A Compact Real-Time Thermal Imaging System Based on Heterogeneous System-on-Chip

Hyun Woo Oh*, Cheol-Ho Choi, Jeong Woo Cha, Hyunmin Choi, Jung-Ho Shin, and Joon Hwan Han
Infra Technology R&D Center
Hanwha Systems, Seongnam, Republic of Korea
Email: {hyunwoo.oh, cheoro1994, jeongwoo.cha, hyunmin.choi, jh.hoya.shin, joonhwan.han}@hanwha.com

***Abstract*—This paper presents a real-time embedded thermal imaging system architecture for compact, energy-efficient, high-quality imaging utilizing heterogeneous system-on-chip (SoC) and uncooled infrared focal plane arrays (IRFPAs). Unlike previous systems that organized separate devices for complex image processing, our system provides integrated image processing support for robust sensor-to-surveillance. The image processing organizes two algorithm stacks: a non-uniformity correction stack to mitigate the distinctive noise vulnerabilities of uncooled IRFPAs, and an image enhancement stack including contrast enhancement and temporal noise filters. We optimized these algorithms for domain-specific factors, including asymmetric multiprocessing (AMP), cache organization, single instruction multiple data (SIMD) instructions, and very long instruction word (VLIW) architectures. The implementation on the TI TDA3x SoC demonstrates that our system can process 640×480, 60 frames per second (FPS) videos at a peak core load of 57.5% while consuming power less than 2.2 W for the entire system, denoting the possibility of processing the 1280×1024, 30 FPS videos from the cutting-edge uncooled IRFPAs. Additionally, our system improves power efficiency by 9.42% and 9.96% at 30 and 60 FPS, respectively, compared to the state-of-the-art when executing similar image processing algorithms.**

***Index Terms*—Thermal Imaging, Image Processing, Heterogeneous Computing, SIMD, Real-Time Embedded Systems.**

I. Introduction

Recent advancements in thermal imaging, driven by the development of uncooled infrared focal plane arrays (IRFPA), have extended their application beyond traditional safety and military uses. These IRFPAs, which operate without thermoelectric coolers, enable more compact and cost-effective designs, opening new possibilities in fields such as advanced driver assistance systems (ADAS) [1], the Internet of Things [2], night vision [3], and computer vision [4]–[6].

However, thermal imaging systems face notable challenges that limit performance and practical utilization. One major challenge is the demanding workload of non-uniformity correction (NUC) required to address uncooled IRFPAs' noise vulnerability. This noise, mainly consisting of fixed-pattern noise (FPN) [7] and thermal drift noise (TDN) [8], significantly degrades image quality [9], [10]. While practical, traditional pixel-wise NUC approaches using calibrated coefficients are computationally intensive due to large memory footprints, limiting use in compact, real-time systems. Though studies have proposed scene-based NUCs reducing the footprints and are optimized for hardware acceleration, those possess adverse effects due to reliance on convergence for guided images [11].

Moreover, achieving high-quality thermal imaging necessitates advanced contrast enhancement algorithms to address the narrow dynamic range of intensity distribution [12]–[14]. These algorithms are crucial not only for enhancing surveillance but also for improving machine learning applications, underscoring the importance of high-performance contrast enhancement [15]. However, these algorithms require substantial computing power, posing a challenge for real-time imaging systems that must balance compactness with consistent operational performance. Prevailing compact imaging systems compromise by limiting their functions to basic NUC algorithms and simple contrast enhancements, e.g., histogram equalization (HE), relegating more advanced algorithms to auxiliary high-performance devices, such as field-programmable gate arrays (FPGA) [3], [16]–[21] and graphics processing units (GPU) [1], [22]. This separation degrades cost-effectiveness and energy efficiency and increases the overall system volume and power consumption. Although integrated solutions on single FPGAs [23]–[25] have been explored, including our previous work [24], they still struggle to manage intensive algorithms.

To address these limitations, we introduce an integrated hardware/software (HW/SW) system architecture for real-time thermal imaging, utilizing an embedded heterogeneous system-on-chip (SoC) and uncooled IRFPA. Our approach consolidates high-performance NUC and advanced image processing algorithms within a compact, energy-efficient framework, eliminating the need for external high-performance devices. Implemented on TI's TDA3x SoCs, our system successfully processes 640×480 resolution videos at 60 frames per second (FPS), utilizing a maximum core load of 57.5% and consuming power less than 2.2 W for the total system. This performance indicates our system's suitability for compact applications and the potential for further refinement to accommodate higher-resolution videos.

The primary contributions of this work are listed as follows:

- An energy-efficient HW/SW architecture for compact, real-time thermal imaging based on heterogeneous SoCs.
- A fully integrated image processing architecture from uncooled IRFPA sensor to surveillance.
- Domain-specific software implementations of NUC and image enhancement algorithms for single instruction multiple data (SIMD) operations and very long instruction word (VLIW) processor architectures, enabling real-time execution of intensive algorithms.

2325-1301/24/$31.00 ©2024 IEEE
DOI 10.1109/RTCSA62462.2024.00023

II. Related Works and Motivations

A. Real-time Thermal Image Processing

1) Compact Thermal Imaging Systems: Compact thermal imaging predominantly relies on FPGAs, as evidenced by the works in [23]–[25]. One state-of-the-art (SOTA) system integrates NUC with two-point correction (TPC) and run-time defect pixel correction (DPC) alongside the HE for contrast enhancement, achieving 640×480 resolution at 30 FPS with a power consumption of 1.8 W for the FPGA chip [23]. Our previous study utilized an SoC FPGA, combining advanced NUC including thermal drift compensation (TDC), TPC, and hybrid DPC, with min-max stretching for contrast, achieving 640×480 at 60 FPS with 2.098 W power for the FPGA [24].

These approaches have advanced compact thermal imaging but have revealed a critical trade-off between economic, energy costs, and performance. Existing systems often rely on basic contrast enhancement techniques, such as the HE and the min-max stretching, which perform only global enhancement but sacrifice local detail, restricting their applicability in diverse scenarios. This necessitates sophisticated algorithms for high-quality imaging and highlights the need for more cost-effective, adaptable platforms, such as heterogeneous SoCs, which reduce economic and energy costs and improve quality.

2) Advanced Contrast Enhancement Techniques: The demand for high-quality thermal imaging across various applications has led to the adoption of advanced techniques such as contrast-limited adaptive histogram equalization (CLAHE). The CLAHE significantly improves image quality by applying tiled HE and interpolating those tiles [26], [27], and has been widely used across various domains, including thermal imaging [26], [27]. Despite its effectiveness, the computational demands of the CLAHE traditionally necessitated powerful hardware, such as FPGAs, for real-time embedded video applications. Pioneering work [18] and subsequent studies [19]–[21] have extended these capabilities, culminating in SOTA systems processing 3840×2160 60 FPS videos [21].

However, relying on FPGA-based solutions poses challenges for developing compact, cost-effective systems. Even lower-cost FPGA solutions, such as [16], have yet to fully address concerns on energy efficiency and overall system costs compared to standalone SoC solutions without auxiliary high-performance devices. Recognizing the potential of embedded lightweight heterogeneous SoCs, our work explores their feasibility for effectively deploying CLAHE. Consequently, this work introduces an optimized SW implementation for real-time processing on a heterogeneous SoC platform.

B. Embedded Heterogeneous SoCs

Various systems based on heterogeneous SoCs have been proposed in response to the demand for balancing throughput, energy efficiency, and cost [28]. For compact systems, some exclude application-level cores and GPUs to save power and reduce size while including microcontroller-level and DSP cores for adequate processing capabilities. The work in [29] applied one such SoC, the TI TDA3x, for ADAS applications, showcasing the efficiency of lightweight heterogeneous SoCs.

1) Software Optimization for DSP: Modern DSPs combine SIMD and VLIW concepts, offering a general-purpose solution with higher energy efficiency than scalar processor cores. This attribute makes them attractive for diverse applications, such as machine learning [30], databases [31], error detection [32], and image processing [33], [34].

Vectorization plays a crucial role in SIMD architectures. Recent works have focused on auto-vectorization at the compiler level [35], [36] to enhance domain-agnostic cases using high-performance superscalar cores based on Intel AVX ISA. However, these enhancements primarily benefit high-performance cores, leaving the potential for further efficiency gains in energy-efficient VLIW processors. We have explored manual techniques to efficiently order the VLIW-optimized vectorization of the compiler, i.e., instruction packing, for lightweight DSPs executing various fixed-point kernels.

2) Vision Accelerator: Work in [37] designed a hardware accelerator, embedded vision engine (EVE), for vision applications. This tightly couples a RISC core (ARP32) with a programmable 16-way vector coprocessor (VCOP) operating in parallel. The VCOP uses a hardwired SIMD datapath connected to cache memory independent of system memory, enabling fast 16-way 32-bit operations and minimizing cache overheads, ideally achieving 4-5× the throughput compared to equivalent DSPs. Our work leverages EVE's computing power for intensive image processing, resulting in lower latency and power consumption than a DSP-only implementation.

III. System Architecture

A. Image Processing Architecture

As illustrated in Fig. 1, our image processing architecture encompasses two primary algorithm stacks: the NUC stack and the image enhancement stack. The former mitigates the intrinsic non-uniformities of IRFPA caused by various origins, while the latter enhances images to achieve multiple objectives.

1) Non-uniformity Correction: The NUC stack consists of the TDC, TPC, flat field correction (FFC), and DPC.

- **TDC**: The TDC mitigates unintended response variations due to temperature changes in the IRFPAs, utilizing a polynomial fitting [38] where the IRFPA temperature (T_{FPA}) is variable. By adjusting the axis using the subtraction of a reference temperature (T_{Ref}) from the T_{FPA}, we reduce the dynamic range of coefficients, minimizing quantization errors.

- **TPC**: The TPC corrects the FPN, which primarily originates from the readout integrated circuit (ROIC) [39], [40]. By applying calibrated gain and offset values, TPC aligns each pixel's response to a reference linear model, alleviating spatial non-uniformity across the sensor.

- **FFC**: Triggered on an as-needed basis, the FFC adjusts for discrepancies between actual responses and our model's predictions. It captures pixel responses from a shuttered flat reference source and then adjusts the TPC offsets to normalize response levels, temporarily pausing the video output.

- **DPC**: The DPC replaces defective pixels by averaging the intensities of surrounding pixels, using both pre-measured maps and run-time threshold-based detection.

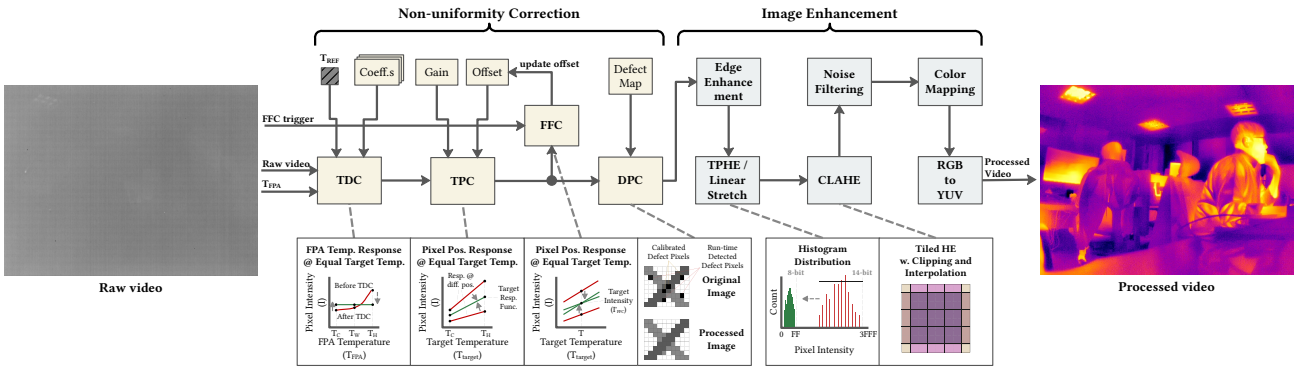


Fig. 1: The proposed image processing architecture for thermal imaging based on long-wave infrared.

Leveraging heterogeneous SoC allows these pixel-wise algorithms to be executed efficiently in real-time, eliminating ghosting artifacts and blurring associated with scene-based NUC approaches [11], [41]. The synergy between the TDC and TPC also decreases the prevalence of FFC activations, enhancing the system's operational continuity and stability without compromising image quality.

2) Image Enhancement: The image enhancement stack improves overall image quality by sequentially executing the EE, contrast enhancement, noise filtering, color mapping, and RGB to YUV conversion (see Fig. 1). The most distinctive feature is the two-layer contrast enhancement techniques: thresholding plateau histogram equalization (TPHE) and CLAHE. This feature aims to enhance the image quality while maintaining throughput for real-time operation.

- **TPHE**: The first layer of contrast enhancement, TPHE, normalizes the 14-bit depth images from the IRFPA to fit an 8-bit depth suitable for standard visual displays. This involves dynamic range compression, crucial for reconciling the inherent disparity between the temperature and visible light domain, caused by the wider temperature sensitivity compared to the general environment's narrower dynamic range (See the raw video in Fig. 1). The TPHE differs from traditional plateau histogram equalization by incorporating a thresholding-based detection mechanism to identify optimal start and end points for histogram adjustment. This strategy minimizes noise and enhances global contrast, making the sparse histogram distributions denser and more suitable for presentation in the visible color space (See Fig. 1).

- **CLAHE**: Following the TPHE, the CLAHE further refines the image contrast by selectively amplifying local details. This layer is tailored to improve visual accuracy and detail without overburdening the system's computing resources. Due to the extensive memory requirements typically associated with the CLAHE—proportional to tile size and exponential to bit depth—the integration can lead to increased cache miss rates and latency. Our architecture addresses this by preceding the CLAHE with the TPHE, reducing the data range and simplifying the histogram complexity, thereby significantly improving processing times and reducing memory overhead.

- **Noise Filtering**: Enhanced contrast can inadvertently increase both temporal and spatial noise—the former due to brightness variations between frames and the latter from heightened detail amplification. Our approach includes two filters designed to mitigate these effects, ensuring that enhancements do not compromise the clarity of the image.

- **Color Mapping & Space Conversion**: The final steps in our image processing sequence involve adapting the enhanced images for display. This involves color mapping adjustments, converting the image format from RGB to YUV, and optimizing the images for various output devices and applications.

B. HW/SW Architecture

1) SoC HW Architecture: To effectively implement our image processing architecture, we have integrated the architecture into the TI TDA3x SoC, which features the two Cortex-M4 cores, the two C66x DSP cores, and the EVE [37].

- **C66x DSP**: The C66x DSP is an 8-way VLIW core with a dual data-path supporting up to 128-bit wide SIMD operations. This allows an efficient trade-off between programming workload and data-level parallelism. We distributed the intensive image processing algorithms to those DSPs.

- **Cortex-M4**: The two Cortex-M4 cores handle less intensive operations such as serial communication, hardware IP control, status checks, and task management. The first core, the system's master, manages the video stream and oversees task distribution across the cores. The second core is tasked with environmental monitoring, such as detecting light conditions and contrast indices, using data extracted during image processing on the DSP cores. This information aids in the real-time configuration of image enhancement algorithms, providing adaptability to diverse environmental conditions.

- **EVE**: The EVE supports histogram acceleration using the VCOP instructions, making the EVE especially suitable for executing histogram-based contrast enhancement algorithms. However, the EVE has drawbacks in the complexity of run-time configuration due to the localized, isolated data-path with scratchpad memories and DMAs for operation (see Fig. 2), requiring stalls for synchronization between the DMA, VCOP, and ARP32. Distinctive SW design flow is another concern.

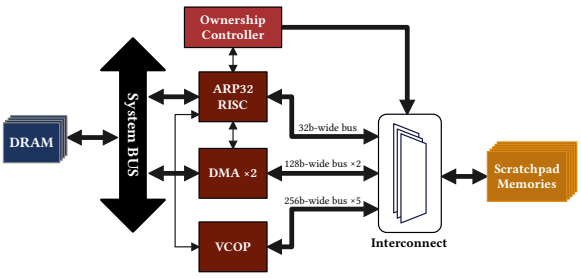


Fig. 2: The data-path architecture of the EVE.

To leverage each processing unit's strengths, we have allocated the advanced contrast enhancement algorithms across the EVE and DSP, with configurable options available at boot time. The EVE has advantages in latency and power due to the more vectorized (up to 16-way SIMD) and isolated low-latency scratchpad memories with 256-bit access, while the DSP offers fast real-time configuration for more adaptability.

This dual approach harnesses the distinct capabilities of each core type within the SoC, optimizing performance and efficiency while maintaining the flexibility needed to handle diverse imaging scenarios effectively.

2) System SW Architecture: In our architecture, we adopted a real-time operating system (RTOS) with asymmetric multi-processing (AMP) to optimize task execution across heterogeneous processing units. Each core manages its own task priorities, which allows for precise workload distribution and minimizes synchronization overheads.

- **Task Scheduling**: To handle the sequential nature of our image processing tasks effectively, we utilize fixed-priority preemptive scheduling. This approach ensures that tasks processing earlier video frames always have higher priority than those processing subsequent frames, thus maintaining a consistent processing order and preventing frame processing delays.

Fig. 3 illustrates the task allocation of the proposed system.

- **Cortex-M4 Roles**: The Cortex-M4 cores manage less computationally intensive but crucial operations. The master loop on the first core oversees communication with external devices and user settings. The second core's configuration loop detects environmental conditions and adjusts the settings of the image processing algorithms on the DSP cores. These cores also manage other tasks such as frame capturing, temporal and spatial noise filtering, and display control.

- **DSP and EVE Workload Distribution**: To distribute workloads efficiently among the DSPs, tasks are assigned not only based on their sequential order but also considering their computational intensity. DSP 0 handles the TDC, TPHE, color mapping, and RGB-to-YUV tasks, while DSP 1 handles the TPC, DPC, edge enhancement (EE), CLAHE, and preprocessing tasks necessary for the EVE. We grouped several sequentially executed algorithms (e.g., TPC+DPC+EE and Color Mapping + RGB-to-YUV) to minimize task creation and data transfer overheads. The CLAHE algorithm is optionally run on the EVE or DSP 1, depending on the system setup selected at boot, allowing flexibility in resource and performance needs.

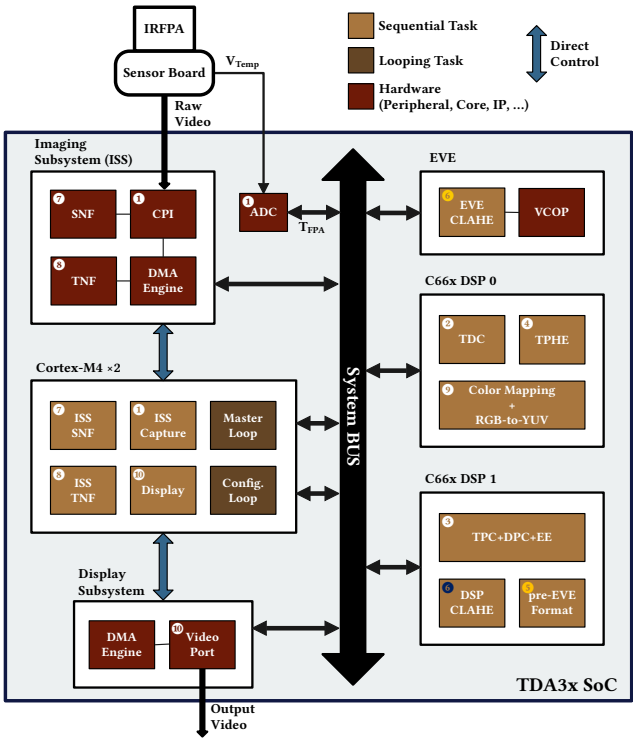


Fig. 3: The task allocation of the proposed system.

These workload distribution approaches aim to leverage the distinct capabilities of each component within the SoC, optimizing performance and efficiency. This ensures our system can adapt to varying operational demands in real-time while maintaining high-quality output.

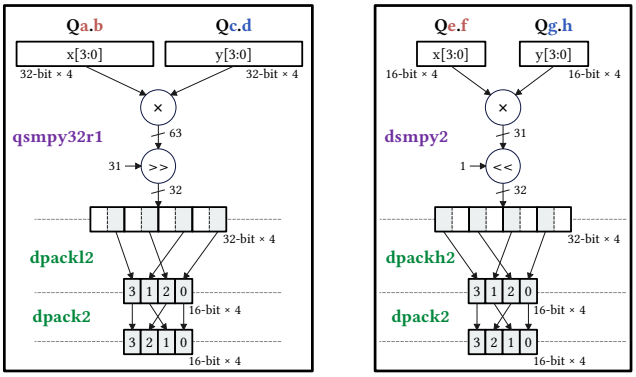
IV. Software Optimization

A. SIMD Kernel for NUC

In our software, compute kernels for image processing algorithms utilize the fixed-point format (q format) to implement real number arithmetic efficiently. Despite the broader dynamic range of floating-point, the q format is favored for its better precision within the small dynamic range [42], typical of image data. Moreover, operations vital for kernel design, such as the dot product and bit-field extraction, are unsupported for the floating-point, necessitating format conversions at the beginning and the end, increasing compute power requirements.

The fact that initial sensor data are integers also makes fixed-point a natural fit for minimizing conversion overheads. Thus, fixed-point arithmetic helps maintain numeric accuracy, critical for image quality, throughput, and energy efficiency.

- **SIMD Optimization**: The pixel data, initially 14-bit unsigned integers stored in 16 bits, are processed as SIMD. This format remains unchanged until contrast enhancement is processed, implying the data bits representing numbers exceeding 16 bits are truncated at the end of each algorithm. The SIMD kernels in Fig. 4 are tailored for diverse fixed-point configurations, accommodating various algorithms efficiently.



(a) (b)

Fig. 4: Compute kernels for fixed-point SIMD multiplication. (**a**) 4-way 32-bit multiplication with conversion to 16-bit. (**b**) 4-way 16-bit multiplication with conversion to 16-bit.

- **Kernel Designs**: The first kernel (Fig. 4(a)) executes the *qsmpy32r1* instruction for 32-bit multiplications within a Q0.31 fixed-point format, performing 4-way 32-bit signed multiplication and shifting right by 31 with rounding to maintain result accuracy. This kernel is adaptable to fixed-point configurations satisfying $a + b = 31$, $c + d = 31$, and $b + d = 31$, with post-multiplication steps extracting 16-bit least significant bits (LSBs) using *dpackl2* and *dpack2*.

The second kernel (Fig. 4(b)) employs the *dsmpy2* 16-bit multiplications, handling overflow by extending results and shifting left by one. This suits any configurations meeting $e + f = 15$, $g+h = 15$, and $f+h = 15$, with *dpackh2* and *dpack2* extracting and ordering 16-bit most significant bits (MSBs).

The rationale behind employing two distinct kernels is to balance accuracy and throughput. The TDC and TPC, the most computationally demanding operations in the NUC algorithm stack, rely on polynomial expressions—quadratic for TDC and linear for TPC—expressed as the following equations.

$$I_{TDC}(i,j) = I_{Raw}(i,j) - \sum_{n=0}^{2} C_n(i,j) \cdot (T_{FPA} - T_{Ref})^{2-n} \tag{1}$$

$$I_{TPC}(i,j) = G(i,j) \cdot I_{TDC}(i,j) + O(i,j) \tag{2}$$

The I_{TDC}, I_{Raw}, and $C_n(i,j)$ refer to the TDC output, raw input, and coefficients, respectively, and I_{TPC}, $G(i,j)$ and $O(i,j)$ refer to the TPC output, gain, and offset, respectively. The coefficients for the squared term at the TDC are subtle, whereas the variables are large. Sole reliance on 16-bit formats leads to critically adverse effects on precision due to the truncation of results. Conversely, small computational adjustments can significantly impact energy use and throughput in pixel-wise operations due to high iteration counts, memory footprint-induced cache misses, and VLIW-related register congestion.

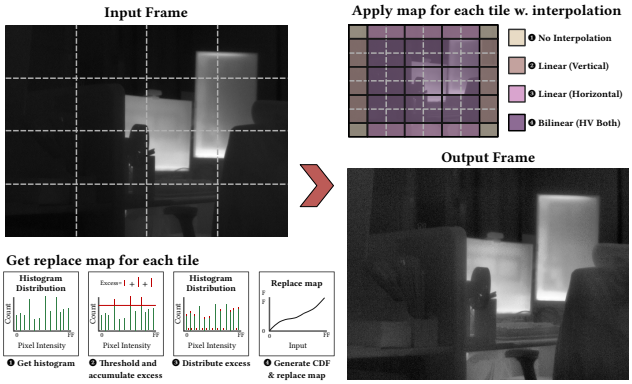


Fig. 5: The entire process of the CLAHE algorithm.

B. CLAHE Optimization

Fig. 5 depicts the entire process of the CLAHE algorithm. The operation of the CLAHE consists of two sequences: replace map generation and applying them with interpolation.

The replace map generation consists of four sub-sequences, executed tile by tile. First, the histogram is created by scanning the tile pixels. Next, each histogram bin is thresholded through clip limit configuration, and each excess value trimmed from thresholding is accumulated. This accumulated value is distributed to other histogram bins. Last, the cumulative distribution function (CDF) is calculated. The generated CDF is normalized to the output bit-depth and used to create the replace map, which is interpolated at the latter sequence.

The basis for separating the replace map generation sequence into sub-sequences is whether the operations can be merged. This separation minimizes the loop overheads. Most of the required arithmetic computations in each loop are vectorized through SIMD instructions. However, thresholding in the second and third sub-sequences requires conditional branching for each pixel in the original pseudo-code to determine whether it exceeds the clip limit. The branch instruction, twice needed for each branch, places five delay slots without branch prediction in C66x ISA. Our kernels are inappropriate for filling these slots due to the intrinsic dependencies. Besides, most arithmetic computations and memory accesses are required for each data in the kernel, blocking vectorization.

1) Thresholding Kernel: We designed the optimized kernel shown in Fig. 6 to achieve two objectives: To avoid the branch overheads and to provide the 4-way SIMD vectorization.

The kernel first finds the minimum value between the clip limit and the histogram bins and generates the thresholded bins using the minimum value. Then, the current excess value is calculated by subtracting thresholded bins from the original bins and accumulating them into the excess vector. Finally, the thresholded bins are stored in the histogram bins. The excess vector is summed after the loop is completed. This kernel efficiently replaces the previous branching-based algorithms, as the kernel includes only three arithmetic instructions that require one cycle for each operation.

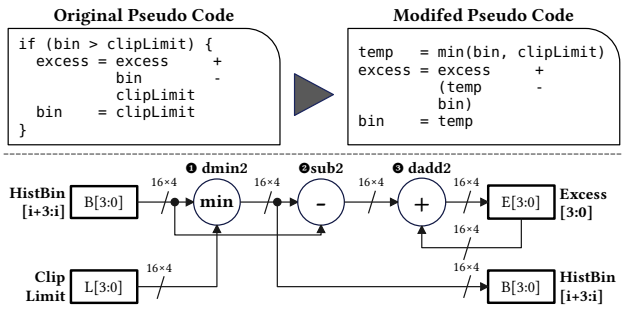


Fig. 6: The kernel for thresholding and excess accumulation in the second sub-sequence of the CLAHE.

```
#### Round One:   Fast Distribution
while (excessTotal > 0xFF) {
  dist16v4 = duplicate_by_four(excessTotal >> 8)
  pass16v4 = {0,0,0,0}
  for (i=0; i < 0x100; i+=4) {
    temp16v4 = min(hist[i+3:i] + dist16v4, clipLimit16v4)
    pass16v4 = pass16v4 + temp16v4 - hist[i+3:i]
    hist[i+3:i] = temp16v4
  }
  excessTotal = excessTotal - sum(pass16v4)
}
#### Round Two:   Distribute by Up to Four
for (j=0; excessTotal > 4; j+=4) {    // j is unsigned 8-bit
  temp16v4 = min(hist[j+3:j] + {1,1,1,1}, clipLimit16v4)
  excessTotal = excessTotal - sum(temp16v4 - hist[j+3:j])
  hist[j+3:j] = temp16v4
}
#### Round Three: Distribute Each by One
for (; excessTotal > 0; j+=1) {
  temp = min(hist[j] + 1, clipLimit)
  excessTotal = excessTotal - temp + hist[j]
}
```

Fig. 7: The pseudo code for excess distribution sub-sequence of the CLAHE (output bit-depth is 8-bit).

In the third sub-sequence, this kernel is also utilized with slight modifications. Fig. 7 presents our excess distribution method, marking differences in green. The kernel differs in adding distribution values before finding the minimum, distributing the excess values with thresholding to the clip limit while accumulating the distributed values in the vector container. This vector is summed and subtracted from the total excess value, marking what excess value is left.

The excess distribution has three rounds of execution. The first round is the fast distribution round, passing the excess value of the total value divided by the bin count for each bin. This round subtracts the distributed values when every iteration for the bins is completed, reducing the iterative computations for throughput. Over-distribution does not occur as integer division always results in a number smaller than or equal to the real division. This round ends when the total excess values are less than the bin count. The second round executes the summation of accumulated values and subtraction from the total excess value for each iteration, adding the sensitivity to prevent over-distribution. The third round, marked as red, does not utilize SIMD, providing the most sensitive operation to prevent over-distribution. This three-round architecture ensures accurate computation while maximizing the throughput.

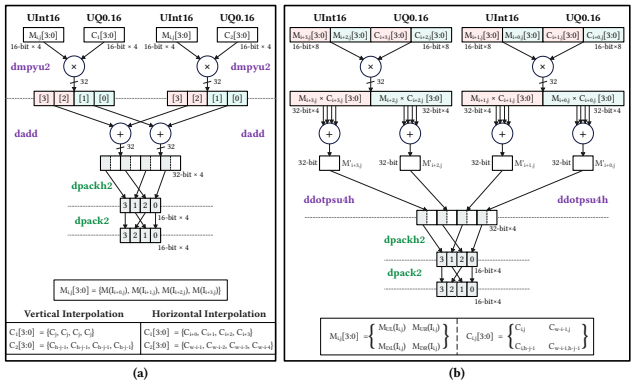


Fig. 8: The kernels for interpolation in the CLAHE. (**a**) Vertical/horizontal linear interpolation. (**b**) Bilinear interpolation.

2) Interpolation Kernel: The interpolation is categorized into four types (see Fig. 5). The four edge parts of the image, halved in width and height relative to the tiles, use the replace map directly without requiring interpolation. The tiles on the vertical and horizontal borders undergo linear interpolation between the adjacent allocated replace maps, which are left and right or top and down, respectively. The central part of the image, away from borders, utilizes bilinear interpolation among the four nearest replace maps for smooth gradient.

Our system calculates the coefficients for interpolation and stores them in unsigned Q0.16 format at system boot time to replace the binary division. The interpolation sequence loads those coefficients and calculates the mapping intensity interpolated between the current replace maps. These operations are also vectorized using the SIMD kernels in Fig. 8.

The linear interpolation includes one 4-way unsigned 16-bit multiplication (*dmpyu2*) and two 2-way 32-bit additions (*dadd2*), generating four outputs at one iteration. For bilinear interpolation, the double 16×4 dot product (*ddotpsu4h*) is utilized, generating two outputs at one kernel iteration. Both kernels use macro operation for the 4-way extraction of 16-bit MSBs (*dpackh2-dpack2*), also used in the NUC kernels.

3) Cache Optimization: We applied cache optimization strategies to improve the performance, shown in Fig. 9. The first optimization is static cache memory partitioning, supported by the C66x core, allowing some shares of the L2 cache to operate as scratchpad memory. By allocating to the L2, the L2 cache miss, which adversely affects the latency due to the characteristics of the dynamic random access memory (DRAM), is eliminated. We allocated each tile's interpolation coefficients and histogram to the L2 cache. The memory required for partitioning is calculated using the following code:

```
1 tileHeight    = height / tileCntV;
2 tileWidth     = width  / tileCntH;
3 tileSize      = tileHeight * tileWidth;
4 tileCnt       = tileCntV   * tileCntH;
5 // Memory sizes for allocation
6 sizeVertLerp  = sizeof(uint16) *   2 * tileHeight;
7 sizeHoriLerp  = sizeof(uint16) *   2 * tileWidth;
8 sizeBiLerp    = sizeof(uint16) *   4 * tileSize;
9 sizeHistogram = sizeof(uint32) * 256 * tileCnt;
```

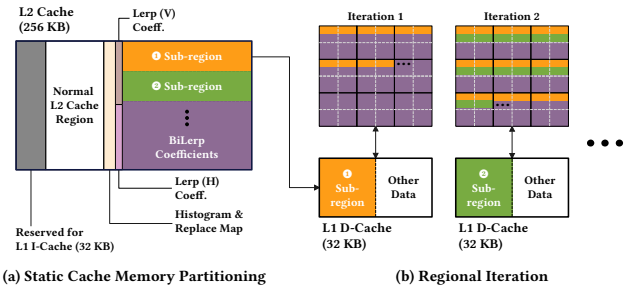


Fig. 9: The cache optimization methods for the CLAHE.

The second optimization is regional iteration, which orders the computation in the interpolation sub-sequence to increase the L1 cache hit rate. Fig. 9 presents the harmonization of the static cache memory partitioning and iteration architecture for the interpolation sub-sequence. We first ordered the computation of each tile by interpolation categories (see Fig. 5), such as non-interpolation, vertical linear interpolation—Lerp (V), horizontal linear interpolation—Lerp (H), and bilinear interpolation—BiLerp. This establishes distinct memory segments between each category, reducing the address variations in continuing memory access iterations. Next, we separated the iteration of bilinear interpolation to match the sub-region of interpolation coefficients, which fit into the L1 data cache, a 32 KB 2-way set associative cache. This causes the kernel to access the regional coefficients repeatedly, minimizing memory address variation and increasing the cache hit rate.

C. VLIW Optimization

The C66x DSP core is an 8-way VLIW machine containing six arithmetic-logical units (ALUs) named *.L*, *.S*, and *.D* and two multiplier units named *.M*, which can operate in parallel. Each unit has its own executable instructions, but the basic arithmetic SIMD instructions are executable for most units except for the *.D*. The *.D* is mainly used for memory access.

The two data paths contain these units individually, organizing eight units and enabling up to 8-way instruction execution in ideal cases. In this architecture, the kernel should simultaneously instruct the eight functional units to maximize the parallelism for enhanced throughput. However, in most cases, the kernel design is not ideal for distributing the computation among the units. Even though modern compilers support optimizations for VLIW architecture, this automatic optimization is often insufficient for thorough distribution.

To maximize parallelism, we manually unrolled each loop that executes the designed compute kernels, maximizing general-purpose register utilization. Fig. 10 presents the concept of loop unrolling for VLIW optimization. Manual unrolling dramatically reduces the iteration count while maintaining the similar VLIW-packed instruction count. This also lowers loop overheads proportional to the iteration count, such as memory array indexing and conditioned branching.

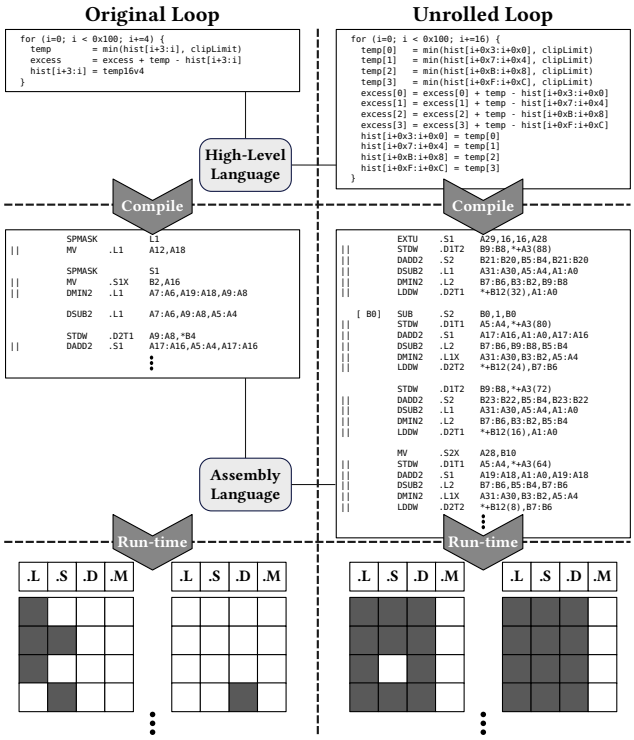


Fig. 10: VLIW optimization based on manual loop unrolling.

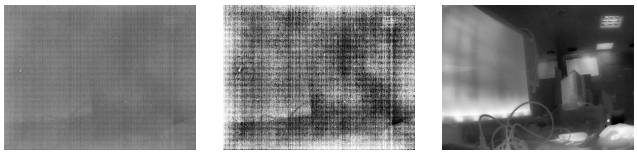


Fig. 11: Comparison of the images from the configurations turning on/off the TPC.

V. Evaluation

This section presents an evaluation of the proposed system, focusing on four key perspectives listed below:

- Visual analysis of the image processing architecture.
- Design space exploration (DSE) of SW optimizations.
- Comparative analysis between the SOTA works.
- Core load analyses to process higher-resolution videos.

We utilized the TDA3MVS SoC coupled with 512 MB LPDDR2 memory for system implementation. The video specification was set to 640×480 at 30 or 60 FPS.

A. Visual Analysis

1) NUC: We evaluated the NUC stack by comparing frames captured with and without the target algorithm, with all other algorithms active except for the EE and noise filtering (see Fig. 11). Contrast enhancements are toggled to improve visualization. The result shows applying the NUC removed the non-uniformities, with no ghosting or blurring observed.

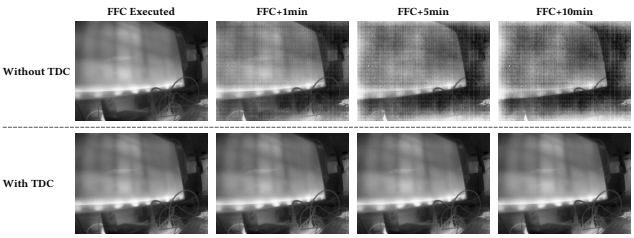


Fig. 12: Comparison of the images over time from the configurations turning on/off the TDC, from immediately after the FFC execution to ten minutes after.

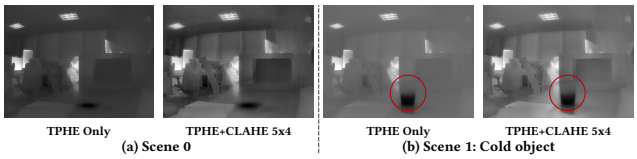


Fig. 13: Comparison of the images from the configurations turning on/off the CLAHE. Other algorithms are enabled.

We assessed the effectiveness of the TDC by comparing frames from the same scene immediately after executing the run-time FFC and ten minutes later (see Fig. 12). Without TDC, images degrade quickly. With TDC, the images remained clear even ten minutes after FFC, demonstrating that TDC noticeably reduces the frequency of FFC interventions, minimizing disruptions during continuous real-time imaging.

2) Contrast Enhancement: Our system's two-layer contrast enhancement algorithm improves on previous single-layer approaches, as shown in Fig. 13. The two-layer algorithm enhances both contrast and locality, making it superior not only to normal scenes (Scene 0) but also to scenes with objects at temperatures that greatly differ from their surroundings (Scene 1). In contrast, the TPHE-only algorithm provides sufficient contrast to detect a human but fails to highlight other elements in the scene. This drawback is more noticeable in the high-temperature contrast scene (Scene 1), denoting the effectiveness of CLAHE in thermal imaging.

B. Design Space Exploration

1) SIMD Kernel: We evaluated the impact of the SIMD kernel on latency within the NUC stack and TPHE, omitting DSP CLAHE, which could not achieve 30 FPS without SIMD. Our kernel reduced TDC and TPC latencies by 40.72% and 74.87%, respectively, thereby decreasing total latency by 46.40% at 30 FPS and 54.48% at 60 FPS, resulting in performance improvements of 1.87× and 2.2×. At 60 FPS, without SIMD, frequent frame receipt exacerbated pipeline interruptions, delaying frame processing until the previous was complete, thus increasing latency. However, the SIMD kernel mitigated these stalls, further decreasing latency. The cumulative CPU load of two DSPs reduced from 58.5% to 28.5% at 30 FPS and from 119.70% to 58.40% at 60 FPS, showing reductions of 51.28% and 51.21%, respectively.

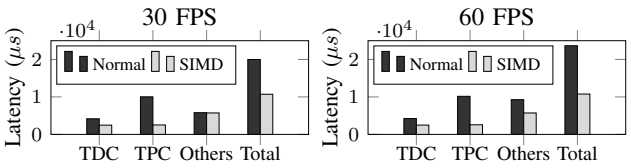


Fig. 14: The latency breakdown on SIMD kernel.

```
#### Find Sub-Region Height Type Count
div = tileHeight / splitCnt
subRgnTypeCnt = 1
while ((tileHeight*subRgnTypeCnt) % splitCnt > 0)
  subRgnTypeCnt++
#### Find Sub-Region Heights
subRgnHeightList = ones(subRgnTypeCnt) * div
rem = ((tileHeight*subRgnTypeCnt) / splitCnt) % subRgnTypeCnt
idx = 0
while (rem-- > 0)
  subRgnHeightList[idx++]++
```

Fig. 15: The algorithm for splitting sub-region.

2) CLAHE Optimization: Using various sub-region settings, we analyzed the impact on system latency when applying regional iteration to bilinear interpolation. The latency results were measured when processing 30 FPS videos while other SW optimizations were fully enabled. The algorithm in Fig. 15 was used for cases where the tile height possesses remainder in integer division. This algorithm diversifies height types to flawlessly process the center parts. For example, if the tile height is 120 and the sub-region count is 32, the heights are $(4, 4, 4, 3)$, processing 15 lines per 4 iterations.

Each tile size setting exhibits a unique loop architecture, as they differ in organizing arithmetic operations and memory footprints. Two key factors affecting latency are cache hit rate and the extent of bilinear interpolation. A smaller tile size improves the L1 cache hit rate, thereby reducing latency. However, the edge parts that require low computational load, i.e., no interpolation (see Fig. 5), decrease proportionally to the tile size. The pixels excluded from the edge parts are replaced not only with linear but also with bilinear interpolations, which are among the most computationally intensive operations in CLAHE. Due to these characteristics, modeling the regional iteration's latency for each setting with only simple, configurable parameters is challenging. This necessitates empirical experiments to determine the latency-optimal sub-region count settings. The results of these are shown in Fig. 16.

Our experiments demonstrated a general decrease in latency as the sub-region count increased, with the most significant reduction observed in the initial incremental step. However, for tile sizes 128×96 and 128×80, increased loop overheads and saturated cache hit rates limit further improvements. The regional iteration achieved an average latency reduction of 16.62% and a maximum reduction of 21.86% in the 80×160 setting, reducing latency by 1920 μs. The lowest observed latency was 5678 μs at a 128×120 tile size with 120 sub-regions setting, underscoring the effectiveness of regional iteration. This could be further improved by cutting the currently indivisible units of the sub-region, the line, into smaller pieces.

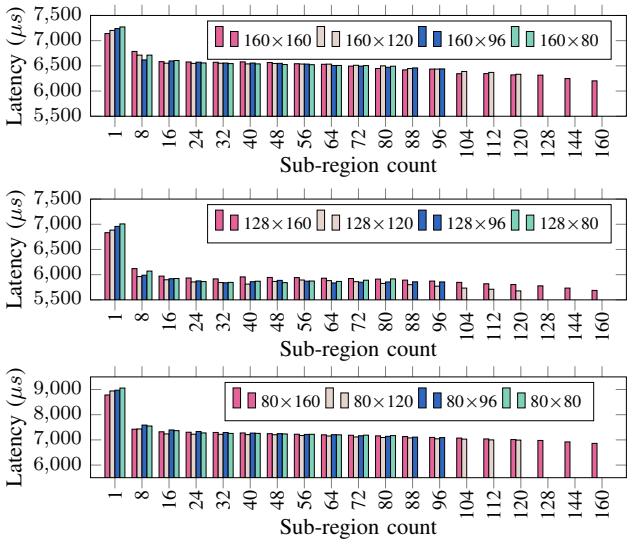


Fig. 16: The latency breakdowns regarding varying sub-region count and tile size configurations.

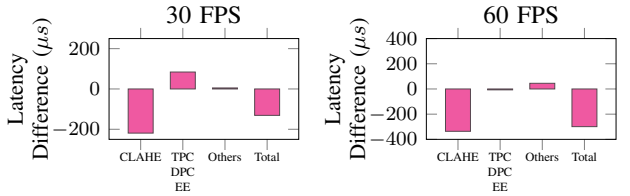


Fig. 17: The average latency impact breakdowns on static cache memory partitioning.

Static cache memory partitioning risks lowering the L2 cache hit rate due to the reduced space available for normal cache operations. This reduction may adversely affect the throughput of the image processing algorithms. To clarify this uncertainty, we conducted the experiment assessing the impact of the partitioning on image processing, particularly the CLAHE and the TPC+DPC+EE tasks on DSP 1 (see Fig. 3). In this experiment, we applied the latency-optimal sub-region count for each tile size configuration, as found in the previous experiment (see Fig .16).

The results in Fig. 17 indicate that the partitioning reduced total latency by 131 μs and 299 μs on average at 30 FPS and 60 FPS, respectively, achieving modest reductions of 0.53% and 1%. Notably, at 60 FPS with a 128×120 tile size, latency decreased by 1265 μs (4.27%), with several configurations achieving over 1 ms reduction while others showed adverse effects. The results vary due to the distinct loop architectures of each setting, which have different workloads and memory footprints, suggesting potential for targeted throughput improvements for specific settings. Further investigation into static cache memory partitioning, such as application to other algorithms, will continue to refine these findings.

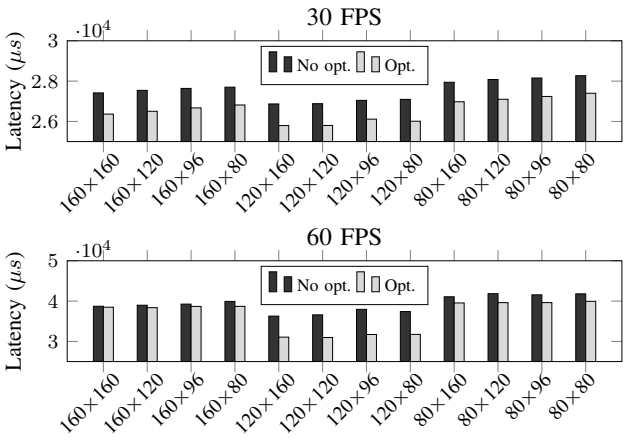


Fig. 18: The latency impact breakdowns of the VLIW optimization on the entire system latency for each tile size setting.

3) VLIW Optimization: We assessed the impact of VLIW optimization on reducing latency by measuring the changes when manual unlooping was enabled and disabled, as illustrated in Fig. 18. Similar to the previous evaluation, the latency-optimal sub-region count settings are applied.

The results present an average latency reduction of 986.91 μs at 30 FPS and 2747.33 μs at 60 FPS, corresponding to 3.72% and 7.18% speedups, respectively. While these results demonstrate consistent performance improvements, the observed speedups were smaller than anticipated. We first expected the speedups to be similar to the SIMD kernels, as the generated assembly codes exhibited notably improved parallelism. However, the actual result shows an even lower impact than the regional iteration, evidenced by the results in Fig. 14 and 16. This result originates from the limited number of data paths, which is only two, causing stalls for memory access. Therefore, according to the comprehensive results, the prior aspects for enhancing throughput appear to be data-level parallelism and cache misses, not instruction-level parallelism.

Nevertheless, specific configurations, such as the 128×96, benefited significantly from this optimization, showing a 16.43% reduction in latency. This improvement is attributable to the characteristics of the image processing pipeline previously identified during the SIMD kernel evaluation. In the future, considering our RTOS architecture, we aim to analyze the correlation between the pipeline and each task's workload.

C. Comparative Analysis

Table I compares our system and the SOTA works, highlighting differences in the system, algorithm stacks, and power consumption metrics. The SOTA works consist of FPGA-based systems, mentioned in Section II-A1 including our previous work [24], and a compact system using a microcontroller unit (MCU) and a camera module [43]. We set the evaluation metric to compare power efficiency as FPS per watt (FPS / W). Our system's power was calculated by measuring energy usage for 30 minutes using a MakerHawk UM34C multimeter.

Works	Syst. Compon.	Resolut.	FPS	NUC Stack	Contrast Enhanc.	Power (W)	Power Effic. (FPS / W)
This work	SoC + IRFPA	640×480	30	TPC+TDC+FFC+DPC	HE-variant+CLAHE HE-variant	1.837[a] / 1.865[b] 1.656	16.33[a] / 16.09[b] 18.12
			60	TPC+TDC+FFC+DPC	HE-variant+CLAHE HE-variant	2.145[a] / 2.160[b] 1.908	27.97[a] / 27.78[b] 31.45
[23]	FPGA + IRFPA	640×480	30 / 111	TPC+FFC+DPC	HE	1.812* / 6.970*	16.56* / 15.96*
[24]	FPGA + IRFPA	640×480	60 / 142	TPC+TDC+FFC+DPC	Min-max stretching	2.098*(60 FPS)	28.60*(60 FPS)
[43]	MCU + Camera	160×120	8.7	Camera dependent	HE-variant	0.920	9.46

[a]EVE CLAHE mode / [b]DSP CLAHE mode. *FPGA on-chip power, other system devices are not measured.

TABLE I: Comparative analysis between this work and SOTA systems.

Core	30 FPS (%)			60 FPS (%)		
	HE[a]	DC[b]	EC[c]	HE[a]	DC[b]	EC[c]
Coretx-M4 0	6.90	14.20	12.30	9.80	21.20	21.00
Coretx-M4 1	1.10	5.60	1.20	1.20	11.80	1.60
DSP 0	23.80	26.40	26.30	46.70	50.50	57.50
DSP 1	8.60	26.20	12.80	17.70	55.70	30.80
EVE	0.70	0.70	6.30	0.70	0.90	12.00

[a]HE-only mode / [b]DSP CLAHE mode / [c]EVE CLAHE mode.

TABLE II: Peak core loads across the processing units.

According to Table I, our system presents enhancements in energy efficiency, especially regarding the composition of the image processing pipeline. When employing the HE-variant for contrast, our approach improves power efficiency by 9.42% and 9.96% at 30 and 60 FPS, respectively, compared to the FPGA-based systems. When the CLAHE is added, slight degradations of -2.84% and -2.87% are measured. This result underscores our system's power efficiency, as the HE requires more computational load than min-max stretching, and the CLAHE requires even more load than the HE-variants.

Even though our power measurements reflect the entire system's consumption, including the additional IRFPA power of up to 300 mW, our system shows better efficiency than the previous works, which provided only the FPGA chip's power usage. This achievement is attributed to the utilization of lightweight heterogeneous SoC and SW optimization, making our system a strong competitor in thermal imaging systems.

Notably, unlike the FPGA-based system [23], the power consumption does not increase proportionally to the FPS, causing the efficiency to be improved at the higher frame rates. This is because the SoC's operating frequency does not differ depending on whether the output FPS changes. This offers further room for power optimization using frequency scaling.

D. Potential for Higher Resolution Video Processing

The load distribution across the different processing units and use cases in our system, presented in Table II, provides insights into its capability to handle higher-resolution videos.

The core utilization at 60 FPS indicates that the DSP 0 is nearing capacity, especially in the DSP CLAHE and EVE CLAHE modes, where the utilization exceeds 50%. Other cores, such as the EVE, show lower utilization, suggesting some tasks could be redistributed or optimized further.

Considering the current resolution of 640×480, scaling up to a higher resolution of 1280×1024 will increase the computational load proportionally, approximately 4.27×. However, the relatively low utilization of some cores, such as EVE (particularly in the HE and the DSP CLAHE modes), hints at the available headroom that could be exploited.

Processing higher resolutions will significantly depend on optimizing current algorithms and effectively redistributing tasks to underutilized cores. We anticipate further throughput gains will be explored by adjusting the workload balance between DSPs and utilizing the EVE more intensively, especially given the EVE's lower utilization rates.

Future work will explore these optimizations to assess the feasibility of supporting higher-resolution video processing within the current HW/SW architecture while minimizing the compromise of the system's performance and energy efficiency. This will involve detailed modeling and profiling of HW/SW architectures and empirical studies, providing better resource utilization through optimizing workload distribution and algorithm configurations.

VI. Conclusion

This paper presents a compact yet high-performance system for real-time thermal imaging. By harmonizing the optimized image processing architecture with the lightweight heterogeneous SoC, we have realized low-cost, high-quality thermal imaging, which includes advanced features such as the CLAHE. Our system processes high-quality thermal video at 640×480 resolution with 14-bit depth, a maximum load of 57.50%, and power consumption below 2.2 W. Additionally, when executing similar image processing, our system presents 9.42% and 9.96% power efficiency improvements at 30 and 60 FPS, respectively, compared to the FPGA-based systems. This indicates the lightweight heterogeneous SoC efficiently eliminates the need for the FPGAs to execute intensive imaging processing algorithms. These optimistic results offer the possibility of processing higher-resolution thermal videos, such as 1280×1024, from the SOTA IRFPAs. The SW optimization techniques enhance the utilization of the DSPs for real-time image processing, making it feasible to handle sophisticated algorithms typically dependent on the FPGAs. We expect our system architecture to be utilized widely for the diverse applications requiring efficient thermal imaging.

References

[1] M. A. Farooq, W. Shariff, and P. Corcoran, "Evaluation of Thermal Imaging on Embedded GPU Platforms for Application in Vehicular Assistance Systems," *IEEE Trans. Intell. Veh.*, vol. 8, no. 2, pp. 1130–1144, Feb. 2023.

[2] A. Naser, A. Lotfi, and J. Zhong, "Multiple Thermal Sensor Array Fusion Toward Enabling Privacy-Preserving Human Monitoring Applications," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 16 677–16 688, Sep. 2022.

[3] B. Kim, M. Kim, and Y. Chae, "Background Registration-Based Adaptive Noise Filtering of LWIR/MWIR Imaging Sensors for UAV Applications," *Sensors*, vol. 18, no. 2, p. 60, Dec. 2017.

[4] N. Kowalczyk, M. Sobotka, and J. Rumiński, "Mask Detection and Classification in Thermal Face Images," *IEEE Access*, vol. 11, pp. 43 349–43 359, 2023.

[5] P. Hurney, P. Waldron, F. Morgan, E. Jones, and M. Glavin, "Review of pedestrian detection techniques in automotive far-infrared video," *IET Intell. Transp. Syst.*, vol. 9, no. 8, pp. 824–832, Oct. 2015.

[6] R. Baghezza, K. Bouchard, A. Bouzouane, and C. Gouin-Vallerand, "Profile Recognition for Accessibility and Inclusivity in Smart Cities Using a Thermal Imaging Sensor in an Embedded System," *IEEE Internet Things J.*, vol. 9, no. 10, pp. 7491–7509, May 2022.

[7] P. A. Coelho, J. E. Tapia, F. Pérez, S. N. Torres, and C. Saavedra, "Infrared light field imaging system free of fixed-pattern noise," *Sci. Rep.*, vol. 7, no. 1, p. 13040, Oct. 2017.

[8] O. Gonzalez-Chavez, D. Cardenas-Garcia, S. Karaman, M. Lizarraga, and J. Salas, "Radiometric Calibration of Digital Counts of Infrared Thermal Cameras," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 11, pp. 4387–4399, Nov. 2019.

[9] A. Rogalski, "Infrared detectors: an overview," *Infrared Phys. Technol.*, vol. 43, no. 3-5, pp. 187–210, Jun. 2002.

[10] D. Scribner, M. Kruer, and J. Killiany, "Infrared focal plane array technology," *Proc. IEEE*, vol. 79, no. 1, pp. 66–85, Jan. 1991.

[11] S. Rong *et al.*, "An improved non-uniformity correction algorithm and its hardware implementation on FPGA," *Infrared Phys. Technol.*, vol. 85, pp. 410–420, Sep. 2017.

[12] B.-j. Wang, S.-q. Liu, Q. Li, and H.-x. Zhou, "A real-time contrast enhancement algorithm for infrared images based on plateau histogram," *Infrared Phys. Technol.*, vol. 48, no. 1, pp. 77–82, Apr. 2006.

[13] F. Branchitta, "Dynamic-range compression and contrast enhancement in infrared imaging systems," *Opt. Eng.*, vol. 47, no. 7, p. 076401, Jul. 2008.

[14] F. Zhang, W. Xie, G. Ma, and Q. Qin, "High dynamic range compression and detail enhancement of infrared images in the gradient domain," *Infrared Phys. Technol.*, vol. 67, pp. 441–454, Nov. 2014.

[15] A. Vidyarthi and A. Malik, "A hybridized modified densenet deep architecture with CLAHE algorithm for humpback whale identification and recognition," *Multimed. Tools Appl.*, vol. 81, no. 14, pp. 19 779–19 793, Jun. 2022.

[16] P. Martínez Cañada, C. Morillas, R. Ureña, J. Gómez López, and F. Pelayo, "Embedded system for contrast enhancement in low-vision," *J. Syst. Archit.*, vol. 59, no. 1, pp. 30–38, Jan. 2013.

[17] E. Lielāmurs, A. Cvetkovs, R. Novickis, and K. Ozols, "Infrared Image Pre-Processing and IR/RGB Registration with FPGA Implementation," *Electronics*, vol. 12, no. 4, p. 882, Feb. 2023.

[18] A. M. Reza, "Realization of the Contrast Limited Adaptive Histogram Equalization (CLAHE) for Real-Time Image Enhancement," *J. VLSI Signal Process.-Syst. Signal Image Video Technol.*, vol. 38, no. 1, pp. 35–44, Aug. 2004.

[19] K. Kokufuta and T. Maruyama, "Real-Time Processing of Contrast Limited Adaptive Histogram Equalization on FPGA," in *Int. Conf. Field Program. Log. Appl. (FPL)*, Milan, Italy, Aug. 2010, pp. 155–158.

[20] B. Unal and A. Akoglu, "Resource efficient real-time processing of Contrast Limited Adaptive Histogram Equalization," in *Int. Conf. Field Program. Log. Appl. (FPL)*, Lausanne, Switzerland, Aug. 2016, pp. 1–8.

[21] T. Kryjak, K. Blachut, H. Szolc, and M. Wasala, "Real-Time CLAHE Algorithm Implementation in SoC FPGA Device for 4K UHD Video Stream," *Electronics*, vol. 11, no. 14, p. 2248, Jul. 2022.

[22] K. Cheng, H. Zhou, H. Qin, D. Zhao, K. Qian, and S. Rong, "An improved non-uniformity correction algorithm and its GPU parallel implementation," *Infrared Phys. Technol.*, vol. 90, pp. 156–163, May 2018.

[23] X. Wang, X. He, X. Zhu, F. Zheng, and J. Zhang, "Lightweight and Real-Time Infrared Image Processor Based on FPGA," *Sensors*, vol. 24, no. 4, p. 1333, Feb. 2024.

[24] H. W. Oh, C.-H. Choi, J. W. Cha, H. Choi, J. H. Han, and J.-H. Shin, "An SoC FPGA-based Integrated Real-time Image Processor for Uncooled Infrared Focal Plane Array," in *26th Euromicro Conf. Digit. Syst. Design (DSD)*, Golem, Albania, Sep. 2023, pp. 660–668.

[25] Q. Liu, H. Wang, H. Bian, and H. Wang, "Dual Mode High Speed Uncooled Short Wave Infrared Camera Based on FPGA," *Journal of Physics: Conference Series*, vol. 1952, no. 3, p. 032042, Jun. 2021.

[26] S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. t. H. Romeny, J. B. Zimmerman, and K. Zuiderveld, "Adaptive histogram equalization and its variations," *Computer Vision, Graphics, and Image Processing*, vol. 39, no. 3, pp. 355–368, 1987.

[27] X. Liang, Y. Tian, S. Yan, K. Wang, C. Guo, and B. Du, "A real-time infrared image enhancement algorithm based on improved CLAHE," in *Int. Conf. Image Video Process. Artif. Intell.*, vol. 10836, 2018, pp. 1–9.

[28] Z. Nikolic, R. Venkatasubramanian, J. A. T. Jones, and P. Labaziewicz, "A scalable heterogeneous multicore architecture for ADAS," in *IEEE Hot Chips 27 Symp. (HCS)*, Cupertino, CA, USA, 2015, pp. 1–32.

[29] P. Viswanath *et al.*, "A Diverse Low Cost High Performance Platform for Advanced Driver Assistance System (ADAS) Applications," in *IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Las Vegas, NV, USA, Jun. 2016, pp. 819–827.

[30] P. Vergés, I. Nunes, M. Heddes, T. Givargis, and A. Nicolau, "Accelerating Permute and N-Gram Operations for Hyperdimensional Learning in Embedded Systems," in *IEEE 29th Int. Conf. Embed. Real-Time Comput. Syst. Appl. (RTCSA)*, Niigata, Japan, Aug. 2023, pp. 253–260.

[31] J. Zhou and K. A. Ross, "Implementing database operations using SIMD instructions," in *ACM Int. Conf. Manag. Data (SIGMOD)*, New York, NY, USA, 2002, p. 145–156.

[32] Z. Chen, A. Nicolau, and A. V. Veidenbaum, "SIMD-based soft error detection," in *ACM Int. Conf. Comput. Front. (CF)*, Como, Italy, May 2016, pp. 45–54.

[33] M. Abeysinghe, J. Villarreal, L. Weaver, and J. Bakos, "OpenVX Graph Optimization for Visual Processor Units," in *IEEE 30th Int. Conf. Embed. Real-Time Comput. Syst. Appl. (RTCSA)*, New York, NY, USA, Jul. 2019, pp. 123–130.

[34] B. Ramesh, A. Bhardwaj, J. Richardson, A. D. George, and H. Lam, "Optimization and evaluation of image- and signal-processing kernels on the TI C6678 multi-core DSP," in *IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Waltham, MA, USA, Sep. 2014, pp. 1–6.

[35] Y. Chen, C. Mendis, M. Carbin, and S. Amarasinghe, "VeGen: a vectorizer generator for SIMD and beyond," in *26th ACM Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, Virtual, USA, Apr. 2021, pp. 902–914.

[36] V. Kandiah, D. Lustig, O. Villa, D. Nellans, and N. Hardavellas, "Parsimony: Enabling SIMD/Vector Programming in Standard Compiler Flows," in *21st ACM/IEEE Int. Symp. Code Gener. Optim. (CGO)*, Montréal, QC, Canada, Feb. 2023, pp. 186–198.

[37] D. K. Mandal *et al.*, "An Embedded Vision Engine (EVE) for automotive vision processing," in *IEEE Int. Symp. Circuits Syst. (ISCAS)*, Melbourne, VIC, Australia, Jun. 2014, pp. 49–52.

[38] G. Bieszczad, T. Orżanowski, T. Sosnowski, and M. Kastek, "Method of detectors offset correction in thermovision camera with uncooled microbolometric focal plane array," in *Electro-Opt. Infrared Syst.: Technol. Appl. VI*, vol. 7481, Berlin, Germany, Sep. 2009, pp. 200–207.

[39] B. Kelleci *et al.*, "Development of 1280x1024 15µm pixel pitch ROIC for MWIR and LWIR IIR detectors," in *Infrared Technol. Appl. XLIX*, Orlando, United States, Jun. 2023, p. 48.

[40] E. Inceturkmen *et al.*, "Development of 1280x1024 10µm pixel pitch ROIC for MWIR IIR detectors," in *Infrared Technol. Appl. XLVIII*, Orlando, United States, May 2022, p. 66.

[41] R. Redlich, M. Figueroa, S. N. Torres, and J. E. Pezoa, "Embedded nonuniformity correction in infrared focal plane arrays using the Constant Range algorithm," *Infrared Phys. Technol.*, vol. 69, pp. 164–173, Mar. 2015.

[42] H. W. Oh, S. An, W. S. Jeong, and S. E. Lee, "RF2P: A Lightweight RISC Processor Optimized for Rapid Migration from IEEE-754 to Posit," in *IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Vienna, Austria, Aug. 2023, pp. 1–6.

[43] I. Stančić, A. Kuzmanić Skelin, J. Musić, and M. Cecić, "The Development of a Cost-Effective Imaging Device Based on Thermographic Technology," *Sensors*, vol. 23, no. 10, p. 4582, May 2023.