https://doi.org/10.2352/EI.2025.37.7.ISS-279

Infinite-ISP: An Open Source Hardware Image Signal Processor Platform for all Imaging Needs

Taimur Bilal, Bakhtawar Amjad, Talha Iqbal, Bilal Zafar, Khurram Usman, Sohaib Imran Bhatti, Abdul Rahman Qureshi, Muqudas Rafiq, Maria Nadeem, Junaid Amjad, Sarfraz Shafi, Kamran Malik, Muhammad Abdullah; 10xEngineers Inc.; Irvine, CA

Abstract

While traditionally not available in the public domain, access to the complete image signal processing (ISP) algorithmic pipeline and its hardware implementation are necessary to enable new imaging use cases and to improve the performance of high level deep learning vision networks. In this paper, we present Infinite-ISP: A complete hardware ISP development suite, comprising of an ISP algorithm development platform, a bit-accurate fixed-point reference model, an ISP register transfer level development platform, an FPGA development workflow and an ISP tuning tool. To aid the hardware development process, we develop end-to-end reference designs for the KV260 vision AI starter kit (AMD Xilinx Kria SOM) and Efinix Ti180J484 kit, with support for 3 image sensors (Sony IMX219, Onsemi AR1335 and Omnivision OV5647) using the Infinite-ISP framework. These ISP reference designs support 10-bit 2592×1536 4 Megapixel (MP) Bayer image sensors with a maximum pixel throughput of 125 MP/s or 30 frames per second. We demonstrate that the image quality of Infinite-ISP is comparable to commercial ISPs found in Skype Certified Cameras and also performs competitively with digital still cameras in terms of the perceptual IQ metrics BRISQUE, PIQE, and NIQE. We envision the Infinite-ISP (available at `https://github.com/10x-Engineers/Infinite-ISP`*) under the permissive open-source license to streamline ISP development from algorithmic design to hardware implementation and to foster community building and further research around it for both academia and industry.*

Introduction

The Image Signal Processor (ISP) inside a digital camera converts raw pixel data from an image sensor into an output image that is an accurate depiction of the captured scene. Traditionally, the ISP design has been manually optimized by ISP *experts* to produce visually pleasing images for human perception. This, however, is not the only role of an ISP from a hardware implementation perspective. The development of a hardware ISP is made even more challenging due to the requirement of multiple complex algorithms working in a pipeline manner while maintaining the modular nature of the processing pipeline so that it can be configured for different scenes, conditions, and image sensors. Moreover, the ISP has to process the image sensor data to achieve a minimum pixel throughput necessary for human consumption while staying within the underlying hardware and power constraints. While classic ISP pipelines have been optimized for human vision, modern ISPs also cater to the demands of computer vision and deep learning networks which sometimes require addition of more complex algorithms to the ISP pipeline and in some cases, an increase in pixel throughput, without forgoing image quality and staying within the power and resources budget.

The ISP algorithms and their hardware implementation have been a closely guarded secret for decades with each camera manufacturer having its own proprietary algorithms and hardware ISP pipeline. Apart from commercial ISP offerings, there are a few open-source alternatives [1, 2, 3, 4] most of which are high-level algorithmic software solutions without corresponding hardware implementations that provide minimal insight into the image quality attainable from the fixed-point hardware pipeline. The few publicly available hardware implementations don't come with ISP tuning tools, bit-accurate fixed-point model, algorithm development model and the documentation necessary for building a vision system around it. OpenISP [1] is a Python based floating-point algorithmic flow and Fast-OpenISP [2] is its C based faster implementation. Both lack fixed-point and hardware implementation of the proposed algorithm pipeline. Demo-ISP consists of Field Programmable Gate Array (FPGA) ports of a hardware ISP on Xilinx Zynq 7000, UltraScale+ and Altera EP4CE6 devices along with their respective firmwares [3]. It, however, does not come with an algorithm development model, a fixed-point bit-accurate model of the FPGA-ported ISP or a tuning tool for the hardware ISP. xkISP is written in C based on the High-Level Synthesis (HLS) framework for seamless hardware inference of the ISP pipeline in Vivado HLS software but lacks the tuning tool, firmware support and the algorithm development model [4].

In the commercial space, hardware ISPs are often developed as soft intellectual property (IP) by many image processing and vision IP core development companies [5, 6, 7, 8, 9]. These solutions are provided as a complete ISP development package along with FPGA Reference Designs [5, 6, 7, 8] and in some cases [9] individual algorithm blocks of the pipeline are also available. Providing such an end-to-end hardware ISP development suite complete with an FPGA reference design, enables the customers such as camera manufacturers and vision System on Chip (SoC) architects to integrate ISP blocks of their choice into their pipeline as per their specific application. These solutions, however, have a high purchase cost, often come with a yearly subscription license and the provided reference designs for the hardware ISP are locked to a particular FPGA vendor. In many cases, the register transfer level (RTL) is encrypted and users are not able to modify the ISP pipeline to suit their application [6]. This lack of a complete end-to-end development framework from a high-level modular algorithm pipeline down to the various components of the hardware implementation and the limitations and high costs of the ones available in the commercial domain motivate our work on Infinite-ISP: A complete development suite for ISPs.

In this paper, we present Infinite-ISP [10]: A comprehensive, modular and tuneable hardware ISP which includes the high-level

algorithm development model of the ISP pipeline, its RTL and FPGA implementation, a bit-accurate fixed-point reference model and a tuning tool. Our major contributions are:

1. **Algorithm Development Model:** We deliver a modular algorithmic pipeline based in Python that includes all the basic blocks for RAW, RGB and YUV pixel processing, performs competitively with open-source and commercial ISPs in terms of image quality metrics and can be enhanced by the community to support further algorithm development.
2. **Reference Model:** We provide a 16-bit fixed-point reference model to bridge the gap between the high level algorithm model and its hardware implementation.
3. **RTL ISP:** Based on the fixed-point reference model, we provide a RTL development platform that has the same modular structure as the algorithm pipeline. This can be extended by the open-source community to incorporate new algorithms in sync with the algorithm pipeline.
4. **FPGA Implementation:** We currently provide the end-to-end ISP FPGA development workflow for the KV260 vision AI starter kit (AMD Xilinx Kria SOM) and the Efinix Ti180J484 kit with support for other FPGAs vendors possible through the collaboration of open-source community.
5. **Verification Framework:** We provide an automated verification framework to ensure that the RTL and FPGA ISPs are bit-accurate against the fixed-point reference model.
6. **ISP Tuning Tool:** We also provide an ISP tuning tool that can be used for configuring the ISP and tuning the image quality for different applications and settings.
7. **ISP Reference Design:** Using the Infinite-ISP framework, we develop a 10-bit pipeline with support for a maximum input image size of 2592×1536 (4 Megapixels (MP)), three image sensors (Sony IMX219, Onsemi AR1335 and Omnivision OV5647) and a maximum throughput of 30 Frames Per Second (FPS). This combined with the availability of the complete ISP development suite, makes the pipeline an ideal target for adaptation and expansion in various applications.

We acknowledge the Demo-ISP [3] for their FPGA implementation on AMD Xilinx Kria KV260 which inspired our initial work on the FPGA ISP. Infinite-ISP, however, builds on [3] and is the first complete ISP development framework in the public domain (supporting multiple FPGA vendors) with the goal of bringing researchers in academia and industry together to enable new vision applications, algorithm development and hardware deployment.

ISP Algorithm & Hardware Co-design Flow

As a hardware ISP development suite, Infinite-ISP provides the tools and the co-design flow methodology (shown in Fig. 1) needed for jointly designing the ISP algorithm and the hardware pipeline using various components of the Infinite-ISP package for a target application. While the hardware ISP development flow seemingly starts with the floating-point algorithm model, the set of feasible algorithms is actually constrained by the hardware ISP specifications and resources which in turn depend on the underlying application. The main hardware specifications are:

1. **Pixel throughput:** With $F = W \times H$ denoting the maximum frame size (width , height) and FPS the maximum frame rate, the pixel throughput $P_{\text{throughput}} = W \times H \times \text{FPS}$.

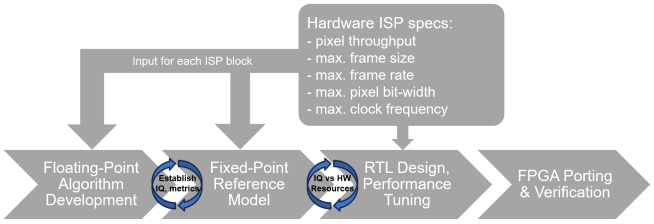

Figure 1. *Illustration of the ISP algorithm and hardware co-design flow and the major hardware specifications limiting the choice of algorithms.*

2. **Pixel bit-depth:** The maximum pixel bit-width determines the resource utilization of the ISP pipeline and the data throughput (Mb/s or Gb/s), which is important for vision SoC designers dealing with bus or memory bandwidths.
3. **Clock Frequency:** The clock frequency determines the ISP pipeline architecture design choice of processing one or multiple pixels per clock (PPC) cycle depending on the desired pixel throughput.

Next, we briefly discuss these hardware constraints in context of the reference design released as part of the Infinite-ISP package [10]. We confine our design to support the digital video signal interface (hsync, vsync, video_data, video_clock) with processing support for 1 PPC only. The pixel throughput $P_{\text{throughput}}$ of the reference ISP pipeline for a 10-bit wide frame of size 2592×1536 (4MP) operating at 30 FPS comes out to 119.44 MP/s. Operating at 1 PPC, the hardware ISP needs to be driven at a minimum clock frequency of 119.44 MHz to achieve the desired specifications. The addition of blanking interval pixels (17 pixels per row, 29 rows per frame for the reference design) increases the minimum clock frequency to 122.5 MHz. Given that each hardware ISP block is capable of processing pixels at a rate of 1 PPC, a clock frequency of 125 MHz should be able to achieve the specifications of our reference ISP.

Infinite-ISP Package

In this section, we provide an overview of the individual components of the Infinite-ISP and the workflow between these components required to develop an ISP for a target application.

ISP Pipeline Overview

The Infinite-ISP Image Pipeline comprises of 20 distinct algorithm blocks pipelined together to transform a RAW image from a Bayer image sensor into an output YUV or RGB image. The complete ISP pipeline, shown in Fig. 2, has four main pixel processing domains:

1. RAW Domain Processing – 10-bit Bayer
2. RGB Domain Processing – 10-bit 3 channel RGB
3. YUV Domain Processing – 8-bit 3 channel YUV
4. YUV/RGB Domain Processing - 8-bit 3 channel YUV/RGB

The ISP pipeline also has a 2A Statistics block, consisting of Auto White Balance (AWB) and Auto Exposure (AE) that can work without CPU intervention if needed. Table 1 describes the various ISP blocks and the algorithms used in the Infinite-ISP pipeline. We refer the reader to the references therein and Infinite-ISP [10] documentation for a detailed description of these algorithms.

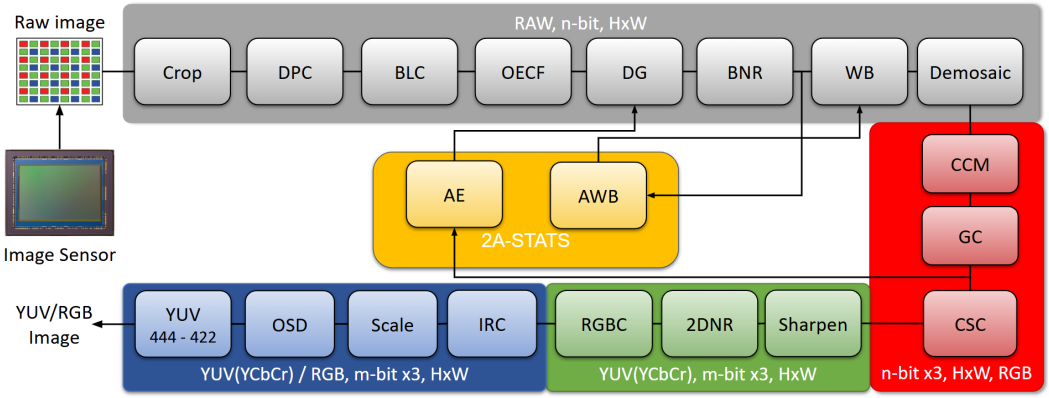

Figure 2. *A functional level block diagram description of the Infinite-ISP algorithm pipeline illustrating the different stages of the ISP.*

ISP Block	Name & Algorithm Description
Crop	Bayer pattern-safe cropping
DPC	Defect Pixel Correction: Modification of [11]
BLC	Black Level Correction: Sensor calibration
OECF	Opto-Electronic Correction Function: LUT from sensor calibration
DG	Digital Gain
BNR	Bayer Noise Reduction; Modified version of joint bilateral filter [12]
AWB	Auto White Balance: Modified gray world [13]
WB	White Balance
Demosaic	CFA Interpolation: Malvar He Cutler [14]
CCM	Color Correction Matrix: Sensor calibration
GC	Gamma Correction
AE	Auto Exposure: Skewness [15]
CSC	Color Space Conversion: BT.601, BT.709
2DNR	Spaial Noise Reduction: Non-Local means [16]
Sharpen	Simple unsharp Mask
RGBC	YUV to RGB Conversion: BT.601, BT.709
IRC	Invalid Region Crop: Simple image cropping
Scale	Image Downscaler: Nearest neighbour
OSD	On-screen display
YUV	Chroma Subsampling (444-422)

Table 1. Infinite-ISP pipeline blocks and algorithms.

Algorithm Development Model

The Infinite-ISP algorithm model is a floating-point implementation of the ISP pipeline shown in Fig. 2 with the primary purpose of algorithm and ISP pipeline structure exploration to establish a baseline image quality and improve on it. While not the major design goal at this stage of the ISP, various parameters like filter sizes of the ISP blocks in window and patch based algorithms are constrained by the hardware resources as described in the co-design section. The algorithm model facilitates the development of various ISP blocks and their effect on the image quality making it possible to identify a set of feasible algorithms which can then move forward for a fixed-point implementation.

Reference Model

The Infinite-ISP reference model is a fixed-point realization of the ISP pipeline obtained by converting all floating-point operands and operations to fixed-point. The floating-point to fixed-point conversion facilitates downstream hardware design, however, the resulting quantization errors negatively impact the image quality performance of the ISP pipeline. This is typically followed by an iterative process to improve the image quality of the fixed-point implementation while keeping in mind the expected hardware resource utilization until it achieves an acceptable performance against its floating-point counterpart.

RTL ISP

The Infinite-ISP RTL is the hardware ISP written in Verilog. The design process of RTL ISP blocks strictly adheres to the hardware co-design specifications. The RTL ISP, shown in Fig. 3, receives input pixels over the digital video signal input interface and outputs the rendered image pixels over the digital video signal output interface. ISP Parameters are fed to the ISP to configure the ISP for different imaging scenarios (variation in image sensor, illumination intensity, scene content etc). The developed RTL ISP block is verified against the reference model using RTL simulation. For the reference ISP design, all 4 Bayer patterns were tested with a test vector set of 8 images of size 2592×1536 obtained in different imaging scenarios (indoor, outdoor, noisy) adding diversity into our verification process. In addition, synthetic images were also used to test some corner cases on a per block basis as well as the entire ISP pipeline.

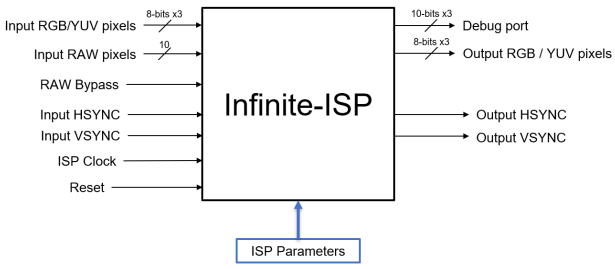

Figure 3. *Illustration of the RTL ISP with its I/O interfaces.*

The Infinite-ISP RTL pipeline has been architecturally divided into subsets: ISP-Lite and VIP (Video Input / Video & Image Processing).

- **ISP-Lite:** This consists of the RAW, RGB and YUV domain

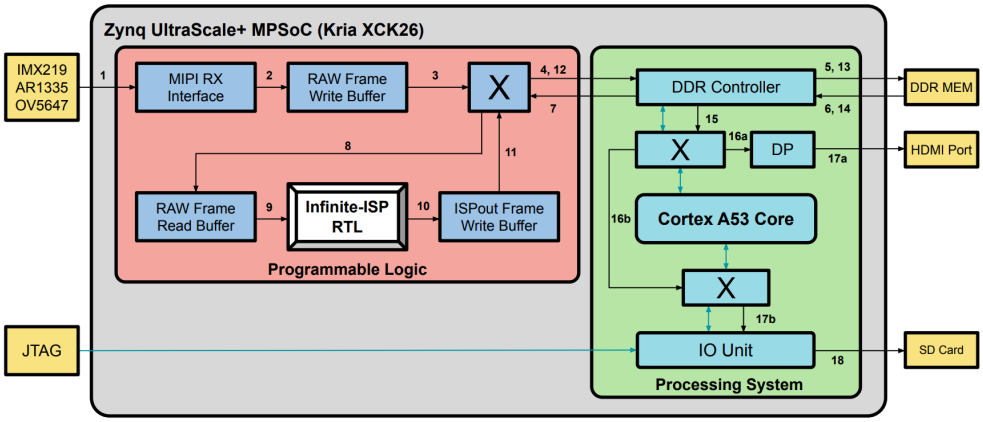

Figure 4. *System diagram for Infinite-ISP Kria reference design. The numbers over the arrows indicate the image pixels dataflow in the ISS i.e., from the image sensor (1) to RTL ISP (9) to the HDMI port (17a) and SD card (18). 'X' mark the AXI-Stream interconnects. For details, refer to [10].*

processing blocks illustrated in Fig. 2 and transforms RAW image sensor pixels to RGB for human perception.

- **VIP:** This consist of YUV/RGB processing blocks in Fig. 2 that deal with remedial processing of the ISP-Lite processed output, handling tasks like output image aspect ratio, down-scaling, on-screen display and output color space selection (RGB/YUV/YUV422).

This architectural split embeds modularity in the ISP design and enables multi-stream output capabilities. Based on the required application, more user-controlled output streams can be obtained by replicating multiple VIP sections at the output of the ISP-Lite, without multiplicative increase in the hardware resource utilization of the ISP. This highlights the Infinite-ISP's utility as a hardware ISP to be ported to vision SoC or FPGA camera systems. The RTL ISP includes the design sources and relevant testbench files. The design can be taken through the ASIC flow or the FPGA flow, depending on the user's requirement. Infinite-ISP currently provides the FPGA flow of the RTL ISP.

FPGA Implementation

Infinite-ISP FPGA implementation constitutes running the FPGA flow for the ISP RTL and implementing a complete camera system by interfacing it with image sensors as well as display and secondary storage devices, serving as a solid demonstration of the performance capabilities and image quality results of the tunable hardware Infinite-ISP. The RTL ISP is designed in a FPGA-vendor agnostic manner. Currently, the RTL ISP has been ported over to two FPGA devices: Xilinx Kria XCK26 and Efinix Ti180. Their end-to-end processing capabilities with Infinite-ISP reference design are:

1. Kria KV260 & Onsemi AR1335: 2048×1536 @ 30 FPS.
2. Efinix Ti180J484 & Sony IMX219: 1952×1112 @ 20 FPS.

For the purpose of this paper, we limit our discussion to the details of Kria KV260 Vision AI Starter Kit (XCK26 FPGA device).

The FPGA implementation involves developing a complete Image Subsystem (ISS) around RTL ISP, consisting of some essential components required to acquire data from the image sensor, process it in the RTL ISP and display it over an HDMI or VGA compatible screen. The image sensor, RTL ISP and display controller hardware support different protocols for pixel data and usually operate on different clock frequencies as well. Thus, the ISS constitutes all the IP cores, logic and clocking required to interface the image sensor to the RTL ISP input and target display to the RTL ISP output to ensure a continuous stream of pixels without any jitter. A simplified ISS is highlighted in Fig. 4.

Resources	LUT	FF	BRAM	DSP
Infinite-ISP RTL	47,867	31,762	50.5	140
Other ISS Logic	38,001	53,874	35	5
Total Used	85,868	85,656	85.5	145
Available	117,120	234,240	144	1,248
Utilization	73.3%	36.6%	59.4%	11.6%

Table 2. FPGA implementation statistics for Kria ISP design.

As derived in the co-design section, the placed and routed RTL ISP has a maximum design clock frequency of 125 MHz on both the Kria and Efinix FPGA implementations. The frame latency from image sensor to the HDMI display is under 100 ms on average. The ISP is initialized by firmware running on the CPU over the AXI4-Lite programming interface. A user menu over serial interface is available (on the Kria KV260 kit) with the following features:

- Image Sensor control: analog, digital gains, integration time
- Lens focus control: only for AR1335 IAS sensor module
- ISP configuration: runtime control of ISP parameters
- Burst capture: RAW-ISP output consecutive frame capture

The burst capture feature allows users to capture a limited number (150-190) of frame pairs of image sensor RAW and their corresponding ISP output into an SD card attached to the Kria KV260 Kit. This is of immense importance for not just the verification of FPGA-deployed ISP against the RM pipeline across consecutive frame inputs, but also allows users across the image and vision community access to simultaneous image sensor RAW and uncompressed ISP output frame pairs at 30 FPS.

ISP Tuning Tool

The Infinite-ISP tuning tool application, shown in Fig. 5, provides the interface for iteratively evaluating the image quality and tuning the Infinite-ISP deployed on the FPGA Platform. It facilitates obtaining crisp rendered output images from the ISP pipeline by tuning the ISP parameters and calculating the correct parameters for BLC, CCM, BNR/2DNR blocks when interfacing

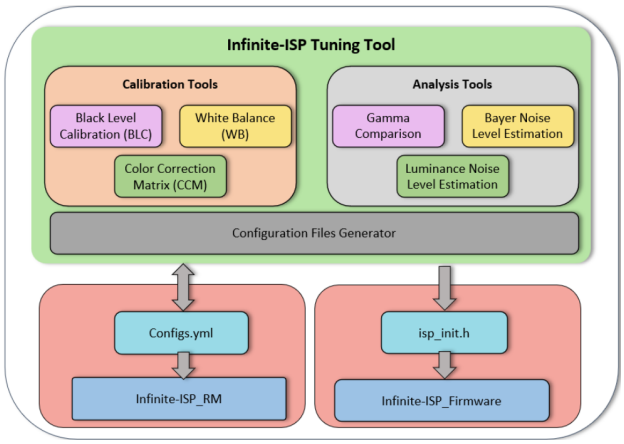

Figure 5. *Infinite-ISP tuning tool for calibration and analysis.*

a new image sensor. The tuning tool connects with the reference model and the FPGA platform and generates a .YAML configuration file for the reference model and also the corresponding ISP parameters file for the FPGA platform. The ISP parameters file is then read by the firmware to configure the FPGA platform. The current release of the tuning tool application consists of:

1. Calibration Tools: BLC, CCM, WB.
2. Analysis Tools: Gamma Compare, Bayer Noise Estimation, Luma Noise Estimation.

Infinite-ISP Image Quality

In this section, we compare the Kria KV260 Infinite-ISP reference design image quality with a Skype certified camera in a controlled indoor environment. We also compare the Infinite-ISP with two other ISPs using the NUS dataset [17].

Infinite-ISP vs Skype Certified Camera

The reference Infinite-ISP design based on the Kria KV260 and the Sony IMX219 image sensor has a resolution of 1080p and operates at 20 FPS. For comparison, we use the 2 MP (1920x1080) Skype certified camera on the Dell P2418HZ LED monitor whose image quality has been tuned by the vendor according to the Skype requirements for video conferencing application. We capture 16 sets of images of the X-rite ColorChecker Classic chart placed at 100-120 cm from the two cameras by varying the illumination intensity from 7 lux to 450 lux. Furthermore, the color temperature is set to either 4000K or 6500K for the captured images. The illumination intensity and color temperature are measured with the Vabira 300 light meter. We use the following metrics for our results:

- **Color accuracy:** Color accuracy is measured using the CIE1976 standard metrics: ΔC_u, ΔC_c and ΔE.
- **White balance:** White balance is analyzed based on saturation error using patch 21 and 22 of the X-rite ColorChecker Classic chart.
- **Noise suppression:** Noise suppression performance is measured in Pixel SNR using patches 19 to 24 (gray patches).

The results for both cameras averaged over the set of 16 images are shown in Table 3. It can be observed that for color rendering accuracy, the Infinite-ISP performs better than the Skype camera in terms of ΔC_c and slightly worse in terms of ΔC_u and ΔE. For white balance accuracy, Infinite-ISP outperforms the Skype camera in terms of the saturation error for both patches and renders a better neutral gray color. The Skype camera achieves better noise suppression than the Infinite-ISP for the three red, green and blue channels in terms of the Pixel SNR. For the luminance channel, however, Infinite-ISP performs better than the Skype camera by more than 3 dB indicating that Infinite-ISP is better able to reduce luma noise (thanks to the 2DNR block) but suffers from chroma noise dominant in low-light conditions. The same observation can also be made at illumination intensity levels of 450, 250 and 30 lux in Fig. 6 which illustrates a subset of the 16 captured images.

Metric	Infinite-ISP	Skype camera
ΔC_u	16.19	10.22
ΔC_c	8.37	9.16
ΔE	19.27	13.03
Saturation error 21	0.0432	0.0791
Saturation error 22	0.0778	0.1151
Pixel SNR red	28.77	36.24
Pixel SNR green	32.99	38.15
Pixel SNR blue	27.44	35.49
Pixel SNR luma	41.85	38.64

Table 3. X-rite ColorChecker Classic chart based analysis of color accuracy, white balance and noise suppression.

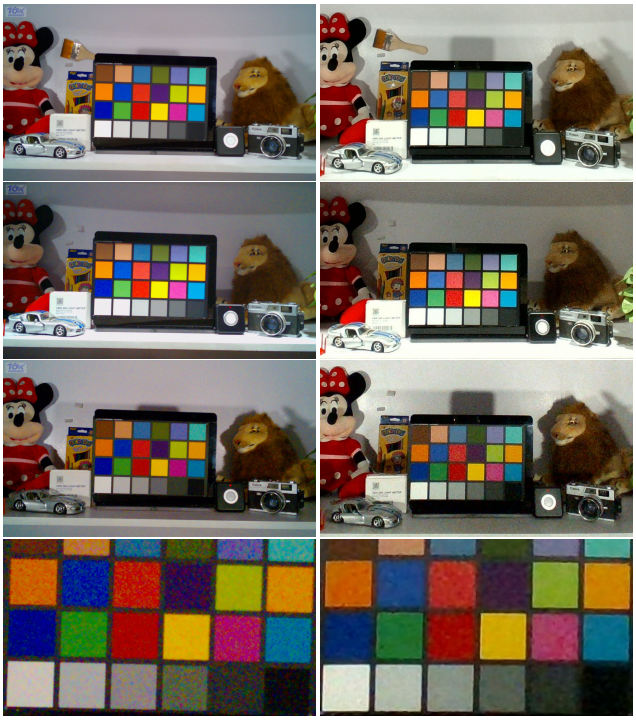

Figure 6. *Visual IQ comparison between the Infinite-ISP (left) and the Skype camera (right) done for 450 lux, 250 lux, 30 lux and 7 lux (zoomed in).*

Perceptual Image Quality Metrics

Next, we compare the Infinite-ISP to a Nikon ISP and Adobe post-processing software ISP using perceptual image quality metrics calculated on 8 images from the NUS dataset [17]. In particular, we compute the following metrics:

- **NIQE:** Naturalness Image Quality Evaluator. Its output

Metric	Nikon	Adobe	Infinite-ISP
NIQE	3.4760	4.3904	4.7844
BRISQUE	42.9328	27.0815	39.2168
PIQE	43.4843	21.3229	52.0134

Table 4. Perceptual IQ metrics for images from NUS dataset.

range is from $0 \rightarrow \infty$.

- **BRISQUE:** Blind / Reference-less Image Spatial Quality Evaluator. Its output range is from $0 \rightarrow 100$.
- **PIQE:** Perceptual Image Quality Evaluator. Its output range is from $0 \rightarrow 100$.

The final results averaged over the 8 images are shown in Table 4. For all three metrics, a lower number corresponds to a better image quality. It can be seen that Adobe achieves the significantly improved performance in terms of BRISQUE and PIQUE due to it being a software ISP without any hardware or time constraints. Infinite-ISP performs better than the Nikon ISP in terms of BRISQUE and achieves comparable performance for the other two metrics. Further development and community involvement can help improve the Infinite-ISP performance by integrating better algorithms into the ISP pipeline.

Infinite-ISP for Image & Vision Community

10xEngineers is actively involved in maintaining this open source project going forward, and we welcome any contributions from the open source community. We conclude this paper by discussing and motivating a few use cases of Infinite-ISP.

Open Source Community

Infinite-ISP is a platform for end-to-end ISP development suited to the needs of researchers, academia and industry alike. With permissive open-source licensing for the entire ISP development flow, users have complete insight into a fully customizable ISP pipeline, accelerating the hardware ISP development for various applications and market segments (surveillance, automotive, broadcast, medical etc). The open-source community can work with the Infinite-ISP to develop and test the performance of novel algorithms and improve on the baseline ISP pipeline. ISP firmware support for Kria KV260 platform can be extended to embedded Linux with libcamera and v4l2 libraries, allowing the software community to rapidly port useful camera applications.

AI-assisted ISP and Video Encoding

Infinite-ISP enables users to develop AI-assisted ISP algorithms that can be readily integrated into the pipeline structure at the algorithm and reference model levels. For hardware ISP development, the pixel data can be directed to Deep Learning Accelerators (DLA) using the RTL ISP thus enabling end-to-end development of AI-assisted ISP algorithms. Infinite-ISP can also be efficiently integrated into vision SoC solutions. The multi-stream architecture of the RTL ISP allows users to generate the required output color space, frame size and chroma-subsampling to interface with a video encoder IP with little to no modification. The modular nature of the RTL ISP paves the way for developing AI-assisted video encoding applications [19] in open-source.

Vision Research and Auto-Camera Tuning

The different components of Infinite-ISP package can be used for developing computer vision applications and smart vision systems. The ISP pipeline can be modified to produce two separate streams: One for human and one for computer vision. The output of the computer vision pipeline can be directly fed to a DLA running AI models for image and video content analytics. Using Infinite-ISP FPGA camera demonstrations, users can make use of the burst capture feature to build custom datasets with untampered raw image sensor data and uncompressed ISP output for their ISP, vision, AI and machine learning work. Infinite-ISP also enables automated tuning of the ISP parameters for metrics other than perceptual image quality. AI model developers can tune the ISP to generate images that maximizes their model's performance, or going a step further they can jointly optimize the AI vision model and the ISP parameters to improve performance [18].

References

[1] cruxOpen, OpenISP, GitHub, 2019. https://github.com/cruxopen/openISP

[2] Qiu Jueqin, fast-OpenISP, GitHub, 2021. https://github.com/QiuJueqin/fast-openISP

[3] bxinquan, DemoISP, GitHub, 2022. https://github.com/bxinquan

[4] openasic-org, xkISP, GitHub, 2022. https://github.com/openasic-org/xkISP

[5] ASICFPGA, Camera ISP. https://www.asicfpga.com/

[6] logicBRICKS, logiISP-UHD Image Signal Processing (ISP) UltraHD Pipeline, 2024. https://www.logicbricks.com/Documentation/Datasheets/IP/logiISP_hds.pdf

[7] Xfuse.AI, Phoenix ISP. https://xfuse.ai/phoenix-details/

[8] GOWIN, GOWIN ISP. https://www.gowinsemi.com/en/market/featured_detail/14/

[9] Lattice Semiconductor, Helion IONOS Image Signal Processing IP Portfolio. https://www.latticesemi.com/products/designsoftwareandip/intellectualproperty/ipcore/helioncores/helionionos

[10] 10x-Engineers, Infinite-ISP, GitHub, 2023. https://github.com/10x-Engineers/Infinite-ISP

[11] Liu Yongji and Yuan Xiaojun, A Design of Dynamic Defective Pixel Correction for Image Sensor, Proc. ICAIIS, 2020, pg. 713-716.

[12] Georg Petschnigg, Richard Szeliski, Maneesh Agrawala, Michael Cohen, Hugues Hoppe, and Kentaro Toyama, Digital photography with flash and no-flash image pairs, ACM Trans. Graph. 23, 2004.

[13] Gershon Buchsbaum, A spatial processor model for object colour perception, J. of the Franklin Institute, 1980, pg. 1-26.

[14] H. S. Malvar, Li-wei He and R. Cutler, High-quality linear interpolation for demosaicing of Bayer-patterned color images, Proc. ICASSP, 2004, pp. iii-485.

[15] Wen Chen and Xinglong Li, Exposure Evaluation Method Based on Histogram Statistics, Proc. EAME, 2017, pg. 290-293.

[16] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel, Non-Local Means Denoising, J. Image Processing On Line, 2011, pg. 208-212.

[17] Dongliang Cheng, Dilip K. Prasad, and Michael S. Brown, Illuminant estimation for color constancy: Why spatial-domain methods work and the role of the color distribution, J. Opt. Soc. Am., 2014.

[18] A. Mosleh, A. Sharma, E. Onzon, F. Mannan, N. Robidoux and F. Heide, Hardware-in-the-Loop End-to-End Optimization of Camera Image Processing Pipelines, CVPR, 2020, pg. 7526-7535.

[19] Hanwha Techwin, WiseStreamIII technology Whitepaper. https://www.hanwhavision.com/wp-content/uploads/2021/10/Whitepaper_WiseStreamIII_210331_EN-1.pdf