

FPGA-based Implementation for Industrial Motion Control System

Questa è la versione Post print del seguente articolo:

Original
FPGA-based Implementation for Industrial Motion Control System / Rubattu, C., Ledda, A., Ratto, F., Jugade, C., Goswami, D., Palumbo, F.. - 100:(2024), pp. 141-147. (2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)) [10.1109/IPDPSW63119.2024.00039].

Availability:
This version is available at: 11388/343469 since: 2025-01-20T15:23:31Z

Publisher:

Published
DOI:10.1109/IPDPSW63119.2024.00039

Terms of use:

Chiunque può accedere liberamente al full text dei lavori resi disponibili come “Open Access”.

Publisher copyright

note finali coverpage

(Article begins on next page)

Camera Ready version of the paper:

FPGA-based Implementation for Industrial Motion Control System

Claudio Rubattu, University of Sassari, *crubattu@uniss.it*

Antonio Ledda, University of Sassari, *aledda3@uniss.it*

Francesco Ratto, University of Sassari, *fratto@uniss.it*

Chaitanya Jugade, Eindhoven University of Technology, *c.jugade@tue.nl*

Dip Goswami, Eindhoven University of Technology, *d.goswami@tue.nl*

Francesca Palumbo, University of Sassari, *francesca.palumbo@unica.it*

This work was supported by ECSEL Joint Undertaking in H2020 project IMOCO4.E, grant agreement No.101007311.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s):

2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS), May 27-31, 2024, San Francisco, California USA.

Please, cite this paper as follow:

Claudio Rubattu, Antonio Ledda, Francesco Ratto, Chaitanya Jugade, Dip Goswami and Francesca Palumbo, "FPGA-based Implementation for Industrial Motion Control System", 2024, 31st Reconfigurable Architectures Workshop (RAW) associated with the 38th Annual IEEE International Parallel and Distributed Processing Symposium (IEEE IPDPS 2024), May 27-31, 2024, San Francisco, California USA. DOI: <https://doi.org/10.5281/zenodo.11352243>.

FPGA-based Implementation for Industrial Motion Control System

Claudio Rubattu*, Antonio Ledda*, Francesco Ratto*, Chaitanya Jugade[§], Dip Goswami[§] and Francesca Palumbo[†]

*University of Sassari, Sassari, Italy 07100

Email: crubattu@uniss.it, aledda3@uniss.it, fratto@uniss.it

[†]University of Cagliari, Cagliari, Italy 09123

Email: francesca.palumbo@unica.it

[§]Eindhoven University of Technology, Eindhoven, Netherlands 5612 AZ

Email: c.jugade@tue.nl, d.goswami@tue.nl

Abstract—In these years of severe chip shortage, it is even more important to improve the efficiency of chip manufacturing. As is well known, manufacturing phases rely on increasingly intelligent production machinery, which must ensure high quality and large volumes. That is true also for die-bonding machines, which are required to satisfy very high standards of speed and accuracy. For this purpose, such devices have started to evaluate the possibility of adopting computer vision algorithms for automatic recognition of wafer positioning and die size.

This paper proposes an FPGA accelerated implementation of one of these algorithms, demonstrating the advantages of using this technology to this end and paving the way towards a larger adoption of this kind of acceleration platform for the different tasks composing modern industrial motion control systems. At the state of the art, such systems are typically managed with software-oriented solutions, which may not be sufficient in the case of highly restrictive requirements in terms of execution time. For this reason, the design flow considered the use of high-level hardware design, which offers a more software-friendly solution to developers without in-depth hardware knowledge. The proposed solution is a state-of-the-art implementation for execution time and resources of programmable logic while enabling higher precision in terms of die position estimation.

I. INTRODUCTION

The advent of Cyber-Physical Systems (CPSs) [1] opened up many new application domains for the usage of advanced computation platforms at the edge, including industrial applications [2]. The H2020 project IMOCO4.E [3] takes part in this context, combining and exploiting novel sensory information, model-based approaches, and Industrial Internet-of-Things philosophies. Its goal is to make mechatronic systems smarter, more configurable, more reliable, while simultaneously improving their performance. Focusing on Industry 4.0, where current manufacturing phases rely on increasingly intelligent production machinery, a relevant case study is the wafer stage of a die-bonding machine [4]. Indeed, we have lately witnessed a severe chip shortage [5] that started with the COVID-19 pandemic and continued with the geopolitical tensions. Despite being relaxed, how it will evolve in the future has yet to be demonstrated. Therefore, it is crucial to improve chip fabrication, and more efficient CPSs can serve the scope.

In [6], it has been demonstrated that leveraging a multi-sensor approach, typical in CPS, and porting advanced processing at the edge is strategic, from the functional perspective, to improve the efficiency of an industrial system such as a die-bonding machine. In that study, a multi-rate multi-sensor fusion algorithm was proposed to flank a traditional system to limit the operational errors that may occur when chips are not accurately positioned. Typically, these systems include industrial sensors (such as linear encoders) that provide feedback on the positioning of the motors that move the machinery, but not on positions at points of interest (which are different because of external disturbances). To limit this issue in [6], accurate, delayed, and slow vision measurements were merged with fast and noisy encoder measurements to achieve higher accuracy and high-speed position estimates that considerably improved the control system's closed-loop performance, providing a sufficient acceleration of the *Vision Algorithm* (VA).

On the physical realization side, the system proposed in [6] could benefit from Field Programmable Gate Arrays (FPGAs) devices that can guarantee the requested Hardware (HW) acceleration, along with execution flexibility [7], and energy efficiency. Generally speaking, their integration on heterogeneous Multi-Processor System on Chip (MPSoCs) opened up a wide range of possibilities exploiting the FPGA potentials along with the Central Processing Unit (CPU) capabilities of managing control flow and communication. Such an evolution made them suitable for different types of application scenarios, including Industry 4.0, where they are already adopted to implement digital control in motion control systems[8], [9], [10], [11]. Nevertheless, the benefits of integrating vision into machine control are limited by latency issues. Some attempts to overcome that by using FPGA-based acceleration platforms are there [12], [13], [14], but they typically remain separate from the control environment [15]. To the authors' knowledge, hardware acceleration using FPGA devices for closed-loop control design is not found in the literature. It is then straightforward, in particular with the advent of modern FPGA platforms offering complete System-on-Module (SoM) support, to embed on the same device also the VA support. SoMs embed on the same board all the core components of an

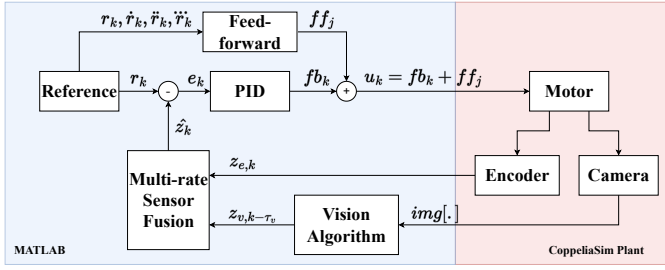


Figure 1: Multi-rate Multi-Sensor system block diagram [6].

embedded processing system (e.g. processor cores, communication interfaces, and memory blocks) along with dedicated ad-hoc peripherals (e.g. camera, embedded MultiMediaCard, etc.) providing specific drivers to use them.

For these reasons, this paper presents an FPGA-based implementation of the VA proposed in [6] to reduce its execution time to improve the precision error of the closed-loop control system. In addition, HW acceleration has been designed to facilitate its adoption by Software (SW) developers as well, since it has involved: i) the AMD Kria K26 SoM, which, for example, can be built-in to the AMD Kria K260V Vision Starter kit for driver-based access to image acquisition peripherals/sensors; ii) high-level HW design SW tools, such as the High-Level Synthesis (HLS) tools, which does not require deep HW expertise and enables time-to-market acceleration.

This work is organized as follows. Section II describes the background context of the reference multi-rate multi-sensor system for industrial motion control [6]. In Section III, a HLS-oriented implementation of a VA and its integration on the AMD Kria K26 SoM is proposed. In Section IV, the results presented concern: i) timing performance of the HW acceleration compared to those of an Arm application processor (included in the AMD Kria K26 SoM) and a commercial x86 processor; ii) HW resource usage associated with the VA accelerator and other system blocks required to handle the acceleration; and iii) precision error for two distinct scenarios with different parameters for external disturbance and vision processing delay. Finally, Section V concludes the paper.

II. MULTI-RATE MULTI-SENSOR SYSTEM IN INDUSTRIAL MOTION CONTROL

Figure 1 presents the block diagram of a closed-loop control system [6] featuring multi-rate multi-sensor fusion, VA, and control structure for industrial motion control application, which represents our case study. It is composed of different blocks (discussed in the following subsections) and its primary objective is to enhance the positioning accuracy of industrial motion control systems operating at various points of control and points of interest by integrating a VA system alongside a linear encoder. This integrated approach effectively mitigates errors stemming from external factors that may disrupt the precise positioning of microchips, factors that cannot be detected solely through the use of a linear measurement system. One of the key conclusions of [6] is that the closed-loop performance of the control system, particularly in terms of

Steady-State Error (SSE) (i.e., e_k at the end of each iteration), Mean Absolute Error (MAE) (equal to $\frac{1}{N} \sum_{k=1}^{N-1} |z_k - \hat{z}_k|$), and Settling Time (ST) (i.e., the time to reach within 2% of the target position), deteriorates as the VA processing becomes slower. Consequently, there is a need to minimize vision delays by providing acceleration of the VA.

A. Proportional-Integrator Derivative (PID) Controller

In the overall closed-loop control structure shown in Figure 1, the PID controller is used as a feedback controller that maintains a desired output by continuously adjusting the control input based on proportional, integral, and derivative terms. fb_k represents the feedback control action, calculated through an equation which takes into account proportional, integral, and derivative controller gains. The error signal is denoted as e_k , and r_k is the reference signal at time instant k .

B. Feed-forward Iterative Learning Controller (ILC)

The ILC is used as a feed-forward controller to improve the performance of systems over multiple cycles or iterations. It learns from past control errors and iteratively adjusts the control input to reduce error, resulting in enhanced tracking accuracy and reduced cumulative error. The ILC is particularly useful in applications with repetitive tasks, such as pick-and-place tasks in semiconductor manufacturing processes, where it can adapt and optimize the control input based on historical performance data. In Figure 1, ff_j is the feed-forward signal where j is the index of a particular repeating task operation.

C. Multi-rate Sensor Fusion (MSF)

The multi-rate Kalman filter is applied to combine data from two sensors: the linear encoder and the vision sensor. Typically, the linear encoder operates at a higher sampling rate, providing position feedback represented as $z_{e,k}$ at intervals of $h_{encoder}$. However, it does not account for external disturbances. In contrast, a vision-based object detection algorithm, such as the one presented below, processes camera images to determine the precise object location, including any disturbances, which is represented as $z_{v,k}$. Vision measurements are collected at regular intervals of h_{vision} sample time, but image processing is computationally intensive. As a result, position information from the image captured at time-instance k , denoted as $img[k]$, becomes available after a delay of τ_v samples. A multi-rate multi-sensor fusion algorithm combines information from both the linear encoder and vision measurements and computes the estimated position information that is used in computing the control action. In particular, the work in [6] demonstrates that minimizing the sampling period of a vision sensor (e.g. at least to 1ms) reduces the error associated with the die position.

D. Vision Algorithm (VA)

The VA is depicted in Figure 2. This block is meant to calculate the distance of the chip from the left edge of the containment plane. The acquisition sensor captures an RGB image that is converted to grayscale. A detection algorithm,

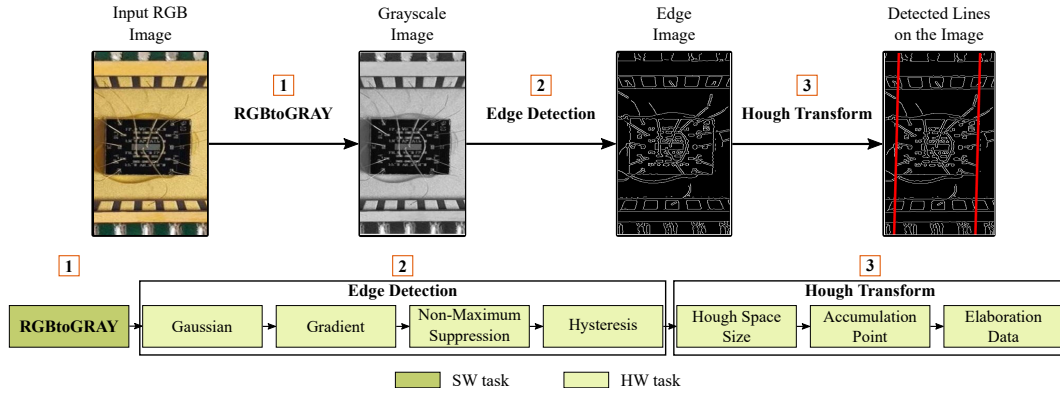


Figure 2: VA pipeline [6] with the proposed HLS-oriented implementation.

based on the Canny edge detector operator [16], is used to enhance the chip edges. Finally, the Hough transform is applied [17]. This latter is used to find lines within an image and it can be tuned to search for lines with specific directions. Considering the reference context, it takes advantage of the fact that the camera is observing rectangular chips with linear edges. Moreover, for both edge detection and the Hough transform, the use case led to their custom implementation. Specifically, only the identification of the vertical edges of the input image and their distance are relevant. Here are some more details on the implementation carried out in this paper:

1) **RGBtoGRAY**: Since the conversion from RGB to gray depends on the resolution of the camera adopted for input acquisition, this step can be removed from the processing pipeline to be accelerated, and possibly given to a SW task that, for example, can adapt it to the resolution of the implemented accelerator (e.g., by exploiting multiple accelerators).

2) **Edge Detection**: The edge detection algorithm highlights the vertical edges of the image by the following 4 steps. At first, the input image is convoluted with a gaussian filter in order to limit noise deriving from the environment in which the input image is captured (*Gaussian*). Then, the Sobel operator is applied to obtain 2 different gradients, both representing the magnitude and direction of the edges (*Gradient*). The next step (*Non-Maximum Suppression*, NMS) calculates the gradient similarity in order to keep only the gradient with the highest value of magnitude. Finally, the most significant edges are detected by applying a threshold function (*Hysteresis*, H).

3) **Hough Transform**: This enables feature extraction of an image. Originally, such an algorithm was created for the detection of simple shapes, such as straight lines, circles, or ellipses, being later extended to identify the positions of arbitrary shapes. In the context of the proposed VA, the Hough transform is applied to find vertical lines with limited slopes to a small range of angles. In this case, the processing of the algorithm involves 3 steps. The former (*Hough Space Size*, HSS) transposes the image resulting from the edge detection into Hough space. Then, the number of lines that occur in correspondence at each point in Hough space is examined and, if greater than a certain threshold, this identifies the position of vertical lines in a 2-dimensional space called Hough plane

(*Accumulation Point*, AP). In the latter, the distances obtained in the Hough plane are converted into pixels related to the identified lines, which can be plotted over the output image and of which the distance to the edge of the image is calculated (*Elaboration Data*, ED).

III. HLS-ORIENTED VA IMPLEMENTATION

High-Level Synthesis (HLS) is the compilation of high-level languages, such as SW programming languages, to FPGAs. This is of paramount importance to the mainstream adoption of FPGAs. HLS can also be seen as the translation of algorithmic descriptions to application-specific HW architectures. Most HLS tools use the C or C++ languages for the algorithmic specification, but other input languages such as Python, MATLAB, or SystemC are used. Examples of C-based commercial HLS tools are Vitis HLS [18] (former Vivado HLS), Stratus HLS [19], Catapult HLS [20], SmartHLS [21] (former LegUp). These tools enable the compilation of C, C++, and/or SystemC code for FPGAs or ASICs and can be effectively combined with other frameworks to provide system-level features support [22], [23]. The resulting HW accelerator can be optimized and customized using directives that specify some desired features to the compiler. The actual syntax and semantics of these directives vary depending on the tool. Common examples of directives (or *#pragma*) are pipelining and unrolling, which allow the resulting HW to execute loop iterations either in pipeline or in parallel, and memory resources, interface, and partitioning, which allow customization of the memory architecture and access. In addition, commercial HLS tools typically offer support for performance estimation and design debugging and verification.

A. IP-Level Implementation

The HW tasks of the VA (described in Section II-D) have been implemented as an Intellectual Property (IP). In this manner, its integration with a suitable MPSoC to follow the processor-coprocessor paradigm can be more easily handled [24], [25]. In the VA IP, each HW task has been defined as a submodule, and optimized with respect to its SW counterpart in order to adhere to the dictates of the HLS. Nevertheless,

HW Task	Optimization Techniques						
	Pipelining	Loop Unrolling	Array Partitioning	Data Sizing	Data Type Conversion	HLS-friendly Functions	Task Pipelining
Gaussian ^P	✓	✓	✓	✓	×	✓	✓
Gradient ^P	✓	✓	✓	✓	×	×	✓
Non-Maximum Suppression & Hysteresis ^P	✓	✓	✓	✓	✓	✓	✓
Hough Space Size ^P	✓	✓	✓	✓	✓	✓	✓
Hough Space Size	✓	✓	✓	✓	×	×	×
Accumulation Point	✓	✓	×	✓	×	×	×
Elaboration Data	✓	✓	×	✓	✓	×	×

Table I: Use of optimization techniques in the HW tasks.

for reasons related to accelerator optimization, tasks *Non-Maximum suppression* and *Hysteresis* have been merged into one block, and task *Hough Space Size* has been divided into two parts that can operate on part or all of the input image, respectively. Table I summarizes the optimization techniques used in each submodule. Pipelined registers have been added in each loop present in the submodules to increase throughput with limited additional area cost. In addition, the decision to totally or partially unroll loops has led to a reduction in latency, at the expense of a multiplication of associated resources. Further optimization in terms of timing has evaluated the use of independent memory blocks to speed up data access in almost all the tasks (exceptions are *Accumulation Point* and *Elaboration Data*). On the other hand, resource limitation has been managed through careful use of the size and type of data related to all registers and memories, and converting floating-point data to fixed-point data when necessary (i.e., *Non-Maximum Suppression & Hysteresis^P*, *Hough Space Size^P*, *Hough Space Size* and *Elaboration Data*). Moreover, the computational complexity implied by some mathematical functions required for processing (such as sine, cosine, arc-tangent, and square root), have been limited with optimized variants provided by HLS libraries or tabulated values. Finally, to reduce the execution time and limit the amount of resources (mainly memory blocks), the execution of tasks that can operate in independent parts of the image (involving edge detection tasks and *Hough Space Size^P*) is handled through a pipeline and linear buffers. Figure 3 shows the execution phases of the VA IP. Tasks that constitute the pipeline are focused on treating one pixel line of the input image at a time and labeled with ^P, while the other tasks operate on the whole image and are performed only one time per frame.

B. System-Level Implementation

In Figure 4, the envisioned test architecture for the H2020 IMOCO4.E project use case is depicted. A host is assigned to control the plant, where there are: i) the die-bonding machine, ii) silicon wafer transported by a conveyor belt, and iii) a camera that generates the input to be processed by the processor-coprocessor system. The processor-coprocessor

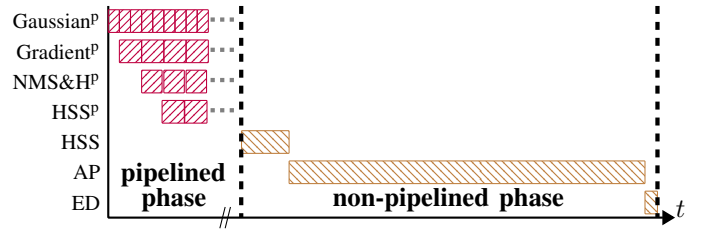


Figure 3: Task execution of the VA IP.

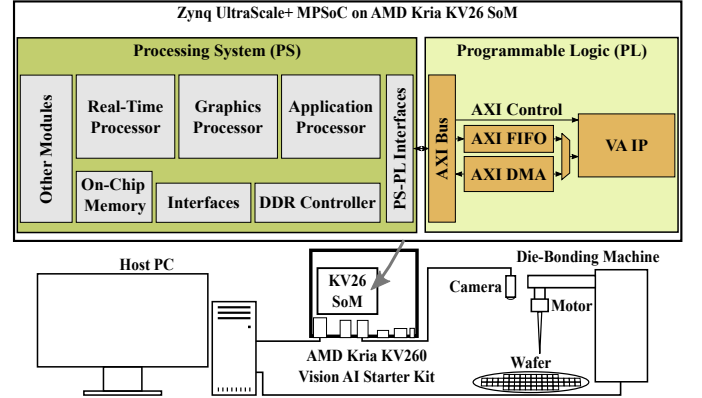


Figure 4: Overview of the envisioned architecture for the target application.

system has been deployed into an AMD Kria K26 SoM, which embeds a ZynqTM UltraScale+TM MPSoC [26]. The MPSoC is composed of the Processing System (PS) and Programmable Logic (PL) communicating through dedicated interfaces. PS consists of a quad-core application processor (which is the only processor used in this work), a dual-core real-time processor, a Graphics Processing Unit (GPU), a memory module, several peripherals, internal interconnections, interfaces with PL, and other modules (such as a power manager and a safety and security module). On the other side, the PL block implementation mainly consists of i) an AXI-based bus, required to enable data and control communication between processor and coprocessor; ii) a First-In First-Out queue (FIFO) for sending data in streaming via AXI-Stream protocol directly from the processor; iii) a Direct Memory Access (DMA) to enable the AXI-Stream transmission of data to the coprocessor while the execution of other tasks can proceed in the processor; and iv) the coprocessor represented by the HW accelerator of the VA.

IV. RESULTS

To assess the VA IP and its integration within a complete processor-coprocessor environment, Vitis HLS and Vivado Design Suite have been adopted (in their version 2023.1). Vitis HLS is a high-level synthesis tool developed by AMD capable of synthesizing Register Transfer Level (RTL) from the OpenCL, C, and C++ language descriptions. All the *#pragmas* described in Section III-A are made available to optimize the design: reduce latency, improve throughput performance, and reduce area and device resource usage of the resulting RTL code [27]. Vivado Design Suite [28] is the

design software for AMD adaptive Systems on a Chip (SoCs) and FPGAs. It includes: Design Entry, Synthesis, Place and Route, Verification/Simulation tools.

The system with the integrated VA IP has been synthesized using Vivado Design Suite at the maximum frequency of 230MHz. Consequently, all the following results have been extracted adopting such a reference frequency. In addition, the VA IP has been customized for the use-case scenario, which requires an input image of size 88x142 pixels (where each element represents a pixel in grayscale format).

A. Execution Time Evaluation

Table II provides a summary of the performance results achieved by comparing HW and SW executions. The SW execution results (SW columns in the table) have been obtained by running 10k executions of the VA SW IP on i) the PS of the Kria KV260 (SW-1), which integrates a quad-core Arm® Cortex®-A53 MPCore™ up to 1.5GHz; and ii) the 6-core AMD Ryzen 5 5600 up to 4.4 GHz (SW-2). For a fair comparison, the execution of the VA involved a single core in the software cases (SW-1 and SW-2) and one accelerator in the hardware case (HW), both evaluated at their maximum operating frequency. Each processing step involved in the Edge Detection and the Hough Transform described in Section II-D has been accelerated, resulting in an overall speed-up of 24.89x and 18.20x compared to SW-1 and SW-2 respectively. Indeed, the constraint set in [6] for the VA to effectively support the MSF of the industrial motion control system presented in Figure 2 is largely met, allowing additional scenarios with better precision error (see Section IV-C). Conversely, evaluated software executions (SW-1 and SW-2) may not meet this time requirement (see their maximum values). This is because the speedup values are skewed in favor of the proposed hardware implementation (especially in those tasks that repeat multiple times per frame, i.e., pipeline tasks), although the higher operating frequency of CPUs may favor speedups closer to 1 (or even in one case lower) for some tasks requiring operations more typical of a general purpose implementation. In any case, the designed VA IP completes in 37.37 us versus the requested 1 ms, and this constraint is met even considering the processor-coprocessor communication overhead. This latter includes the latency needed by the DMA to move the image to be processed in the coprocessing unit. In addition, the normalized execution time is about 3 ns/pixel, which is within the values present in state-of-the-art task acceleration related to Canny edge detection [29] and the Hough transform [30].

B. Resource Usage Evaluation

In terms of resources, the system requires a low percentage of PL usage, mainly dedicated to the VA IP (see Figure 5). These values reflect the use of optimizations applied from the original software code (as shown in Table I). In particular, the utilization of 16.48% Look-Up Tables (LUTs) is mainly attributable to the loop unrolling, which leads to the multiplication of hardware blocks required for parallel processing. In addition, the wide use of the pipeline directive certainly affects

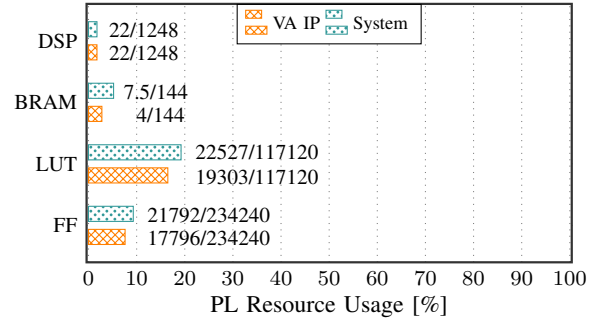


Figure 5: PL resource usage of VA IP and system (@230MHz) in terms of FF, LUT, BRAM and DSP of the AMD Kria K26 SoM.

the number of registers: 7.60% of Flip Flop (FF). Although the input image has been processed by the accelerator one pixel line at a time, 4 blocks of 36kb RAM corresponding to 2.78% of the Block Random Access Memory (BRAM) in the target FPGA have been needed. A limited amount (i.e., 1.76%) of Digital Signal Processors (DSPs) has been required due to the presence of typical image processing operations contained in the tasks *Gradient* and *Hough Space Size*. At the system level, further resources are associated with the presence of DMA, FIFO, and blocks for clock, reset, and interconnect management. Nevertheless, the VA IP enables to port additional functionalities on the co-processing unit (for example, as the MSF described in Section II-C) or multiple instances of the VA IP.

C. Precision Error Evaluation

To evaluate the effectiveness of the closed-loop control system which incorporates the multi-rate multi-sensor fusion II-C and employs edge detection and Hough transform II-D as a VA, we perform design-space exploration across different parameters and scenarios. Our evaluation considers high-precision industrial motion control applications such as pick-and-place tasks in semiconductor manufacturing systems. As done in [6], Table III presents the closed-loop control system's performance in terms of Steady-State Error (SSE), Mean Absolute Error (MAE), and Settling Time (ST).

In the pick-and-place task, ideally, five semiconductor dies are arranged consecutively with uniform spacing. The camera's region of interest focuses on the first die at $z_0 = 0\mu m$. The goal is to center the first die at a target position of $220\mu m$ by the end of the first iteration. In another scenario, external disturbances affect the die position, and in such cases, position inaccuracy cannot be measured by the linear encoder sensor, which only tracks the position of the wafer table where the semiconductor die is placed. However, the vision sensor can accurately measure the true position of the die using captured images and a VA.

Table III depicts two distinct scenarios. In the first scenario, an initial disturbance of $\Delta = 20\mu m$ is applied to the first semiconductor die. Subsequently, disturbances are applied to dies two, three, and four, with decreasing steps of $5\mu m$. The fifth semiconductor die does not experience any disturbance. In the second scenario, an initial disturbance of $\Delta = 40\mu m$

HW Task	Execution Times [ns]								Speedup		
	SW-1 (on Arm Cortex-A53)				SW-2 (on AMD Ryzen 5 5600)				HW	HW	
	μ	σ	min	max	μ	σ	min	max		vs SW-1	vs SW-2
Gaussian ^P	1.65E3	28.93	1.40E3	0.07E6	1.14E3	97.75	1.00E3	0.25E6	108.70	15.18x	10.49x
Gradient ^P	2.48E3	209.95	2.11E3	0.09E6	2.29E3	232.62	2.00E3	0.21E6	217.40	11.41x	10.53x
Non-Maximum Suppression ^P & Hysteresis ^P	1.17E3	2.01	1.09E3	0.09E6	757.98	44.19	600.00	0.26E6	200.01	5.85x	3.79x
Hough Space Size ^P	596.13	0.65	480.00	0.06E6	288.61	17.30	200.00	20.50E3	221.75	2.69x	1.30x
Hough Space Size	7.31E3	262.58	6.89E3	0.04E6	2.33E3	90.00	2.20E3	9.30E3	465.00	15.72x	5.01x
Accumulation Point	5.93E3	139.17	5.84E3	0.01E6	4.35E3	192.09	4.20E3	37.50E3	3.48E3	1.70x	1.25x
Elaboration Data	187.07	87.07	160.00	6.98E3	101.85	67.08	100.00	500.00	122.00	1.53x	0.83x
Total	0.93E6	0.03E6	0.93E6	1.02E6	0.68E6	54.02E3	0.67E6	1.28E6	37.37E3	24.89x	18.20x

Table II: Comparison between SW execution time (μ medium value, σ standard deviation, min minimum value, and max maximum value) and HW execution time (using a clock frequency of 230 MHz). *HW vs SW-1/2* indicate the HW speed-up compared to the SW-1/2 μ values.

Scenario 1: $\Delta_1 = 20, \Delta_2 = 15, \Delta_3 = 10, \Delta_4 = 5,$ and $\Delta_5 = 0$.								
$h_{encoder}$ [ms]	h_{vision} [ms]		Die position					
			1	2	3	4	5	
0.125	0.125	SSE [μm]	0.138	0.162	0.208	0.11	0.0312	
		MAE [μm]	0.826	1.216	1.362	0.364	0.022	
		ST [ms]	25.32	29.43	24.38	27.44	29.51	
0.125	0.25	SSE [μm]	0.200	0.280	0.326	0.191	0.0501	
		MAE [μm]	1.164	1.620	1.836	0.232	0.035	
		ST [ms]	18.62	25.34	30.43	25.73	20.49	
0.125	0.5	SSE [μm]	0.431	0.331	0.418	0.112	0.151	
		MAE [μm]	2.724	2.913	2.663	1.960	0.086	
		ST [ms]	19.32	23.52	30.11	28.12	23.16	
Scenario 2: $\Delta_1 = 40, \Delta_2 = 20, \Delta_3 = 10, \Delta_4 = 5,$ and $\Delta_5 = 0$.								
$h_{encoder}$ [ms]	h_{vision} [ms]		Die position					
			1	2	3	4	5	
0.125	0.125	SSE [μm]	0.246	0.283	0.220	0.186	0.065	
		MAE [μm]	0.932	1.463	1.530	0.364	0.030	
		ST [ms]	27.11	30.43	26.72	29.31	31.47	
0.125	0.25	SSE [μm]	0.259	1.910	0.511	0.231	0.091	
		MAE [μm]	1.466	1.814	2.313	1.112	0.070	
		ST [ms]	19.47	32.40	33.61	28.60	23.73	
0.125	0.5	SSE [μm]	0.611	0.530	0.502	0.311	0.220	
		MAE [μm]	3.180	3.412	3.120	1.170	0.090	
		ST [ms]	24.30	27.33	20.01	33.28	29.61	

Table III: Closed-loop performance of multi-rate sensor fusion in different scenarios. All the deviations (Δ_i), where $i = \{1, 2, 3, 4, 5\}$ for 5 dies are in μm .

is applied to the initial semiconductor die. Afterward, disturbances are applied to dies two, three, and four as well, with decreasing steps by a factor of 2. The fifth semiconductor die does not encounter any disturbance. In these scenarios, we consider the linear encoder operating sampling period as $h_{encoder} = 0.125ms$ and different VA sampling periods as $h_{vision} = 0.125ms, 0.25ms, 0.5ms$. Table III presents the performance in both scenarios. The SSE values are generally smaller than $1\mu m$ in almost all cases, although they may vary based on the nature of the disturbances. Notably, the performance in terms of SSE, MAE, and ST degrades when $h_{vision} = 0.5ms$ compared to the case with $h_{vision} = 0.125ms$.

Therefore, it is desirable to reduce vision delay and have a shorter h_{vision} , and the execution time performance of the proposed accelerator can meet such a sampling time.

V. CONCLUSION

This work offers further evidence of how a wider adoption of heterogeneous platforms, which combine elements of general-purpose processing with programmable logic, can help in performing various tasks required by modern industrial motion control systems. In particular, the main objective of this work was to demonstrate the possibility of improving system performance in terms of precision error as required by H2020 project IMOCO4.E, thanks to the acceleration of the VA block, by utilizing a coprocessing unit while leaving enough room to pack a complete vision-based control system on the same computing platform. Performance results shows as the proposed HLS-oriented solution can satisfy the use-case timing requirement, enabling an improvement in terms of precision error. To achieve this, two strategies have been followed to encourage software developers applied in such industrial sector to consider the proposed implementation flow. The former concerns the adoption of a target device, the AMD Kria KV260, which is built for advanced vision application development without requiring complex hardware design knowledge. Whilst it is a “non-production” version, not intended to be the final device in a commercial solution, the AMD K26 family of this SoM includes also a production version with more connectors, an eMMC non-volatile memory (allowing applications to be booted entirely on the SoM rather than from the SD card) and a larger temperature range. Moreover, it does have a complete operating system support to manage the entire software execution and facilitate the I/O. The latter was to start from a typical software-friendly implementation, and proceed to a more hardware-oriented version through the use of HLS. HLS-based design enables an easier approach to hardware acceleration for such developers, who can gradually focus on the various types of optimization. Finally, since the acceleration performance largely meets the requirements of the use case with synthetic input images, the proposed implementation can eventually be improved during a future assessment phase with realistic images and at the same time limit the precision error.

ACKNOWLEDGMENTS

This work was supported by ECSEL Joint Undertaking in H2020 project IMOCO4.E, grant agreement No.101007311.

REFERENCES

- [1] E. A. Lee, "Cyber physical systems: Design challenges," in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, 2008, pp. 363–369.
- [2] S. J. Oks, A. Fritzsche, and K. M. Möslin, "Engineering industrial cyber-physical systems: An application map based method," *Procedia CIRP*, vol. 72, pp. 456–461, 2018, 51st CIRP Conference on Manufacturing Systems.
- [3] S. Mohamed *et al.*, "The IMOCO4.E reference framework for intelligent motion control systems," in *ETFA*, 2023.
- [4] G. Van der Veen, J. Stokkermans, N. Mooren, and T. Oomen, "How learning control supports industry 4.0 in semiconductor manufacturing," 2020.
- [5] Power & Beyond. (2023) Global chip shortage 2023 - updates in June. <https://www.power-and-beyond.com/global-chip-shortage-2023-updates-in-june-a-878ccf26f2bb3b7c58ffec27b42b262f/>.
- [6] C. Jugade, D. Hartgers, S. Mohamed, D. Goswami, A. Nelson, G. v. d. Veen, and K. Goossens, "Improved positioning precision using a multi-rate multi-sensor in industrial motion control systems," in *2023 European Control Conference (ECC)*, 2023, pp. 1–8.
- [7] F. Ratto, S. Esposito, C. Sau, L. Raffo, and F. Palumbo, "Multithread accelerators on fpgas: A dataflow-based approach," in *13th Workshop on Parallel Programming and Run-Time Management Techniques for Many-Core Architectures and 11th Workshop on Design Tools and Architectures for Multicore Embedded Computing Platforms, PARMA-DITAM 2022, June 22, 2022, Budapest, Hungary*, vol. 100, 2022, pp. 6:1–6:14.
- [8] X. Shao and D. Sun, "A fpga-based motion control ic design," in *2005 IEEE International Conference on Industrial Technology*, 2005, pp. 131–136.
- [9] M. A. M. Prado, J. R. Reséndiz, D. C. T. Pérez, C. M. T. Hernández, and G. H. Ruiz, "Motion control with fpga," in *Field*, G. Dekoulis, Ed. Rijeka: IntechOpen, 2017, ch. 3.
- [10] C. Sau, T. Fanni, C. Rubattu, L. Fanni, L. Raffo, and F. Palumbo, "Feasibility study and porting of the damped least square algorithm on FPGA," *IEEE Access*, vol. 8, pp. 175 483–175 500, 2020.
- [11] J. Wang, M. Li, W. Jiang, Y. Huang, and R. Lin, "A design of fpga-based neural network pid controller for motion control system," *Sensors*, vol. 22, no. 3, 2022.
- [12] S. Jin, D. Kim, T. Song, Q. N. Le, and J. W. Jeon, "An fpga-based motion-vision integrated system for real-time machine vision applications," in *2009 IEEE International Symposium on Industrial Electronics*, 2009, pp. 700–705.
- [13] J. Y. Mori, J. Arias-Garcia, C. Sánchez-Ferreira, D. M. Muñoz, C. H. Llanos, and J. M. S. T. Motta, "An fpga-based omnidirectional vision sensor for motion detection on mobile robots," *Int. J. Reconfigurable Comput.*, vol. 2012, pp. 148 190:1–148 190:16, 2012.
- [14] G. J. García, C. A. Jara, J. Pomares, A. Alabdo, L. M. Poggi, and F. Torres, "A survey on fpga-based sensor systems: Towards intelligent and reconfigurable low-power sensors for computer vision, control and signal processing," *Sensors*, vol. 14, no. 4, pp. 6247–6278, 2014.
- [15] Motion Control Tips. (2021) The benefits of integrating vision into machine control. <https://www.motioncontrolltips.com/the-benefits-of-integrating-vision-into-machine-control/>.
- [16] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.
- [17] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, 1972.
- [18] AMD. (2023) Vitis HLS. <https://www.xilinx.com/products/design-tools/vitis/vitis-hls.html>.
- [19] Cadence. (2024) Stratus High-Level Synthesis. https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-level-synthesis.html.
- [20] Siemens. (2024) Catapult High-Level Synthesis and Verification. <https://eda.sw.siemens.com/en-US/ic/catapult-high-level-synthesis/>.
- [21] Microchip. (2024) SmartHLS Compiler Software. <https://www.microchip.com/en-us/products/fpgas-and-plds/fpga-and-soc-design-tools/smarthls-compiler>.
- [22] C. Rubattu, F. Palumbo, C. Sau, R. Salvador, J. Sérot, K. Desnos, L. Raffo, and M. Pelcat, "Dataflow-functional high-level synthesis for coarse-grained reconfigurable accelerators," *IEEE Embedded Systems Letters*, vol. 11, no. 3, pp. 69–72, 2019.
- [23] F. Ratto, T. Fanni, L. Raffo, and C. Sau, "Mutual impact between clock gating and high level synthesis in reconfigurable hardware accelerators," *Electronics*, vol. 10, no. 1, 2021.
- [24] C. Rubattu, W. Errico, and K. Marinis, "A low-overhead axi bridge on spacewire for multi-device spacecraft systems," in *2022 International SpaceWire & SpaceFibre Conference (ISC)*, 2022.
- [25] C. Sau, L. Fanni, P. Meloni, L. Raffo, and F. Palumbo, "Reconfigurable coprocessors synthesis in the MPEG-RVC domain," in *International Conference on ReConfigurable Computing and FPGAs, ReConFig 2015, Riviera Maya, Mexico, December 7-9, 2015*, M. Hübner, M. B. Gokhale, and R. Cumpulido, Eds. IEEE, 2015, pp. 1–8.
- [26] A. Xilinx. (2024) Kria KV260 Vision AI Starter Kit. <https://www.xilinx.com/products/som/kria/kv260-vision-starter-kit.html>.
- [27] AMD. (2023) Vitis High-Level Synthesis User Guide (UG1399). <https://docs.xilinx.com/r/2023.1-English/ug1399-vitis-hls/HLS-Pragmas>.
- [28] ——. (2024) Vivado Design Suite. <https://www.xilinx.com/products/design-tools/vivado.html>.
- [29] L. Guo and S. Wu, "Fpga implementation of a real-time edge detection system based on an improved canny algorithm," *Applied Sciences*, vol. 13, no. 2, 2023.
- [30] I. El Hajjouji, S. Mars, Z. Asrih, and A. El Mourabit, "A novel fpga implementation of hough transform for straight lane detection," *Engineering Science and Technology, an International Journal*, vol. 23, no. 2, pp. 274–280, 2020.