Received: 13 October 2022 | Revised: 9 November 2022 | Accepted: 14 November 2022

DOI: 10.1049/cds2.12135

IET Circuits, Devices & Systems

The Institution of Engineering and Technology WILEY

ORIGINAL RESEARCH

Automated quantification system for vision through polymer-dispersed liquid crystal double-glazed windows: Circuit implementation

Mohammed Lami | **Faris Al-naemi** | **Walid Issa**

Industry & Innovation Research Institute, Sheffield Hallam University, Sheffield, UK

Correspondence

Mohammed Lami, Industry & Innovation Research Institute, Sheffield Hallam University, Sheffield S1 1WB, UK.
Email: b8018683@my.shu.ac.uk

Abstract

Polymer-dispersed liquid crystal automated quantification system for vision through polymer-dispersed liquid crystal double-glazed windows: Circuit implementation (PDLC)-windows played an essential role in providing a visual comfort for occupants in commercial buildings recently. PDLC windows adjust the visible transparency of the glazing to control the daylight accessed to internal environments. A former study proposed an algorithm to quantify the vision through the PDLC glazing in terms of image contrast. The quantification algorithm determines the minimum level of transparency that maintains a comfortable vision through the window. This study introduced the implementation of a real-time automated system that achieves the vision quantification process. Firstly, system on-chip was utilised to realise the quantification algorithm, including contrast estimation. Secondly, the contrast determination action was re-implemented using MATLAB, Cortex-A9 microcontroller, and Cyclone V field programmable gate array field programmable gate array-chip. The implemented systems were evaluated based on the latency, throughput, power consumption, and cost.

KEYWORDS

Cortex-A9 microcontroller, FPGA, PDLC glazing, smart windows, system on-chip, vision quantification

1 | INTRODUCTION

Conventional windows in the external envelope of commercial buildings are no longer passive elements. Numerous studies proposed different modifications on the existed designs of windows to be smart elements in buildings recently [1]. The new forms add important roles to the windows to enhance the living environment for the occupants. Modern windows are feasible in renewable energy systems as they offer better thermal and daylight performance [2–4]; in addition, the windows are utilised for power generation in building-integrated photovoltaic windows [5].

For instance, Guo and Zhang [6] employed a multilayer glazing window as a heat recovery device. The study utilised the exhaust air from the internal environment to ventilate the window and enhance the thermal performance. The study realised an overall energy saving of 38.2%. A similar methodology was applied by Lami et al. [7] to ventilate a double-glazed window in commercial buildings. The ventilation process led to an energy saving of 83.1% and a declination in the heat transfer coefficient of 3.82 to 2.36 W/m2 K compared to the conventional double-glazed windows. Kim at el. [8] optimised the physical properties of an electrochromic smart window for the highest possible energy saving. Solar radiation through the window was reduced due to the shading effect of the electrochromic glazing, resulting in a maximum annual energy saving of 45% when the window was oriented to the south. The thermal, daylighting, and electrical behaviour performance of a semi-transparent photovoltaic (STPV) window was investigated by Olivieri et al. [9]. The windows were built from STPV modules that differ in visible transparency (10%–40%) and solar heat gain coefficient (SHGC) (0.655–0.734). The study concluded that the electrical generation efficiency was increased with the decreasing of the transparency;

however, this led to a reduction in the illuminance of the internal environment.

The new features in smart windows required special algorithms to accomplish their functions. To implement these algorithms, special software and hardware are needed. For example, Ismail et al. [10] used FORTRAN programming language to implement a numerical code by using a PC. The code was used to assess the thermal performance of three proposed windows. However, other programming languages are more efficient than FORTRAN such as an open-source C-language, and it is widely used in modern embedded systems nowadays [11]. Other studies employed MATLAB software tool [12] to implement their developed algorithms. These algorithms determine the transparency level of a polymer-dispersed liquid crystal (PDLC) windows [13] or to control the operation of multilayer ventilated windows [14, 15]. But the industrial applications require standalone embedded systems [16] that compensate the need for PC-based software. For example, Dobrojevic and Bacanin [17] utilised a Raspberry Pi-based embedded system [25] with Python programming language [18] to control an intelligent home over Internet of Things (IoT). Elkholy et al. [19] implemented a field programmable gate array field programmable gate array (FPGA)-based energy management system for a three sources hybrid microgrid. The FPGA system showed a high response against the changes of the load. A grid interfaced solar photovoltaic (GIPV) system has been developed by Kumar et al. [20] based on an ARM Cortex-M4 core. The ARM core controls other peripheral entities such as the voltage source converter and the DC-DC boost converter to realise the interface between the solar grid and PC.

However, in some applications when the processed data has a relatively high size such as images or videos, then it is essential to employ high processing speed embedded systems. Liu and Feng [21] proposed a modified object edge detection (Zynq) algorithm. The new algorithm was designed using a special SoC FPGA-based hardware platform. The proposed Zynq algorithm exploits the hardware acceleration blocks on the FPGA chips. Jaskolka et al. [22] emphasised the high capability of the Raspberry Pi 3 SoC system to implement image and video processing tasks, and it can replace the MATLAB-based realisation method. In addition to its low cost, Raspberry Pi 3 uses the freely available open-source Python programming language. Pereira [23] introduced four scenarios to implement an image processing algorithm. The algorithm was dedicated to detecting cracks task in buildings for civil applications. The scenarios included realising the proposed algorithm using image processing toolbox in MATLAB and without it, using PC-based C/C++ coding and Raspberry Pi-3 implementation. The processing time for the crack detection process was calculated in different scenarios. The shortest execution time was when the MATLAB image processing toolbox was employed. Wang [24] proposed a neural network method for English vocabulary translation. Relying on the FPGA Xilinx type chip raised the accuracy of vocabulary detection to 98%, whereas the existing accuracy for the FUZZY method is 92.35%.

Implementations of digital circuits can be evaluated in different bases. The processing speed of a circuit reflects the rate of data manipulation, and it can be measured in terms of the latency or throughput. For instance, Ahmad et al. [25] introduced a method to determine the cost of FPGA realisation for a given function. The study used the proposed method to evaluate the FPGA implementation of different circuits based on the latency and throughput. The experimental result showed that the latency and throughput of four mathematical operations were 2382 clock cycles and 0.178 mega inversions per second, respectively. Paunski et al. [26] compared the performance of different single board SoC systems for automated applications. The performance of Raspberry Pi-3, Rock64 and LattePanda was observed under the same operational conditions. LattePanda system showed the lowest power consumption of 3.6 W while the power consumed by the Raspberry Pi-3 was 7.5 W. In addition, Rock64 system achieved a face detection algorithm in 311 s; however, the same task was executed in 757 s by the Raspberry Pi-3 system.

Lami et al. [27] introduced a system to quantify the vision through a polymer-dispersed liquid crystal double-glazed window in commercial buildings. The study concluded that it is possible to quantify the vision through windows glazing in terms of image contrast [28, 29]. The experimental part of the study included the procedures of vision quantification process. Vision quantification process contains two main steps: Firstly, 16 digital images were taken; each image was taken at a different level of visible transparency of the PDLC film. Then the contrast for the images is determined individually. Secondly, the preferred vision level is derived from the contrast/transparency curve based on the preferred vision ratio (PRV), which was set by the user. A MATLAB-based code was used to estimate contrast values. The experiment was executed every 30 min during the daytime manually.

However, the proposed quantification process should be addressed by a standalone automated system and does not depend on a PC-based software in practical applications. In addition, the process should be executed spontaneously and within a shorter time than 30 min. Shortening the repetition time is necessary to cover the variation of outdoor illumination in real-time. Therefore, this paper proposed an automated system to handle the vision quantification algorithm. Firstly, an algorithm was introduced to mimic the quantification process of the previous work including contrast calculations [27]. Secondly, an ARM System on Chip (SoC) system-based was developed to realise the proposed algorithm; the SoC system was tested experimentally. Thirdly, the proposed contrast estimation task was re-implemented using alternative systems: MATLAB, ARM Cortex-A9 microcontroller, and Altera Cyclone V FPGA chip-based implementation. Fourthly, the performance of the implemented systems was evaluated in different basis: latency, throughput, clock consumption, power consumption, and cost. This paper has the following contributions:

a. Introducing an implementation method for the proposed vision quantification algorithm to fit practical applications with more optimised execution time.
b. Assessment of the algorithm execution and producing analytical comparison between different solutions

2 | METHOD AND THE PROPOSED SYSTEM

Since the purpose of this work is to realise the algorithm proposed by Lami et al. [27], firstly it is essential to go through the following steps that are employed to achieve the quantification process:

1. An image processing model is responsible for determining the contrast of a set of 16 images that were manually captured through the PDLC-window. Each image was taken at a certain level of visible transparency for the PDLC film.

Contrast computation was relied on the RMS contrast technique [30], as illustrated in Equations (1) and (2):

$$\text{RMS Contrast} = \sqrt{\frac{1}{M * N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \left(I_{ij} - \overline{I}\right)^2} \quad (1)$$

Where:

M and N are the column and row numbers of an image I respectively
I_{ij} is the pixel value which lies in the ith column and the jth row, and
$\overline{I}$ is the average brightness of all pixels of the image I and it can be estimated in Equation (2) in the range of [0, 1]:

$$\overline{I} = \frac{1}{M * N} \sum_{i=0}^{M*N-1} I_i \quad (2)$$

2. A mathematical model in charge of computing the voltage level is required to drive the PDLC film. This will maintain a certain level of visible transparency based on a preferred vision ratio (PVR) set by the user.

The contrast values obtained by the image processing model is sketched against the corresponding voltage as shown in Figure 1. As depicted in the figure, contrast values were drawn at ac voltage values: 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 120, 140, 160, 180 and 200 in volt. Some important points on the curve were defined as below:

- C_{min} and C_{max} are the minimum and maximum contrast values, respectively.
- $(v_{\text{th}}, C_{\text{th}})$ is the threshold contrast point where C_{th} is the threshold contrast value at which contrast values start saturating and v_{th} is the corresponding voltage. The threshold contrast value was determined by Equation (3):

$$\text{Threshold Contrast } (C_{\text{th}}) = 0.707\ (C_{\text{max}} - C_{\text{min}}) \quad (3)$$

To estimate the voltage (v_{PVR}) for the PDLC film that provides a preferred visible transparency, two points on the contrast curve should be known (C_a, v_a) and (C_b, v_b); the two points lie directly before and after the preferred point (C_{pvr}, v_{pvr}). The quantification algorithm should track the latter point based on the preferred vision ratio (PVR) that is set by the user. The range of PVR starts from 0 (provide minimum allowed visibility through the window), while the maximum value of PVR is 1 (provide maximum available visibility through the window). However, the instantaneous value for the PDLC film (v_{PVR}) was estimated by Equation (4):

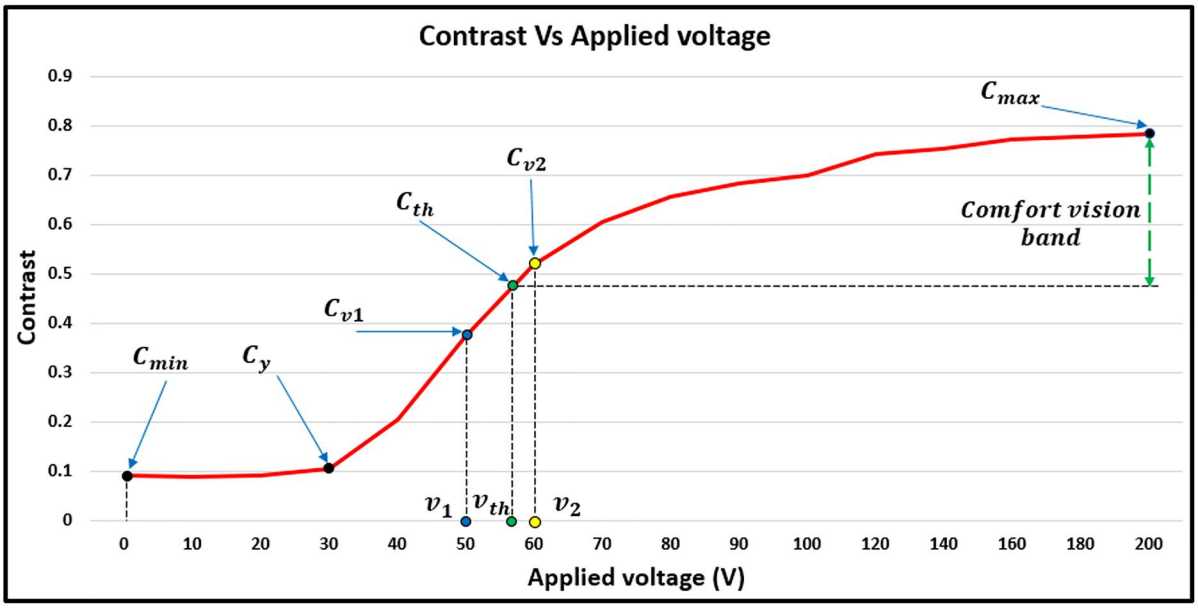

FIGURE 1 Contrast curve

$$v_{\mathrm{PVR}} = v_b - \left[(v_b - v_a)\left(\frac{C_b - (\mathrm{PVR}(C_{\max} - C_{\mathrm{th}}) + C_{\mathrm{th}})}{C_b - C_a}\right)\right] \ldots\ldots 0 \leq \mathrm{PVR} < 1 \quad (4)$$

This study relies on the workflow as shown in Figure 2. Suitable hardware and software were prepared to implement the quantification action. The proposed algorithm will be introduced in the next sections; a system on-chip (SoC) system was utilised to implement the quantification algorithm. On the other hand, three alternative methods were employed to implement the quantification process: MATLAB-based, microcontroller MC-based, and field programmable gate array technique FPGA-based implementation methods. Finally, the performance of the realised systems was evaluated and compared comprehensively.

The proposed quantification algorithm is shown in Figure 3. The quantification algorithm proposed by the previous study (Lami et al. [27]) has multiple stages: image capturing, contrast computation, estimation of PDLC voltage, and repetition time estimation. However, the previous study did not have whole parts of the proposed algorithm in one chart to show how the system can work to achieve it. Instead, the previous study executes different step manually. In addition, the previous study did not add the repetition time step. Therefore, the present study gathered all the mentioned parts of the quantification process and added the repetition time stage as shown in Figure 3. Since the whole process should be executed within a specific time (T_r) periodically, the algorithm starts by storing the current time (t_1). Then a group of images should be taken and stored in a storage area. The main controller instructs a PDLC power driver to generate the first voltage level (0 V) as mentioned earlier. After setting the ordered voltage, a high-definition HD camera takes one shot through the PDLC window and store it at a memory storage device. The controller instructs the power driver to generate the next voltage level, and the camera will take another shot through the window. The operation is repeated for the remaining images. Finally, the 16 images should be stored at the memory of the system; each image is taken at a certain voltage level as mentioned earlier.

The next part of the process is responsible for contrast computation. The stored images should be fetched individually to the RAM, where the controller can access and process them. The controller extracts the RGB data of the images and computes the contrast according to Equations (1) and (2). At this stage, image processing implementation has been accomplished; the system now determines all the 16 (contrast/voltage) points on the contrast curve as explained earlier in Figure 1.

The next action is estimating the voltage required to drive the PDLC film to maintain the preferred transparency based on the PVR ratio. The controller first determines the milestone points on the contrast curve and then determines the corresponding voltage according to the Equations (3) and (4). The voltage level is then derived by the controller to be sent to the PDLC power driver; the voltage level could be a binary code that reflects the preferred voltage value estimated by Equation (4). The power driver will encode it and generate the request voltage for the PDLC film. Finally, it is important to record the elapsed time at this step ($t_2 - t_1$). Then the system decides to wait for (T_r − elapsed time) to complete the full cycle (T_r) before re-executing the quantification algorithm.

3 | EXPERIMENTAL SETUP

An experiment with four scenarios has been executed to realise the circuitry for the quantification process controller. The following is a demonstration of the experimental scenarios, including purpose, design, hardware, and software setup.

3.1 | Scenario 1: System on-chip SoC-based controller

A SoC-based system Raspberry Pi-3 (RPi-3) was deployed to realise the quantification process in Figure 3. Therefore, the

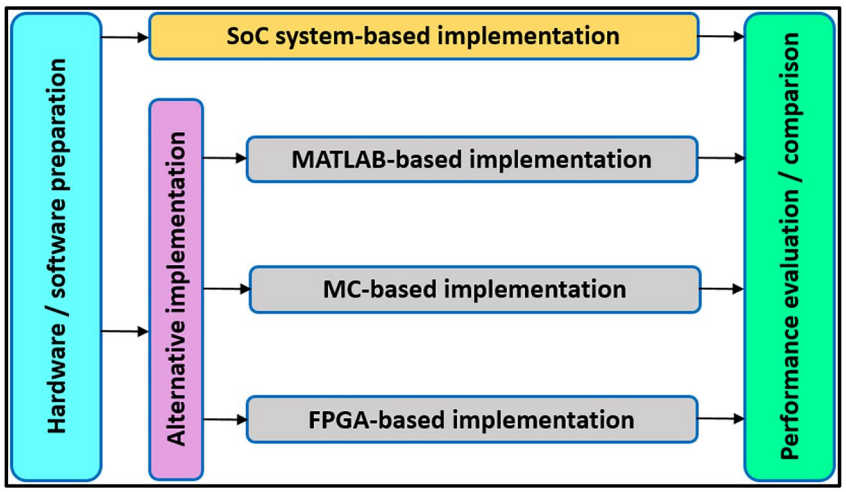

FIGURE 2 Workflow

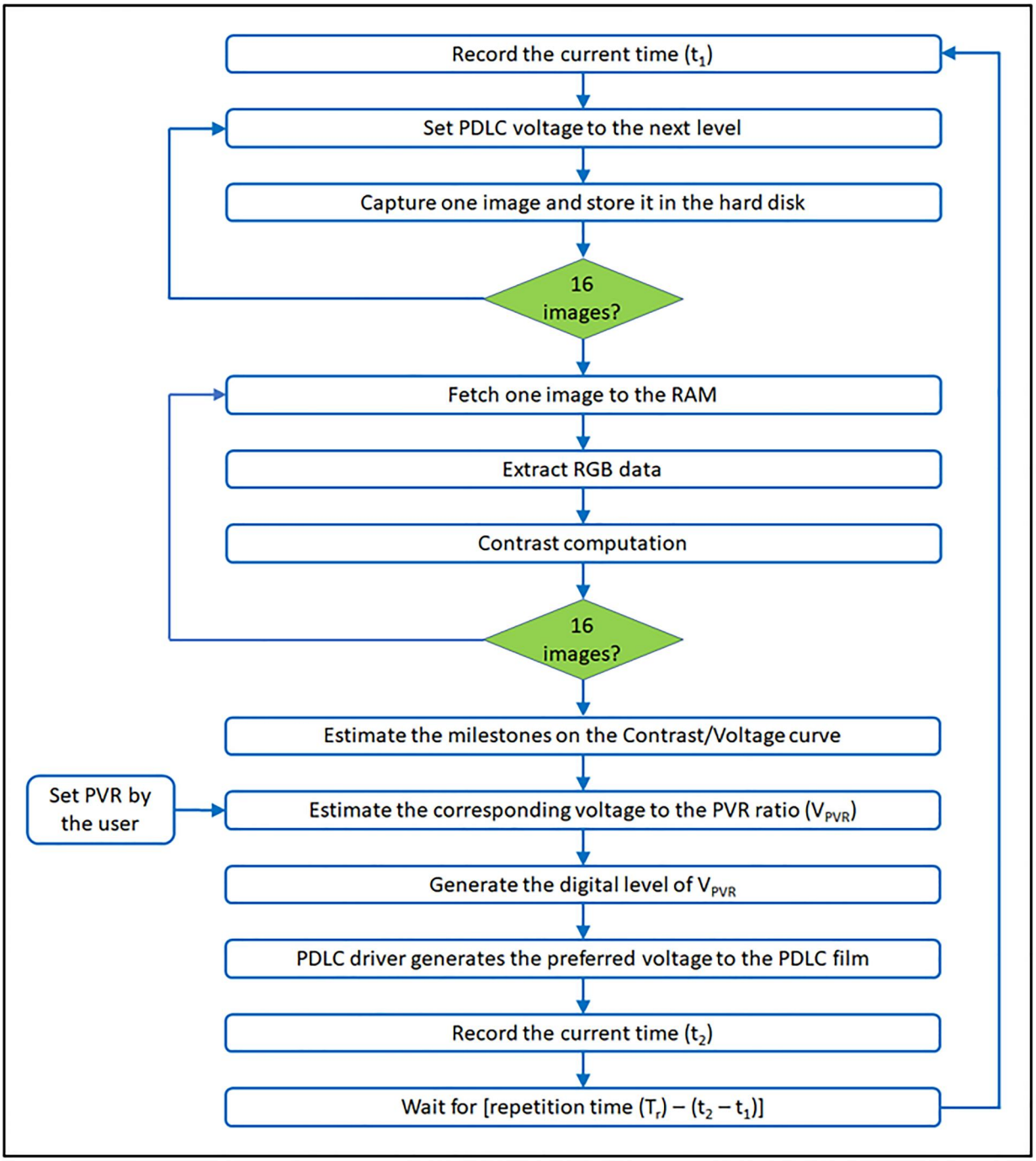

FIGURE 3 The proposed quantification algorithm

hardware and software were prepared as shown in Table 1. The Raspberry Pi-3 board [31] provides an open source SoC system managed by Linux. The quantification algorithm in Figure 3 was interpreted into a Python code to be executed by the SoC system. The Python code controls the peripheral devices to achieve the proposed process. The images were taken by an HD camera, which is compatible with the SoC system. The voltage level decided by the SoC system is a 10-bit digital signal. In other words, the range of the PDLC AC voltage is divided into (2^{10} or 1024) level. The digital signal (SoC decision) is sent to an ATmega328P-based microcontroller (Arduino board). The latter microcontroller is responsible for generating a pulse-width-modulated (PWM) signal based on the received decision. The PWM signal is forwarded to the power driver to generate the preferred voltage level for the PDLC film. A PC was used to monitor the operating system of the SoC system. Figure 4 depicts the setup of scenario 1.

TABLE 1 Experimental setup

Scenario	Item	Specifications	
1	SoC system	CPU	Quad core ARM Cortex A-53 (1.5 GHz)
		RAM	1 GB DDR2 SDRAM
		Hard disk	External SD card (16 GB)
		Operating system	Linux
		Programing language	Python 3
	HD camera	Module	Raspberry Pi 5 Megapixel camera
		Resolution	Up to 2592 × 1944
	Power driver	Module	Triac-based AC dimmer
		Rating	AC current 16 A, 600 V
		Microcontroller	ATmega328P- based MC Arduino UNO
2	Software	WINDOWS-based MATLAB version 2022	
	Programing language	MATLAB-based programing language	
	Operating system	64-bit WINDOWS 10	
	PC	Intel quad Core i7 2.2 GHz, 8 GB RAM	
3	Microcontroller	Dual core ARM Cortex-A9 microcontroller (50 MHz)	
	Software	ARM Development Studio IDE 2022.1	
	Programing language	'C'	
4	FPGA chip	Altera Cyclone V	
	Software	Quartus Prime 21.1 (Lite Edition)	
	Programing language	VHDL	

Abbreviation: FPGA, field programmable gate array.

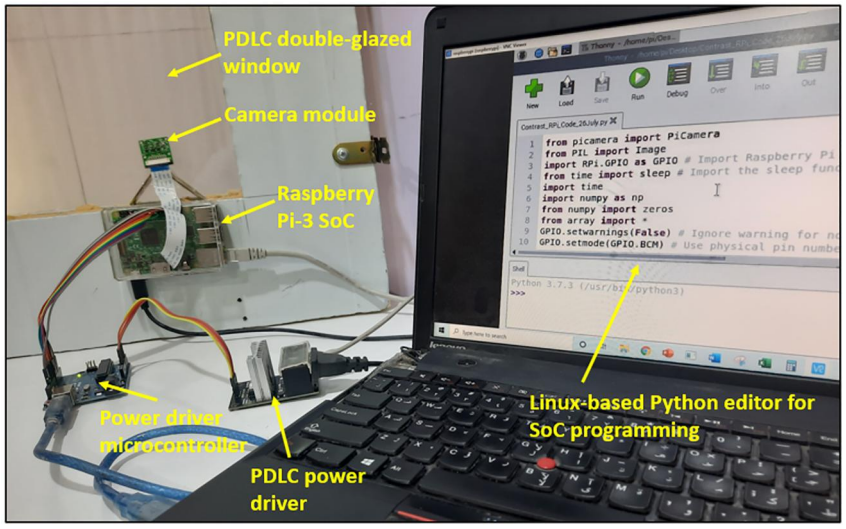

FIGURE 4 Experimental setup for scenario 1

3.2 | Scenario 2: MATLAB-based contrast estimator

In this scenario, a MATLAB code has been prepared to implement only the contrast computation section in the proposed quantification process. The code starts by reading the RGB data of the images and compute the contrast individually. Table 1 demonstrates more details about the hardware and software setup of scenario 2.

3.3 | Scenario 3: Microcontroller-based contrast estimator

In this scenario, an ARM Cortex-A9 microcontroller has been utilised to realise the contrast estimator part in the proposed quantification algorithm. The microcontroller is a built-in Altera Cyclone V FPGA chip. The microcontroller relates to a client via a bare-metal bus [32]. The ARM Development Studio IDE software (in the client side) mediates between the

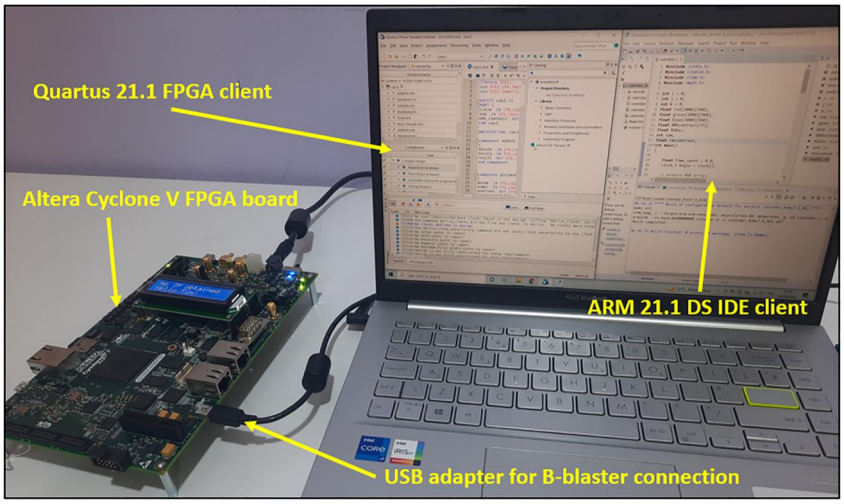

FIGURE 5 Experimental setup for scenario 3 and 4

programming code and the microcontroller. A code was written using the well-known 'C' programming language to execute the contrast action. Figure 5 and Table 1 show more details about the setup of scenario 3.

3.4 | Scenario 4: FPGA-based contrast estimator

In scenario 4, the field programmable gate array technique was employed to build the contrast estimator circuit. Altera Cyclone V provides 41,910 configurable logic blocks (CLBs), which were utilised to construct the contrast estimator. The FPGA design is managed by the Quartus software [33].

The estimator circuit shown in Figure 6 was prototyped by the FPGA chip to determine the contrast value for one image at a time. The design relies on the contrast Equations (1) and (2). The intellectual properties (IPs) in the library of Quartus software were used in the design. The top-level design of the estimator circuit was described using Very High Scale Integrated Circuit Hardware Description Language (VHDL) [34]. The detailed timing diagram is depicted in Figure 6. The contrast estimation process starts by resetting the output of all components in the circuit. Once the accumulator (ACC1) determines the summation of the Equation (2) ($\overline{I} = \frac{1}{M*N} \sum_{i=0}^{M*N-1} I_i$), the divider (DIV1) circuit divides the summation value by the number of pixels times three (3MN). At this stage, the average brightness $\overline{I}$ of the input image I has been computed. This step requires (3MN) clock cycles to accumulate the RGB data for all pixels, in addition to one clock for the division operation. The term $(\sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (I_{ij} - \overline{I})^2)$ in Equation (1) is determined by the accumulators (ACC2 and ACC3), the divider (DIV2), the subtractor (SUB) and the multiplier circuits. For every pixel, the average pixel brightness (I_{ij}) is estimated by accumulating the RGB data for a particular pixel (by ACC2); then it is divided by three (by DIV2). The average brightness of all pixels ($\overline{I}$) is subtracted from the average brightness of the current pixel I_{ij} by the subtractor (SUB). The computed difference of the current pixel is squared by the multiplier (MUL). The squared difference $(I_{ij} - \overline{I})^2$ for all pixels is accumulated by the accumulator (ACC3). This step requires eight clock cycles for one pixel; hence, it consumes (8MN) clock cycles. The accumulated output of (ACC3) is divided by the number of pixels ($M * N$) by the divider (DIV3). Finally, the RMS contrast value is obtained by finding the square root for the output of (DIV3) by the square root circuit (SQRT). It is assumed that the clock signals for different entities are produced by a separate clock distribution circuit. Each entity in the FPGA circuit is triggered by a clock signal at appropriate times to achieve the mathematical computation sequence of Equations (1) and (2). The number of consumed clocks were shown in the timing diagram of Figure 6 for different stages in terms of column M and row N of the image. The clock distribution circuit receives a universal clock signal from the clock source on the FPGA chip.

4 | RESULTS AND DISCUSSION

The objectives of this section are:

1. Demonstrating and comparing the experimental results of the implemented scenarios mentioned in Table 1.
2. Investigating the feasibility of the implemented systems in terms of latency or processing-speed, throughput, power consumption, and the cost.

The Raspberry camera was attached to the SoC system and fastened securely in front of the window. The power driver was connected to the SoC system, and its output was connected to the PDLC film. Before executing the Python code, the time required by the SoC system to take one image was investigated as shown in Figure 7. This time is necessary to know as it is

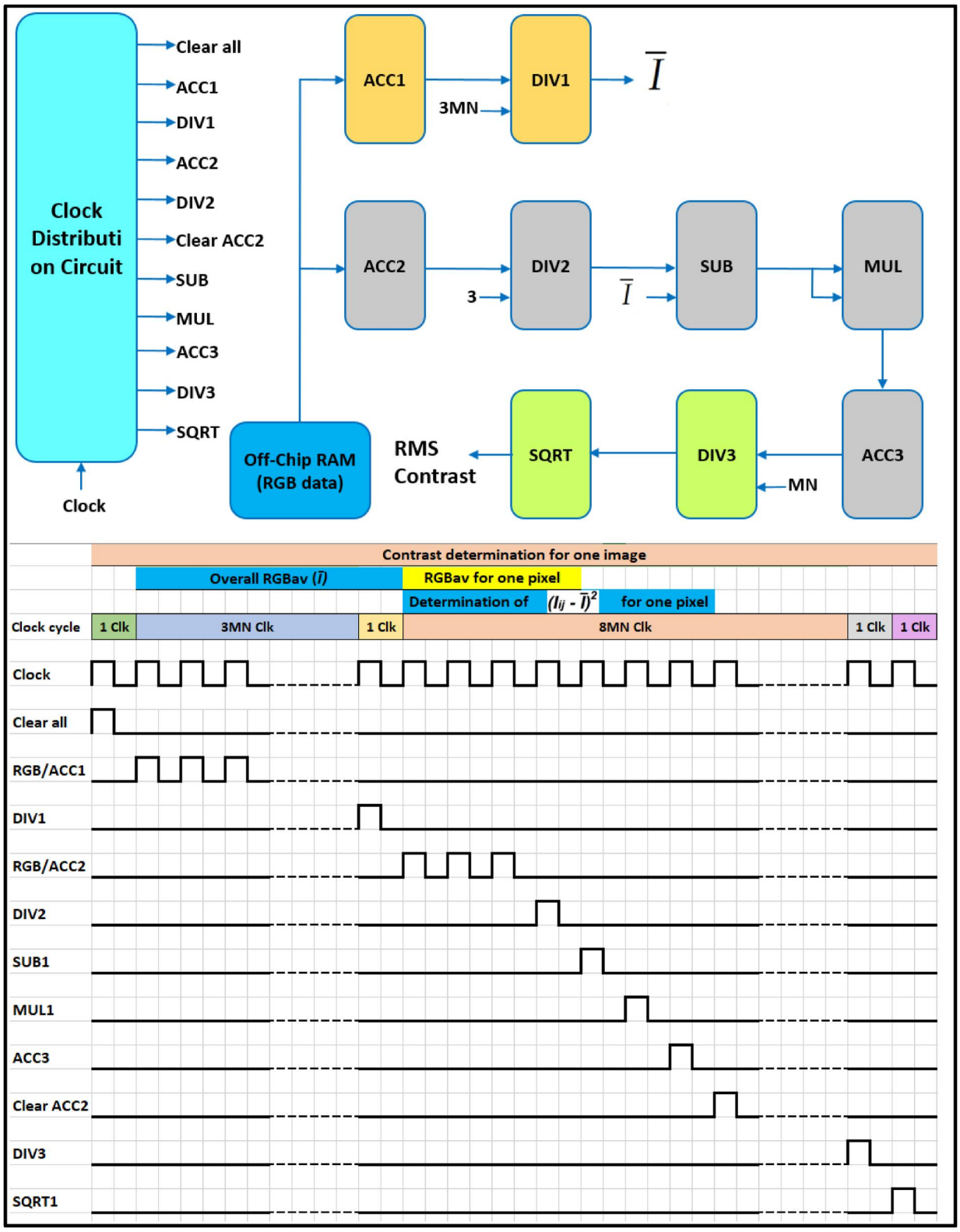

FIGURE 6 FPGA-based contrast estimator circuit and the timing diagram. FPGA, field programmable gate array

considered as a delay which should be added to the overall latency of the system. The following steps were followed to define the period for each step:

1. To capture one image, the SoC system generates the code that reflects the required voltage level and sends it to the power driver. The power driver needs a time to adjust its

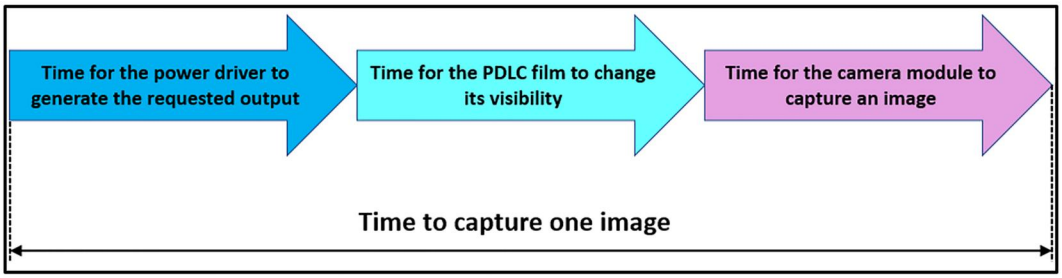

FIGURE 7 Time to capture one image

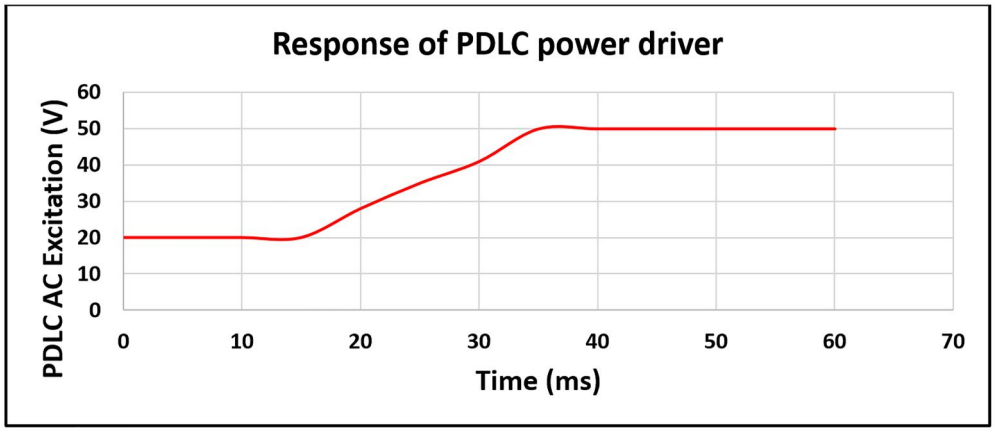

FIGURE 8 Time response of the power driver

output according to the requested level. A separate experiment was performed to examine the time response of the power driver. A well calibrated AC voltage sensor was prepared to observe the output of the power driver. While the output is at 20 V, the power driver was requested to generate a 50 V (20V and 50V were arbitrary chosen); the power driver responds as shown in Figure 8. The time response revealed that the output was jumped to 50 V within 20 msec.

2. Regarding the time response of the PDLC film against input voltage change, it can rely on the previous research examination of this topic. Pozhidaev et al. [35] concluded that the transition time of a PDLC film from opaque to translucent state is 10 msec approximately.
3. A Python code was prepared (in a separate experiment) to inspect the time taken by the camera module to take an image. The same SoC system and camera module of the experiment were used. The test showed that the camera needs 531 msec to take one image. This time includes image capturing and saving the image in a particular directory on the memory of the SoC system.

Consequently, it can be concluded that the SoC system required 561 msec (20 msec + 10 msec + 531 msec) to capture one image or approximately 8.98 s for 16 images. This time will be considered as a latency in addition to latency due to the other tasks of the proposed quantification algorithm.

After running the Python code of scenario 1, the SoC system executed the quantification process; 16 1080 × 960 images were taken and stored at the memory. The image group was used for contrast computation in this scenario as well as the other scenarios. The obtained contrast values were applied to the mathematical model of the quantification process to generate the preferred voltage level. The PVR ratio was set to an arbitrary value. Finally, the power driver fed the PDLC film the preferred voltage according to the SoC decision. The throughput of the experimental systems can be estimated based on the size of input data and the latency.

$$\text{Throughput of vision quantification SoC system} = \frac{\text{Data size}}{\text{Latency}} \text{ (bit per sec)} \quad (5)$$

The input data in all the experimental scenarios is 16 images. The experimental systems deal with the RGB data of the images. Thus,

$$\text{Input data size} = 3KMN \text{ (bit)} \quad (6)$$

Where M and N are column and row numbers of the image pixels, and K is the number of bits per colour; in this case, $K = 8$ as the input images are 8-bit-coloured images.

The Python code was suitably organised to observe the time consumed by different steps while executing the programme. The result was as follows:

- Image capturing and storing (at SoC memory) time = 8.98 s.
- Image fetching (from the hard disk to the RAM) = 0.12 s.
- Contrast computation time = 451.69 s.
- Estimating voltage level for the power driver = 0.02 s.
- Approximately 0.86 Mb/sec throughput.

According to the second scenario, the 16 images that were taken by the camera in the first experiment (scenario 1) were stored at the hard disk of a Windows-based PC. The location of the image group was properly set in the MATLAB code so that the programme can access them during the execution. The MATLAB software utilised dual CPU cores from the PC processing resources while running the programme. The code included a 'TIC-TOC' instructions to estimate the execution time for the contrast calculation section. The programme consumed 2.98 s to compute the contrast values of the images. The recorded throughput of the system is 133.6 Mb/sec.

For the third scenario, the images taken in the first scenario were applied to the ARM A9 microcontroller system for contrast computation. The storage directory of Images was stored was included in the 'C' code to be accessed by the programme. The code also included instructions to estimate the time consumed for contrast calculation. A USB blaster link was initiated between the ARM development studio IDE software (PC side) and the on-chip CPUs (on Cyclone V FPGA board). The code was compiled by a 'C' compiler to generate the source code to be executed by the microcontroller. The running of the code showed that the contrast computation action consumed 8.36 s. The observed throughput was 47.62 Mb/sec.

Regarding the fourth scenario, a new FPGA Cyclone V chip-based project was created in Quartus prime software. The hardware of the digital circuit of Figure 6 was described using VHDL language. Intellectual properties were imported from the library of Quartus software to implement the circuit. A VHDL code was written as a top-level design that contains the detailed description of the wiring between different entities of the circuit. The off-chip memory in Figure 6 was excluded from the FPGA design, as it is located outside the FPGA chip. The RAM holds the RGB data of the 16 images obtained from scenario 1. The off-chip memory (an SD RAM) and the FPGA chip are existing on a 'Cyclone V FPGA board'. The project was compiled, and the following facts were extracted from the Quartus software about hardware resource utilisation:

- Configurable logic block utilisation: 2493 out of 41,910 (≈6%).
- Digital signal processing block utilisation: 3 out of 112 (≈3%).
- Clock source utilisation: on-chip clock source (50 MHz).

Referring to the timing diagram of the FPGA design in Figure 6, the number of consumed clock cycles for the contrast computation of one image is as follows:

$$\begin{aligned}&\text{Clock cycles for Contrast computation}\\&= (11MN + 4)\ \text{clock cycle}\end{aligned} \tag{7}$$

Where M and N are column and row numbers of the image pixels. Then the latency of the FPGA contrast estimator is as follows:

$$\text{Latency of FPGA contrast estimator} = \frac{11MN + 4}{F_{CLK}} \tag{8}$$

As the taken photos in all scenarios have a resolution of (1080 × 960) and the circuit used the 50 MHz (0.02 μsec cycle time), the latency of the FPGA contrast estimator is (0.228 s). As a result, the throughput is 109.14 Mb/sec. However, the results of the implemented scenarios have been listed in Table 2.

The following sections will evaluate and compare the implemented systems in the experimental scenarios according to the obtained result. Different point of views will be discussed to point out the pros and cons of each system.

4.1 | Latency

The experimental result of the Raspberry Pi-3 SoC system shows that the contrast determination process consumed 451.69 s. In other words, it caused 98% of the whole programme's latency. The observed execution time to run the whole algorithm is 7.68 min, However, this latency will not meet the practical requirements when the preferred repetition

TABLE 2 Results summary

Implementation method	For 16 images			For 1 image
	SoC system	MATLAB	Cortex-A9 MC	FPGA
Contrast computation (s)	451.69 s	2.98	8.36	0.228
Throughput (Mb/s)	0.86	133.6	47.62	109.14
Other facts	Time for other parts of the algorithm = 9.12 s	---	---	Configurable logic block utilisation (6%) DSP block utilisation (3%)

Abbreviation: FPGA, field programmable gate array.

time (T_r) of the quantification process is less than it. Therefore, an alternative solution should be considered to reduce vision quantification running time. Since the effective delay was in contrast estimation, the implemented 'contrast estimation systems' in the remaining scenarios will be examined to resolve this issue.

MATLAB implementation provided an effective reduction in the latency of 2.98 s. Consequently, the quantification process will be accomplished within ($T_r = 12.1$ s) compared to ($T_r = 460.81$ s) in the case of the Raspberry Pi-3 SoC system. On the other side, the ARM Cortex-A9 bare-metal implementation gives a powerful performance. The latency was dropped to 3.8% compared to the Raspberry Pi-3 SoC system. This offers a reduction in the quantification process time to ($T_r = 17.48$ s). The fourth scenario depicted a decline in the contrast computation to 0.228 s, and it is 1981 times faster than the SoC system in the first scenario. In conclusion, the less latency system the higher processing speed system in the FPGA contrast estimator.

4.2 | Throughput

The throughput of a system is reversely proportional to its latency. Hence, the less latency, the more the throughput. For example, the utilised SoC system showed the least throughput of 0.86 Mb/sec among other systems as it works with the highest latency. In contrast, the maximum throughput that occurred is 109.14 Mb/sec by the FPGA implementation technique at minimal latency. However, the presented FPGA design achieves the contrast computation for one image only. In the following sections, the possibility of using the residual resource in the chip to achieve the contrast computation for the 16 images will be discussed. Therefore, the throughput and other design parameters of the FPGA circuit will be redeclared later.

4.3 | Power consumption

To evaluate the power consumption of the presented experimental systems, they can be classified into two groups. The Raspberry SoC system in scenario 1, and the PC-based MATLAB system in scenario 2 can be identified as computers. While the ARM Cortex-A9 microcontroller and the FPGA system are considered as non-computers, previous studies showed that the power consumption of normal PCs is greater than that for Raspberry SoC systems. For instance, Bekaroo and Santokhee [36] investigated the power consumption of different kinds of computers; the computers under examination were performing a number of tasks. They concluded that a laptop computer consumes 17.68 W which is four times more that the consumption of the Raspberry SoC system. On the other hand, former studies concluded that the power consumption of the FPGA-based DSP implementation is less than DSP processors. For example, Bai et al. [37] implemented a DSP algorithm by using an Altera FPGA chip and another method by using a DSP processor from Texas Instruments [38]. They concluded that the FPGA-based system consumed 2.4 less power that the commercial DSP processor.

4.4 | Cost

The cost of the implemented experimental systems is greatly varied. For instance, the Raspberry Pi SoC systems have the least cost with only $35 [39]. Numerous studies classified the FPGA chips as a low-cost implementation method for their DSP applications [40, 41]. However, the contrast estimator systems in scenarios 3 and 4 were implemented by the Altera cyclone V SoC development kit. The latter development kit costs $1795 from official organisations [42]. Regarding MATLAB implementation, since it is essential to use the software and the hardware permanently, the cost will be inflated. However, MATLAB's perpetual licence costs $2350 from the official websites [43]. Moreover, the cost of a normal PC will be added to the cost of MATLAB implementation. In conclusion, the best ranking of the experimental scenarios based on the cost is Raspberry Pi-3 system, Cyclone V (scenarios 3 and 4), and MATLAB implementation.

The ARM Cortex-A53 microcontroller (used in SoC system of scenario 1) has advanced capabilities more than the Cortex-A9 microcontroller (used in scenario 3). For instance, the quadcore 64-bit data bus in A53 MC versus the dual-core 32-bit data bus in A9 MC [44, 45]. However, the results showed that ARM A9 MC achieved a contrast estimation process with less latency and higher throughput compared to A53 MC. To investigate the behaviour of A53 MC in the SoC system, the CPU utilisation was monitored during the programme execution. It was found that the Linux operation system allocated only one CPU to execute the Python programme, while other cores were employed to run other tasks of the SoC system. On the other hand, all available resources of ARM A9 MC were dedicated to handle contrast computation. In contrast, former studies emphasised that the Raspberry Pi SoC system is not the most efficient choice to achieve mathematical tasks [26].

In digital circuits, latency and power dissipation are directly proportional to the number of clock signals applied to the gate of CMOS transistors as cited in former studies [46, 47]. To calculate the number of clock cycles consumed in the presented experimental systems, the clock frequency (F_{CLK}) and the latency should be known. Then,

$$\text{Clock consumption} = \text{Latency} * F_{\mathrm{CLK}} \tag{9}$$

Relying on Equation (9) and the given experimental results, the contrast computation consumed $691.2 * 10^9$, $6.56 * 10^9$, $418 * 10^6$, and $11.4 * 10^6$ clock cycles in scenarios 1–4, respectively. In other words, the more clock consumption for a particular function the more the latency and power dissipation. Therefore, there was a decline in the performance of the system in the case of SoC and MATLAB, and ARM A9 MC

implementation. An essential difference is recognised between FPGA implementation and other methods in the clock consumption tactic. Experimental scenarios 1–3 employed a microcontroller or a microprocessor to achieve the contrast computation action. The mathematical process was interpreted into instructions. The CPU requires one or more clock cycles to complete one instruction. As a result, the number of clock cycles will be inflated. In contrast, FPGA implementation utilised the available configurable logic blocks on the chip to achieve the contrast estimator circuit. The $11.4 * 10^6$ clock cycles that were consumed by the FPGA circuit were required to accomplish the summation loops in Equations (1) and (2). This led to a minimal number of clock cycle utilisations and the least latency and power consumption [48].

The configuration capability in an FPGA chip architecture allows us to apply the preferred designs on it in Ref. [49]. In addition, once an FPGA chip is configured, the chip acquires the features of hardware implementation. In other words, the latency of an implemented system is determined by the delay of the input/output paths in FPGA chips [50], typically in nanoseconds or less. In contrast, the latency of software implementation methods is determined by the number of clocks required to execute the implemented algorithm, such as in microcontrollers, PCs, and SoC systems; typically a high delay compared to hardware implementation [26]. Therefore, FPGA can serve the implementation of image processing tasks more than other methods as emphasised in the literature. Mo, Z. et al. [51] implemented an odour recognition algorithm that mimics human nose. They used the Raspberry Pi 3 SoC system, PCs with different kinds of processors, and the FPGA technique. The SoC system consumed 4.6 ms to execute the algorithm. For PC implementation, the best case was 2.1 ms using i7-10875H processor. However, the FPGA system showed an exceptional performance with 57.3 µs. According to the literature, FPGA implementation is the most efficient method when it is used in complex computation algorithms. The deviation in the performance between FPGA and other implementation methods in the literature is comparable to the result of the present study. The latency of the proposed system was declined from 451.69 s in the SoC system to only 0.228 s in the FPGA system.

A hardware acceleration technique can be accomplished in FPGA chips to achieve more reduction on the latency. Designers use the available hardware resources on the chip to build a circuit in parallel paths. The parallel designs are used in DSP applications to increase the computation efficiency [52]. This technique can be applied to the presented FPGA circuit of scenario 4. The circuit determines the contrast for one image and only 6% of the area of the chip was utilised. Hence, the circuit can be repeated 16 times in parallel; each circuit estimates the contrast of one image. The RGB data and timing signals will be applied to all circuits simultaneously. Thus, the overall latency will not change (0.228 s) while the throughput will be multiplied to 16 times (1.75 Gb/sec).

Further parallelism can be done on the presented FPGA circuit. The two summations in Equations (1) and (2) caused the effective percentage of the latency in the circuit. If the summation ($\sum_{i=0}^{M*N-1} I_i$) is executed in N parallel branches, then the N accumulator and N divider will be needed. Consequently, the latency of the summation will be dropped N times. Similarly, the latency of the other summation can be reduced and mitigate the overall delay. However, the hardware resources of the utilised Altera Cyclone V FPGA chip (41,910 logic block) may not be adequate to implement this action. Alternatively, an FPGA chip with bigger hardware resources should be adopted to accommodate the additional components.

To sum up, Figure 9 summarises the behaviour of the proposed system in this work. The Raspberry Pi-3 SoC system showed a poor calculation efficiency with high latency and power dissipation. However, if it can achieve the quantification algorithm within the repetition time (T_r), then it can be chosen due to the low cost. The MATLAB and Cortex-A9 MC implementation methods showed a closed performance with a few seconds latency. Cyclone V FPGA chip implementation had an exceptional behaviour with the least latency and highest throughput. Moreover, the proposed vision quantification algorithm can be achieved with the least repetition time ($T_r = 9.14$ sec) with the FPGA design.

Among the presented systems in this work, FPGA showed the best performance in terms of latency, throughput, and power dissipation. Furthermore, it meets the requirement of quantification of vision through PDLC-windows. However, the performance of the proposed systems will be declined if the input data size is extended. Currently, the processed data is 8-bit colour images with a resolution of (1080*960). When it is required to increase an image size (for more contrast accuracy, for example 1920*1080 and 16-bit colour images), additional computational complexity will be necessary. Similarly, the system should handle further mathematical operations if the number of images (to sketch the contrast curve) is increased.

A moving cloud can shade the building within few seconds. In this case, the PDLC visibility is not a match with the new condition unless the proposed system responds in real-time. The system reaction towards the outdoor change is by amending the PDLC excitation via the vision quantification algorithm. Therefore, the repetition time T_r should be as shortest as possible to cover the outdoor illumination condition instantaneously. If the presented FPGA vision quantification system is applied, then the proposed algorithm will be repeated in $T_r = 9.14$ sec.

If an unexpected object moves in front of the glazing while in the image-capturing step of the quantification process, it will lead to an unfamiliar form of the contrast curve. As a result, the outcome of quantification operation will be impacted effectively. This drawback can be averted by adding an inspection stage in the quantification process. An inspector will be necessary to test contrast values by comparing them with the standard pattern of the contrast curve. If the difference is greater than a threshold level, then the quantification procedures should be restarted. However, the inspection stage can

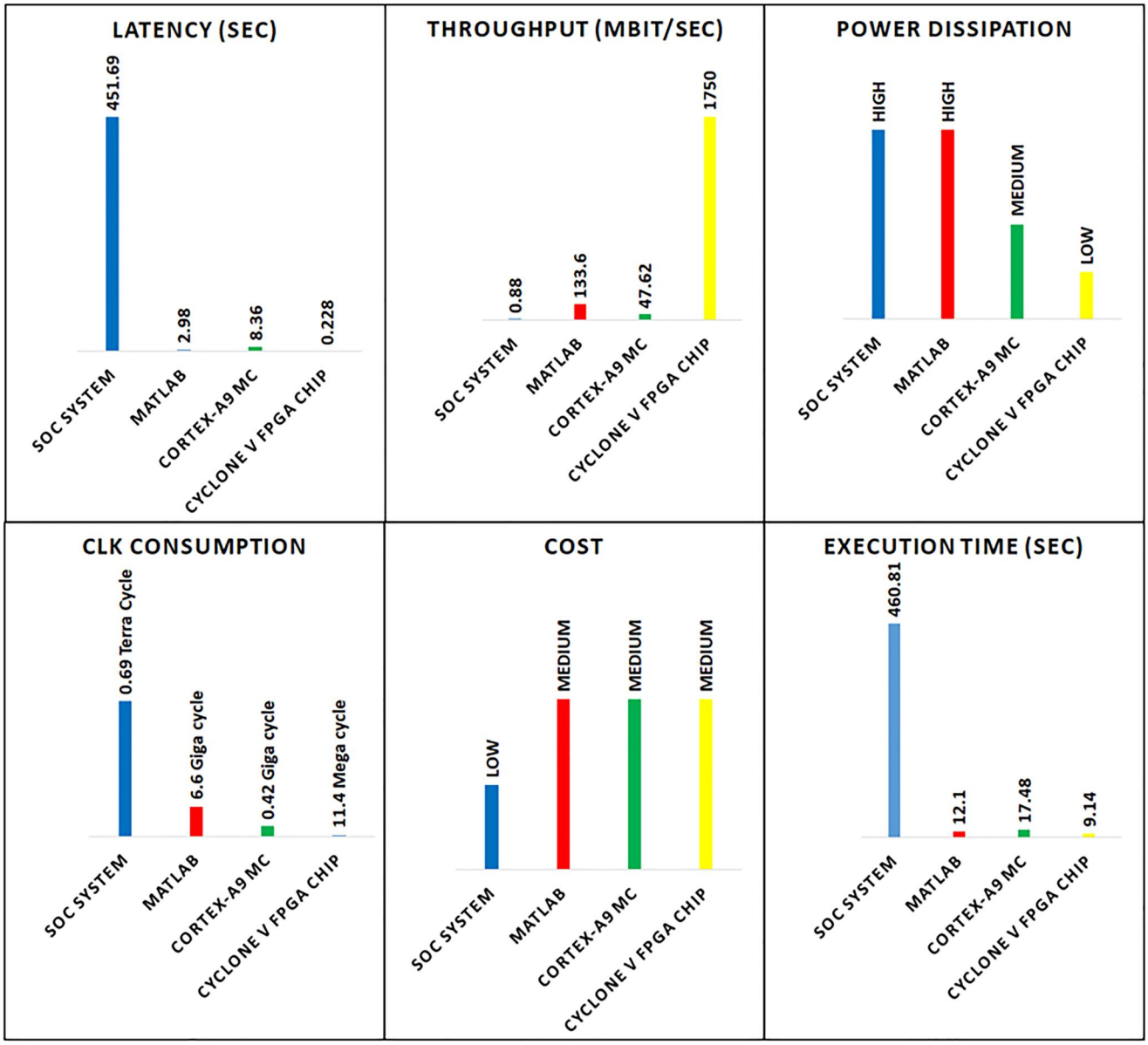

FIGURE 9 Result summary

add more latency to the system and reduce the overall computational efficiency.

If one of the proposed systems is employed in a commercial building, the visible transparency of the glazing will be changed from a transparent to an opaque condition every time the quantification algorithm is being executed, that is, within a time of (T_r). However, this will create an uncomfortable visual experience for the occupants, especially if the repetition time is short. To overcome this situation, the quantification algorithm can be run on a test window positioned in a non-occupied area in the building. The decision of the controller will be publicised to the building's glazing.

Since the proposed systems quantify the vision through PDLC glazing in terms of visible transparency, it is possible to quantify the vision through other types of glazing. Therefore, the proposed systems are applicable for smart windows that offer changeable optical properties [53, 54]. This provides a real-time assessment of vision quality for a better visual linking between the occupants and external views.

5 | CONCLUSION

To conclude, this study introduced different implementation methods for a visible transparency controller for PDLC windows in commercial buildings. The automated controller executes a formerly proposed vision quantification algorithm in real-time. A set of images were taken for a view through a PDLC window by the controller. The main objective of the realised system is to analyse the vision contrast of the images and estimate the suitable voltage for the PDLC film. The voltage level determines the visibility through the PDLC

window based on user preferences. In addition, the automated system determines the minimum level of glazing visibility that sustains a visual comfort towards the external scenes. The main outcome of this work can be illustrated in the following:

1. The study emphasised that the processing cost and power consumption of the Raspberry Pi-3 SoC system is high, and it may not be eligible to handle the proposed quantification algorithm.
2. MATLAB and ARM Cortex-A9 MC implementations showed a closed contrast computation efficiency of 2.98 and 8.36 s, respectively. However, the power and clock cycle consumption are relatively high.
3. ARM Cyclone V FPGA-chip implementation is the most efficient system with a latency of 0.228 s and low power consumption.
4. The minimum repetition time for the proposed quantification algorithm was achieved by the ARM Cyclone V FPGA system of $T_r = 9.14$ sec.

In commercial buildings, using the implemented automated system determines the comfortable visible band for smart windows. The proposed system is applicable not only to PDLC windows but also can be used with other kinds of smart glazing whenever there is a need to assess vision quality through the glazing. In addition to earlier research on smart windows, the real-time observability and controllability of the proposed system optimise the visual comfort in internal environments of commercial buildings. In future work, a complete FPGA implementation will be considered. The digital camera and PDLC power driver systems can be interfaced with the FPGA chip. In addition, the delay of the camera and PDLC driver is high (8.9 s) compared to contrast computation by the FPGA chip (0.228 s); therefore, if alternative high response devices are used, the overall system latency will be mitigated. Moving scenes through the PDLC-window will be considered in the next step. A peripheral system will be created to detect the instantaneous changes in the views. The latter system will analyse the correlation between repetitive images to insure a standard form of contrast curve.

AUTHOR CONTRIBUTIONS

Mohammed Lami: Conceptualisation; Data curation, Formal analysis; Investigation; Methodology; Resources; Validation; Writing – original draft. **Faris Al-naemi:** Conceptualisation; Resources; Supervision; Writing – review & editing. **Walid Issa:** Conceptualisation; Supervision; Writing – review & editing.

ACKNOWLEDGEMENTS

For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) license to any author-accepted manuscript version arising from this submission.

CONFLICT OF INTEREST

The authors have no competing interests to declare that are relevant to the content of this article. The authors declare that they have no conflict of interest.

DATA AVAILABILITY STATEMENT

The paper has no associated data.

ORCID

Mohammed Lami https://orcid.org/0000-0002-9319-9068

REFERENCES

1. Au, B.W.C., Chan, K.Y.: Towards an all-solid-state electrochromic device: a review of solid-state electrolytes and the way forward. Polymers. 14(12), 2458 (2022). MDPI. https://doi.org/10.3390/polym14122458
2. Wu, J., et al.: Coupled optical-electrical-thermal analysis of a semi-transparent photovoltaic glazing façade under building shadow. Appl. Energy. 292, 2021 (2021). https://doi.org/10.1016/j.apenergy.2021.116884
3. Wang, C., et al.: The study of a double-skin ventilated window integrated with CdTe cells in a rural building. Energy. 215, 119043 (2021). https://doi.org/10.1016/j.energy.2020.119043
4. Kim, J.H., Han, S.H.: Indoor daylight performances of optimized transmittances with electrochromic-applied kinetic louvers. Buildings. 12(3), 263 (2022). https://doi.org/10.3390/buildings12030263
5. Basher, M.K., Alam, M.N.E., Alameh, K.: Design, development, and characterization of low distortion advanced semitransparent photovoltaic glass for buildings applications. Energies. 14(13), 3929 (2021). https://doi.org/10.3390/en14133929
6. Guo, J., Zhang, C.: Utilization of window system as exhaust air heat recovery device and its energy performance evaluation: a comparative study. Energies. 15(9), 3116 (2022). https://doi.org/10.3390/en15093116
7. Lami, M., et al.: Optimisation of wasted air utilisation in thermal loss reduction in double-glazed windows of commercial buildings in cold regions. Int. J. Energy Environ. Eng. 13 (2022). https://doi.org/10.1007/s40095-022-00499-0
8. Kim, J.H., Hong, J., Han, S.H.: Optimized physical properties of electrochromic smart windows to reduce cooling and heating loads of office buildings. Sustainability. 13(4), 1–30 (2021). https://doi.org/10.3390/su13041815
9. Olivieri, L., et al.: Integral energy performance characterization of semi-transparent photovoltaic elements for building integration under real operation conditions. Energy Build. 68, 280–291 (2014). PARTA. https://doi.org/10.1016/j.enbuild.2013.09.035
10. Ismail, K.A.R., et al.: Experimental investigation on ventilated window with reflective film and development of correlations. Sol. Energy. 230, 421–434 (2021). https://doi.org/10.1016/j.solener.2021.10.061
11. Pereira, R., et al.: Ranking programming languages by energy efficiency. Sci. Comput. Program. 205, 102609 (2021). https://doi.org/10.1016/j.scico.2021.102609
12. MATLAB. 9.7.0.1190202 (R2020a); The MathWorks Inc., Natick, (2020)
13. Dabbagh, M., Krarti, M.: Optimal control strategies for switchable transparent insulation systems applied to smart windows for us residential buildings. Energies. 14(10), 2917 (2021). https://doi.org/10.3390/en14102917
14. Ke, W., et al.: Comparative analysis on the electrical and thermal performance of two CdTe multi-layer ventilated windows with and without a middle PCM layer: a preliminary numerical study. Renew. Energy. 189, 1306–1323 (2022). https://doi.org/10.1016/j.renene.2022.03.090
15. Wang, C., et al.: Design and performance investigation of a novel double-skin ventilated window integrated with air-purifying blind. Energy. 254, 124476 (2022). https://doi.org/10.1016/j.energy.2022.124476
16. Krishnamoorthy, R., Krishnan, K., Bharatiraja, C.: Deployment of IoT for smart home application and embedded real-time control system. Mater. Today Proc. 45, 2777–2783 (2021). https://doi.org/10.1016/j.matpr.2020.11.741
17. Dobrojevic, M., Bacanin, N.: IoT as a backbone of intelligent homestead automation. Electronics. 11(7), 1004 (2022). MDPI. https://doi.org/10.3390/electronics11071004
18. Python Software Foundation (2018). Python. Retrieved from. https://www.python.org/

19. Elkholy, M.H., et al.: Smart centralized energy management system for autonomous microgrid using FPGA. Appl. Energy. 317, 119164 (2022). https://doi.org/10.1016/j.apenergy.2022.119164
20. Kumar, A., et al.: Design, analysis and implementation of electronically interfaced photovoltaic system using ARM Cortex-M4 microcontroller. Comput. Electr. Eng. 98, 107701 (2022). https://doi.org/10.1016/j.compeleceng.2022.107701
21. Liu, J., Feng, J.: Design of embedded digital image processing system based on ZYNQ. Microprocess. Microsyst. 83, 104005 (2021). https://doi.org/10.1016/j.micpro.2021.104005
22. Jaskolka, K., et al.: A Python-based laboratory course for image and video signal processing on embedded systems. Heliyon. 5(10), e02560 (2019). https://doi.org/10.1016/j.heliyon.2019.e02560
23. Pereira, F.C., Pereira, C.E.: Embedded image processing systems for automatic recognition of cracks using UAVs. IFAC-PapersOnLine. 28(10), 16–21 (2015). https://doi.org/10.1016/j.ifacol.2015.08.101
24. Wang, X.: Image recognition of English vocabulary translation based on FPGA high-performance algorithm. Microprocess. Microsyst. 80, 103542 (2021). https://doi.org/10.1016/j.micpro.2020.103542
25. Ahmad, A.H.S., Riazuddin, A., Khan, S.A.: An FPGA cost estimation technique for design space exploration (DSE). In: 17th IEEE International Multi Topic Conference: Collaborative and Sustainable Development of Technologies, pp. 378–382. IEEE INMIC 2014 - Proceedings (2015). https://doi.org/10.1109/INMIC.2014.7097369
26. Paunski, Y.K., Angelov, G.T.: Performance and power consumption analysis of low-cost single board computers in educational robotics. IFAC-PapersOnLine. 52(25), 424–428 (2019). https://doi.org/10.1016/j.ifacol.2019.12.575
27. Lami, M., et al.: Quantifying of vision through polymer dispersed liquid crystal double-glazed window. Energies. 15(9), 3196 (2022). https://doi.org/10.3390/en15093196
28. Ariateja, D., Ardiyanto, I., Soesanti, I.: A review of contrast enhancement techniques in digital image processing. In: 2018 4th International Conference on Science and Technology (ICST), pp. 1–6 (2018). https://doi.org/10.1109/ICSTC.2018.8528579
29. Beghdadi, A., et al.: A critical analysis on perceptual contrast and its use in visual information analysis and processing. IEEE Access. 8, 156929—156953 (2020). https://doi.org/10.1109/ACCESS.2020.3019350
30. Ionescu, C., Fosalau, C., Petrisor, D.: A study of changes in image contrast with various algorithms. In: EPE 2014 - Proceedings of the 2014 International Conference and Exposition on Electrical and Power Engineering, pp. 100–104 (2014). https://doi.org/10.1109/ICEPE.2014.6969876
31. Raspberry Pi Foundation.: Raspberry Pi - Teach, Learn, and Make with Raspberry Pi (2018). Retrieved from https://www.raspberrypi.org/
32. Strohmer, B., et al.: ROS-enabled hardware framework for experimental robotics. In: 2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig), pp. 1–2 (2019). https://doi.org/10.1109/ReConFig48160.2019.8994770
33. https://www.intel.com/content/www/us/en/products/details/fpga/development-tools/quartus-prime/resource.html
34. Harris, S.L., Harris, D.: 4 - hardware description languages. In: Harris, S. L., Harris, D. (eds.) Digital Design and Computer Architecture, pp. 170–235. Morgan Kaufmann (2022). https://doi.org/10.1016/B978-0-12-820064-3.00004-0
35. Pozhidaev, E.P., et al.: Polymer dispersed liquid crystals with electrically controlled light scattering in the visible and near-infrared ranges. Opt. Mater. Express. 10(12), 3030 (2020). https://doi.org/10.1364/ome.410163
36. Bekaroo, G., Santokhee, A.: Power consumption of the Raspberry Pi: a comparative analysis. In: 2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies, pp. 361–366. EmergiTech (2016). https://doi.org/10.1109/EmergiTech.2016.7737367
37. Bai, Y., et al.: FPGA vs DSP: a throughput and power efficiency comparison for Hierarchical Enumerative Coding. In: IEEE/IFIP International Conference on VLSI and System-On-Chip, pp. 318–321. VLSI-SoC (2013). https://doi.org/10.1109/VLSI-SoC.2013.6673300
38. Texas Instruments Inc.: TMS320C6A816x C6-Integra DSP+ARM Processors. Literature number: SPRS680B, (2011)
39. Bolaños, F., LeDue, J.M., Murphy, T.H.: Cost effective raspberry pi-based radio frequency identification tagging of mice suitable for automated in vivo imaging. J. Neurosci. Methods. 276, 79–83 (2017). https://doi.org/10.1016/j.jneumeth.2016.11.011
40. Mostafa Saad, E., et al.: FPGA-Based Implementation of a Low Cost and Area Real-Time Motion Detection Redundant Independent Files (RIF): A Technique for Reducing Storage and Resources in Big Data Replication View Project Computer Aided Diagnosis View Project FPGA-BASED IMPLEMENTATION of A LOW COST and AREA REAL-TIME MOTION DETECTION (2008). [Online]. https://www.researchgate.net/publication/224326286
41. Wirtz, S.F., et al.: Development of a low-cost FPGA-based measurement system for real-time processing of acoustic emission data: proof of concept using control of pulsed laser ablation in liquids. Sensors. 18(6), 1775 (2018). https://doi.org/10.3390/s18061775
42. https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English%26CategoryNo=167%26No=819
43. https://www.mathworks.com/pricing-licensing.html
44. Wanza Weloli, J., et al.: Efficiency modeling and exploration of 64-bit ARM compute nodes for exascale. Microprocess. Microsyst. 53, 68–80 (2017). https://doi.org/10.1016/j.micpro.2017.06.019
45. Aviles, P.M., et al.: Evaluating reliability through soft error triggered exceptions at ARM Cortex-A9 microprocessor. Microelectron. Reliab. 126, 114323 (2021). https://doi.org/10.1016/j.microrel.2021.114323
46. Lee, C.Y., Lin, T.Y., Chang, R.G.: Power-aware code scheduling assisted with power gating and DVS. Future Generat. Comput. Syst. 34, 66–75 (2014). https://doi.org/10.1016/j.future.2013.12.011
47. Nouri, S., Rossi, D., Nurmi, J.: Power mitigation of a heterogeneous multicore architecture on FPGA/ASIC by DFS/DVFS techniques. Microprocess. Microsyst. 63, 259–268 (2018). https://doi.org/10.1016/j.micpro.2018.09.010
48. HajiRassouliha, A., et al.: Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms. Signal Process. Image Commun. 68, 101–119 (2018). https://doi.org/10.1016/j.image.2018.07.007
49. Pyrgas, L., Kitsos, P., Skodras, A.: Compact FPGA architectures for the two-band fast discrete Hartley transform. Microprocess. Microsyst. 61, 117–125 (2018). https://doi.org/10.1016/j.micpro.2018.06.002
50. Prakash, A., et al.: FPGA-aware techniques for rapid generation of profitable custom instructions. Microprocess. Microsyst. 37(3), 259–269 (2013). https://doi.org/10.1016/j.micpro.2013.02.002
51. Mo, Z., et al.: FPGA implementation for odor identification with depthwise separable convolutional neural network. Sensors. 21(3), 1–19 (2021). https://doi.org/10.3390/s21030832
52. Unnikrishnan, K.S., Madhavan, S.: Parallel computation using DSP slices in FPGA. Procedia Technol. 24, 1127–1134 (2016). https://doi.org/10.1016/j.protcy.2016.05.064
53. Oh, M., Tae, S., Hwang, S.: Analysis of heating and cooling loads of electrochromic glazing in high-rise residential buildings in South Korea. Sustainability. 10(4), 1121 (2018). https://doi.org/10.3390/su10041121
54. Shchegolkov, A.V., et al.: A brief overview of electrochromic materials and related devices: a nanostructured materials perspective. Nanomaterials. 11(9), 2376 (2021). MDPI. https://doi.org/10.3390/nano11092376

How to cite this article: Lami, M., Al-naemi, F., Issa, W.: Automated quantification system for vision through polymer-dispersed liquid crystal double-glazed windows: Circuit implementation. IET Circuits Devices Syst. 1–15 (2022). https://doi.org/10.1049/cds2.12135