7t that are used for AI inference, and at the end of the section we discuss how these features are combined to determine the capability of the device for low latency AI inference.

A. Machine Learning Processor block

The Speedster7t1500 FPGA contains 2560 Machine Learning Processor blocks in the device. These hardened silicon components can run at up to 750 MHz, providing a total headline TOPS figure of 61.44 TOPS at INT8. The Machine Learning Processor contains integrated Block RAM and Multiply Accumulate circuitry. This enables maximum performance to be achieved, as all high bandwidth data flows are kept within the Machine Learning Processor block, vastly reducing the routing overhead on the FPGA when moving weights and activations into the compute elements of the device. The Machine Learning Processor block includes a cascade path between adjacent blocks to share memories and data for weights or activation data and implementation of efficient data structures such as systolic array architectures.

The architecture of the Machine Learning Processor block is shown in Figure 1.

B. Speedster7t Numerical Precision

Quantizing models for inference is a widely used technique, in which compute operations are carried out in a format which is less computationally expensive. This approach has been applied to a wide variety of models including BERT [4], ResNet and GNMT [5], and sees adoption in widely recognised machine learning benchmarks [6].

The quantization process includes one or both of:

1) Reducing the number of bits of the datatype. e.g. use 8 bits instead of 32 bits.

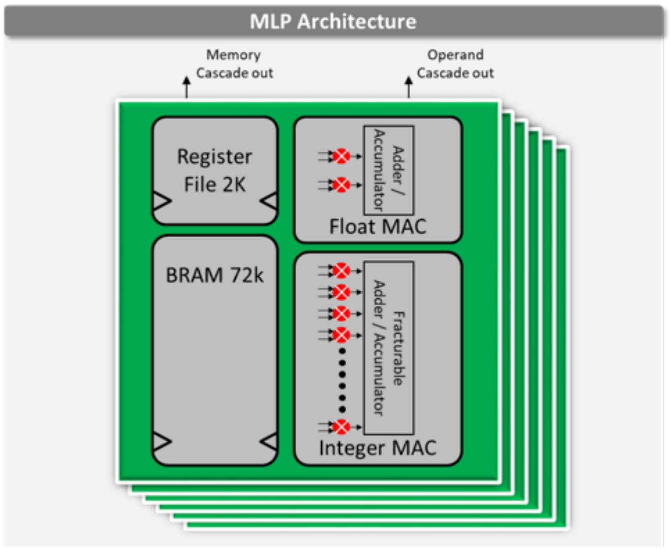

Fig. 1. MLP Architecture on Achronix Speedster7t FPGA

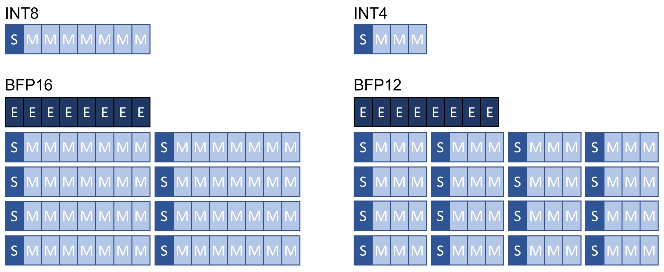

Fig. 2. Block Floating Point Quantization Format

2) Using a less expensive format. e.g. use integer instead of floating-point.

Quantizing to integers is a popular choice for AI inference. Running inference at INTX (most often INT8) is widely used for deployment including for models from the domains of Machine Translation [7], Automatic Speech Recognition [8], Computer Vision [9] and NLP embeddings [5].

Formats other than INTX and FP16 are becoming more widely used as hardware vendors accommodate them. For example, BrainFloat16 is supported by Google TPUs and Intel Xeon CPUs. Recently, Block Floating Point formats have been demonstrated by Microsoft [10], enabling networks to retain accuracy with a post training quantization flow, and more efficient hardware computation. The block floating point formats share a single exponent value across a block of mantissa values, typically 8 or 16. This scheme provides improved dynamic range over fixed-point arithmetic with accuracy approaching that of traditional floating point, but compute efficiency equivalent to integer processing. Figure 2 illustrates the Block Floating Point format.

The Achronix Speedster7t FPGA can support a range of numerical formats due to its fully fracturable integer MAC within the Machine Learning Processor blocks on the FPGA. The Machine Learning Processor supports operations in INT16, INT8 or INT4 multiplications. The addition of the floating point MAC enables the device to support floating point formats fp16, fp24 and bf16. Combining both MACs enables use of Block Floating Point formats of arbitrary block sizes, enabling the device to implement BFP16 and BFP12 formats, among others. Table I shows the achievable TOPS for supported numerical formats and the associated memory compression factor. In this paper we refer to BFP16 with a block size of 8 and BFP12 with a block size of 16.

# Format	Headline TOPS S7t1500	Memory Compression Factor (relative to fp32)
fp24	7.68	1.33
fp16	7.68	2
bf16	7.68	2
INT16	15.36	2
BFP16	61.44	3.56
INT8	61.44	4
BFP12	122.88	7.11
INT4	122.88	8

TABLE I
COMPUTE CAPABILITY OF SPEEDSTER7T1500 AT DIFFERENT NUMERICAL FORMATS

As well as improving compute efficiency, lower precision formats also reduce the memory bandwidth requirements of an inference platform, which for memory bound processing and low latency applications can lead to a linear improvement in performance.

C. GDDR6 memory

The BRAMs included in the Machine Learning Processor blocks of the Speedster7t provide a total on chip memory resource of 195 Mbits, sufficient to hold an AI model of under 24 MBytes at INT8 in persistent on chip storage. When neural network sizes exceed that threshold, model parameters need to be stored in larger external memories. In the case of the Speedster7t, the device supports external GDDR6 memory of up-to 32 GB in size. The GDDR6 memory is shown in Figure 3. This enables much larger models to be processed, with a bandwidth of 4 Tb/s available to move model parameters onto the device, to support low latency processing.

To understand the performance capability of a device for AI inference, it is important to consider both compute performance and memory performance in combination. This is particularly important for understanding performance limitations for low latency deployments or for where networks are memory intensive. Some examples include real time streaming ASR and transformer model architectures. CNNs typically require low memory bandwidth, even at small batch sizes and so inference solutions for CNNs are not optimized for high bandwidth memory.

D. Hardened Network On Chip

Having a large external memory bandwidth is only useful if model parameters can be moved into the compute engines on the device without losing efficiency. The Speedster7t contains a hardened 2D Network on chip (2D-NoC) capable of transporting 20 Tb/s across the FPGA. This enables data to be

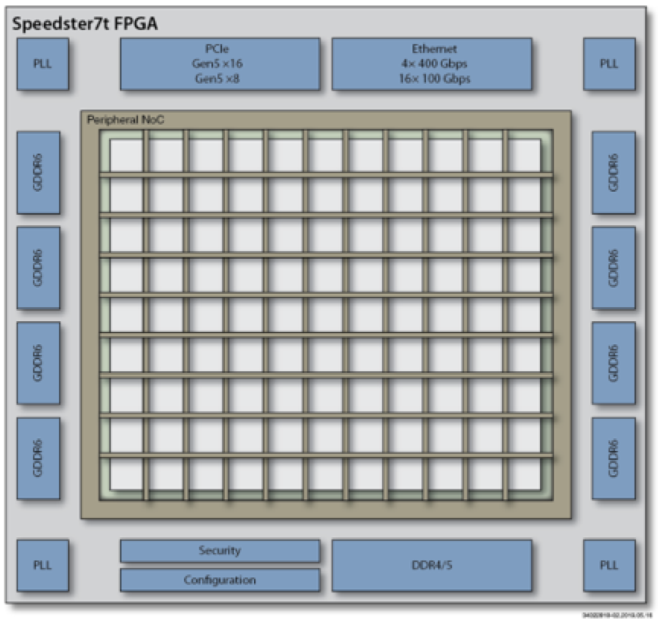

Fig. 3. External interfaces of Achronix Speedster7t FPGA

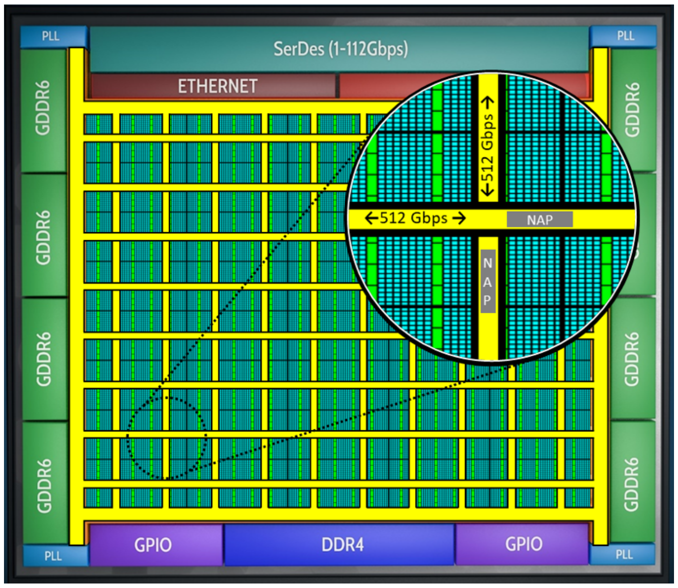

Fig. 4. 2D-NoC in Achronix Speedster7t FPGA

moved from external memory into compute engines and across the fabric without limiting compute access to external memory bandwidth. Moreover this 2D-NoC reduces the need to use fabric routing for the purposes of data movement, leaving routing resource free in the fabric and enabling better timing results.

The 2D-NoC is shown in Figure 4.

III. An AI Overlay Architecture for the Speedster7t

In order to benchmark the Speedster7t family, we create an AI inference overlay design for the Speedster7t1500 device

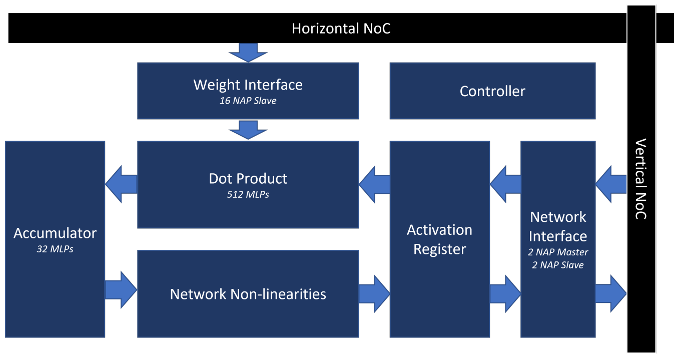

Fig. 5. The MAU Core Architecture on Achronix Speedster7t FPGA

running on a Bittware S7t-VG6 VectorPath PCIe Accelerator Card. The overlay is built using the Myrtle.ai programmable MAU Accelerator® architecture. The MAU Core is a programmable processing engine for deep neural networks, that overlays the FPGA fabric to provide a flexible and run time configurable inference engine. We place 4 MAU Cores, optimized for the Achronix Speedster7t1500 FPGA onto the card and use this to demonstrate achieved utilization and timing results.

The design uses 4 MAU Accelerator® Cores, with each core containing 512 Machine Learning Processor (MLP) blocks to form the central dot product circuit required in all machine learning inference. The full design uses 80% of all available MLPs for dot product compute, with the remaining 20% used either for additional compute operations and non linearities, or as block rams only.

The design is implemented using BFP16 format, as defined in Figure 2. This uses a mantissa size of 8 bits and an 8 bit exponent with a block size of 8. This gives good model accuracy and can be applied to models after training, without loss of network accuracy, simplifying the user flow of the architecture overlay.

We floorplan the MAU Accelerator® Cores in the Achronix ACE® software to ensure that a high level of logic utilization can be achieved, whilst retaining a high clock frequency.

The overlay architecture has a headline TOPS capability of 36.4 TOPS, which is 59.2% of the Headline INT8 compute capability of the FPGA fabric. This high efficiency figure is enabled by use of a double clocking scheme to enable the MLP blocks to run at 560MHz, 74.7% of the 750MHz rated Fmax of these components, while implementing all logic in fabric at 280MHz.

The architecture of the Achronix Speedster7t FPGA optimized MAU Accelerator® Core is shown in Figure 5. This has all the required functionality to implement AI benchmarks for GEMV and Multi-layer Perceptron (MLP) operations.

A. Double clocking scheme for Machine Learning Processors

A double clocking scheme is used to clock the Machine Learning Processor blocks in the FPGA. This allows the hardened silicon components to run at a higher clock frequency than the fabric, achieving an operating point close to the design

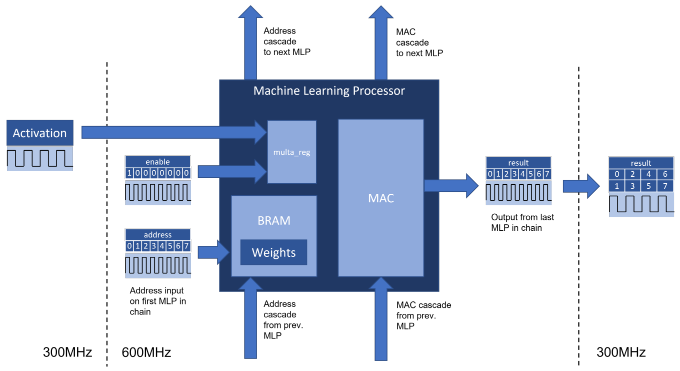
Address cascade to next MLP
MAC cascade to next MLP
Machine Learning Processor
Activation
enable
1 0 0 0 0 0 0 0
multa_reg
MAC
BRAM
Weights
address
0 1 2 3 4 5 6 7
Address input on first MLP in chain
Address cascade from prev. MLP
MAC cascade from prev. MLP
result
0 1 2 3 4 5 6 7
Output from last MLP in chain
result
0 2 4 6
1 3 5 7
300MHz
600MHz
300MHz

Fig. 6. Double Clocking Scheme for MAU Core Architecture on Achronix Speedster7t FPGA

specification of these blocks. This is possible due to the tight coupling of BRAM and MAC elements within the Machine Learning Processor block, that enables data transfer of weights into the MAC via dedicated routing within the block. This dedicated routing between BRAM and MAC for transferring weights in the Machine Learning Processor carries 177 Tb/s, 16 times that on the activation input.

The dot product circuit at the centre of the MAU Accelerator® Core multiplies together a 256x256 matrix and a 256 vector. The matrix is typically used for weights and the vector for the activations. The activation vector is held constant over 8 cycles on the 560 MHz clock domain, while the weights are read from the BRAM that is tightly-coupled to the MAC and change every cycle. The Machine Learning Processor blocks are arranged in 16 columns of 32, cascaded so that each column computes two BFP16 dot products of size 256.

The activation vector is distributed to the Machine Learning Processor blocks by the activation fanout component, which also handles delaying the activations to align with the partial sums travelling up the cascade between Machine Learning Processor blocks. The indices for the weights to be read from the BRAMs are fed at the bottom of the column, and the dot product results are output at the top of the column. The output is on the 560 MHz domain, with values on adjacent clock cycles concatenated and transferred to the 280 MHz domain by a logic deserializer.

The Machine Learning Processor blocks are floorplanned to two outer columns in each core, reserving the central column for use by other elements of the design and network specific operations. This can be seen in the layout in Figure 7.

B. Use of 2D Network on Chip

The 2D Network on Chip (2D-NoC) is used to move data around the design, reducing the need for fabric routing resource to be used for data movement. The 2D-NoC is used to move inference data between CPU and FPGA, via the PCIe interface; to move weights between GDDR6 and MAU Accelerator® Cores at runtime, enabling large networks to be stored in off chip memory; and to move inference data between MAU Accelerator® Cores on the chip, enabling data to passed

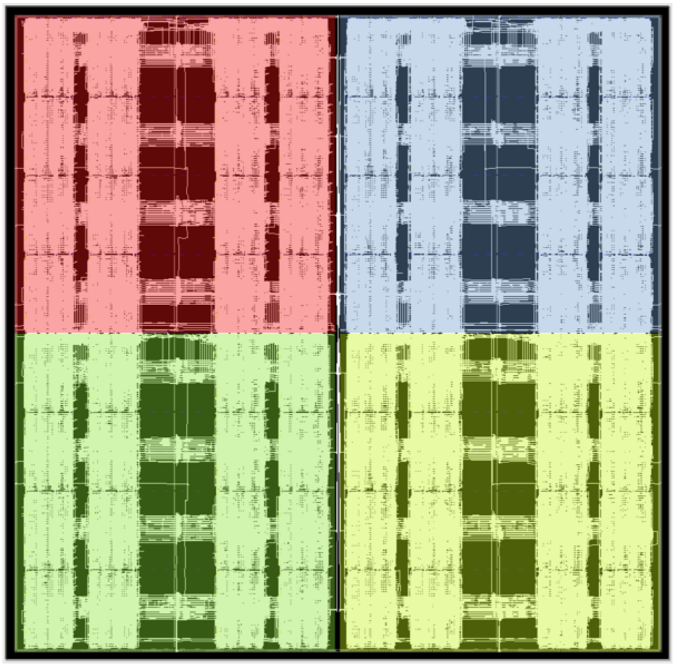

Fig. 7. Floorplan of dot product circuit on Achronix Speedster7t FPGA

between cores implementing different layer operations, or splitting matrix operations across cores.

Each core has 16 Network Access Points (NAP) for loading weights from GDDR6, these provide up to 1.12 Tb/s at 280 MHz, which is greater than the available 1 Tb/s available memory bandwidth allocated to each core. This ensures that maximum performance can be achieved for memory bound networks and low latency operating scenarios. Each core has 16 NAP Slave connections spread over 4 hNoC rows.

For host-to-core and core-to-core data transfers, the design has 2 NAP Master and 2 NAP Slave connections per core. This provides a data bandwidth of 143.36 Gbps between cores. This is sufficient for most operations to be transferred between cores, with the exception of very small matrix computations.

# Resource	Utilization	Percentage Utilization
MLP	2,400	93.8%
BRAM	2,368	92.5%
LUT	57,510	8.3%
DFF	182,887	13.2%
NAP Master	8	10.0%
NAP Slave	72	90.0%

TABLE II
RESOURCE UTILIZATION OF MAU ACCELERATOR® CORE OVERLAY

C. Device Utilization

The utilization of the device for the accelerator overlay is as shown in Table II. This shows that a very high utilization of Machine Learning Processor blocks and BRAM is used to achieve high AI computation, while only requiring a low resource use of LUTs and DFFs in the fabric. This allows room for additional functions to be implemented in fabric,

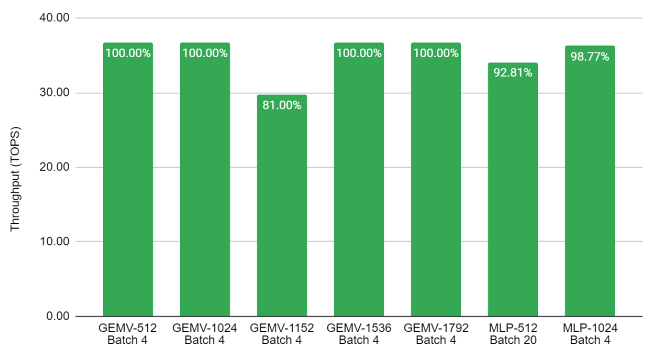

Fig. 8. Throughput of key AI Benchmarks on Achronix S7t-VG6 VectorPath Accelerator Card. Results were measured on a C2 speed-grade device at 560 MHz. Labels show utilization as a percentage of the headline performance of the MAU Accelerator®.

and it reduces the routing congestion in the fabric, leading to higher achievable clock frequencies in the fabric.

IV. AI Benchmarking Results

A. Methodology

We benchmark two simple operations on the Speedster7t to give an indication of what performance can be achieved for AI networks. We implement a GEMV benchmark and a Multi-layer Perceptron (MLP) benchmark. The GEMV demonstrates performance for operations that are central to all AI benchmarks.

The GEMV-N benchmark computes $\mathbf{Ax}+\mathbf{y}$, where $\mathbf{A}$ is a square matrix of dimension N, and $\mathbf{x}$ and $\mathbf{y}$ are N-vectors. The MLP-N benchmark computes a 5-layer Multi-layer Perceptron, where each layer is defined as $\text{layer}_i(\mathbf{x_i}) = \mathbf{W_i}\mathbf{x_i}+\mathbf{b_i}$ where $\mathbf{W_i}$ is a $N \times N$ square matrix. Each layer feeds into the following, $\mathbf{x_{i+1}} = \text{layer}_i(\mathbf{x_i})$, and $\mathbf{x_0}$ is the input.

We use hardware cycle counters to measure time to compute a particular benchmark and infer throughput from steady-state measurements. We do not time the data transfer between CPU and FPGA, as this would dominate the processing time for these very small benchmark operations. All benchmarks are run with constants persisted in BRAMs after initially being loaded from external GDDR6 memory.

B. Compute Performance

Figure 8 shows high performance across various sizes of matrix for both GEMV and MLP. Benchmarks are parallelized across four cores, so Batch 4 means one inference running independently on each core. For Batch 20 results, each core is running 5 pipelined inferences. The overlay architecture is used to 100% efficiency for GEMV benchmarks, where the matrix is divisible by 512.

C. Achieved Roofline Performance

Considering an acceleration platform for low latency and real time applications, Figure 9 shows the achieved roofline performance for the Achronix Speedster7t1500 device. Networks with an arithmetic intensity of over 70 operations/byte can

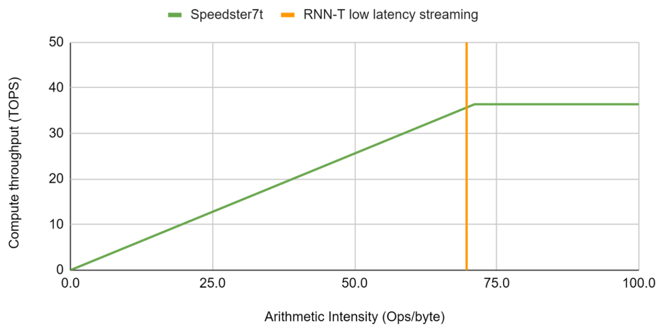

Fig. 9. Achronix Roofline Performance and RNN-T operating point, using headline memory bandwidth

be implemented at 36.4 TOPS on the device. The RNN-T operating point is used as an example to highlight processing needs of an accelerator processing the MLPerf RNN-T reference model in a streaming deployment at batch size 8 and 80ms chunk size.

V. Conclusion

This paper demonstrates the new AI optimised FPGA from Achronix, the Speedster7t1500, is able to achieve 59.2% efficiency and 36.4 TOPS when applied to key AI benchmarks. The AI optimized architecture enables a high compute clock frequency, and the high external memory bandwidth positions this device well for low-latency workloads.

References

[1] J. Fowers, K. Ovtcharov, M. Papamichael, T. Massengill, M. Liu, D. Lo, S. Alkalay, M. Haselman, L. Adams, M. Ghandi, S. Heil, P. Patel, A. Sapek, G. Weisz, L. Woods, S. Lanka, S. K. Reinhardt, A. M. Caulfield, E. S. Chung, and D. Burger, "A configurable cloud-scale dnn processor for real-time ai," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, 2018, pp. 1–14.

[2] Xilinx, "Xilinx alveo accelerators power sk telecom's real-time ai-based physical intrusion and theft detection service," https://www.xilinx.com/news/media-kits/sk-telecom-deploys-xilinx-fpgas-for-ai-acceleration-achieves-5x-performance-16x-performance-html, 2019.

[3] ——, "Kuaishou provides asr services for large-scale, live-stream broadcast and video apps with xilinx," https://www.xilinx.com/content/dam/xilinx/publications/powered-by-xilinx/kuaishou-case-study.pdf, 2021.

[4] H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius, "Integer quantization for deep learning inference: Principles and empirical evaluation," *arXiv preprint arXiv:2004.09602*, 2020.

[5] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat, "Q8bert: Quantized 8bit bert," in *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*. IEEE, 2019, pp. 36–39.

[6] V. J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, C.-J. Wu, B. Anderson, M. Breughe, M. Charlebois, W. Chou *et al.*, "Mlperf inference benchmark," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020, pp. 446–459.

[7] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[8] Y. He, T. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. Sim, T. Bagby, S.-y. Chang, K. Rao, and A. Gruenstein, "Streaming end-to-end speech recognition for mobile devices," 05 2019, pp. 6381–6385.

[9] S. Wu, G. Li, F. Chen, and L. Shi, "Training and inference with integers in deep neural networks," *arXiv preprint arXiv:1802.04680*, 2018.

[10] B. Darvish Rouhani, D. Lo, R. Zhao, M. Liu, J. Fowers, K. Ovtcharov, A. Vinogradsky, S. Massengill, L. Yang, R. Bittner, A. Forin, H. Zhu, T. Na, P. Patel, S. Che, L. Chand Koppaka, X. SONG, S. Som, K. Das, S. T, S. Reinhardt, S. Lanka, E. Chung, and D. Burger, "Pushing the limits of narrow precision inferencing at cloud scale with microsoft floating point," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 10 271–10 281. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/747e32ab0fea7fbd2ad9ec03daa3f840-Paper.pdf