Within-Camera Multilayer Perceptron DVS Denoising

A. Rios-Navarro, [2,1,†], S. Guo, [3,1†], G Abarajithan [4,†], K. Vijayakumar[4,†], A. Linares-Barranco, [2,*], T. Aarrestad, [4,*], R. Kastner, [4,1,*], and T. Delbruck,[1,*]

[1]Sensors Group, Institute of Neuroinformatics, Univ. of Zurich and ETH Zurich, Switzerland — [2]Robotic and Tech of Computers group, SCORE lab, ETSI-EPS, Univ. of Seville (USE), Spain — [3]College of Electronic Engineering, National University of Defense Technology (NUDT), China — [4]Inst. of Particle Physics and Astrophysics, ETH Zurich (ETH), Switzerland — [5]Univ. of California, San Diego (UCSD), USA — [†]These authors contributed equally — [*]Contact authors emails: tobi@ini.uzh.ch, arios@us.es, alinares@atc.us.es, thea.aarrestad@cern.ch,

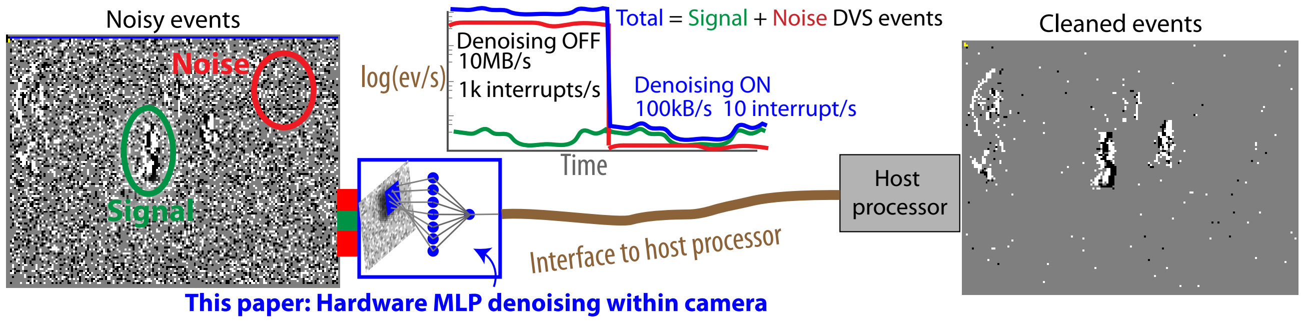

Figure 1. **Advantages of in-camera denoising**: The MultiLayer Perceptron denoising Filter (**MLPF**) reduces the background noise rate in this surveillance scene by a factor of more than 100X while only blocking about 25% of the true signal events created by the moving people. The MLPF accurately and quickly discriminates signal and noise events. Building the filter using a hardware-accelerated neural network implemented alongside the camera's logic circuits dramatically reduces the host processing requirements. In low-light scenes where event camera noise increases, the data rate is reduced from 10MB/s to 100kB/s. With a USB buffer size of 10k events, the host processor interrupt rate reduces from 1 kHz to less than 10 Hz, allowing the processor to mostly sleep during idle periods.

Abstract

In-camera event denoising reduces the data rate of event cameras by filtering out noise at the source. A lightweight multilayer perceptron denoising filter (MLPF) provides state-of-the-art low-cost denoising accuracy. It processes a small neighborhood of pixels from the timestamp image around each event to discriminate signal and noise events. This paper proposes two digital logic implementations of the MLPF denoiser and quantifies their resource cost, power, and latency. The hardware MLPF quantizes the weights and hidden unit activations to 4 bits and has about 1k weights with about 40% sparsity. The Area-Under-Curve Receiver Operating Characteristic accuracy is nearly indistinguishable from that of the floating point network. The FPGA MLPF processes each event in 10 clock cycles. In FPGA, it uses 3.5k flip flops and 11.5k LUTs. Our ASIC implementation in 65nm digital technology for a 346×260 *pixel camera occupies an area of 4.3mm*^{2} *and consumes 4nJ of energy per event at event rates up to 25MHz. The MLPF can be easily integrated into an event camera using an FPGA or as an ASIC directly on the camera chip or in the same package. This denoising could dramatically reduce the energy consumed by the communication and host processor and open new areas of always-on event camera application under scavenged and battery power.*

Code: https://github.com/SensorsINI/dnd_hls

1. Introduction

Event cameras [1] based on the Dynamic Vision Sensor (**DVS**)[*] pixel [2] are useful for vision problems that require high dynamic range and face the fundamental trade-off between latency and power in frame-based cameras [3]. DVS pixels produce events that signal

[*]The arxiv preprint version of this paper includes links and uses PDF tooltip popups for acronyms that some PDF viewers (Adobe and SumatraPDF version ≤3.00) can show.

the change in brightness but also produce Background Activity (**BA**) noise, particularly under low light conditions, where noise rates can increase by a factor of 100 [4]–[7].

Denoising algorithms aim to filter out the noise without removing any signal events. Denoising can be performed on the host computer, but then the camera must transmit all the raw signal and noise events, consuming more power in the communication bus (e.g. USB) and on the host processor, which must remain awake all the time to remove the noise.

Fig. 1 shows how denoising within the camera can dramatically reduce the energy consumed by the communication and host processor. Without denoising, the host processor must constantly remain awake and busy even in the absence of any real signal events. Accurate denoising within the camera preserves most signal events but filters out nearly all the noise events. Thus during quiescent periods where the raw camera output is dominated by noise events, the host processor interrupt rate can be reduced by 2 orders of magnitude, allowing it to sleep most of the time and thus open new areas of always-on event camera application under scavenged and battery power.

We showed in [5] how a tiny MLPF denoises with greater accuracy than prior handcrafted correlation-based denoisers. Our MLPF runs in batch mode on a desktop Graphics Processing Unit (**GPU**) with a measured throughput of about 10^6ev/s[2], but clearly this loses its potential benefit of reducing system-level power because it requires continuous activation of an expensive and power-hungry GPU. Our work here to realize the MLPF within the camera builds on the developments of ultra-quick classifiers from the particle physics community, which require latencies more than 1,000 times quicker (i.e. ns to μs).

2. Novel Contributions

1. Our work reports the first use of ultra-quick latency hardware neural networks developed for particle physics to enable event-by-event accurate event camera denoising.

2. Sec. 5.1 shows how to train the MLPF so that the signal-noise discrimination accuracy with 4-bit weight and activation precision is nearly as good as the floating point version reported in [5].

3. Sec. 5 describes the first Field Programmable Gate Array (**FPGA**) and Application Specific Integrated Circuit (**ASIC**) implementations of the MLPF denoiser.

[2]In batches of 1000 events on NVIDIA RTX 2080 SUPER

4. Our ASIC implementation of an event camera denoiser is, to our knowledge, the first reported. Our ASIC MLPF consumes only a fraction of the power of recently reported event cameras.

3. Background and Related Work

Correlation-based Nearest Neighbor (**NNb**) denoising algorithms, such as the Background Activity Filter (**BAF**) [8] and the improved SpatioTemporal Correlation Filter (**STCF**) [5] have accuracies that are competitive with complex software algorithms and large neural network denoisers [9]–[14]. These correlation denoisers require only a handful of operations per event and have a memory cost about the same as the number of pixels. However, in [5], we showed that training a tiny Multilayer Perceptron (**MLP**) to discriminate signal versus noise results in a significant improvement in discrimination accuracy, particularly at high noise rates and for more challenging denoising, such as driving, where a moving camera creates a denser structure.

In [5], we constructed datasets of known signal and noise events and measured the denoising accuracy using a Receiver Operating Characteristic (**ROC**) curve and computing the Area Under the Curve (**AUC**).

Fig. 2 compares the accuracy of the MLPF against other low-cost fast denoisers. Fig. 2A shows that at False Positive Rate (**FPR**)=0.1, the True Positive Rate (**TPR**) of MLPF is approximately 25% better than the next best STCF, and is 2X better than the popular BAF. Fig. 2B visually compares the MLPF with the STCF on the *driving* dataset with identical TPR settings of their thresholds. The differences are subtle, but the inset shows how STCF allows noise to pass that MLPF blocks. Fig. 2C shows that the MLPF maintains AUC better than other denoisers as the noise rate increases.

Previous implementations of hardware event camera denoising are [15]–[20][3] The FPGA Order(N) Filter (**ONF**) [16] and HashHeat [18] require much less memory than the number of pixels that are practical for cameras that only need to denoise scenes with sparse activity regions; however, [5] showed that these denoisers have poor accuracy for dense scenes where the memory is overwritten too quickly (see, for example, the DWF curve in Fig. 2). The FPGA BAF reported in [17] — also implemented within some Inivation event cameras — is useful, but its accuracy and resiliance to high noise rates are much worse than the MLPF proposed here; see Fig. 2. The mixed-signal

[3]We exclude denoisers that build a binary image and denoise it[21], [22] because they discard event timing and event count information.

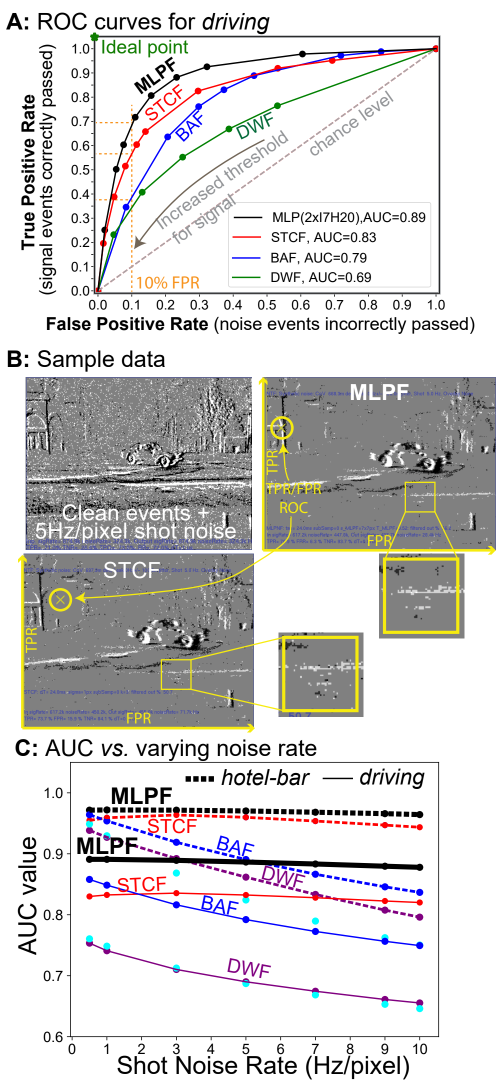

Figure 2. **A:** ROC curves for the proposed MLPF compared with other denoisers STCF, BAF, and Double Window Filter (**DWF**) [5]. The signal vs. noise discrimination threshold is swept to generate each curve. Evaluated on the *driving* dataset from [5] from which this figure is adapted. **B:** Sample data with closeup. **C:** Dependence of overall accuracy metric AUC on noise rate.

subsampled BAF of [15] does not scale to advanced digital processes. The IIR filter array of [19] severely subsamples the pixel array onto its 2D memory, resulting in significant spatial artifacts. The Less Data Same Information (**LDSI**) [20] is a two-layer retina-inspired denoiser that likely has a high latency.

Table 1. Hardware MLPF specifications.

$W \times H$	DVS width×height	346×260[a]
s^2_{MLPF}	input patch	7^2px
N_{MLPF}	# hidden units	10
τ	age window	64 ms[b]
	Quantization bits	(weights/activations)[c]
	Input units	4+1/4+1
	Hidden units	4+1/4
	Output unit	4+1/15+1
	Threshold T_{MLPF}	15+1
	Accumulators	16.6[d]
	Network	
	Num. weights+bias	1001
	Sparsity	40%[e]
	Accuracy (AUC)	0.87[f]

[a] The Timestamp+Polarity Image (**TPI**) memory has size $W \times H \times 18$ bits. The 18 bits are composed of a 16-bit ms timestamp and 2-bit polarity. The ms timestamp is obtaind by right-shifting the μs timestamp by 10 bits. [b] The age window τ is quantized power of 2 from 1 ms to 256 ms. [c] 4+1 means 4 fraction + 1 sign bit. 4 means unsigned 4-bit fraction. [d] All neurons use default QKeras signed 16-bit accumulators with 6-bit integer part. The neuron activation function quantizes this activation to produce the output value. [e] Percent of non-zero weights. [f] Evaluated on combined datasets from [5].

4. Event camera denoising

4.1. Denoising cost metrics

A practical physical implementation of denoising must consider a quartet of Figure of Merits (**FOM**): discrimination accuracy, silicon area, power, and latency. These FOMs generally trade-off against each other, requiring choosing a balance between them. High accuracy is needed to remove noise but not signal. A small area is important to minimize cost. Low latency is needed to discriminate an event before the next one arrives. Low power is essential for power-constrained applications and to minimize camera heating, which increases the photodiode's dark current.

4.2. Denoising accuracy metrics (ROC and AUC)

As discussed in [5], denoising event camera output consists of making a binary discrimination between signal and noise for each event. Positive classification means that the event is classified as a signal event. The ROC measurement of discrimination trade-off plots the TPR and FPR over all thresholds. Ideal denoising achieves zero FPR (noise misclassified as signal) and perfect TPR=1 (signal correctly classified as signal), resulting in AUC=1. A higher AUC means a better classifier. The optimum TPR and FPR depend on the application: An always-on surveillance system might favor small FPR (large noise suppression), while a mobile robot might favor high TPR (high signal retention). We adopt the metric AUC used in [5] to remove bias by a particular choice of threshold. The AUC is a scalar measure of the ROC curve.

4.3. The MLPF

The MLPF is a lightweight classifier trained on labeled data. It achieves good denoising accuracy by detecting spatiotemporal structural cues that can be helpful in discriminating signal versus noise events. For example, it infers that an event is a likely signal event because it is part of a moving edge or corner.

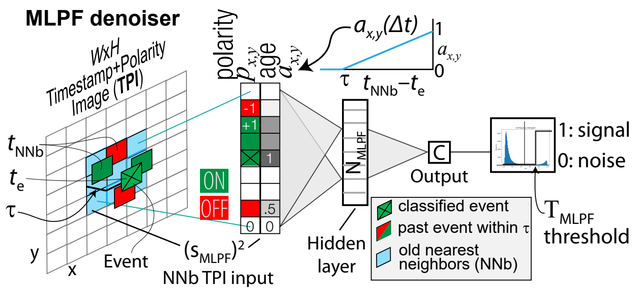

Figure 3. MLPF input and network.

Fig. 3 illustrates the MLPF. It drives a single hidden layer of N_{MLPF} neurons from an input patch of $(s_{\text{MLPF}})^2$ pixels of the TPI from the neighborhood around the event that is to be classified. The TPI is an $W \times H$ pixel 2D memory that holds the latest timestamp (in ms) and the ON or OFF brightness change event polarity corresponding to each DVS pixel. Table 1 lists values used in this paper.

Recent events are important; older events are less relevant, so the $a_{\text{x,y}}$ input channel encodes the age of NNb events as a type of time surface [1]: $a_{x,y}$ is calculated from each TPI pixel by the function illustrated in Fig. 3 where t_e is the timestamp of the event e that is to be classified, and t_{NNb} is the timestamp of the most recent previous event stored in the TPI corresponding to a NNb pixel. $a_{x,y}$ approaches 1 for recent events and decays to 0 for older events. τ is the time window parameter.

Polarities of past events are also informative because a moving edge usually produces identical polarities, so the the $p_{\text{x,y}}$ input provides the signed NNb polarities, using -1 for OFF, +1 for ON and 0 for events older than τ. The $p_{\text{x,y}}$ at the central pixel is from the classified event to provide the necessary information to determine whether the classified event has the same polarity as the past events in the NNb.

The MLPF with $s_{\text{MLPF}} = 7$ pixels has $7 \times 7 \times 2 = 98$ input units (polarity and age) and a single hidden

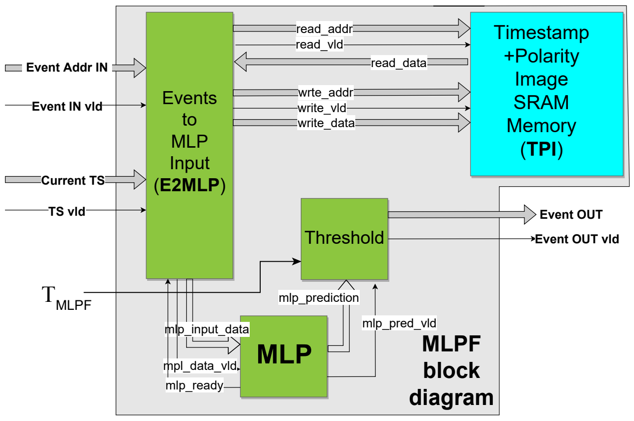

Figure 4. MLPF FPGA and ASIC block diagram.

layer of $N_{\text{MLPF}} = 10$ neurons, and there are about 1k weights. Hidden neurons use a Rectified Linear Unit (**ReLU**) activation function, and the final output C uses a sigmoid activation function. We threshold C against T_{MLPF} to form the final binary discrimination, signaling that the event is signal or noise. T_{MLPF} adjusts the TPR/FPR trade-off.

5. Implementation

Fig. 4 illustrates the three major components of the MLPF hardware implementation: Events to MLP Input (**E2MLP**), MLP, and TPI. The E2MLP block receives the event generated by the DVS and its timestamp and produces the activation vector for the MLP. To calculate this activation vector, the coordinates of the current event are used to access the TPI memory and read the timestamps and polarity of the events in the 7×7 neighborhood of the current event. As the readings are taken, the age of the neighborhood event is calculated from the $a_{\text{x,y}}$ equation illustrated in Fig. 3 [5]. The activation vector is composed of 98 elements, where the first 49 are the ages and the last 49 are the polarities of the neighborhood events. After the activation vector is generated, the TPI memory is updated with the current event's timestamp and polarity information.

5.1. Dataset, Training, and Network quantization

Our training dataset combined the *hotel-bar* and *driving* noise datasets from [5]. These datasets were chosen to cover the range of applications from sparse surveillance to dense mobile robotics event camera output. To these clean DVS recordings we add both simulated and pre-recorded DVS BA noise events. We train the network to optimally discriminate signal and noise events. Appendix F of the supplementary material of [5] details the floating point MLPF training procedure (we removed the dropout layer here). We used exactly the same dataset and training procedure except for quantization-aware training as detailed next.

Applying model compression at training time is crucial to minimize the area and maximize the accuracy of the model. We rely on quantization-aware training through the QKeras[4] library. Through drop-in replacement of standard Keras layers, this library allows users to create quantized versions of the Keras equivalent. Doing so at training time optimizes accuracy by allowing the network to adjust the model weights according to the precision tailored for the hardware.

Fig. 5 lists the network code. The quantization function `quantized_bits` takes as input the number of bits to quantize to, the number of integer bits, as well as an optional parameter *alpha*, which is used to change the absolute scale of the weights while keeping them quantized within the same bit width. In our case, we quantize the kernel and bias to a bit width of 4 with zero integer bits and set *alpha* to 1 (meaning that no scaling is applied). The inputs, weights, activations, and output are quantized to a fixed point decimal ranging from -1 to 1[5]. Note that the function `quantized_bits` behaves differently from the fixed point numbers used in Vivado High Level Synthesis (**HLS**). For `quantized_bits(bitwidth, integer bits)`, the HLS equivalent is `ap_fixed(bitwidth, integer bits+1)`. The experimental results in Fig. 6 show that the use of quantified weights and activations has almost no effect on the AUC value, except for 2 bits. Due to the limited diversity of the MLPF output caused by the 2-bit input and weights of the last dense layer, sweeping thresholds barely affect the ROC curve and the AUC.

For our implementation, we chose 4-bit weight and activation quantization. For simplicity of design, the input of the network is quantized to 4 bits for age a and polarity p channels. The age is computed by right shifting the Δt time difference by some number of bits according to the desired τ. The polarity is represented as 0 or ± 1. The final output is quantized to 16 bits for a fine discrimination threshold. The chosen model (Fig. 5) achieves AUC of 0.87 on the *driving* dataset, close to the floating point AUC of 0.88 (Fig. 6).

5.2. Firmware implementation with `hls4ml`

We use the `hls4ml` library [23] to generate an FPGA implementation of the MLP. `hls4ml` converts a given Deep Neural Network (**DNN**) into HLS code,

[4] https://github.com/google/qkeras.

[5] Actually to the closest possible value to 1, e.g. 0.875 for 4-bit signed-value quantization

Figure 5. QKeras code for the network.

```
inputs = Input(shape=[98, ], name='input')
x = QActivation("quantized_bits(4,0,alpha=1)",
    name="qact0")(inputs)
x = QDense(10, input_shape=(98, ),
    kernel_quantizer=quantized_bits(4,0,alpha
    =1),bias_quantizer=quantized_bits(4,0,alpha
    =1),name="fc1")(x)
x = QActivation("quantized_relu(4,0)", name="
    relu1")(x)
x = QDense(1, kernel_quantizer=quantized_bits
    (4,0,alpha=1),bias_quantizer=quantized_bits
    (4,0,alpha=1),name="fc2")(x)
x = Activation("sigmoid", name="doutput")(x)
model = Model(inputs, x)
```

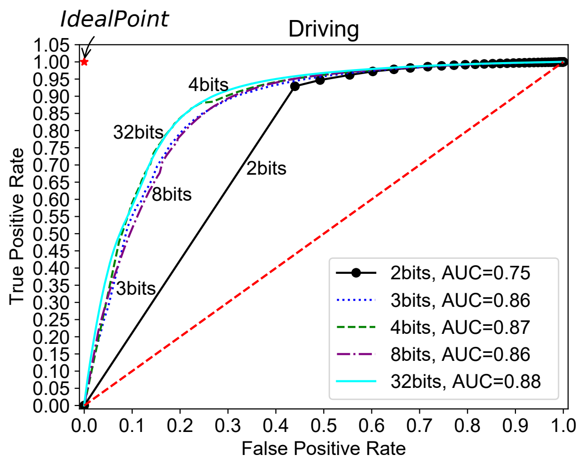

Figure 6. ROC curves for quantized MLPFs compared with floating point 32bits network. Evaluated on the more challenging *driving* dataset from [5].

which uses a C-like syntax decorated with hardware-specific optimizations such as pipelining and custom data types. The HLS compiler then translates the HLS code into the FPGA firmware. In contrast to conventional Register Transfer Logic (**RTL**) logic design, HLS lies at a higher level of abstraction, making it faster to specify and generate custom FPGA architectures, since it is a subset of C coding and thus allows greater productivity, but it has limitations for designs that require complex interconnected circuits.

The `hls4ml` library interfaces to QKeras [24] to provide a translation of the quantized weights of the MLPF model using quantization-aware training into the correct fixed-point equivalent values on the FPGA implementation. A developer can compile the HLS code for the network in a few seconds from the trained QKeras model.

`hls4ml` was developed for low-latency and high-throughput applications. Thus, hls4ml generates a custom implementation for each layer and the entire network is spatially instantiated on the FPGA logic. Our MLPF is implemented in a fully parallel manner; all multiplications are performed in parallel. The output neuron activation is computed as a combinational logic weighted sum of the hidden units, which themselves are computed as combinational logic weighted sums of input unit activations. The weights are frozen into the connections, and the network must be resynthesized when they need to be changed. Zero weights do not use hardware resources.

The Vivado HLS compiler limits arrays to 1024 elements, which is less than what the hidden layer of the 98I-20H-1O MLPF in [5] would require to be fully unrolled[6]. Therefore, to reach the lowest possible latency, we trained an MLPF with $N_{\text{MLPF}} = 10$ hidden units. Using 10 hidden units instead of 20 and the chosen quantization (Table 1) reduces the AUC from the floating point 0.89 to 0.87. The low precision of the weights allows all multiplications in the network to be computed by LookUp Tables (**LUT**) rather than the more power-hungry hardware multiplier units.

Since we do not depend on having an output between zero and one (this is only necessary at training time), we removed this final sigmoid activation function and instead placed a threshold on the activation of the output unit. This saves one clock cycle.

5.3. FPGA realization

Fig. 7 shows our setup to implement MLPF using a DAVIS346 prototype camera [25], [26] connected by Address Event Protocol (**AER**) cable to a custom Xilinx xc7z100 FPGA development board [27].

Our first implementation of the E2MLP module used dual read port Block RAM (**BRAM**) to read the 49 TPI pixels in 25 clock cycles. By adopting a memory organized as in Liu's EDFLOW camera [28], the columns are read in parallel, in only 7 clock cycles. The key to making this scheme work was the correct use of the Vivado HLS pragma *array_partition* as in [28]. This pragma partitions the array into smaller arrays; in this case, a separate BRAM is for each sensor column. The resulting RTL has multiple smaller memories instead of one large memory. This pragma effectively increases the amount of read and write ports for storage. The resulting BRAM usage increases because it requires more memory instances and registers.

The activations generated by the E2MLP module are used by the MLP module to perform inference on these data and generate a prediction. To decide whether the current event is considered to signal or noise, a threshold is applied to the calculated prediction.

[6]The array would have to be partitioned into 98*20=1,960 elements, which is more than the tool allows.

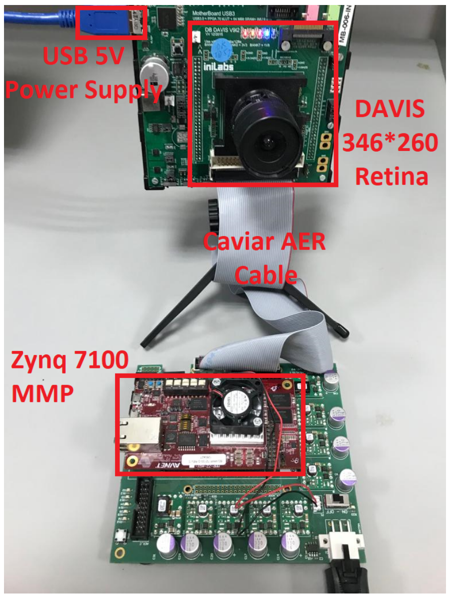

Figure 7. The DAVIS camera connected to the Zynq XC7Z100 FPGA development board that holds the MLPF.

For the FPGA implementations, the E2MLP, MLP and TPI blocks are written in Vivado HLS 2020.1. We synthesized our solution for two different FPGAs, a decade-old midrange Xilinx Zynq XC7Z100 and an entry-level Zynq Ultrascale+ ZU3CG.

Both of these System on Chip (**SoC**) FPGAs are attractive candidates for building an event camera because their processors can run a simplified Linux kernel and thus the camera could form a discrete smart camera node in a robot.

5.4. ASIC realization

We wrote an RTL (SystemVerilog) implementation of the MLP and E2MLP blocks in Fig. 4 to circumvent current Vivado HLS limitations in generating ASIC compatible RTL. We target a conventional 65nm digital process technology from TSMC, since it is economical, widely available, and because we had access to it for this project. We used Cadence Genus for the synthesis and Cadence Innovus for the place and route. Fig. 8 shows the layout of the circuit without memory. It was obtained using the low power library, applying the clock-gating technique (for 43% of inferred flip-flops) and reducing the power ring size to minimize its area. Our MLP architecture also allows us to reuse the multipliers, adders, and registers, minimizing the chip area and power consumption. We use data gat-

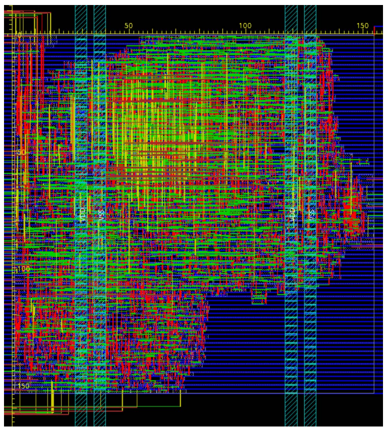

Figure 8. ASIC MLPF E2MLP+MLP layout in 65nm digital technology. The ruler units are microns. The logic area is 0.022mm^2 (67% density). The bonding pads are not included since this block would normally be placed adjacent to the sensor core. In our current version, the SRAM blocks are not included.

ing to prevent the gates and flip-flops from switching on clock cycles without new data. We measured an improvement of 20% in power consumption with the use of clock gating. Our RTL design and scripts are released as open source under a GPL license.

Our ASIC architecture differs from the FPGA implementation to address the following trade-offs. The FPGA implementation uses one BRAM per column of the image, to be able to read any arbitrary 7×7 neighborhood within 7 clock cycles. This is possible since these Static RAMs (**SRAM**) are synthesized as groups of much smaller BRAMs already available on the FPGA. Since SRAMs are custom generated on silicon, similar ASIC implementation with one SRAM per column would result in a much higher area and consume more power due to the duplication of control circuitry for each SRAM.

Therefore, in our ASIC design, we utilize two 1W1R SRAMs[7]. We can read two pixels of the TPI per cycle, compared to seven per cycle in our FPGA design. As a result, the latency of the ASIC implementation is 33 clock cycles, compared to 10 clock cycles in FPGA. Since our ASIC implementation has a much higher clock frequency, it is faster with a smaller latency (40 ns) compared to the FPGA (43 ns).

6. Results

Table 2 lists the main FOMs of the FPGA and ASIC implementations of the MLPF in terms of maximum clock frequency, latency in clock cycles and nanoseconds, resources used, power, and energy per event. FPGA resources consist of LookUp Tables (**LUT**), Flip-Flops (**FF**), and Block RAMs (**BRAM**). A LUT is a programmable memory element (at synthesis time) that implements combinational logic. A FF is a single bit memory element. ASIC resources are measured by area. We did not measure FPGA power consumption because it is completely dominated by standby power. The Table 2 notes provide details for many of the specifications.

The MLP inference latency is only 3 cycles, giving a total FPGA latency of 10 cycles. In our FPGA implementations, synthesis generated the TPI as multiple dual port BRAM memories and needs 400 BRAM_18K. This memory has 89960 words ($W \times H$ pixels) with 18-bit word size, where 16 bits are used to store the timestamp and 2 bits for the event polarity.

Our ASIC implementation area is dominated by SRAM. We estimated a standby power of about 35 mW, consisting mostly of SRAM leakage. The energy per event is 4 nJ, so a noise event rate of 1 MHz would burn another 4 mW.

7. Conclusion

This paper proposes two logic circuit implementations of a multilayer perceptron event camera denoiser.

The FPGA implementation discriminates each event in 10 clock cycles or 100 ns with the clock frequency of 100 MHz, allowing the denoising to occur at event rates of 10 MHz. For higher noise rates, the denoiser can bypass or block events that arrive while it is busy.

We showed two implementations mapped to SoC FPGAs that would be practical for integrating MLPF denoising within an embedded DVS camera. Using an individual BRAM for each column of the pixel array, we showed how the MLP input can be formed for each event in only 7 clock cycles, enabling 10-cycle discrimination. The ASIC block, which could be integrated into future DVS chips, has a longer latency in clock cycles, but since the block runs at a much higher clock frequency, it can classify each event in 40 ns, enabling event-by-event denoising up to rates of 25 MHz. This ASIC block power consumption is estimated to be on the order of 40 mW, which is a fraction of the power of recently reported event cameras.[8] At low noise rates, the power consumption drops to essentially the SRAM leakage. Our current design uses standard SRAM; by exchanging this memory for lower leakage low power memory, we believe it would be possible to reduce the standby power by an order of magnitude.

[7]Our memory compiler could not generate dual-read-port SRAM, so we used two single port SRAMs

[8]Recent event camera chips [29]–[31] report sensor power consumption from 60 mW to 525 mW.

Table 2. FOMs of the hardware MLPFs for Table 1 parameters.

	Xlinx Zynq XC7Z100		Xlinx Zynq U+ ZU3CG		65nm ASIC. Vdd=1.08V	
Max. Clock Freq.	100 MHz		236 MHz		833 MHz	
Latency	cycles	ns	cycles	ns	cycles	ns
E2MLP	7	70	7	$\sim$30	30	36
MLP	3	30	3	$\sim$13	3	3.6
Total	10	100	10	$\sim$43	33	40

	Xlinx Zynq xc7z100			Xlinx Zynq U+ ZU3CG			65nm ASIC	
	LUT	FF	BRAM	LUT	FF	BRAM	Logic area	SRAM area
Resources	18.4k	3.9k	400[a]	24.6k	3.8k	400[a]	0.022mm^2	4.3[b]mm^2
Resource %	6%	0.72%	26%	34%	2%	92%	–	–
Power	Not relevant						40mW[c]	
Energy/event	Not relevant						4nJ[d]	

[a] Used for TPI. [b] Our two SRAMs each run at 1 GHz with 1.0V supply. They have 18 bit word width and 2048 words per bank. Maximum possible is 2048 per bank using multiple banks to store the TPI. The entire design requires $W \times H/2048 \approx 44$ such SRAM banks. [c] Assumes 1 MHz event rate and 4 nJ per event (1.2 nJ MLP + 2.4 nJ SRAM +0.4 nJ E2MLP), plus 35 mW SRAM leakage. [d] Assumes one MLP inference and 50 SRAM accesses. The energy per event is computed by $E = P \times T \times N$, where P is the power consumption of the block, T is the period of the clock and N is the required number of clock cycles per event.

Table 3 compares the hardware MLPF with other FPGA hardware denoisers in terms of accuracy, memory cost, and throughput (there is no other ASIC implementation to compare with). As discussed in Sec. 3, except for the BAF in [17], other denoisers seek to minimize memory, which in general leads to poor discrimination accuracy at high noise rates. The MLPF has by far the best AUC accuracy on the sparse *hotel-bar* and dense *driving* datasets of [5]. Although MLPF is not the fastest, its maximum denoising rate of $\approx$ 25 MHz is suitable for recent event cameras.

Recent industrial event camera chips [29]–[31] feature event pixels that are less than 5 μm and are built in stacked technologies with an optical top part using 90nm technology and a digital bottom part using feature sizes down to 22nm technology. The digital layer area is several times larger than the sensor area. For this paper, we designed the MLPF ASIC block for 65nm technology; In 22nm technology, the logic area would be smaller, but the SRAM required to hold TPI would need to be several Mpx. The dominant TPI SRAM would occupy an area of about 1 mm^2/Mpx[9]. It would likely be possible to use subsampling onto a smaller-resolution TPI [15] too much loss of denoising accuracy. The MLPF would comprise only a small part of these designs. However, since these recent event camera chips use some form of sampled, frame-like sparse readout of the brightness change events, it is unclear to what extent they can use fine event timing for denoising like MLPF.

[9]Based on 6T cell with area $130F^2$, where F = 22nm is the feature size. The total SRAM area would be 130 $\times$ (22e-9m)^2 $\times$ 1e6px $\times$ 18bits $\times$ 1e6mm^2/m^2 = 1mm^2.

Table 3. Comparison with other hardware DVS denoisers.

Denoiser	***hotel-bar* AUC[a]**	***driving* AUC[a]**	**Mem(#)[b]**	**Max. Event Rate[c] MHz**
MLPF FPGA	**0.96**	**0.87**	$W \times H$	23
MLPF ASIC	”	”	$W \times H$	25
BAF [17]	0.89	0.79	$W \times H$	3.6
ONF [16]	0.01[e]	0.01[e]	$2 \times (W + H)$	3
HashHeat [18]	0.67	0.56[f]	**128**	100
IIRs [19]	NA	NA	$W \times H$[g]	**385**
LDSI [20]	NA	NA	$W \times H$	3?[d]

[a] Evaluated at shot noise rate of 5 Hz/pixel using method in [5].
[b] Memory cells for $W \times H$ pixel DVS assuming cells each have on the order of 18-36 bits and that no subsampling as in [8], [15], [19] is used to reduce memory at the cost of reduced denoising accuracy.
[c] Maximum event rate for denoising all events. [d] The LDSI was measured at 50 MHz clock and reported to run at up to 177 MHz clock frequency. The throughput was not reported except to state it could run up to the maximum event rate of [2] which is 1 MHz.
[e] The ONF AUC is very low because it cannot achieve large TPR and FPR even with very low threshold. [f] The HashHeat AUC is derived from sweeping its threshold compared with the heat values stored in a 128-element vector.
[g] The IIRs denoiser actually subsampled the array to 20x20 pixel memory, causing significant artifacts.

Acknowledgements

This work was supported by the Swiss SNF projects SCIDVS (2O0O21_185069/1), VIPS (2082-l_181010/1), and PZ00P2_201594, and by the Spanish grant MINDROB (PID2019-105556GB-C33/AEI/10.13039/501100011033). We thank Dr. Min Liu and Prof. Shih-Chii Liu (INI), Prof. Mingu Kang (UCSD), and anonymous reviewers for their suggestions to improve the paper.

References

[1] G. Gallego, T. Delbruck, G. M. Orchard, *et al.*, "Event-based vision: A survey," en, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PP, Jul. 2020, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2020.3008413 (cit. on pp. 1, 4).

[2] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128x128 120 dB 15 us latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008, ISSN: 0018-9200, 1558-173X. DOI: 10.1109/jssc.2007.914337 (cit. on pp. 1, 8).

[3] S.-C. Liu, B. Rueckauer, E. Ceolini, A. Huber, and T. Delbruck, "Event-Driven sensing for efficient perception: Vision and audition algorithms," *IEEE Signal Process. Mag.*, vol. 36, no. 6, pp. 29–37, Nov. 2019, ISSN: 1558-0792. DOI: 10.1109/MSP.2019.2928127 (cit. on p. 1).

[4] Y. Hu, S.-C. Liu, and T. Delbruck, "V2e: From video frames to realistic DVS events," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2021, pp. 1312–1321. DOI: 10.1109/CVPRW53098.2021.00144 (cit. on p. 2).

[5] S. Guo and T. Delbruck, "Low cost and latency event camera background activity denoising," en, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PP, Feb. 2022, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2022.3152999 (cit. on pp. 2–6, 8).

[6] T. Finateu, A. Niwa, D. Matolin, *et al.*, "5.10 a 1280x720 back-illuminated stacked temporal contrast event-based vision sensor with 4.86um pixels, 1.066GEPS readout, programmable event-rate controller and compressive data-formatting pipeline," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, San Francisco, CA, USA: IEEE, Feb. 2020, pp. 112–114, ISBN: 9781728132051. DOI: 10.1109/isscc19947.2020.9063149 (cit. on p. 2).

[7] R. Graca and T. Delbruck, "Unravelling the paradox of intensity-dependent DVS noise," in *2021 International Image Sensors Workshop (IISW 2021)*, online, 2021, (accepted). [Online]. Available: https://imagesensors.org/Past%20Workshops/2021%20Workshop/2021%20Papers/R25.pdf (cit. on p. 2).

[8] T. Delbruck, "Frame-free dynamic digital vision," in *Proceedings of Intl. Symp. on Secure-Life Electronics, Advanced Electronics for Quality Life and Society*, vol. 1, Tokyo, Japan: University of Tokyo, 2008, pp. 21–26. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.192.2794&rep=rep1&type=pdf (cit. on pp. 2, 8).

[9] J. Wu, C. Ma, L. Li, W. Dong, and G. Shi, "Probabilistic undirected graph based denoising method for dynamic vision sensor," *IEEE Trans. Multimedia*, pp. 1–1, 2020, ISSN: 1941-0077. DOI: 10.1109/TMM.2020.2993957 (cit. on p. 2).

[10] Y. Alkendi, R. Azzam, A. Ayyad, S. Javed, L. Seneviratne, and Y. Zweiri, "Neuromorphic camera denoising using graph neural network-driven transformers," Dec. 2021. arXiv: 2112.09685 [cs.CV]. [Online]. Available: http://arxiv.org/abs/2112.09685 (cit. on p. 2).

[11] R. W. Baldwin, M. Almatrafi, V. Asari, and K. Hirakawa, "Event probability mask (EPM) and event denoising convolutional neural network (EDnCNN) for neuromorphic cameras," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA: IEEE, Jun. 2020, ISBN: 9781728171685. DOI: 10.1109/cvpr42600.2020.00177 (cit. on p. 2).

[12] Z. Zhang, J. Suo, and Q. Dai, "Denoising of event-based sensors with deep neural networks," en, in *Optoelectronic Imaging and Multimedia Technology VIII*, vol. 11897, SPIE, Oct. 2021, pp. 203–209. DOI: 10.1117/12.2602742 (cit. on p. 2).

[13] P. Duan, Z. W. Wang, X. Zhou, Y. Ma, and B. Shi, "EventZoom: Learning to denoise and super resolve neuromorphic events," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. [Online]. Available: http://ci.idm.pku.edu.cn/CVPR21a.pdf (cit. on p. 2).

[14] H. Fang, J. Wu, L. Li, J. Hou, W. Dong, and G. Shi, "AEDNet: Asynchronous event denoising with Spatial-Temporal correlation among irregular data," in *Proceedings of the 30th ACM International Conference on Multimedia*, ser. MM '22, Lisboa, Portugal: Association for Computing Machinery, Oct. 2022, pp. 1427–1435, ISBN: 9781450392037. DOI: 10.1145/3503161.3548048 (cit. on p. 2).

[15] H. Liu, C. Brandli, C. Li, S. Liu, and T. Delbruck, "Design of a spatiotemporal correlation filter for event-based sensors," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2015, pp. 722–725. DOI: 10.1109/ISCAS.2015.7168735 (cit. on pp. 2, 3, 8).

[16] A. Khodamoradi and R. Kastner, "O(N)-Space spatiotemporal filter for reducing noise in neuromorphic vision sensors," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. 99, pp. 1–1, 2018. DOI: 10.1109/TETC.2017.2788865 (cit. on pp. 2, 8).

[17] A. Linares-Barranco, F. Perez-Pena, D. P. Moeys, *et al.*, "Low latency Event-Based filtering and feature extraction for dynamic vision sensors in Real-Time FPGA applications," *IEEE Access*, vol. 7, pp. 134 926–134 942, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2941282 (cit. on pp. 2, 8).

[18] S. Guo, Z. Kang, L. Wang, S. Li, and W. Xu, "HashHeat: An O(C) complexity hashing-based filter for dynamic vision sensor," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, ieeexplore.ieee.org, Jan. 2020, pp. 452–457. DOI: 10.1109/ASP-DAC47756.2020.9045268 (cit. on pp. 2, 8).

[19] M. Kowalczyk and T. Kryjak, "Hardware architecture for high throughput event visual data filtering with matrix of IIR filters algorithm," in *2022 25th Euromicro Conference on Digital System Design (DSD)*, Aug. 2022, pp. 284–291. DOI: 10.1109/DSD57027.2022.00046 (cit. on pp. 2, 3, 8).

[20] J. Barrios-Aviles, A. Rosado-Munoz, L. D. Medus, M. Bataller-Mompean, and J. F. Guerrero-Martinez, "Less data same information for Event-Based sensors: A bioinspired filtering and data reduction algorithm," en, *Sensors*, vol. 18, no. 12, Nov. 2018, ISSN: 1424-8220. DOI: 10.3390/s18124122 (cit. on pp. 2, 3, 8).

[21] S. K. Bose, D. Singla, and A. Basu, "A 51.3 TOPS/W, 134.4 GOPS in-memory binary image filtering in 65nm CMOS," Jul. 2021. arXiv: 2107.11723 [eess.IV]. [Online]. Available: http://arxiv.org/abs/2107.11723 (cit. on p. 2).

[22] X. Cheng, H. Zhu, J. Liu, M. Wang, and X. Zeng, "An efficient Markov random field based denoising approach for dynamic vision sensor," in *2021 IEEE 14th International Conference on ASIC (ASICON)*, ieeexplore.ieee.org, Oct. 2021, pp. 1–4. DOI: 10.1109/ASICON52560.2021.9620426 (cit. on p. 2).

[23] J. Duarte *et al.*, "Fast inference of deep neural networks in FPGAs for particle physics," *J. Instrum.*, vol. 13, no. 07, P07027, 2018. DOI: 10.1088/1748-0221/13/07/P07027 (cit. on p. 5).

[24] C. N. Coelho, A. Kuusela, S. Li, *et al.*, "Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors," en, *Nature Machine Intelligence*, pp. 1–12, Jun. 2021, ISSN: 2522-5839, 2522-5839. DOI: 10.1038/s42256-021-00356-5 (cit. on p. 5).

[25] C. Brandli, R. Berner, M. Yang, S. Liu, and T. Delbruck, "A 240x180 130 dB 3 us latency global shutter spatiotemporal vision sensor," *IEEE J. Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, Oct. 2014, ISSN: 0018-9200, 1558-173X. DOI: 10.1109/JSSC.2014.2342715 (cit. on p. 6).

[26] G. Taverni, D. Paul Moeys, C. Li, *et al.*, "Front and back illuminated dynamic and active pixel vision sensors comparison," *IEEE Trans. Circuits Syst. Express Briefs*, vol. 65, no. 5, pp. 677–681, May 2018, ISSN: 1558-3791. DOI: 10.1109/TCSII.2018.2824899 (cit. on p. 6).

[27] A. Linares-Barranco, A. Rios-Navarro, S. Canas-Moreno, E. Pinero-Fuentes, R. Tapiador-Morales, and T. Delbruck, "Dynamic vision sensor integration on fpga-based cnn accelerators for high-speed visual classification," in *International Conference on Neuromorphic Systems 2021*, ser. ICONS 2021, Knoxville, TN, USA: Association for Computing Machinery, 2021, ISBN: 9781450386913. DOI: 10.1145/3477145.3477167 (cit. on p. 6).

[28] M. Liu and T. Delbruck, "EDFLOW: Event driven optical flow camera with keypoint detection and adaptive block matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 9, pp. 5776–5789, Sep. 2022, ISSN: 1558-2205, 1558-2205. DOI: 10.1109/TCSVT.2022.3156653 (cit. on p. 6).

[29] M. Guo, S. Chen, Z. Gao, *et al.*, "A 3-Wafer-Stacked hybrid 15MPixel CIS + 1 MPixel EVS with 4.6GEvent/s readout, In-Pixel TDC and On-Chip ISP and ESP function," in *2023 IEEE International Solid- State Circuits Conference (ISSCC)*, San Francisco, CA, USA: IEEE, Feb. 2023, pp. 90–92. DOI: 10.1109/isscc42615.2023.10067476 (cit. on pp. 7, 8).

[30] K. Kodama, Y. Sato, Y. Yorikado, *et al.*, "1.22um 35.6mpixel RGB hybrid Event-Based vision sensor with 4.88um-Pitch event pixels and up to 10K event frame rate by adaptive control on event sparsity," in *2023 IEEE International Solid- State Circuits Conference (ISSCC)*, Feb. 2023, pp. 92–94. DOI: 10.1109/ISSCC42615.2023.10067520 (cit. on pp. 7, 8).

[31] A. Niwa, F. Mochizuki, R. Berner, *et al.*, "A 2.97um-Pitch Event-Based vision sensor with shared pixel Front-End circuitry and Low-Noise intensity readout mode," in *2023 IEEE International Solid- State Circuits Conference (ISSCC)*, Feb. 2023, pp. 4–6. DOI: 10.1109/ISSCC42615.2023.10067566 (cit. on pp. 7, 8).