Image synchronization for 3D application using the NanEye sensor

Ricardo M. Sousa[a,b], Martin Wäny[b], Pedro Santos[b], Morgado-Dias [a,c]
[a] University of Madeira, Rua dos Ferreiros 9000-082, Funchal, Portugal
[b]Awaiba Lda, Madeira Tecnopolo 9020-105, Funchal, Portugal
[c] Madeira Interactive Technologies Institute, Madeira Tecnopolo 9020-105, Funchal, Portugal

Further author information:
Ricardo M. Sousa: E-mail: ricardoandresousa@hotmail.com
Martin Wäny: E-mail: waeny@awaiba.com

ABSTRACT

Based on Awaiba's NanEye CMOS image sensor family and a FPGA platform with USB3 interface, the aim of this paper is to demonstrate a novel technique to perfectly synchronize up to 8 individual self-timed cameras. Minimal form factor self-timed camera modules of 1 mm x 1 mm or smaller do not generally allow external synchronization. However, for stereo vision or 3D reconstruction with multiple cameras as well as for applications requiring pulsed illumination it is required to synchronize multiple cameras. In this work, the challenge to synchronize multiple self-timed cameras with only 4 wire interface has been solved by adaptively regulating the power supply for each of the cameras to synchronize their frame rate and frame phase. To that effect, a control core was created to constantly monitor the operating frequency of each camera by measuring the line period in each frame based on a well-defined sampling signal. The frequency is adjusted by varying the voltage level applied to the sensor based on the error between the measured line period and the desired line period. To ensure phase synchronization between frames of multiple cameras, a Master-Slave interface was implemented. A single camera is defined as the Master entity, with its operating frequency being controlled directly through a PC based interface. The remaining cameras are setup in Slave mode and are interfaced directly with the Master camera control module. This enables the remaining cameras to monitor its line and frame period and adjust their own to achieve phase and frequency synchronization. The result of this work will allow the realization of smaller than 3mm diameter 3D stereo vision equipment in medical endoscopic context, such as endoscopic surgical robotic or micro invasive surgery.

Keywords: Multi-Camera Synchronization, Control and Regulation Theory, Stereo Vision, CMOS Image Sensors, FPGA, High-Speed USB3 Interface

1. INTRODUCTION

The simultaneous synchronization of video signals from multiple cameras is a necessity for 3D image sensor based applications. To ensure that the output of these applications is correct, the existence of synchronization between the video sequences is essential. The absence of this condition may imply frame desynchronization or frame loss, which can lead to distortions in the video sequences displayed to the user [1]. As a consequence, it is not possible to combine the different frames without using an external memory to store the set of video frames [1], [2]. If the video streams are synchronized, a 3D reconstruction algorithm can merge the set of frames into a single image, and interpret the differences to determine depth. The ability to perceive the depth is what allows the system user to see where objects are in relation to the set of cameras [3].

The increase in the number cameras in a video capture system entails a large volume of video data available for processing. The simultaneous operation of 8 image sensors, transmitting approximately 40 frames/s, implies that a software control application would have to simultaneously process 320 frames every second and perform frequency and phase control on each of the 8 cameras, entailing a notable computational cost and communication latency [4]–[6]. In addition to that, in order to implement a faster and finer sensor operational frequency control it is necessary to implement a hardware level control. This way the delays and consequently the overall system error are minimized by synthesizing dedicated circuits to process and correct the phase-frequency error, in parallel. A way to implement this control at the level of hardware is by means of a FPGA platform converting computationally intensive functions in RTL (*Register Transfer*

Level) circuits. The configurable logic allows the creation of dedicated control architectures laid out as parallel logic circuits, enabling delay time reduction, faster processing and overall phase-frequency error minimization [7].

The goal proposed in this project was to synchronize the operation of multiple digital cameras (up to 8), allowing, among other applications, the realization of a smaller than 3 mm stereoscopic 3D vision system designated NanEye Stereo, applicable to medical imaging such as endoscopy, robotic assisted surgery and dental imaging.

2. MULTI-CAMERA SYNCHRONIZATION MECHANISMS

In [1] it is referred that video synchronization can be performed by software or hardware. The following topics explain the advantages and disadvantages of each of these methods and refer to some implementation examples of hardware synchronization.

1.1 Software synchronization

Software synchronization uses visual clues or visible characteristics, such as lighting changes, visible tags on moving objects, or object trajectories, to perform video sequence synchronization. This has the advantage of enabling video synchronization without dedicated hardware and without the need of having cameras physically close to one another by opening the possibility of using a network of cameras based on wireless technology [8]. On the other hand, software synchronization requires a significant amount of memory and post processing, in addition the algorithms used must be adapted to the specific application of the camera network [1], [2].

1.2 Hardware synchronization

Hardware synchronization uses mostly wired control and data interfaces to connect to each camera. In externally clocked cameras the control can be performed by timing the trigger pulse and varying the external clock frequency fed to the camera module. Other configurations require the use of memory to store the video data and adjust the deviations.

The advantages of using hardware synchronization are that the synchronization error is exactly known and that the amount of memory and post processing can be minimized or completely eliminated. On the other hand, these advantages are offset by the need for dedicated and possibly expensive hardware. In addition to this, the cameras should be able to receive and recognize the synchronization signals. This implies the use of wires between the control module and each one of the cameras (or between cameras), which is not viable for wireless applications or when the distance between cameras is too great [1], [8].

In [9] an FPGA based real-time processing of stereoscopic high definition video is demonstrated. On this application, the FPGA sends a trigger signal for each camera, adjusting the frame rate in order to synchronize the video streams. However, the pixels corresponding to each camera are not necessarily received at the same time, due, in great part, to small differences in phase between the two pixel clocks. To synchronize the two cameras with a pixel level error, it was necessary to temporarily store the video streams in FIFO buffers to compensate for the temporal deviations. The same principle is used in [2]. However, in this case external video decoders are placed between the cameras and the FPGA platform. In this case, it is assumed that the cameras both operate at the same frequency and that all electrical connections have exactly the same dimensions. Each video decoder captures the corresponding video stream and generates its own clock signal based on the camera's horizontal synchronization signal, bringing both streams to the same clock domain. Both streams are linked to FIFOs, in the FPGA, which act as data buffers to compensate for stream misalignment.

Another multi-camera system application is industrial assembly line inspection. The work described in [10] demonstrates the need of a multiple camera system for fault detection in transparent glass tubes using multiple vision angles. Real time visual inspection systems usually consists of the use of cameras operated with the same common pulse and controlled by a centralized system. The architecture described in [10] follows the same principle using a trigger signal to drive the capture, preprocessing, and video data transmission process, in each camera. The particularity on this system is that all data is sent to a main camera, which then forwards it to a centralized computer system.

Another approach to this problem was used in [11]. This involved sending a synchronization signal to each camera to start the image capture process. However, the innovation on this system was to include a timestamp in each captured frame with a unique code that translated the exact moment in which the frame was captured. The video data was stored in external memory and after post-processing, the timestamps on each frame were recovered, allowing the frame grabber to collect and align the frames, ensuring synchronization. Other examples of synchronization by hardware are listed in [12] and [13].

However, all these implementations have in common the fact that they use synchronization signals (triggers) and/or memories to align the captured frames. Making these inviable solutions taking into account the target hardware constraints. Firstly the NanEye image sensor, described in [14], is a self-timed camera, meaning the pixel clock signal is generated internally on the sensor, through a voltage controlled ring oscillator (*Voltage Controlled Oscilator* - VCO). Since the NanEye image sensor is self-timed (internal clock) and operates autonomously, there are no trigger or external clock pins. This configuration enables the creation of camera modules with smaller footprints through the reduction of the external pins needed on the camera. On this particular sensor there are only 4 pins: 2 used for power supply, between V_{CC} and GND, and 2 for data communication, through a LVDS semi-duplex interface. The disadvantage is that typically the internally generated clock signal has a notable jitter level, adding to that the fact that the pixel clock frequency has a strong dependency on the ambient temperature due to the use of a ring oscillator [15], [16]. Secondly, despite the FPGA platforms having external memory (SDRAM or Flash) and the instantiation of RAMs and FIFOs internally on the logic is possible, this was not considered a valid option in view of the fact that this project was intended to create a synchronization system that actively modulated the camera behavior.

3. PROPOSED TOPOLOGY

The topology of this proposal is presented in Fig. 1. Its function is to adjust the frequency of operation of the camera based on the line period signal, Line Valid (LVAL), and adjust the phase difference between frames on the basis of the frame period signal, Frame Valid (FVAL). It is composed of a controller block (in the context of this work designated Frequency Comparator), a digitally controlled oscillator (DCO) and a deserialization module. The architecture of the phase-frequency control system based on the line and frame period of an image is an original proposal of this paper. It is motivated by the need to create a multi-camera synchronization system that uses a single fixed reference, in this case, a clock signal with constant frequency, to measure the image sensor frame rate.

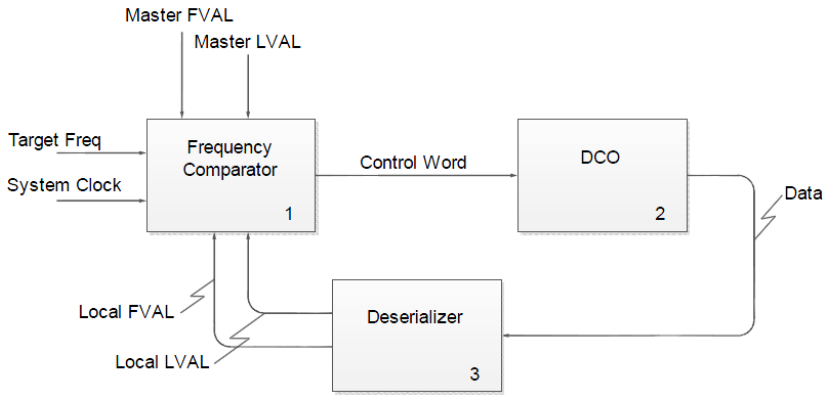


Figure 1 - Block diagram of the frame rate control system.

The Manchester coded data stream from the image sensor is, in a first phase, decoded and then converted from serial to parallel format and outputted to the USB interface. At the output of the Deserializer block the line and frame synchronization signals can be obtained. The Frequency Comparator block is used to compare the LVAL period with a reference period defined externally, which corresponds to a given target frequency (or frame rate). In addition a comparison is made with an external FVAL signal to determine the phase error between the local frame and the outer frame from another NanEye camera. The output is a control word used to drive the DCO and modulate the voltage applied to the sensor. Being that the DCO is a DAC coupled to the image sensor that, for the purpose of behavioral analysis, can be modeled as a VCO.

Relating this to the previously introduced state-of-the-art [2], [9]–[13], this architecture is distinguished by performing an active synchronization control by increasing or decreasing the frame rate of each sensor until the multiple video streams are synchronous in time. This has the clear advantage of discarding the use of memory elements to store the individual frames. Moreover, this control architecture theoretically renders the system independent from the cable length, a problematic factor in [2].

4. FREQUENCY COMPARATOR

At an early stage the frequency control was only performed on a single camera by adjusting the applied voltage. To that effect only LVAL is analyzed. It is possible to ascertain the line's period by counting the number of rising edge transitions of the system clock signal common to all the logic on the FPGA. This process is illustrated in the diagram of Fig. 2. Since the clock signal has a constant frequency, given by a 48 MHz crystal oscillator, a fixed reference by which to determine the sensor's frequency error can be obtained. The count number obtained through this method can then be compared to the expected for a given frequency. Knowing that in one line 249 Pixel Periods of data plus 3 Pixel Periods at 0 are transmitted, both encoded in 12 bit, the following relationship can be obtained,

$$Target_{Clk} = \frac{(12\ bits \times 252\ PP) \times T_{Target\ Freq}}{T_{System\ Clock}}, \tag{1}$$

being $T_{Target\ Freq}$, the value of the period corresponding to a clock signal with the desired pixel clock frequency, and $T_{System\ Clock}$ the period of the system clock signal.

For a better understanding consider the following example: it is intended that the sensor transmits data at a rate of 44 fps, using a 48 MHz reference clock signal. According to the NanEye 2C specification [14], a transmitted frame contains a total of 250 rows of pixel data plus 3 more lines reserved for sensor configuration, so the period of a line is given by,

$$T_{Line} = \frac{T_{Frame}}{253\ lines}\ (s), \tag{2}$$

replacing the values,

$$T_{Line} = \frac{1/44\ fps}{253\ lines} = 89{,}83\ \mu s. \tag{3}$$

Considering this result and replacing it in the relation previously deducted in (1) is obtained,

$$Target_{Clk} = \frac{T_{Line}}{T_{System\ Clock}} = \frac{89{,}83\ \mu S}{1/48\ MHz} \cong 4312\ clocks, \tag{4}$$

the result being the number of clock transitions that the system will have to detect so that the image sensor transmits 44 frames per second given the conditions specified above. If the count value is above this value this means that the camera is operating at a lower frequency than desired, so the control module should increase the applied voltage to compensate. On the other hand, if the count is below the specified target, it implies that the camera is running at a higher frame rate than intended, conversely the applied voltage should decrease. Thus the system can adjust the sensor operation to any value within its operating range. This allows it to be independent of ambient temperature and cable length.

However, this method alone does not guarantee phase synchronization between multiple cameras. As illustrated in Fig. 3, by connecting multiple NanEyes in parallel and performing frequency control using only the method described above, it was found that although they may be running at the same frequency, there is a phase difference between the frames read from each of the sensors. To overcome this it is then necessary to make an adjustment to the operating frequency of each sensor in order to gradually adjust its phase with a given reference FVAL. In order to achieve this, small adjustments to $\mathrm{Target_{Clk}}$ can be made, by adding or subtracting a default value to this reference, as shown in Fig. 4. Considering one of the sensors as the reference (Master) and a second sensor as dependent (Slave), it is possible to adjust the frame phase between cameras by detecting the instant the local FVAL signal transitions and comparing it with the transition of an FVAL signal from another camera considered as the reference, or Master. If the local FVAL's falling edge transition occurs before the reference signal, this means that the video stream from the local camera is in advance in relation to the latter. Consequently, it is necessary to reduce the sensor frequency by increasing $\mathrm{Target_{Clk}}$ by a suitable amount

(initially and in the case of the example, the adjustment is made in unitary steps). On the other hand, if the local FVAL transition occurs after the Master signal, this implies that the local frame is lagging in relation to Master FVAL. This way, it is necessary to increase the operational frequency by reducing the $\mathrm{Target}_{\mathrm{Clk}}$ to correct the error.

Yet, although this method ensures phase and frequency synchronization between frames, it does not take into account the adjustment speed depending on error magnitude. For a better understanding of the problem consider the following example: the local sensor frame (Slave FVAL) is in advance in relation to the frame obtained from the sensor set as Master by the equivalent of 100 periods of the clock signal. As illustrated in Fig. 5, if the phase correction is made by 1 clock period added to $\mathrm{Target}_{\mathrm{Clk}}$, then it is found that the system takes a considerable amount of time to correct the phase error. For this reason it was necessary to create a mechanism that would allow the control system to make a quick phase adjustment while maintaining stability and low phase-frequency error. For this it was then necessary to monitor the magnitude of the error between the two frames. If the error was found to be above a defined threshold, the adjustment would be faster (i.e.: 30 times the clock signal added or subtracted from the number of target clock cycles, $\mathrm{Target}_{\mathrm{Clk}}$), on the other hand, if the error was smaller than the defined threshold the adjustment would be much smoother to guarantee low oscillation and small error. As illustrated in Fig. 6, to achieve this effect a XOR gate and a counter were used. At the input of the XOR gate were placed the two frame signals, Master FVAL and Slave FVAL. At the output a signal can be obtained whose duty cycle is proportional to the synchronization error between the two frames. Again using a counter it is possible to effectively measure the error's amplitude by detecting the rising edge transitions of a clock signal within the High period of the FVAL XOR signal.

Setting a threshold value between high and small error, in practice a correction system using fixed discrete correction steps can be implement. Experimentally, the amplitude of the correction and the limit of transition between the higher correction step and the smoother correction step, were defined. In the example given in Fig. 6, the fast step is 30 if the "XOR width" is greater than the set limit (in this case one and a half times the low period of the Master FVAL signal), and the slow step is 1 if lesser than the limit. The value 30, used as the quick step, was determined empirically as the optimal value that allowed the fastest phase adjustment, in the presence of a large error, without destabilizing the system. In practice, the use of these two levels of adjustment proved to be sufficient to ensure the speed required for the application in question (the synchronism is achieved within 2 s). However, if the requirements of the application changed the inclusion of more adjustment levels to decrease the correction time, could be justified. This process and all the control method described in this section is illustrated in the flowchart of Fig. 7.

By assigning one of the cameras the role of Master, implies that its frame rate is directly controlled by the graphical user interface through command reception from a PC based USB3 interface. The remaining cameras connected to the module are configured as Slaves, and are not directly controlled from the interface, instead receive control signals (Master FVAL and Master LVAL) from the Master camera. The frame signal is used in the phase synchronization process in accordance to the method described above. The line signal is used by the cameras in Slave mode to measure the operational frequency of the Master camera, by determining the Master LVAL's period. On the basis of the result the applied voltage to the local sensor is set to synchronize its frequency with the Master camera.

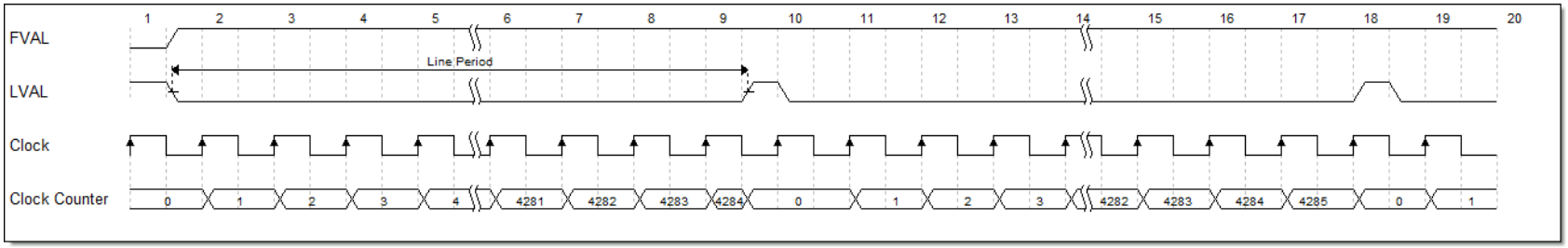


Figure 2 – Time diagram of the line period count.

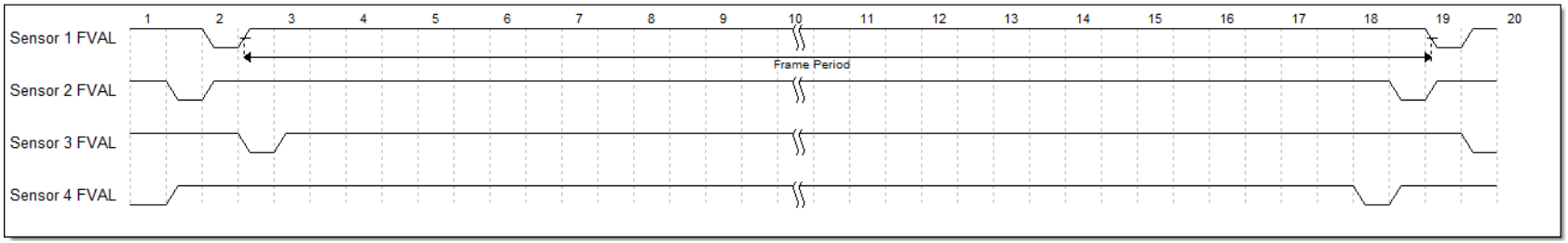


Figure 3 – Time diagram of the phase difference between frames.

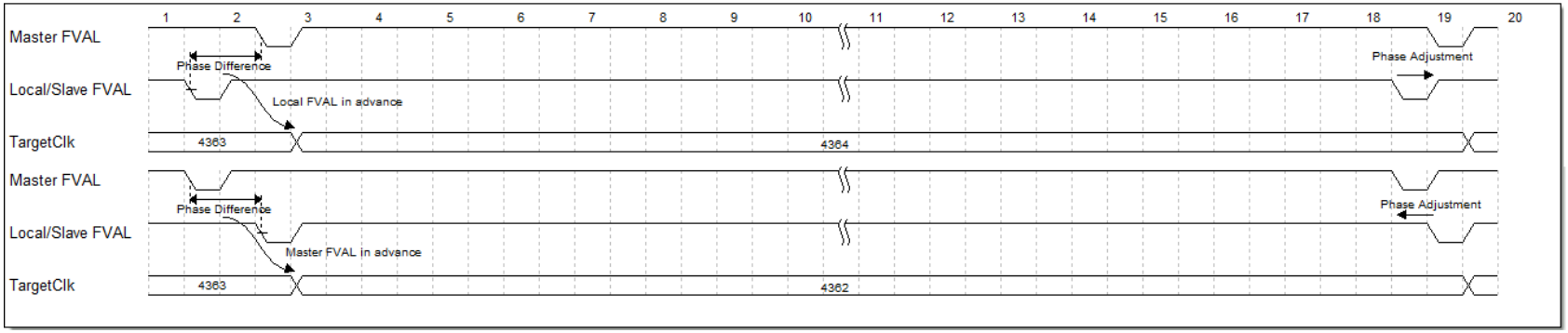


Figure 4 – Time diagram of the frames phase adjustment.

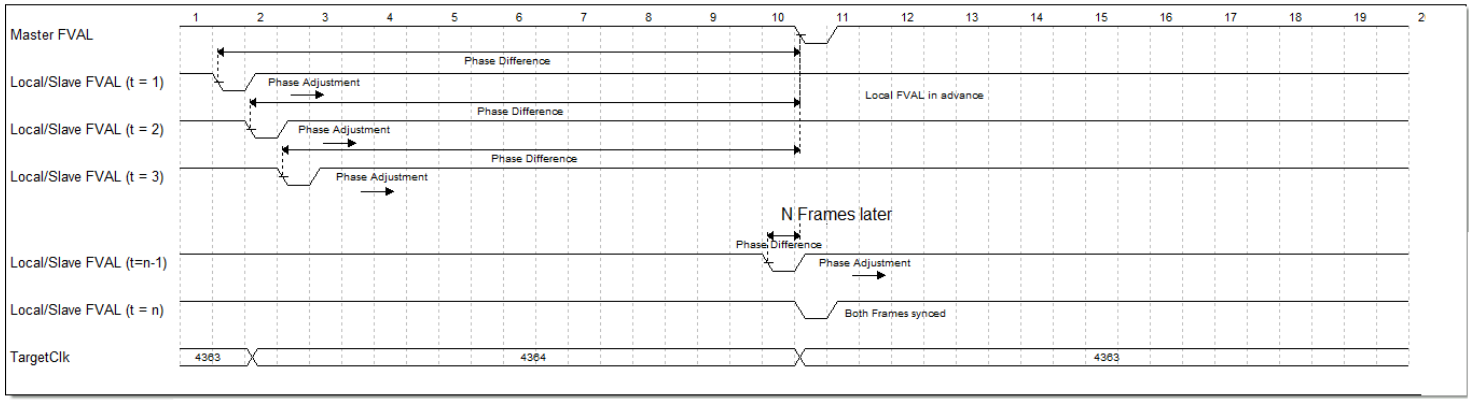


Figure 5 – Slow temporal phase adjustment between frames.

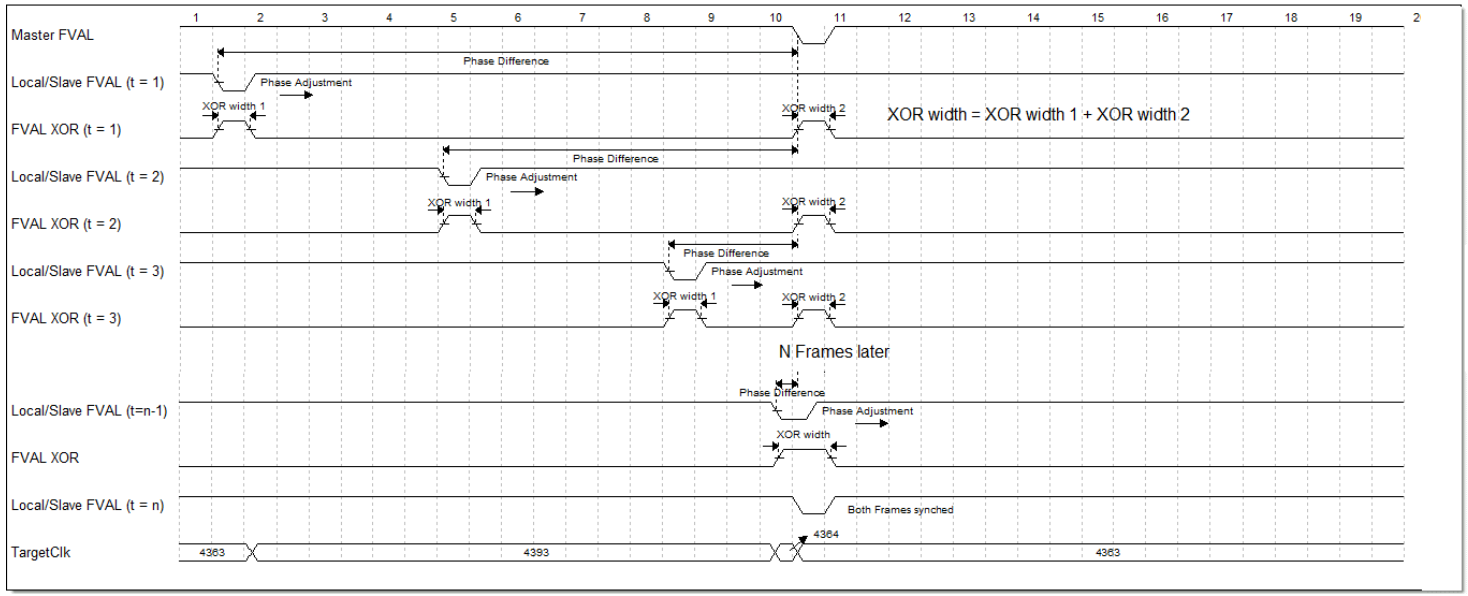


Figure 6 - Fast temporal phase adjustment between frames.

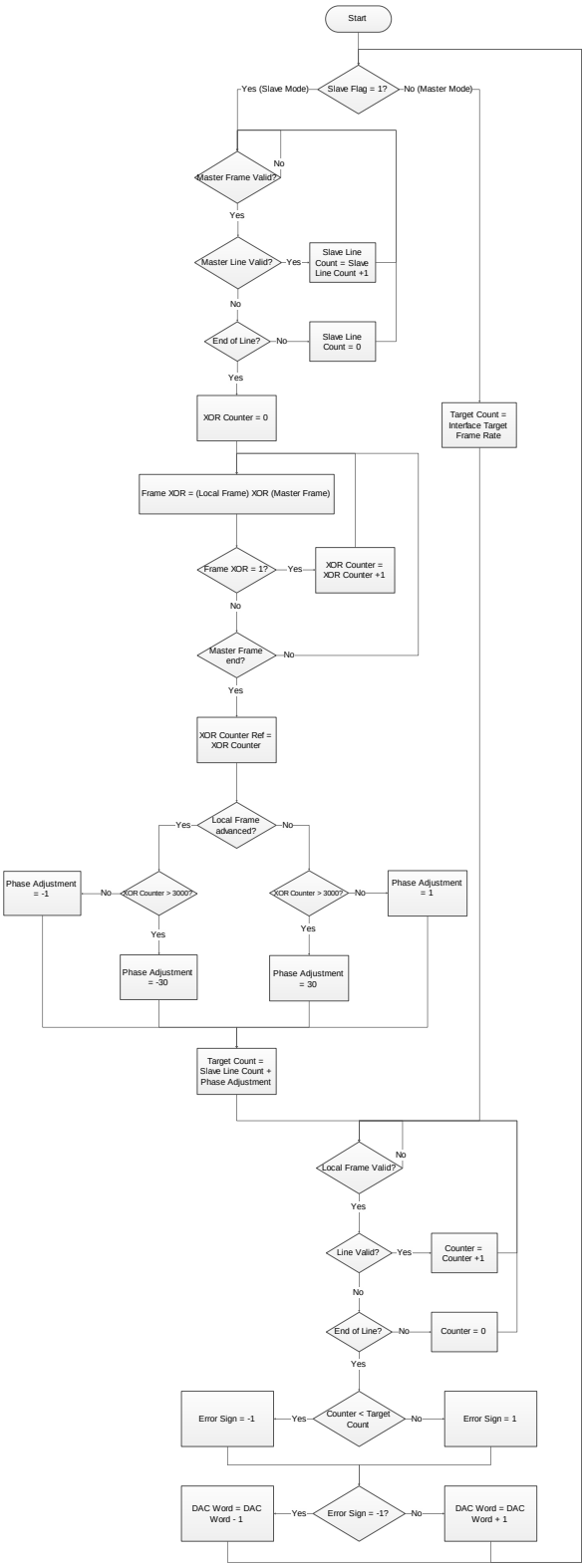


Figure 7 – Flowchart of the Frequency Comparator control module.

5. PRACTICAL IMPLEMENTATION

A Frequency Comparator control module is generated for each of the sensors connected to the FPGA. One of the sensors is defined as the Master, by setting its Slave Flag input as '0', the other instantiations of the control module receive the Slave Flag = '1'. The integration of the Frequency Comparator with the remaining logic follows the general structure shown in Fig. 8. The sensors B and C adjust their frame rate to follow the operational frequency imposed by sensor A. The latter adjusts its frame rate in accordance with the target word frequency received from the graphical user interface.

The decoding and control architecture of each sensor is composed of: a Frequency Comparator, with the task of modulating the voltage to be applied to the sensor on the basis of the target frequency received from the graphical interface (Master mode), or on the basis of the reference signals Master FVAL and Master LVAL (Slave mode); a DCO, composed of a digitally controlled ADC and the sensor itself represented by a VCO, a Deserializer responsible for decoding the stream of data from the sensor the vsync and hsync signals; and a block called Image Out, which makes the adaptation of pixel data for transmission on the USB interface.

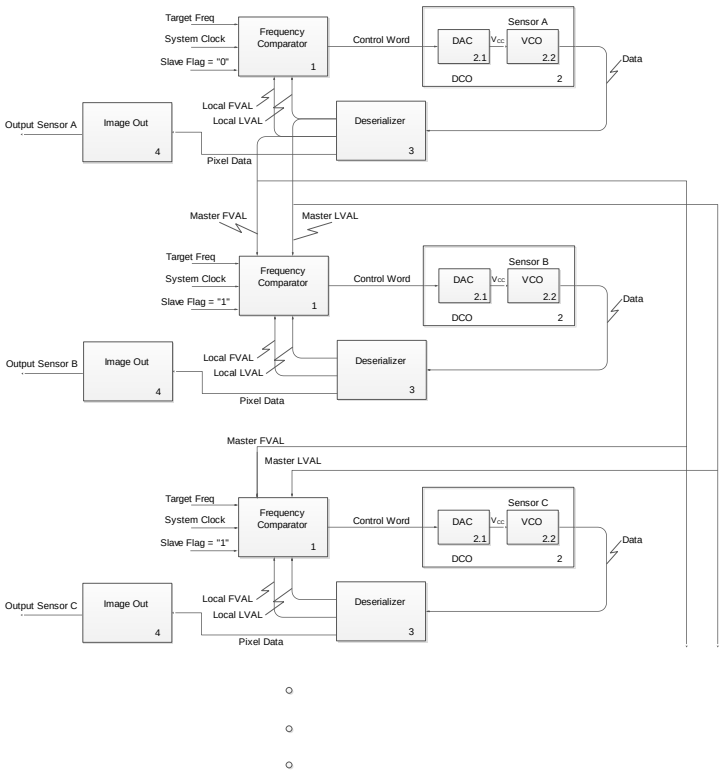


Figure 8 – Integration of the *Frequency Comparator* control module on the existing FPGA firmware.

6. RESULTS

The obtained results are presented concerning first the system's performance regarding the phase-frequency alignment error and then the system's behavior when submitted to various ambient temperatures.

6.1 Phase alignment

To perform the phase-frequency alignment it was necessary to implement the synchronization algorithm described in section 4. In practice this requires the exchange of the Master/Slave control signals. On a first stage this was performed on 2 NanoUSB2 platforms [17] (equipped with a Xilinx Spartan 3E FPGA and an USB 2.0 interface – Fig. 9), and able to connect one camera per platform. The control signals were exchanged between the two platforms through IO ports

configured as bi-directional buffers on the FPGA logic. On a second stage the algorithm was implemented on a DisposcopeUSB3 platform [18] (equipped with a Xilinx Spartan 6 FPGA and an USB 3.0 interface – Fig. 10) and able to connect up to 8 NanEye cameras (4 tested) on an adapter shield. The Master/Slave control signals are routed within the FPGA logic as seen on Fig. 8. The images from the various cameras are displayed in a single graphical interface, designated Awaiba Quadviewer (Fig. 11).

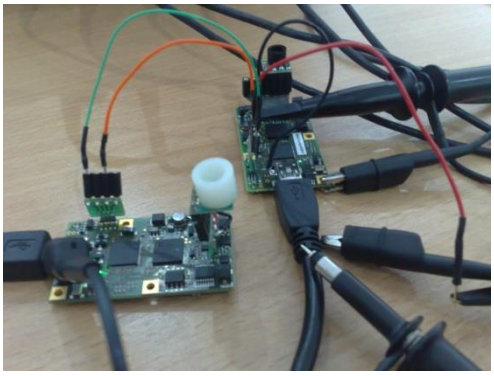


Figure 9 – Master/Slave interface implemented on two NanoUSB2 platforms (Stage 1 implementation).

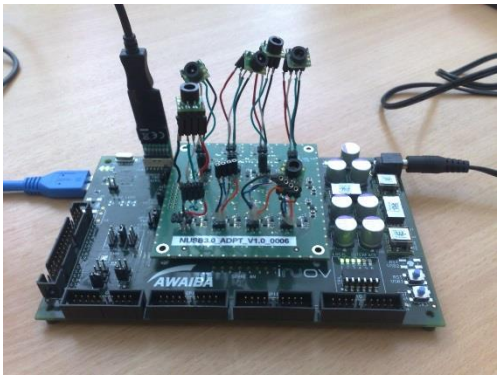


Figure 10 – Multiple NanEyes connected on a DisposcopeUSB3 platform (Stage 2 implementation).

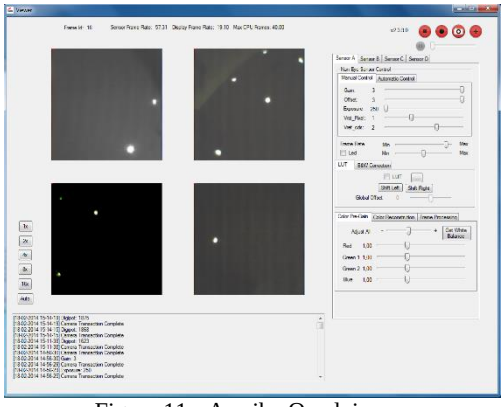


Figure 11 - Awaiba Quadviewer.

With this configuration it is possible to obtain the phase-frequency synchronicity of the decoded frames. The synchronization performed through this setting can be observed in Fig. 12 and in Fig. 13 the phase error obtained through this method can be seen.

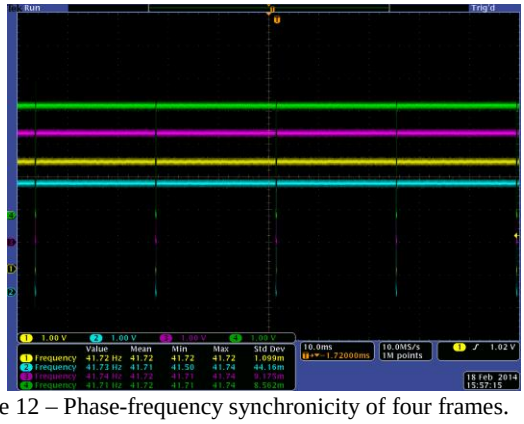


Figure 12 – Phase-frequency synchronicity of four frames.

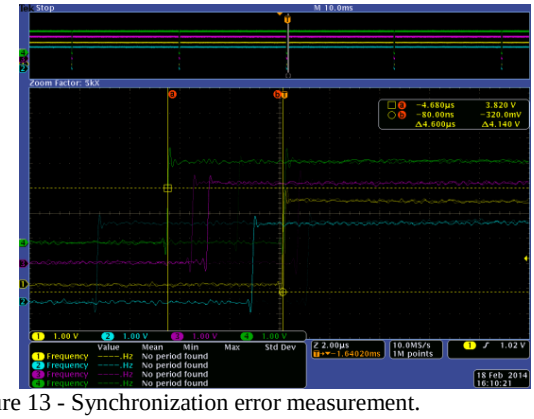


Figure 13 - Synchronization error measurement.

From the observation of both figures it was concluded that the synchronization error is small in relation to the frame period. Experimentally through successive measurements, the average phase error between frames was measured on the NanoUSB2 platform, as $Avg(\varphi_{Error})_{NanoUSB2} = 3.82$ µs. In Table 1, along with this, it is included the maximum and the minimum error.

Table 1 - Phase errors between frames in NanoUSB2 platform (comparison with simulation results).

NanoUSB2 Platform		**MATLAB/SIMULINK**	
$Max(\varphi_{Error})_{NanoUSB2}$	5.80 µs	$Max(\varphi_{Error})_{Simulink}$	95.00 µs
$Min(\varphi_{Error})_{NanoUSB2}$	0.40 µs	$Min(\varphi_{Error})_{Simulink}$	16.67 µs
$Avg(\varphi_{Error})_{NanoUSB2}$	3.82 µs	$Avg(\varphi_{Error})_{Simulink}$	54.63 µs

Comparing the results with the simulation it can be verified that the practical error is much lower. This is due to the fact that the model used in the simulation is more pessimistic than the reality, in particular, the delays introduced by components such as the DAC were entered considering the worst case scenario. Adding to these results the errors obtained on the stage 2 implementation, Table 2 is obtained.

Table 2 - Phase errors obtained on both test platforms.

NanoUSB2 Platform		**DisposcopeUSB3 Platform**	
$Max(\varphi_{Error})_{NanoUSB2}$	5.80 µs	$Max(\varphi_{Error})_{DisposcopeUSB3}$	8.76 µs
$Min(\varphi_{Error})_{NanoUSB2}$	0.40 µs	$Min(\varphi_{Error})_{DisposcopeUSB3}$	0.58 µs
$Avg(\varphi_{Error})_{NanoUSB2}$	3.82 µs	$Avg(\varphi_{Error})_{DisposcopeUSB3}$	3.77 µs

On a universe of 22 samples, the average phase error obtained on the DisposcopeUSB3 platform was found to be, $Avg(\varphi_{Error})_{DisposcopeUSB3} = 3.77$ µs. Even though on the stage 1 implementation the control signals are routed between boards, the phase errors obtained on both FPGA platforms are quite similar. Taking into account that the nominal frame rate on a NanEye camera is 44 fps, this means that the average phase error, on the DisposcopeUSB3 platform, is approximately 0.017 %.

6.2 Temperature tests

The image sensor's oscillation frequency has a strong temperature dependence due to the effects referred to in [15] and [16], and by which, it becomes necessary to subject the sensor to a range of temperature conditions. Two versions of the NanEye image sensor, 2B and 2C, were subjected to a temperature gradient of 50 °C on two NanoUSB2 platforms, designated Board 1 and Board 2. The aim was to observe the applied voltage variation to the image sensor (V_{CC}) as a function of the ambient temperature (T) in such a way that the sensor frame rate remained constant at 48 fps. In the graph of Fig. 14, and in Fig. 15 a) through d) can be observed the behavior of the image sensor, for the above specified test conditions.

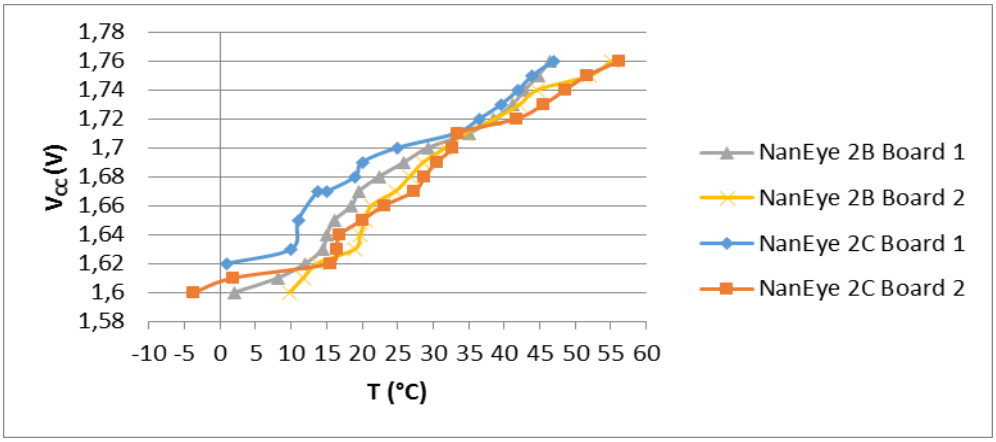
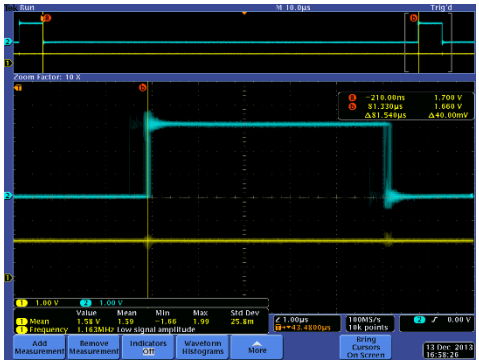
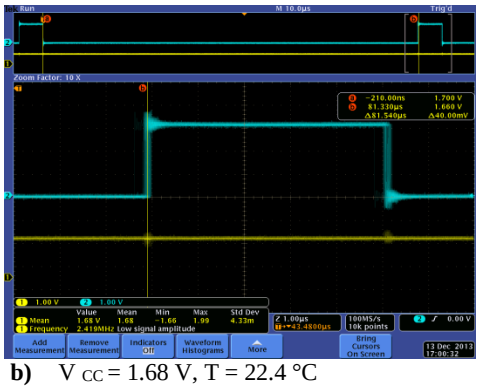


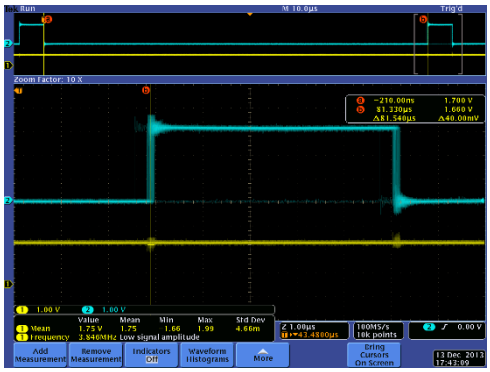
Figure 14 - Variation of the voltage applied to the sensor (version 2B and 2C) as a function of the ambient temperature.



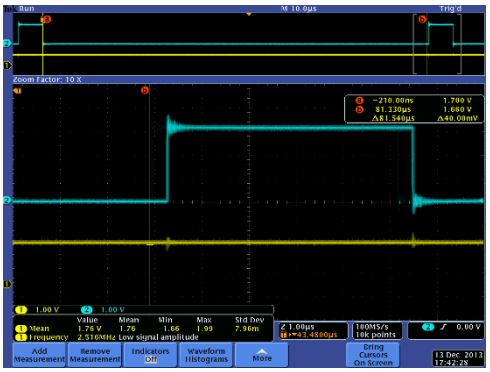
a) V_{CC} = 1.58 V, T = 2 °C



b) V_{CC} = 1.68 V, T = 22.4 °C



c) V_{CC} = 1.75 V, T = 44.8 °C



d) V_{CC} = 1.76 V, T = 46.4 °C

Figure 15 – Control voltage adjustment according to the ambient temperature. The bottom waveform, channel 1 signal, represents the applied voltage to the image sensor. The top waveform, channel 2 signal, represents the image line period signal decoded from the data stream.

Observing Fig. 15 a) to c) it is verified that the FPGA controlled applied voltage dynamically adapts to the ambient temperature. At a lower temperature the applied voltage is reduced, 1.58 V for a temperature of 2 degrees Celsius. With the increase in temperature the system automatically adjusts maintaining a constant line rate with a low period of 81.33 µs, which is equivalent to the transmission of data at a rate of approximately 48 fps. In Fig. 15 d), however, a gap between the desired and measured line rate was observable. This was due to the self-imposed 1.76 V voltage supply limitation. As the voltage applied to the sensor could not increase, the sensor could no longer operate at the desired frequency due to the temperature increase, recovering again to the target frequency as soon as the temperature dropped to a voltage range below 1.76 V. This voltage limit was imposed in order to demonstrate precisely this phenomenon and to prove the system's ability to recover the synchronism with the target frame rate.

Similar tests were performed when operating 4 cameras simultaneously (as seen in Fig. 16). The purpose was to determine if the control system would maintain phase-frequency synchronization over large temperature differences between cameras. In example, as seen in Fig. 17, one of the cameras was submitted to sub-zero temperatures causing it to freeze, whilst another was subjected to temperatures around 50 ºC.

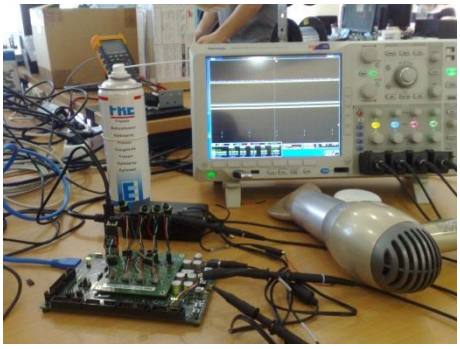


Figure 16 – Setup of the temperature test system.

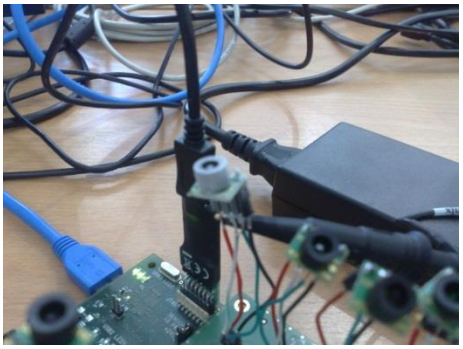


Figure 17 - NanEye camera submitted to sub-zero temperatures.

In Fig. 18 a) and b) the effective synchronization of the cameras can be observed. The Top waveforms, scope channels 3 and 4, represent the voltage levels applied to A and B sensors, respectively. The bottom waveforms represent the FVAL signals corresponding to sensor A, on channel 2, and sensor B, on channel 1.

a) $T_{Sensor\ A}$ = 22.5 °C
$T_{Sensor\ B}$ = -1.7 °C

b) $T_{Sensor\ A}$ = 59.7 °C
$T_{Sensor\ B}$ = 0.9 °C

Figure 18 – Phase-frequency synchronization between two NanEye cameras operating at different ambient temperatures.

For a better understanding, the obtained results were condensed on four different test cases represented on Table 3.

Table 3 – Phase-frequency synchronization temperature test cases.

Sensor A			**Sensor B**			
Temperature (°C)	**V_{CC} Voltage (V)**	**Frame Rate (fps)**	**Temperature (°C)**	**V_{CC} Voltage (V)**	**Frame Rate (fps)**	**Instantaneous φ_{Error} (µs)**
22.9	1.81	46.54	22.9	1.83	46.52	25
-1.2	1.68	46.52	23.1	1.83	46.53	7
22.5	1.83	46.52	-1.7	1.69	46.52	8
59.7	1.91	46.52	0.9	1.70	46.52	4

In the first test case, both the sensors are at a similar temperature, hence the applied voltage to each one of these is also similar, allowing both cameras to maintain phase-frequency alignment at a constant 46.5 fps frame rate. In the second situation, sensor A was submitted to a sub-zero temperature. As a result the control system reduces the applied voltage to compensate. In test case 3 (Fig. 18 a)), the roles reverse. Sensor B is submitted to temperatures below 0 °C while sensor A is at ambient temperature. Under these conditions the instantaneous phase error was found to be 8 µs. On the last case, which is illustrated in Fig. 18 b), the total temperature difference between sensors was about 60 °C. It was observed that phase-frequency synchronism was maintained regardless of this large temperature gradient. It is to be noted that the same principle of operation can be applied when dealing with cameras placed at different lengths. The voltage drop on the power supply cable is automatically compensated by the control system meaning the frequency control is independent from the cable's length.

7. CONCLUSIONS

The frequency control method presented on this paper demonstrated its ability to dynamically control the effective frame rate on a NanEye image sensor by automatically modulating the imposed supply voltage. This enables it to achieve phase-frequency synchronism between multiple cameras regardless of: ambient temperature (tests conclude that synchronism is maintained even with temperature gradients of 60 ºC), cable length, sensor version, and FPGA platform. It is also capable of performing frame phase synchronization with an average error of approximately 0.017% of the total frame period. However the system's operation is limited by the operating range of the sensor. One of the characteristics of this architecture relates to the limited voltage dynamic range of the sensors, which is between 1.6 V and 2.4 V. This means that for temperature extremes the Frequency Comparator is not capable of performing the compensation leading to a frequency error that may cause desynchronization between the video streams. However, this is not a limitation of the control system in itself, but rather of the voltage supply range the sensor can be subjected.

ACKNOWLEDGEMENTS

We thank all the Awaiba staff for their encouragement and support. Their technical knowledge and expertise were invaluable contributions to the inception and continued development of this work. The authors would like to acknowledge the Portuguese Foundation for Science and Technology for their support through project PEst-OE/EEI/LA0009/2011. Also acknowledged is the Funding Program + Conhecimento II: Incentive System to Research and Technological Development and Innovation of Madeira Region II, through the project Vision 3D – MADFDR – 01 – 0190 – FEDER – 000014.

REFERENCES

[1] D. Wedge, D. Huynh, and P. Kovesi, "Video Sequence Synchronization", University of Western Australia, (2007).
[2] W. Kaczurba, "FPGA-Based System Combines Two Video Streams to Provide 3D Video," Analog Dialogue no. 47–12, 1–5 (2013).
[3] H. P. Moravec, "Robot Spatial Perception by Stereoscopic Vision and 3D Evidence Grids," Carnegie Mellon University, (1996).
[4] Y. Zhao and I. E. G. Richardson, "Complexity management for video encoders," Proc. tenth ACM Int. Conf. Multimed. - Multimed. '02, 647 (2002).
[5] L. Zhang and W. Dong, "Spatial-Temporal Color Video Reconstruction from Noisy CFA Sequence," IEEE Trans. Circuits Syst. Video Technol. vol. 20 no. 6, 1–18 (2010).
[6] C. S. Hannangara, I. E. Richardson, and A. J. Miller, "Computational complexity management of a real-time h.264/avc encoder," IEEE Trans. Circuits Syst. Video Technol. vol. 18 no. 9, 1–11 (2007).
[7] G. Stitt, "Are field-programmable gate arrays ready for the mainstream?," IEEE Micro vol. 31 no. 6, 58–63 (2011).
[8] G. Litos, X. Zabulis, and G. Triantafyllidis, "Synchronous Image Acquisition based on Network Synchronization," Conference on Computer Vision and Pattern Recognition Workshop CVPRW '06 no. c, (2006).
[9] P. Greisen, S. Heinzle, M. Gross, and A. P. Burg, "An FPGA-based processing pipeline for high-definition stereo video," EURASIP J. Image Video Process. vol. 2011 no. 1, 18 (2011).
[10] N. Flores-Guzmán, J. H. Sossa-Azuela, and R. Bizuet-García, "A Decision and Communication Management Methodology for embedded Multi-smart Camera systems, applied to real-time inspection in lamps production," Second ACM/IEEE International Conference on Distributed Smart Cameras, 1–10 (2008).
[11] T. Kanade, P. J. Narayanan, and P. W. Rander, "Virtualized Reality: Being Mobile in a Visual Scene," Int. Conf. on Artificial Reality and Tele-Existence/Conf. on Virtual Reality Software and Technology, 273–285 (1995).
[12] Silicon Labs, "AN377 - Timing and Synchronization in Broadcast Video," (2009).
[13] J. Xia, X. Zhao, and W. Guo, "Research on the method of multi-channel video acquisition and display based on FPGA," I.J. Comput. Netw. Inf. Secur. vol. 1 no. November, 17–23 (2010).
[14] M. Wäny and E. Reis, "NanEye_2D ASIC SPECIFICATION," Awaiba vol. 41 no. 2.00.6, 1–24 (2014).
[15] R. Kumar and V. Kursun, "Impact of temperature fluctuations on circuit characteristics in 180nm and 65nm CMOS technologies," 2006 IEEE Int. Symp. Circuits Syst. no. V, 4 (2006).
[16] S. Suman and B. P. Singh, "Ring Oscillator Based CMOS Temperature Sensor Design," Int. J. Sci. Technol. Res. vol. 1 no. 4, 76–81 (2012).
[17] Awaiba, "NanoUSB2.1 Product Specification," no. 1.0.0, 1–8 (2013).
[18] V. Bexiga and F. Piedade, "Evaluation Board Interfaces," Awaiba no. 0.0.1, 1–36 (2013).