# Real-Time, Low-Latency Image Processing with High Throughput on a Multi-Core SoC

Barath Ramesh, Alan D. George, Herman Lam
NSF Center for High-Performance Reconfigurable Computing (CHREC)
Department of Electrical and Computer Engineering
University of Florida
Gainesville, FL 32611-6200
{ramesh, george, hlam}@chrec.org

***Abstract*—Real-time, low-latency, image processing with high throughput is vital for many time-critical applications in fields such as medical imaging, robotics, and wearable computers. Traditionally, FPGAs have often been employed to meet these requirements. However, due to the productivity challenges, using FPGAs may not be viable in some cases. Alternatively, the typical approach of processing an image on a CPU device at a frame level increases the processing latency of the computer-vision system. In this paper, we explore the feasibility of processing an input image at a *sub-frame* level on a multi-core SoC device, the 66AK2H12 from TI. Using *sub-frame* processing, we optimize and evaluate the performance of four kernels commonly used in image-processing pipelines. Our results show, for all four kernels in our study, that real-time, low-latency image processing with high throughput for HD-resolution images can be achieved using the multiple DSP cores available in the 66AK2H12 device.**

## I. Introduction

Many time-critical applications in fields such as medical imaging, robotics, and wearable computers require actions to be performed based on the processed image obtained from a computer-vision system. In such applications, there is a need for real-time, low-latency image processing with high throughput. Traditionally, designers have often employed reconfigurable-logic devices such as field-programmable gate arrays (FPGAs) to meet these requirements. However, due to the need for fast turnaround-times with embedded-systems design, FPGA-based platforms may not be a viable option in some cases. Alternatively, a frame-based approach has been employed on fixed-logic CPU devices for processing input images from an image sensor. However, the typical approach of processing an image at the frame level on such devices increases the processing latency of computer-vision systems. In this paper, we explore the feasibility of processing an input image at a *sub-frame* level on a fixed-logic, multi-core device, the 66AK2H12 from Texas Instruments [1]. The 66AK2H12 is a system-on-chip (SoC) device with a quad-core ARM Cortex-A15 and an eight-core, high-performance DSP processor well known for power efficiency in image-processing applications.

In this study, we employ a *sub-frame* approach to optimally map and study the throughput and the latency of commonly used computer-vision algorithms using the available eight DSP cores of the 66AK2H12 device. A *sub-frame* is defined as a set of consecutive lines of the input image. The image-processing kernels chosen for our study are non-uniformity correction (NUC), Sobel $3 \times 3$ edge detection, automatic gain control (AGC), and gamma correction (GC). These kernels are chosen because they are commonly used in image-processing pipelines within applications for our domain areas of interest.

The metric for real-time throughput in image processing is well defined in literature as *greater than* 30 frames-per-second (FPS). However, there is no well-defined metric for low latency. In [2] and [3], image processing is considered to have low latency, when the latency is on the order of $\Theta\left(\textit{number of lines} \times \textit{line rate}\right)$, where $\Theta$ represents the upper bound, *number of lines* is the size of a sub-frame, and *line rate* for HD-resolution images is approximately 30 $\mu$s. Thus, we use the aforementioned metrics to study the throughput and the latency of our design.

The contribution of this paper stems from the optimization and analysis of the four key computer-vision kernels on the available eight DSP cores of the 66AK2H12 device. Showcasing a data-parallel design for an image-processing pipeline that uses all the four kernels, our results for HD-resolution images show that high-throughput, low-latency image processing can be achieved using a single DSP core of the 66AK2H12 device at approximately 1W of power. Furthermore, we show that our design is scalable across the available eight DSP cores of the 66AK2H12 device. Finally, since the performance of all the kernels in our study is memory-bound, we compute the effective external memory bandwidth utilization of our DSP-optimized design for kernels to quantitatively analyze the throughput of our design.

The reminder of the paper is organized as follows. Section II reviews the related studies on real-time, low-latency image processing. Section III gives a brief overview of the 66AK2H12 device architecture. Section IV presents an overview of the kernels that are benchmarked for our study. In Section V, we describe how the kernels are optimally mapped onto the 66AK2H12 device architecture. Section VI presents the benchmarking results obtained for the kernels, and Section VII concludes the paper with key results.

## II. Related Research

There has been much work published on real-time, low-latency image processing with high throughput using FPGAs (e.g., [4]–[10]). In [4], the authors propose an FPGA-based architecture for edge-detection algorithms to achieve high throughput, low latency, and reduced memory requirement. In [6], the authors develop an embedded computer-vision hardware using an FPGA and a mobile CPU for real-time, low-latency image processing in robotics. Although our latencies on the 66AK2H12 device are significantly higher than latencies that can be obtained on FPGA-based platforms in prior works (e.g., [4]–[10]), one has to account for the productivity benefits of employing a fixed-logic device over reconfigurable-logic devices.

The study of high-throughput, low-latency image processing on fixed-logic, multi-core devices is limited in the literature. In [3], the authors demonstrate real-time, low-latency image processing for temporal adaptive filtering of ultrasound images on a DSP device. Although the approach in [3] is similar to ours, their study is limited to a single DSP core with low-resolution images of size $512 \times 350$. Our studies show that high-throughput, low-latency image processing can be achieved on a fixed-logic, multi-core SoC like the 66AK2H12 at a comparable power to that of the FPGA. We also show that our design is scalable across the multiple DSP cores of the 66AK2H12 device for HD-resolution images.

## III. 66AK2H12 Architecture Overview

The 66AK2H12 is an SoC with a quad-core ARM Cortex-A15 and an octal-core C66x DSP, with both fixed-point and floating-point precision capabilities. As shown in the architecture diagram in Figure 1, the C66x CorePac is the heart of the device providing high-performance computing through instruction- and data-level parallelism. The C66x CorePac consists of eight C66x DSP cores, each of which is equipped with 32 KB of L1 and 1 MB of L2 on-chip memory. The on-chip memories can be configured as part RAM and/or cache. The EDMA peripheral enables efficient direct memory access (DMA) between DDR3 and on-chip memory without CPU involvement. The ability to configure the on-chip memory as part RAM and/or cache along with EDMA allows the developer to tune the architecture to their application or kernel needs.

## IV. Kernel Overview

This section gives a brief overview of the kernels that were studied on the 66AK2H12 device for our studies. We describe the use-case and the functionality of each kernel.

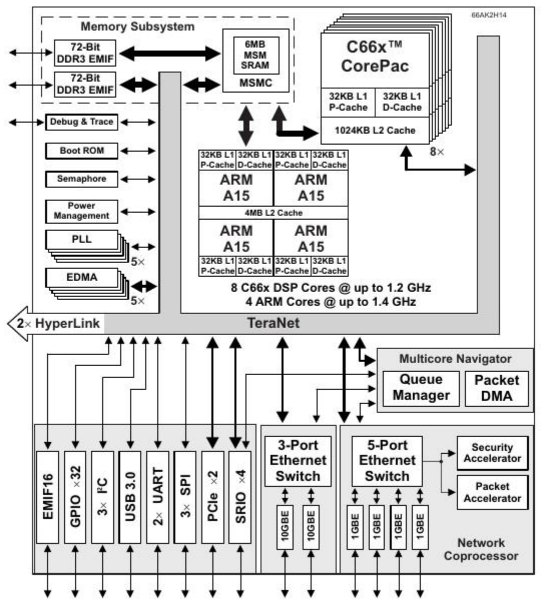

Fig. 1. 66AK2H12 device architecture [1]

### A. Non-Uniformity Correction (NUC)

The image sensors in cameras suffer from a bias called fixed pattern noise (FPN). The bias typically manifests as vertical aberrations on the image which reduces image quality. In order to reduce FPN and improve image quality, NUC must be employed on the raw-pixel data obtained from the image sensor. There are two components associated with the FPN found in image sensors, the offset component and the gain component. The calibration-based method is one of the most commonly used technique for NUC. In the calibration-based method, the FPN is commonly modeled as shown in Eq. 1. From Eq. 1, $a$ represents the gain and $b$ represents the offset component of the FPN respectively. Also, from Eq. 1, for a given pixel $i$, $y_i$ is the output pixel-intensity value and $x_i$ is the input pixel-intensity value. The NUC parameters $a$ and $b$ are typically obtained by calibrating the image sensor to output a dark image.

$$y_i = x_i \times a_i + b_i \tag{1}$$

### B. Automatic Gain Control (AGC)

AGC is employed to maintain the mean intensity of images displayed from the image sensor at a constant value. The AGC system increases the image sensor's gain if the scene is too dim and decreases the image sensor's gain if the scene is too bright based on the mean intensity of the image. The task of the AGC kernel is to calculate the mean intensities of images from the image sensor. For an input image $A$ of resolution $x \times y$, the mean intensity $M$ is calculated as shown in Eq. 2. From Eq. 2, $A(i, j)$ is the intensity level of a given pixel located at $(i, j)$ of the input image $A$.

$$M = \frac{1}{x \times y} \sum_{j=0}^{y-1} \sum_{i=0}^{x-1} A(i,j) \tag{2}$$

### C. *Sobel* 3×3 *Edge Detection*

Sobel 3×3 edge detection is one of the most commonly used edge-detection algorithm. As shown in Eq. 3 and Eq. 4, the input to the kernel consists of an image $A$ that is traversed by sliding two gradient masks, $G_x$ and $G_y$, of size $3 \times 3$ across all non-edge subspaces of $A$. Effectively, the Sobel 3×3 edge-detection kernel can be described as two, 2D convolutions (denoted as $*$) performed over the input image $A$. The first convolution $\nabla G_x$ is performed using the mask $G_x$ to detect the gradient changes along the $x$-direction of the input image $A$. The second convolution $\nabla G_y$ is performed using the mask $G_y$ to detect the gradient changes along the $y$-direction of the input image $A$. The final edge-detected image $\nabla G_{xy}$ is as shown in Eq. 4.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{3}$$

$$\begin{aligned} \nabla G_x &= G_x * A \\ \nabla G_y &= G_y * A \\ \nabla G_{xy} &= \sqrt{\nabla G_x^2 + \nabla G_y^2} \end{aligned} \tag{4}$$

### D. *Gamma Correction (GC)*

GC is a commonly used post-processing algorithm in the image-processing domain. Opto-electronic devices that capture and display digital images exhibit subtle nonlinear properties that require some form of correction. Gamma correction is an inexpensive technique employed to improve image quality in real-world, computer-vision systems. The most commonly used model for correction is the power-law curve dictated by the parameter gamma ($\gamma$). As shown in Eq. 5, the input pixel-intensity level $I$, represented using $n$ bits, is gamma-corrected to an output pixel-intensity level $I'$, based on the $\gamma$ value.

$$I' = (255) \times \left(\frac{I}{2^n}\right)^{\gamma} \tag{5}$$

## V. Mapping Kernels onto 66AK2H12 Device for High-Throughput, Low-Latency Image Processing

Low-latency image processing in computer-vision systems requires the input image to be processed with limited buffering. Figure 2 shows the software architecture for low-latency image processing on the 66AK2H12 device. Typically, computer-vision systems with multi-core processors process the input image over a complete frame using the available processing cores. With our approach, the input image is processed at a sub-frame level for achieving low-latency processing as shown in Figure 2. As mentioned before, a sub-frame is defined as a set of consecutive lines of the input image. Every sub-frame is processed using the available eight C66x DSP cores of the 66AK2H12 device. The size of the sub-frame is dependent on the number of cores used to process the input image. In our case, the size of a sub-frame is set at 3 lines/core for equal distribution of lines in HD-resolution images (1080 lines) across the available eight C66x DSP cores of the 66AK2H12 device.

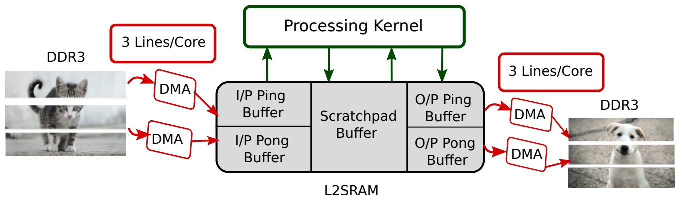

Fig. 2. Software architecture for low-latency processing

The L2 on-chip memory of the C66x DSP cores is partitioned into part RAM and part cache, with the RAM section referred to as L2SRAM. The L1 on-chip data memory is configured as cache. As shown in Figure 2, we employ ping-pong line buffers in L2SRAM to transfer the input sub-frame from DDR3 via DMA, and also to output the processed sub-frame to DDR3 via DMA. Thus, we mask the DMA-transfer time of a sub-frame behind the compute time of the kernel.

In our study, the processing latency is defined as the time interval between the start of DMA transfer for the first sub-frame to the end of DMA transfer for the processed first sub-frame. Thus, we process the input image as multiple sub-frames, and in the process decrease the latency involved to output the processed image data from the kernel.

In the following sub-sections, we describe the optimizations performed to map the kernels in our study in a low-latency fashion on the 66AK2H12 device. We also perform memory-level and SIMD optimizations to improve the performance of the kernels on the C66x DSP core.

### A. *Non-Uniformity Correction*

For the NUC kernel, the NUC parameters $a$ and $b$ of a given pixel are required to correct the intensity value of that pixel in a given sub-frame, thus, making the kernel memory-intensive. In our design, to improve the utilization of available DDR3 bandwidth, for every sub-frame of the input image that we DMA into L2SRAM, we also DMA the corresponding sub-frame's NUC parameters into L2SRAM. Also, the NUC parameters, $a$ and $b$ of a given pixel, are concatenated and stored in the external DDR3 memory for ease of DMA.

### B. *Automatic Gain Control*

The core of the AGC algorithm is reduction operation, that is, reducing the input-image matrix into a single value using summation. The performance of the reduction

```
for (i = 0; i < chunkSize; i+=8)
    {
      imgIn_8 = _amem8(&imgIn[i]);
      _amem8(&imgOut[i]) = imgIn_8;

      imgIn1_4 = (imgIn_8 ) & 0xFFFFFFFF;
      imgIn2_4 = (imgIn_8 >> 32) & 0xFFFFFFFF;

      imgIn3210 = _unpkbu4(imgIn1_4);
      imgIn7654 = _unpkbu4(imgIn2_4);

      sum_4 = _dadd2(imgIn3210, imgIn7654);
      sum1_2 = _unpkh2(_loll(sum_4));
      sum2_2 = _unpkhu2(_hill(sum_4));

      sum12 = _dadd(sum1_2, sum2_2);
      sum_t = _dadd(sum_t, sum12);
    }

sum_opt = _hill(sum_t) + _loll(sum_t);
```

Fig. 3. C code with intrinsic optimizations for AGC kernel

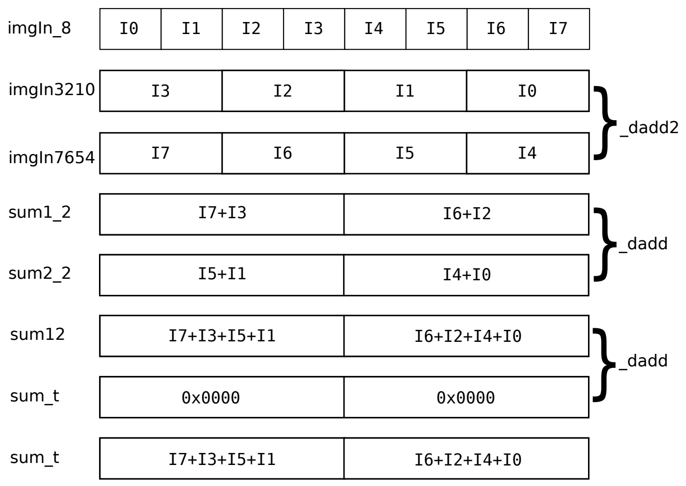

Fig. 4. Optimized AGC kernel using SIMD intrinsics

operation is memory-bound. We perform intrinsic-based optimizations in our design to improve the utilization of available DDR3 bandwidth. The SIMD intrinsics [11], *amem8*, *unpkbu4*, *dadd2*, *unpkh2*, *unpkhu2*, *dadd*, and *dadd2*, allow for octal-word processing (8-bit precision for pixels) on the C66x DSP core.

Figure 3 shows an excerpt of our *C code* for the primary computation loop with intrinsic optimizations for the AGC kernel. Intrinsics reduce the iteration count of the ***for*** loop by eight times. Also, after the compiler pipelines the ***for*** loop, the iteration interval for our C code is twice that of the optimized code with SIMD intrinsics. Thus, our SIMD optimizations result in improving the throughput and the latency of the AGC kernel. Figure 4 shows how the reduction operation is performed for the first iteration of the ***for*** loop shown in Figure 3, where $I0-I7$ represent the first eight pixels of the input sub-frame.

### C. *Sobel* $3\times3$ *Edge Detection*

The core of the Sobel $3\times3$ edge-detection algorithm is a 2D convolution operation. Hence, the kernel requires reading the surrounding pixel values of the input pixel to produce the processed output pixel. Since the Sobel $3\times3$ edge-detection mask has a radius of one, the input sub-frame size of the kernel is 5 lines/core and the output sub-frame size of the kernel is 3 lines/core. Optimized code with intrinsics available in TI's C66x IMGLIB [12] is used for performing the Sobel $3\times3$ edge-detection algorithm on the data available in L2SRAM to improve throughput. In our current method for Sobel $3\times3$ edge detection, we read every line of the input image twice to output the edge-detected image. To improve design-efficiency, circular buffer of few KB (sub-frame size for HD-resolution images) is required in L2SRAM, which is not trivial to optimally design on a fixed-logic device. We plan to explore this in our future work.

### D. *Gamma Correction*

The GC kernel is a one-to-one intensity mapping of a given pixel value to another pixel value. To improve DDR3 bandwidth utilization, the intensity mapping can be pre-computed based on Eq. 5 and stored in a lookup table (LUT). In our DSP-optimized design, we pre-compute and store the LUT values inside L2SRAM of the C66x DSP core. For a given sub-frame, we perform a lookup of intensity values of the pixels in the sub-frame to produce the gamma-corrected sub-frame.

## VI. Results

In our study, all kernels were evaluated using an EVMK2H evaluation module. The EVMK2H has a 66AK2H12 device clocked at 1 GHz and equipped with 1 GB of DDR3 memory. After the design of kernels was optimized for a single C66x DSP core, OpenCL was used to parallelize the kernels across all the eight C6xx DSP cores of the 66AK2H12 device. The performance of our DSP-optimized design was tested using an 8-bit grayscale image of resolution $1920\times1080$. The input and the output images were stored in DDR3 memory. The performance measurements of our design include FPS and latency, which is described as follows:

1) ***FPS***: Time taken to process entire input image in terms of throughput.
2) ***Latency***: Time taken to output first sub-frame of processed image.

The performance of all the kernels in our study is memory-bound. In order to quantitatively analyze the performance of our DSP-optimized design, we calculate the effective DDR3 memory bandwidth utilization $\alpha$, as shown in Eq.6. From Eq. 6, the image resolution in our case is $1920\times1080$ and $N$ is the number of bytes per pixel that is read/written to DDR3 for a given kernel. Also, the DDR3 memory is clocked at 500 MHz with a word length of 8 bytes, which translates to a maximum theoretical bandwidth of 8 GB/sec.

$$\alpha = 100\times\left(\frac{\text{FPS}\times\left(\text{Image resoultion}\right)\times\left(N\ \text{bytes}\right)}{\text{DDR3 Bandwidth}}\right) \tag{6}$$

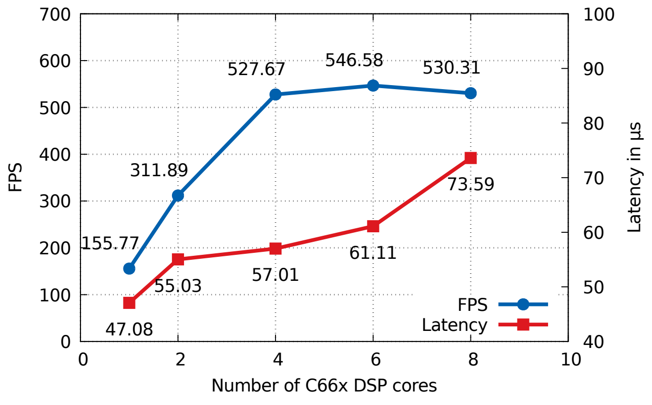

Fig. 5. Throughput and latency of NUC kernel

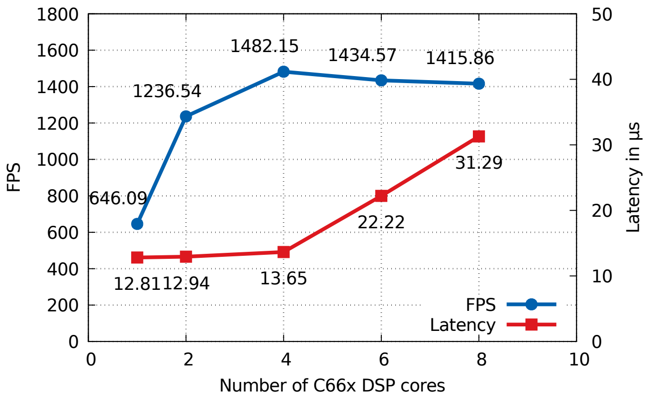

Fig. 6. Throughput and latency of AGC kernel

### *A. Non-Uniformity Correction*

Figure 5 shows the throughput and the latency of the NUC kernel across varying C66x DSP cores. We achieve real-time throughput of 155.77 FPS using a single C66x DSP core and high throughput of 546.58 FPS using six C66x DSP cores. The throughput of the kernel does not show linear speedup beyond four C66x DSP cores and saturates at six C66x DSP cores. Increasing the number of C66x DSP cores increases the amount of data read from DDR3 (pixel data and NUC parameters), thus, resulting in a bottleneck for achieving near-linear speedup with more than four C66x DSP cores. From Eq. 6, for the NUC kernel, $N$ is calculated to be 6: one byte for pixel read, one byte for pixel write, and four bytes to read the NUC parameters $a$ and $b$ for each pixel. Using Eq. 6, the effective DDR3 bandwidth utilization of our design peaks at 85.00% with six C66x DSP cores.

For the NUC kernel, we achieve a minimum latency of 47.08 $\mu$s using a single C66x DSP core and a maximum processing latency of 73.59 $\mu$s using eight C66x DSP cores. Recall from Section I, our latency results are in the order of $\Theta\left(\textit{number of lines} \times \textit{line rate}\right)$ and satisfy the constraint for low-latency image processing. The latency increases with more C66x DSP cores as the amount of data read from DDR3 increases with more cores. Thus, we achieve real-time, low-latency image processing, with high throughput for the NUC kernel.

### *B. Automatic Gain Control*

The AGC kernel is a reduction-style kernel and hence the performance of the kernel is memory-bound. Figure 6 shows the throughput and the latency of our intrinsic-optimized AGC kernel with varying C66x DSP cores. We achieve high throughput of 1482.15 FPS for the kernel using four C66x DSP cores. The effective DDR3 bandwidth utilization of the AGC kernel using four C66x DSP cores is 76.83%. Using more C66x DSP cores does not yield speedup once the DDR3 memory bandwidth is saturated.

For the AGC kernel, we achieve a minimum latency of 12.81 $\mu$s using a single C66x DSP core and a maximum latency of 31.29 $\mu$s using eight C66x DSP cores. Our throughput and latency results for the AGC kernel show that the 66AK2H12 device is a suitable candidate for real-time, low-latency image processing, with high throughput for *reduction-style kernels*.

### *C. Sobel* $3 \times 3$ *Edge Detection*

Figure 7 shows the throughput and the latency of the Sobel kernel across varying C66x DSP cores. We achieve high throughput of 1084.77 FPS for the Sobel kernel using eight C66x DSP cores. The throughput of the Sobel kernel scales linearly until four C66x DSP cores and saturates at eight C66x DSP cores. Our effective DDR3 bandwidth utilization peaks at 56.23% with eight C66x DSP cores. As mentioned before, in our design of the kernel, we read every line of the input image twice. Thus, we do not saturate the available DDR3 memory bandwidth from the algorithm's perspective.

For the Sobel $3 \times 3$ edge-detection kernel, we achieve a minimum latency of 26.56 $\mu$s using a single C66x DSP core and a maximum latency of 50.45 $\mu$s using eight C66x DSP cores. The Sobel algorithm is a convolution-style kernel. Thus, our results show that real-time, low-latency image processing with high throughput can be achieved on the 66AK2H12 device for *convolution-style kernels*.

### *D. Gamma Correction*

The GC kernel is a one-to-one mapping of pixel intensity values. Figure 8 shows the throughput and the latency of our DSP-optimized GC kernel with varying C66x DSP cores. We achieve high throughput of 1365.68 FPS for the GC kernel using six C66x DSP cores. The effective DDR3 bandwidth utilization of the kernel using six C66x DSP cores is 70.80%.

For the GC kernel, we achieve minimum latency of 26.67 $\mu$s using a single C66x DSP core and a maximum latency of 29.87 $\mu$s using eight C66x DSP cores. The kernel does not require any computation, but a simple lookup of incoming pixel values. With the LUT stored in L2SRAM of each of

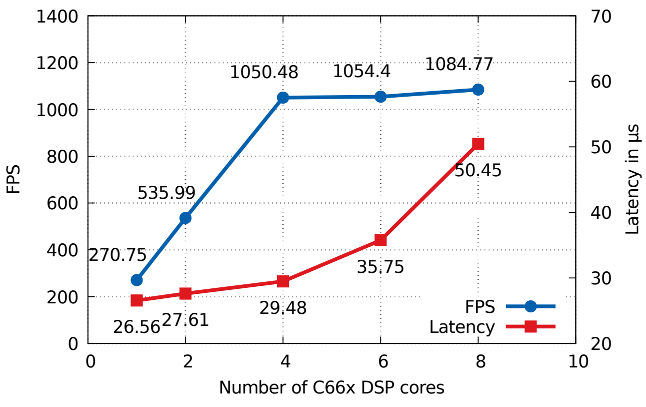

Fig. 7. Throughput and latency of Sobel 3x3 edge-detection kernel

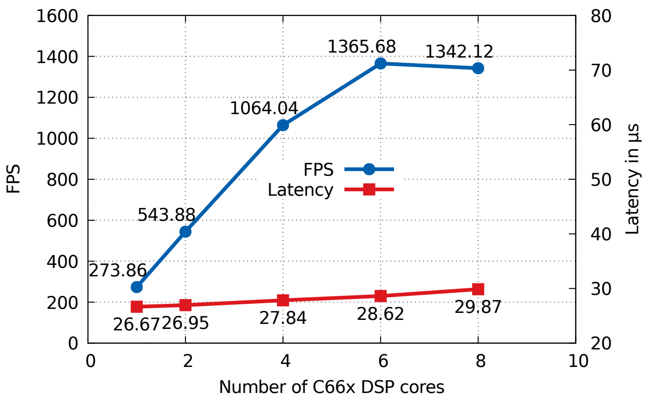

Fig. 8. Throughput and latency of GC kernel

the C66x DSP cores, our results show that real-time, low-latency image processing with high throughput can be achieved on the 66AK2H12 device for a post-processing kernel like the GC kernel.

## *E. Image-Processing Pipeline*

For a final demonstration, the four kernels in our study are connected in a pipelined fashion as shown in Figure 9. Figure 10 shows the throughput and the latency of the image-processing pipeline with all the four kernels in our study. We achieve real-time throughput of 68.92 FPS with a single C66x DSP core and high throughput of 484.32 FPS using eight C66x DSP cores. The throughput of the image-processing pipeline scales almost linearly across the available eight C66x DSP cores.

For the image-processing pipeline, we achieve a minimum latency of 91.56 $\mu$s using a single C66x DSP core and a maximum latency of 107.73 $\mu$s using eight C66x DSP cores. Considering that a single C66x DSP core consumes approximately 1W of power [13], we achieve real-time, low-latency image processing with high throughput for commonly used computer-vision kernels in an image-processing pipeline at low power.

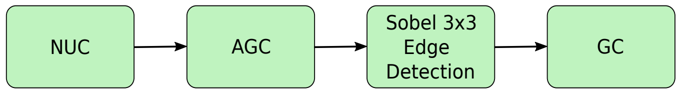

Fig. 9. Kernels in image-processing pipeline

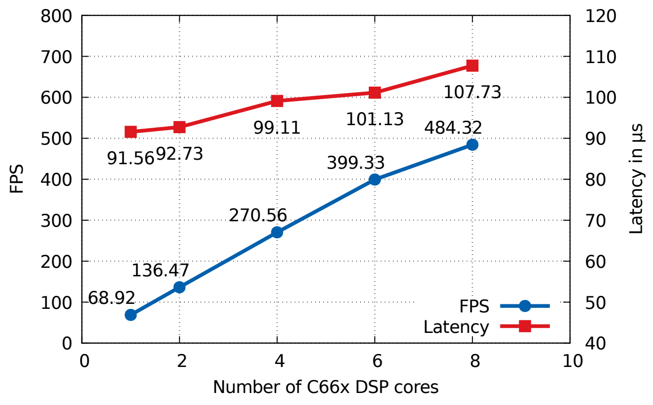

Fig. 10. Throughput and latency of image-processing pipeline

## VII. Conclusions

Real-time, low-latency image processing with high throughput is vital for many time-critical applications in fields such as medical imaging, robotics, and wearable computers. The typical approach of processing the input image by frame increases the processing latency of computer-vision systems. In this paper, we explore the feasibility of processing the input image at a sub-frame level on the fixed-logic, multi-core 66AK2H12 SoC device. Also, we analyze the performance of the kernels studied in terms of throughput, latency, scalability, and effective DDR3 bandwidth utilization.

Our results for the image-processing pipeline with all the four kernels in our study show that real-time throughput of 68.92 FPS and a low latency of 91.56 $\mu$s can be achieved at approximately 1W of power. Furthermore, we achieve high throughput of 484.32 FPS and a low latency of 107.73 $\mu$s using eight C66x DSP cores for the image-processing pipeline. The DSPs are well known for better power efficiency in image-processing applications over conventional CPUs. Although FPGA-based platforms achieve significantly better latencies than the DSP, one has to account for the productivity benefits of employing a fixed-logic device over reconfigurable-logic devices. In summary, the current-generation, multi-core SoCs, such as the TI's 66AK2H12, can provide a viable solution for today's real-time, time-critical, vision-based applications.

## Acknowledgments

This work was supported by CHREC members and the I/UCRC Program of the National Science Foundation under Grant No. IIP-1161022. The authors would like to thank Gongyu Wang, doctoral candidate at CHREC for his feedback during this work.

## REFERENCES

[1] Multicore dsp+arm keystone ii system-on-chip (soc). [Online]. Available: http://www.ti.com/lit/ds/symlink/66ak2h12.pdf

[2] Z. Zivkovic, V. Kliger, R. Kleihorst, A. Danilin, B. Schueler, G. Arturi, C.-C. Chang, and H. Aghajan, "Toward low latency gesture control using smart camera network," in *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, June 2008, pp. 1–8.

[3] O. Bockenbach, I. Wainwright, M. Ali, and M. Nadeski, "Achieving low latency, reduced memory footprint and low power consumption with data streaming," in *High Performance Extreme Computing Conference (HPEC), 2015 IEEE*, Sept 2015, pp. 1–7.

[4] P. R. Possa, S. A. Mahmoudi, N. Harb, C. Valderrama, and P. Manneback, "A multi-resolution fpga-based architecture for real-time edge and corner detection," *IEEE Transactions on Computers*, vol. 63, no. 10, pp. 2376–2388, Oct 2014.

[5] M. Kraft, M. Fularz, and A. Kasinski, "System on chip coprocessors for high speed image feature detection and matching," in *Proceedings of the 13th International Conference on Advanced Concepts for Intelligent Vision Systems*, ser. ACIVS'11. Springer-Verlag, 2011, pp. 599–610.

[6] D. Honegger, H. Oleynikova, and M. Pollefeys, "Real-time and low latency embedded computer vision hardware based on a combination of fpga and mobile cpu," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 4930–4935.

[7] H. HÃijbert, B. Stabernack, and F. Zilly, "Architecture of a low latency image rectification engine for stereoscopic 3-d hdtv processing," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 5, pp. 813–822, May 2013.

[8] N. Petrovsky, A. Stankevich, and A. Petrovsky, "Fpsoc using xilinx zynq for medical image coding based on the quaternionic paraunitary filter banks," in *2015 17th International Conference on E-health Networking, Application Services (HealthCom)*, Oct 2015, pp. 596–597.

[9] O. Benderli, Y. C. Tekmen, and N. Ismailoglu, "A real-time, low latency, fpga implementation of the 2-d discrete wavelet transformation for streaming image applications," in *Digital System Design, 2003. Proceedings. Euromicro Symposium on*, Sept 2003, pp. 384–389.

[10] Z. Yu, L. Claesen, Y. Pan, A. Motten, Y. Wang, and X. Yan, "Soc processor for real-time object labeling in life camera streams with low line level latency," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, June 2014, pp. 345–348.

[11] Tms320c6000 optimizing compiler, user guide. [Online]. Available: http://www.ti.com/lit/ug/spru187u/spru187u.pdf

[12] Tms320c64x+ dsp image/video processing library, programmer's guide. [Online]. Available: http://www.ti.com/lit/ug/spruf30a/spruf30a.pdf

[13] D. Wang and M. Ali, "Synthetic aperture radar on low power multi-core digital signal processor," in *High Performance Extreme Computing (HPEC), 2012 IEEE Conference on*, Sept 2012, pp. 1–6.