EUROPEAN WORKSHOP ON ON-BOARD DATA PROCESING (OBDP2021), 14-17 JUNE 2021

USING THE VECTORBLOX™ ACCELERATOR SOFTWARE DEVELOPMENT KIT TO CREATE PROGRAMMABLE AI/ML APPLICATIONS IN RADIATION TOLERANT (RT) POLARFIRE® FPGAs

Aaron Severance [(1)], Diptesh Nandi [(1)], Ken O'Neill [(1)]

[(1)]Microchip Technology Inc, 2355 W Chandler Blvd, Chandler AZ, 85224, USA
Aaron.Severance@Microchip.com
Diptesh.Nandi@Microchip.com
Ken.ONeill@Microchip.com

ABSTRACT

In this paper, we describe the VectorBlox Accelerator software development kit (SDK) and how it is used to optimize and convert trained Artificial Intelligence (AI) models, targeting power-optimized 28 nm field programmable gate arrays (FPGAs). Neural networks are sourced from a variety of supported input frameworks, such as TensorFlow, Caffe, ONNX, and PyTorch. The VectorBlox Accelerator SDK performs a three-step conversion flow to optimize the networks, calibrate and scale them to 8-bit representation, and finally create an image for implementation on the radiation tolerant (RT) PolarFire FPGA.

Power-efficient implementation of the neural network on the FPGA is achieved by a soft IP core called CoreVectorBlox. This soft IP core comprises a RISC-V processor and firmware, a vector processor, and a convolutional neural network accelerator, which consists of a two-dimensional array of processing elements, making use of the multiply-accumulate blocks in the RT PolarFire FPGA.

By implementing neural networks in a matrix processor programmed in the fabric of the radiation-tolerant RT PolarFire FPGA, the networks can be iterated and changed without resynthesizing the FPGA, resulting in convenient, programmable low-power AI applications that can be dynamically changed at runtime.

Examples of performance and utilization of a variety of neural networks sourced from TensorFlow, Caffe, ONNX and PyTorch will be provided, for implementations both with and without triple module redundancy which may be desired for radiation mitigation purposes.

The radiation-tolerant RT PolarFire FPGA will be described, with emphasis on radiation test data and schedules for qualification and flight models.

1. BACKGROUND – AI IN SPACE

The challenge faced by designers of space vehicle payload instruments is one of increasing demand for information gathered on orbit. Sensor resolutions and frame rates are increasing, and now satellite payload instruments are generating data at hundreds of giga bits per second. Downlink bandwidth is not increasing as quickly, which means that sensor data must be processed into useful information on orbit, in the payload instruments. This creates demand for ever larger, faster, and more feature-rich FPGAs. It also gives rise to an emerging requirement for sophisticated AI and machine learning (ML) capabilities to enable efficient processing of payload data, and for autonomous decision making on orbit.

2. AI/ML TRENDS AND RT POLARFIRE FPGA TECHNOLOGY

Artificial Intelligence and Machine Learning (AI/ML) are making their way into embedded devices to enable smart applications in equipment where power, footprint, and thermal constraints have historically limited their adoption. AI/ML is particularly good at recognition in the audio and video domain, allowing sensors and devices to make decisions and react to the environment around them. Smart embedded vision applications are currently dominated by convolutional neural networks (CNNs), which pass an input through a cascade of convolutional layers to classify an image or find and identify regions of interest. Neural networks typically are created using 32-bit floating point arithmetic operators but are tolerant to minor deviations and can be deployed using 8-bit integer math with minimal loss of precision.

Processing for neural networks is split up among training and inference. Training is the process of learning from large datasets and is done offline before deploying an AI solution in a remote equipment, such as in a space vehicle. Inference takes an existing trained

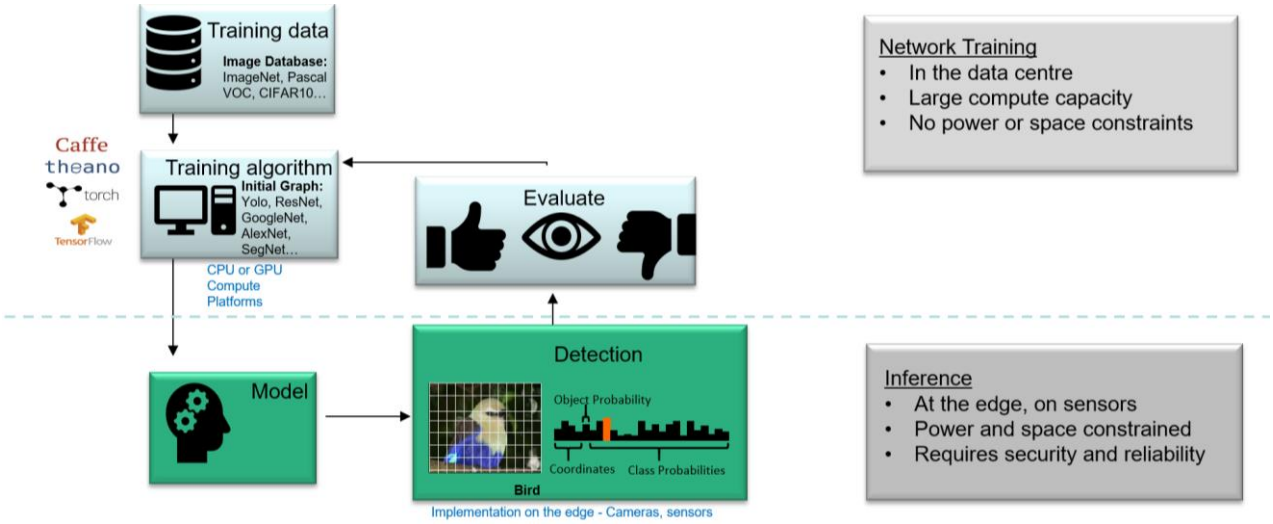

Figure 1: Neural Networking at the Edge

model and an input source to produce a result such as a classification score. Figure 1 illustrates the process. The VectorBlox Neural Network IP and SDK allow for trained networks to be run on RT PolarFire FPGAs in a flexible, power-efficient manner.

RT PolarFire FPGAs are designed for low power and high flexibility, making them ideal for integration in equipment performing neural network inferencing. FPGAs give an integration advantage over standalone AI/ML solutions as they can integrate multiple functions such as sensor interfacing and network control. This can reduce the total bill of materials, footprint, power consumption and thermal output of an instrument. RT PolarFire FPGAs are particularly well suited for low power remote sensing applications. Compared to competing devices they use non-volatile configuration memory instead of SRAM-based configuration memory, which lowers their static power consumption in addition to preventing radiation-induced configuration upsets. They also support 8-bit multiply-add operations in their math blocks making them efficient for neural network inference.

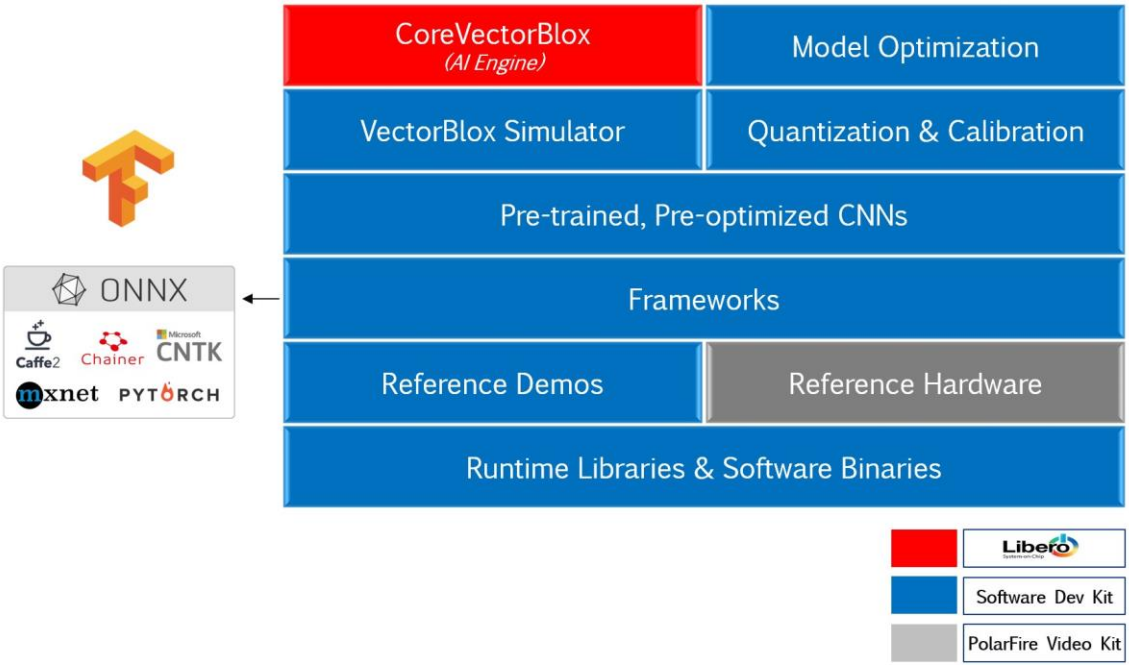

Figure 2: VectorBlox AI Solution

3. VECTORBLOX AI SOLUTION [1]

The VectorBlox AI Solution from Microchip, shown in Figure 2, is a combination of the CoreVectorBlox Neural Network IP core which runs on PolarFire and RT PolarFire FPGAs, and the VectorBlox Accelerator SDK which converts trained models to 8-bit integer models that the CoreVectorBlox IP core can process. The VectorBlox Accelerator SDK also includes a bit-accurate simulator that runs on most x86 PCs to allow for early testing of quantized model accuracy and early application integration. There is a reference hardware platform using the PolarFire Video Kit which uses a video camera as input and outputs to an HDMI display.

4. OVERLAY ARCHITECTURE AND NETWORK SWITCHING

One key feature of the VectorBlox AI Solution is that it uses an overlay architecture on top of the PolarFire or RT PolarFire FPGA. An overlay is not designed for a specific network but rather can run different networks with the same FPGA configuration. This means that when iterating on network architectures and retraining networks, the FPGA design does not need to be resynthesized. Instead, the VectorBlox Accelerator SDK creates a new model file that will run on the same FPGA configuration once the model is loaded onto memory attached to the FPGA.

This also means that multiple networks can be supported with the same FPGA configuration at runtime, enabling applications to switch between networks being run dynamically. For instance, an application could use a fast, low-accuracy network for initial detection and then upon finding something of interest run a higher-accuracy network that has fewer false positives. Or, an application can find regions of interest in an image and then run another algorithm on those regions.

5. SUPPORTED INPUT FRAMEWORKS AND NETWORK LAYER TYPES

The VectorBlox Accelerator SDK is designed to take in networks from multiple input frameworks and produce a common output representation. Networks in TensorFlow, Caffe, ONNX, Keras, OpenVINO, and Darknet form can be directly input into the SDK flow. The CoreVectorBlox IP core overlay architecture can support a variety of network architectures and layer types as it has at its core a flexible vector processor, which will be explained in more detail later. This also means that new network layer types can be added via updates to the VectorBlox Accelerator SDK. The layer types currently supported are shown in Table 1.

Table 1: Layers Supported by the VectorBlox Accelerator SDK

Layer Name	Notes
Add	
AvgPool	
Clamp	
Concat	
Constant	
Convolution	
Gather	Only Supported if output is constant
GroupConvolution	
Interpolate	
LRN	Very slow implementation.
MatMul	
MaxPool	
Multiply	Only Supported if multiplying each channel by a scalar
PReLU	
Pad	Only constant padding is supported
Parameter	
RegionYolo	Removed from graph. Must be handled with post processing
ReLU	
ReorgYolo	
Reshape	
Result	
SoftMax	Only supported if output layer
ShapeOf	
Squeeze	
TopK	Only Supported on axis==1, mode ==max
Transpose	Supported if node can be replaced by a reshape or order parameter is equal to (0,2,3,1)
Unsqueeze	

6. CORE VECTORBLOX NEURAL NETWORK IP

The CoreVectorBlox Neural Network IP core is an FPGA overlay consisting of a RISC-V scalar microcontroller, VectorBlox MXP vector processor, and a 2D grid of processing elements for processing convolutional and dense layers. The core has an AXI4 memory-mapped host port for reading and writing data from external memory, and an AXI4-lite client port for an external host to control network processing. It can be configured in multiple size configurations to trade off FPGA resource utilization for performance.

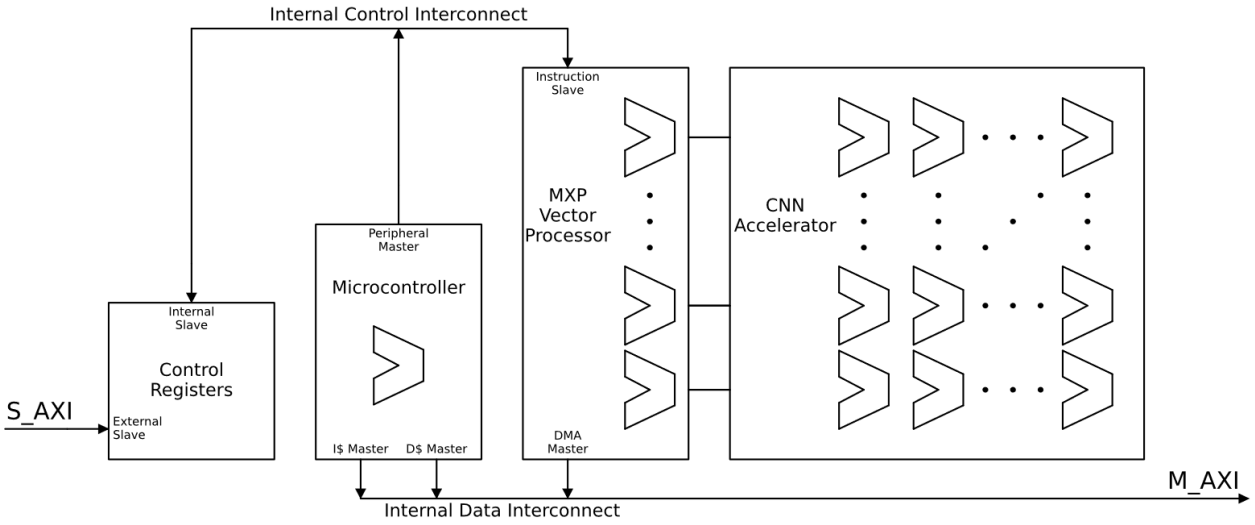

Figure 3: VectorBlox™IP Architecture

7. ARCHITECTURE

Figure 3 gives an overview of the architecture of the CoreVectorBlox IP. Incoming control signals from an external host set control registers which describe the addresses of network inputs, outputs, parameters and weights (bundled together into a Binary Large OBject or BLOB) and can start processing networks and check status. The microcontroller is a RISC-V soft processor that communicates with the control registers and can issue instructions to the MXP vector processor as well as do lightweight and irregular computation. It runs firmware that is distributed as a BLOB and therefore can be field upgraded; if a new network layer type is needed that is not supported, the VectorBlox SDK can be upgraded to support it and a new firmware BLOB can be downloaded to the FPGA-attached memory without needing to resynthesize a new FPGA design.

The MXP is a vector processor with a scratchpad architecture. It runs under control of the RISC-V microcontroller, either directly or by replaying instruction traces that are recorded when a network is first run. The MXP vector instructions are data parallel, performing the same operation on many elements within a vector, and can be of a variable length up to the entire size of the MXP scratchpad memory. Vector instructions are executed on multiple parallel arithmetic and logic units (ALUs); the datapath width depends on which size configuration of CoreVectorBlox IP is used. Vector operations may take multiple cycles to complete if they are longer than the datapath is wide; wider datapaths will process instructions more quickly. The MXP has a direct memory access (DMA) engine for loading and storing data from/to external memory, which can bring in large amounts of data from memory with single instructions. Striding is supported for bringing in partial maps when full maps will not fit in the scratchpad memory. DMA and regular operations can proceed in parallel, and memory accesses are pipelined so that a new set of activations and weights will be brought in parallel to processing the current set of activations and weights.

The CNN accelerator is a two-dimensional grid of processing elements (PEs) which have a multiply-accumulate unit and a small local RAM to store partial sums. It can operate in a pointwise mode and a depthwise mode. In the pointwise mode the two dimensions correspond to parallel output maps being worked on and parallel elements within an output map, respectively. A 32x32 array of PEs can work on 32 elements of 32 output maps simultaneously. Additionally, the local RAM can be used to switch between more output elements and/or maps, allowing for greater data reuse of input data. Data comes in from the MXP as two inputs, weights and activations, which get broadcast along the two axes to the PEs. The PEs multiply weights by activations and accumulate them into their local RAM. When computation is complete, data goes into a shift register and the output maps are read out sequentially. The depthwise mode is similar to pointwise mode, but because depthwise convolution does not operate across multiple input maps the weights come in from a separate small RAM and the first dimension is parallel kernel elements rather than parallel input maps.

The CoreVectorBlox IP has two clocks; a base clock which runs at ~150 MHz and a datapath clock that runs at twice the base clock rate. External interfaces are clocked by the base clock. The RISC-V microcontroller

and interface logic run at the base clock rate, as well as the MXP decode logic, as these are complex pipelines with feedback loops. The MXP scratchpad memory, execution units, and the CNN accelerator all operate at the 2x datapath clock for improved performance per area, as they are all feed-forward pipelines that are easy to extend for higher frequencies. To match the decode logic to the datapath logic the MXP datapath and CNN accelerator appear to be a virtual width that is twice the actual physical datapath width. For example, if the datapath width is virtually 256-bits wide, it is physically 128-bits but processes a contiguous 256-bits over two consecutive datapath cycles. The two clocks are synchronous to each other, but the 2x clock uses negative-edge flip flops (vs. the positive-edge flip flops of the base clock) to ease hold timing when transferring data between the clocks.

Table 2: VectorBlox™ IP Configurations

Configuration	Vector Processor Width	Vector Scratchpad	CNN Accelerator Array Size
V250	128-bit	64 kB	16x16
V500	256-bit	128 kB	16x32
V1000	256-bit	256 kB	32x32

CoreVectorBlox IP comes in three configurations: V250, V500 and V1000. Table 2 lists how the MXP vector processor width, MXP scratchpad memory size and CNN accelerator size vary between the three configurations. A table with look-up-table (LUT) counts can be found in the results section.

8. VECTORBLOX ACCELERATOR SDK

The VectorBlox Accelerator SDK takes a user network and converts it to a BLOB that can run on the CoreVectorBlox IP. Additionally, it provides a bit-accurate functional simulator that runs on most x86 PCs and can be used to evaluate the accuracy of the converted network and allow for early application development without needing an FPGA. The SDK is created using Python® and can run under Windows® or Linux®.

9. NEURAL NETWORK CONVERSION FLOW

The conversion flow consists of three steps: 1) intermediate representation (IR) conversion and general optimizations; 2) calibration; and, 3) 8-bit integer BLOB generation. The conversion flow is shown in Figure 4.

The IR conversion and optimization stage takes in a network from supported input frameworks (TensorFlow, Caffe, etc.) and converts it to a common IR using the open source OpenVINO tools. At this stage, generic optimizations for inference are performed such as folding batch normalization into other layers. The calibration stage finds the scaling factors needed to convert a 32-bit floating-point model to 8-bit integer math to balance the amount of saturation that occurs with results that are too big for the 8-bit representation against the loss of precision that occurs when weights are too small and important information is lost. Input data is used to find the minimum and maximum values seen during processing, as well as to perform bias correction to set the mean value seen across maps in the quantized model as close as possible to that of the original floating-point model. Biases are applied on a per-layer basis, while scaling factors are applied per-map.

The 8-bit integer BLOB generation takes the input model and calibration information and produces a memory image that can be run on the x86 simulator or on the programmed FPGA. The BLOB consists of hyperparameters (layers and sublayers, where layers are major operations such as convolution and sublayers are minor operations that can be grouped together such as activation functions) and weights. The hyperparameters also describe how to schedule the network on the hardware; for instance, layers which do not fit entirely into the CoreVectorBlox scratchpad are brought into the accelerator in chunks and processed piecemeal. This schedule will be different for the different size configurations (V250, V500 and V1000) though the results will be bit identical.

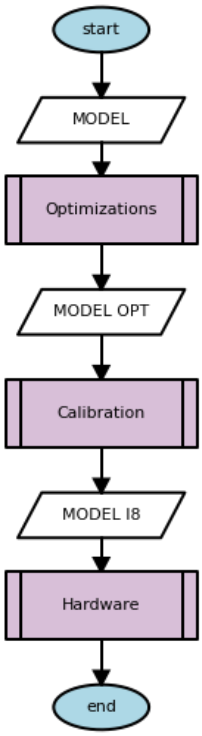

Figure 4: VectorBlox SDK Conversion Flow

10. SIMULATOR

The bit-accurate simulator is a functional C model of the MXP vector processor and CNN accelerator array. It can be called as a separate program or inserted into a user's program with either a C or Python interface.

The simulator works by emulating the MXP at an instruction level. It parses the network BLOB using the same firmware that runs on the FPGA implementation (though compiled for x86 instead of RISC-V). The simulator models the CoreVectorBlox IP internal state including the MXP scratchpad memory and CNN accelerator accumulators but does not model the MXP's instruction pipeline, hazards or memory unit. Instructions are executed using native host instructions wherever possible; for instance, a vector addition becomes a for-loop of scalar additions on the x86 host (which can even be compiled into SIMD instructions in some cases).

11. RESULTS

Table 3 shows the conversion accuracy of the 8-bit model quantized by the VectorBlox Accelerator SDK relative to the original floating-point models. The first column is a simple 8-bit quantization based on the minimum and maximum values observed. The second column is the accuracy achieved in hardware using the output of the VectorBlox Accelerator SDK (including bias correction) and the final column is the reference 32-bit floating point accuracy. For image classification top1 results are used for accuracy; for object detection networks 11-point mAP accuracy scores are used. The VectorBlox Accelerator SDK achieves state-of-the art conversion accuracy from floating-point to 8-bit integer, with typical accuracy loss under 1% from the reference model. Table 4 gives results area and performance across the three size configurations of the CoreVectorBlox IP.

Table 3: VectorBlox IP Conversion Accuracy

Input Framework	Model	Accuracy		
		Simple 8-bit	VectorBlox SDK 8-bit	Floating Point 32-bit
Caffe	Squeezenet 1.1	58.8	58.6	59.2
TensorFlow	Mobilenet v1	68.8	71.2	71.6
TensorFlow	Mobilenet v2	69.0	71.6	72.0
ONNX	Resnet18 v1	71.4	72.8	72.8
PyTorch	Resnet50	75.2	75.0	75.0
Darknet	TinyYOLO v2 VOC	54.2	54.4	55.1
Darknet	TinyYOLO v3 CPC	39.5	40.4	40.9

The size numbers are listed in thousands of look-up tables (kLUTs) as well as the percentage of LUTs available on the RT PolarFire RTPF500T FPGA. The maximum frequency (Fmax) for each configuration along with the size of the CNN accelerator array determines the peak giga operations per second (GOPs) that the configuration can deliver. Fmax is listed for the base clock; the MXP datapath and CNN accelerator array operate at 2x the base clock internally. Power numbers are listed in mW/GOP for relative comparison; this number is the total power determined as the dynamic power used by the CoreVectorBlox IP along with the static power used by the portion of the FPGA occupied by the CoreVectorBlox IP. Performance is listed for two popular computer vision networks, Mobilenet-v1 and TinyYOLO-v3 in frames per second (FPS).

There is a small frequency degradation going from the V250 up to the V1000, mainly due to the difficulty of placing the CNN accelerator array and routing signals across it as its size increases. Power efficiency increases at larger size configurations as the fixed power costs of the control and interface logic are amortized over more GOPs from the accelerator array. Performance scales well per GOP, close to linearly for TinyYOLO-v3. For Mobilenet-v1 there is close to linear performance per GOP scaling from V250 to V500, with a smaller increase from V500 to V1000 since the depthwise layers used in Mobilenet do not increase in performance with CNN accelerator array depth.

Table 4: VectorBlox™ IP Performance, Without Triple Module Redundancy (TMR)

CoreVectorBlox Configuration	Size		Fmax (base) MHz	Peak GOPs	mW/GOP	Performance (FPS)	
	kLUTs	% of RTPF500T				Mobilenet v1	TinyYOLO v3
V250	28	6%	154	79	7.0	26.2	9.1
V500	48	10%	143	146	6.4	47.7	16.6
V1000	63	13%	136	279	5.1	68.0	26.5

Table 5: VectorBlox IP Performance, With Synthesized Local TMR (Preliminary Results)

CoreVectorBlox Configuration	Size			Fmax (base) MHz	Peak GOPs	mW/GOP	Performance (FPS)	
	kLUTs	kDFF	% of RTPF500T				Mobilenet v1	TinyYOLO v3
V250	56	66	14%	102	52	-	17.4	6.0
V500	97	116	24%	86	88	-	28.7	10.0
V1000	136	161	33%	76	156	-	38.0	14.8

Table 5 shows some preliminary utilization and performance data with synthesized local triple module redundancy (TMR) applied to the entire CoreVectorBlox IP core. In addition to number of kLUTs, the table includes an assessment of the number of thousands of d-type flip-flops (kDFF) that are consumed in the TMR implementation. As expected, the number of flip-flops consumed exceeds the number of LUTs consumed, and there is a decrease in performance of around 35% to 45%. This is observable in the decreased peak GOP rate and the lower FPS rates compared to the non-TMR implementations in Table 4. Power has not yet been measured for the TMR implementation. Note that no radiation testing has yet been performed on the CoreVectorBlox IP, either with or without TMR, so it is not clear whether TMR gives any significant improvement in operation of the IP in a radiation environment.

Table 6 shows power consumed in PolarFire FPGAs for three different implementations of the CoreVectorBlox IP. We have the dynamic power consumed by the IP running at peak GOPs, the static power consumed by the portion of the FPGA where the IP resides, and the total power. The final column in the table shows the total power, normalized to mW per GOP.

Table 6: VectorBlox IP Power Consumption Breakout

CoreVectorBlox Configuration	Peak GOPs	Dynamic Power (mW)	Static Power (mW)	Total Power (mW)	Total Power (mW/GOP)
V250	79	387	65	452	7.1
V500	146	698	127	825	6.4
V1000	279	1094	206	1300	5.1

12. RT POLARFIRE FPGA [2]

RT PolarFire is Microchip's latest radiation tolerant FPGA. It offers a substantial increase in density and performance, relative to pre-existing radiation tolerant FPGAs, to address the growing need for computational throughput on orbit. In common with all Microchip radiation tolerant FPGAs, it exhibits a complete absence of radiation-induced configuration upsets, permitting operation without the need to monitor, repair or reload the FPGA configuration in space. The power consumption of RT PolarFire is significantly lower than any other FPGA at its density level. This enables use of power supply components with lower cost and smaller footprint than would be the case with an FPGA with higher power consumption. It also results in considerable savings due to reduction or elimination of costly thermal management solutions designed to dissipate excess heat in sensitive satellite payload instruments. Finally, RT PolarFire is integrated into a hermetically sealed ceramic package which enables qualification to Qualified Manufacturers List (QML) class V, as required by the most demanding space programs. The main features of RT PolarFire are listed in Table 7.

Table 7: Features of RT PolarFire FPGAs

RT PolarFire FPGA	RTPF500T
DFF (TMR)	0
DFF (Non-TMR)	481K
4-Input Look-Up Tables (LUTs)	481K
Mathblocks (18x18 MACC)	1,480
Total RAM	33 Mbits
uPROM	513 Kbits
Serdes Transceivers (250 Mbps – 10 Gbps)	24
I/O (HSIO / GPIO)	584 (324 / 260)
On-Orbit Reprogramming	Supported 500 cycles max.
Package	1509 Ceramic Column Grid Array
Qualification (Planned)	Mil Std 883 QML Class Q QML Class V

In total ionizing dose radiation effects testing, RT PolarFire FPGAs have shown minimal degradation in performance at 100 kRAD, and an increase in leakage current of 10% to 15%. Single event effects (SEE) have been measured in proton and heavy ion radiation. Radiation test results are summarized in Table 8. Additional radiation testing is planned in 2021 and 2022.

Engineering models of RT PolarFire FPGAs are available at the time of writing. QML qualification is in progress, with completion of QML class Q qualification anticipated in 2022 and QML class V in 2023.

Table 8: RT PolarFire Radiation Characteristics

Total Ionizing Dose (TID)	100kRAD
Configuration Upsets	Absent Tested to > 80 MeV-cm2/mg
Single Event Latch-Up	LET_{TH} 80 MeV-cm2/mg with GPIO operating at 1.8V
Single Event Upsets (Unprotected DFF)	1E-7 errors/bit-day, GEO SolarMin
Single Event Upsets (DFF With Local TMR)	1E-11 errors/bit-day, GEO SolarMin
On-Orbit Reprogramming	Supported 500 cycles max.

13. CONCLUSION

The VectorBlox Accelerator SDK and CoreVectorBlox IP provide an easy-to-use and flexible way to implement neural networks on power-efficient RT PolarFire FPGAs. The overlay architecture allows for iteration on and changing of neural networks without resynthesizing FPGA designs, including dynamically changing networks at runtime. The core uses standard AXI interfaces making it easy to add to existing designs. The VectorBlox Accelerator SDK can take in user networks from a variety of network frameworks and quantize to 8-bit integer math with very little loss in accuracy. Together, the VectorBlox AI Solution and RT PolarFire FPGAs enable users to implement low-power machine learning and artificial intelligence solutions in space-flight instruments, overcoming difficult power and thermal constraints.

14. REFERENCES

[1] Microchip web site, VectorBlox page https://www.microchip.com/en-us/products/fpgas-and-plds/fpga-design-resources

[2] Microchip web site, RT PolarFire page https://www.microchip.com/en-us/products/fpgas-and-plds/fpgas/polarfire-fpgas