Multi-camera synchronization core implemented on USB3 based FPGA platform

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

ABSTRACT

Centered on Awaiba's NanEye CMOS image sensor family and a FPGA platform with USB3 interface, the aim of this paper is to demonstrate a new technique to synchronize up to 8 individual self-timed cameras with minimal error. Small form factor self-timed camera modules of 1 mm x 1 mm or smaller do not normally 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 of synchronizing multiple self-timed cameras with only 4 wire interface has been solved by adaptively regulating the power supply for each of the cameras. 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, a Master-Slave interface was implemented. A single camera is defined as the Master, 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 implementation 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

3D image processing requires the synchronization of video signals from multiple cameras. To this end, synchronization between video signals is necessary. If this is not achieved, frame desynchronization or frame loss can occur, 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]. Thus 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.

The increase in the number of 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. By doing this, the delays and consequently the overall system error are minimized by synthesizing dedicated parallel circuits to process and correct the phase-frequency error. A possible solution to implement this control at the hardware level can be achieved using a FPGA platform and 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]. However the target hardware's constraints have to be taken into account. Firstly the NanEye image sensor, described in [8], is a self-timed camera, meaning that the pixel clock signal is generated internally on the sensor, through a voltage controlled ring oscillator (*Voltage Controlled Oscillator* - 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 high jitter level and the pixel clock frequency has a strong dependency on the ambient temperature due to the use of a ring oscillator [9], [10]. Secondly, despite the FPGA platforms having external memory (SDRAM and Flash) and the instantiation of RAMs and FIFOs is possible, this was not considered a valid option considering that this project was intended to create a synchronization system that actively modulated the camera behavior.

2. PROPOSED TOPOLOGY

Fig. 1 shows the proposed topology for the frame rate control. The function of this structure 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 based on the frame period signal, Frame Valid (FVAL). The topology is composed of a controller block (called 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 work. 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.

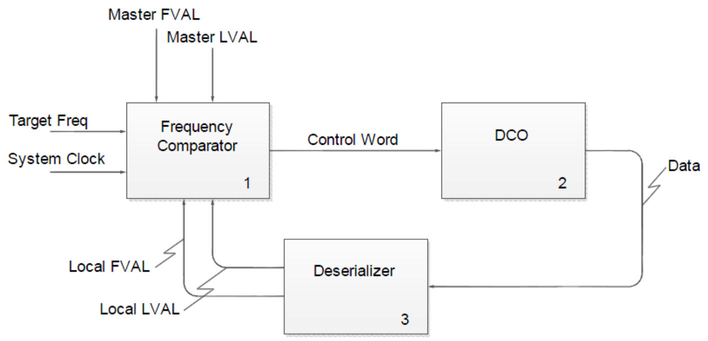

Figure 1 - Block diagram of the frame rate control system.

The image sensor data is transmitted through a LVDS interface using a phase encoding scheme. It is then decoded and converted from serial to parallel format and outputted to the USB interface. From 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.

3. FREQUENCY COMPARATOR

The first stage of development focused on the frequency control of a single camera by dynamically adjusting the applied voltage. To realize this goal, only the camera's LVAL needs to be analyzed. One way to perform this is to ascertain the line's period by counting the number of rising edge transitions of a well-known clock signal. 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 value 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 [8], 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,

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

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

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

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 the calculated target, this means that the camera is operating at a lower frequency than desired, so the control module should increment the applied voltage to compensate. On the other hand, if the count is below the designated target, it infers that the camera is running at a higher frame rate than intended, conversely the applied voltage should be reduced. 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. Nonetheless, this technique alone does not ensure phase synchronization between various image sensors. By associating multiple NanEyes in parallel and performing frequency control employing just the method portrayed above, it was observed that in spite of the fact that 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 $\text{Target}_{\text{Clk}}$ can be made, by adding or subtracting a default value to this reference. Considering one of the sensors as the reference (Master) and a second sensor as dependent (Slave), it is possible to correct the frame phase misalignment by detecting the instant the local FVAL signal transitions and comparing it with the transitioning of the Master FVAL. 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's frequency by increasing the $\text{Target}_{\text{Clk}}$ by a suitable amount (initially the adjustment was made in unitary steps). On the other hand, if the Slave FVAL transition occurs after the Master signal, this implies that the local frame is lagging in relation to the Master FVAL. This way, it is necessary to increase the pixel clock frequency by reducing the $\text{Target}_{\text{Clk}}$ to correct the error. However, even though this technique ensures phase and frequency synchronization between frames, it does not take into account the time needed for the misalignment to be corrected. As illustrated in Fig. 3, it can easily be concluded that a considerable amount of time is required to perform the phase correction if only 1 clock period is added to $\text{Target}_{\text{Clk}}$. As such it was necessary to create a mechanism that would allow the control system to make a fast phase adjustment when a high phase error was detected. If the error was found to be above a defined threshold, the adjustment would be sped up by adding or subtracting 30 clock periods to the number of target clock cycles, $\text{Target}_{\text{Clk}}$. On the other hand, if the error was smaller than the defined threshold the adjustment would be unitary to guarantee low oscillation and small error. As exemplified in Fig. 4, 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, a correction system using fixed discrete correction steps can be implement. Experimentally, the amplitude of the correction and the transition limit between the higher correction step and the unitary correction step, were defined. In the example given in Fig. 4, the fast step is 30 if the "XOR width" is greater than the set limit (which is 1.5 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.

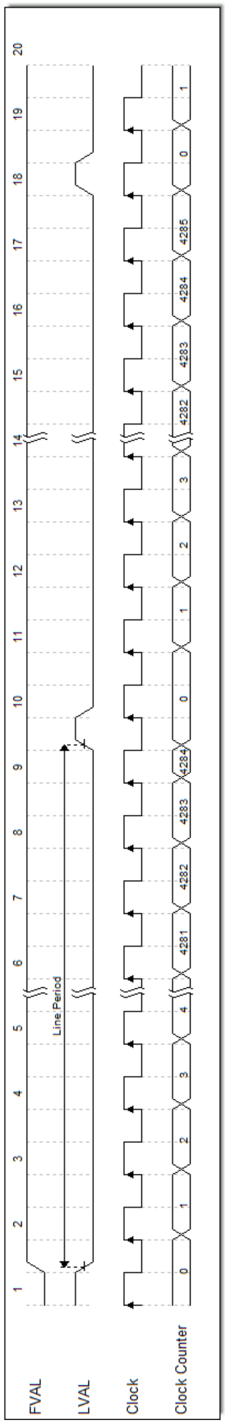

Figure 2 – Time diagram of the line period count.

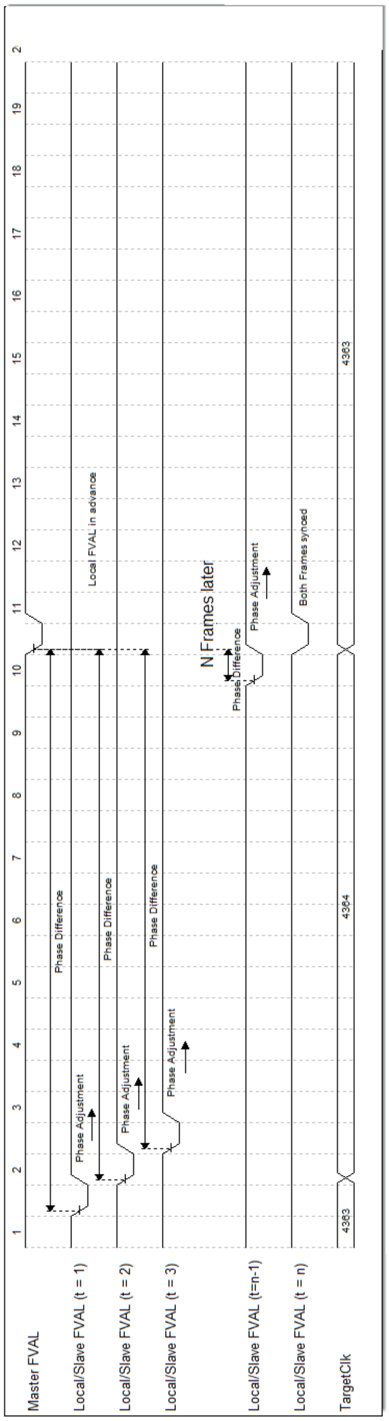

Figure 3 – Slow temporal phase adjustment between frames.

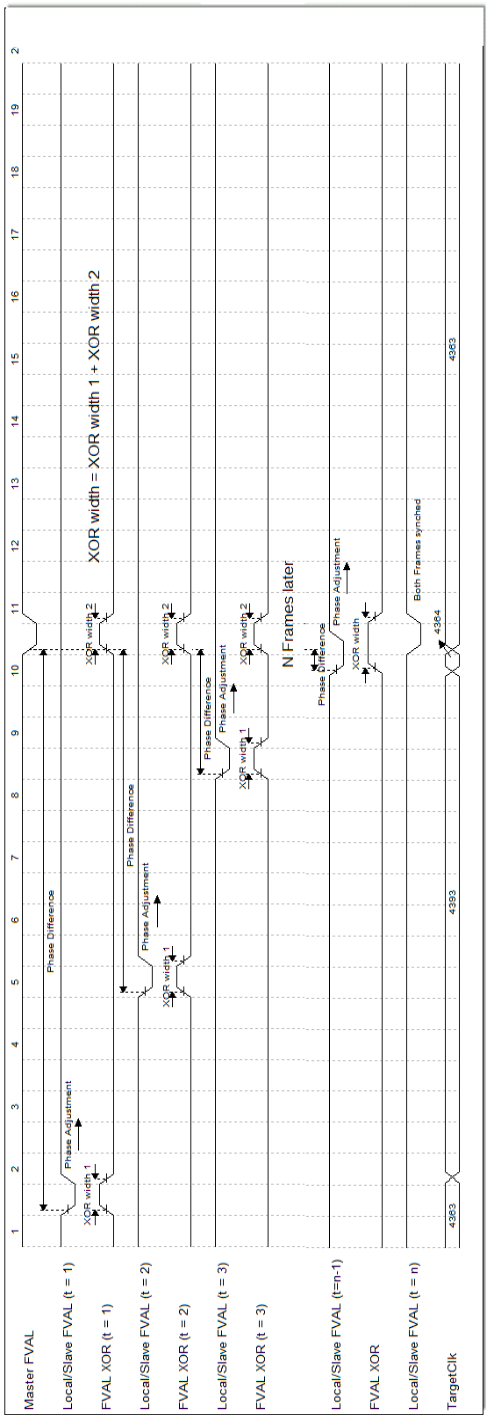

Figure 4 - Fast temporal phase adjustment between frames.

4. PRACTICAL IMPLEMENTATION

Assigning one of the cameras the role of Master, implies that its frame rate is directly controlled by the PC based GUI through command exchange over the USB3 interface. The remaining cameras are configured as Slaves, and are not directly controlled from the interface, instead they receive control signals (Master FVAL and Master LVAL) from the Master camera. The frame period 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 frequency of the Master camera. The applied voltage to the slave sensor is set to synchronize its frequency with the Master camera based on the result.

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. 5. 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, which modulates the voltage to be applied to the sensor based on the target word frequency (Master mode), or based on 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 and generating the vsync and hsync signals; and a block called Image Out, which makes the adaptation of the pixel data for transmission over the USB interface.

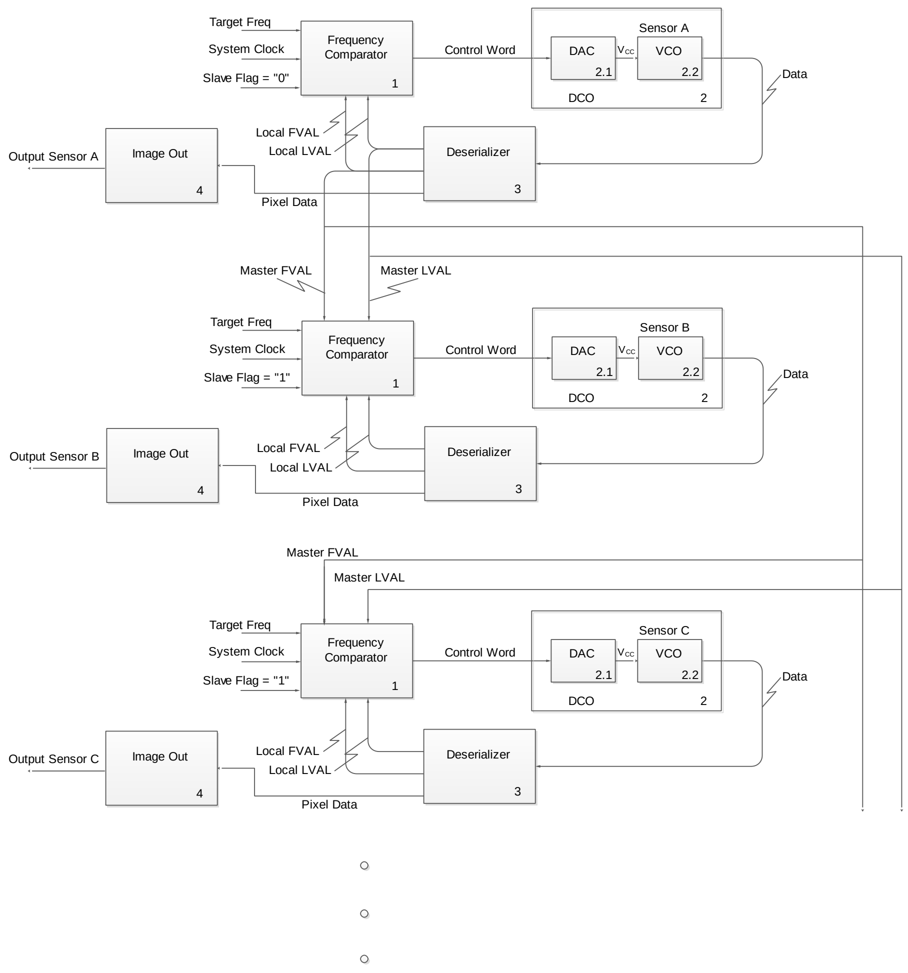

Figure 5 – Integration of the *Frequency Comparator* control module on the existing FPGA firmware.

5. RESULTS

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

5.1. Phase alignment

In practice, the phase-frequency synchronization algorithm described in section 3, requires the exchange of the Master/Slave control signals. On a preliminary stage this was implemented on 2 NanoUSB2 platforms [11] (equipped with a Xilinx Spartan 3E FPGA and an USB 2.0 interface), and able to connect one camera per platform. The control signals were exchanged between platforms through the available GPIO ports. On a second stage the algorithm was implemented on a DisposcopeUSB3 platform [12] (equipped with a Xilinx Spartan 6 FPGA and an USB 3.0 interface) and able to connect up to 8 NanEye cameras (tested with 4) through an adapter board. The Master/Slave control signals are routed within the FPGA logic as seen on Fig. 5. With this configuration it is possible to obtain the phase-frequency alignment of the decoded frames. The synchronization performed through this setting can be observed in Fig. 6 and Fig. 7. On the latter the phase error obtained through this method can be observed.

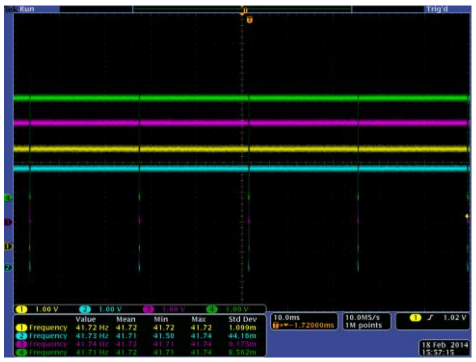
Figure 6 – Phase-frequency synchronicity of four frames.

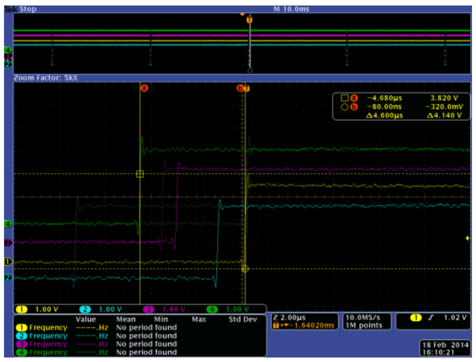
Figure 7 - Synchronization error measurement.

Observing both figures it can be concluded that the synchronization error is quite 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. Adding to this the results obtained on the stage 2 implementation, Table 1 is obtained.

Table 1 - 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 %.

5.2. Temperature tests

The use of a ring oscillator to generate the pixel clock implies that the data stream's frequency is strongly dependent on temperature [9], [10]. As such, it becomes necessary to subject the sensor to a range of temperature conditions to evaluate the control system's performance. 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 goal was to observe the applied voltage variation (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. 8 can be observed the behavior of the image sensor, for the above specified test conditions.

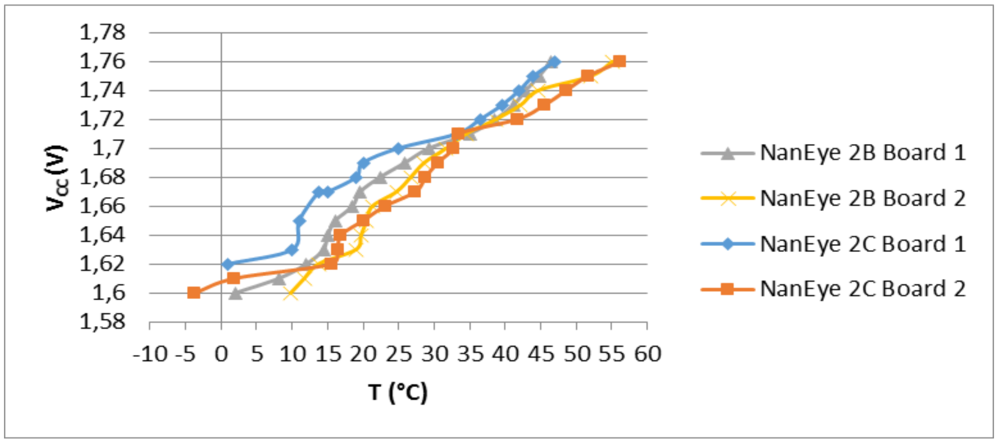

Figure 8 - Variation of the voltage applied to the sensor (version 2B and 2C) as a function of the ambient temperature.

Similar tests were performed when operating 4 cameras simultaneously. The purpose was to determine if the control system would maintain phase-frequency synchronization over large temperature differences between cameras. For a better understanding, the obtained results were condensed on four different test cases represented on Table 2.

Table 2 – 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

On the first situation, portrayed on Table 2, both sensors are at equivalent temperatures, therefore the applied voltage to these is also similar, allowing both cameras to maintain phase-frequency alignment at a constant 46.5 fps frame rate. In situation number 2 and 3, one sensor is submitted to sub-zero temperatures while the other is maintained at ambient temperature. As a result the control system reduces the applied voltage to the frozen sensor to compensate the phase offset. On the last case, the total temperature difference between sensors was approximately 60 °C. It was observed that phase-frequency synchronism was maintained regardless of the large temperature gradient. 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.

6. CONCLUSIONS

The control algorithm presented on this paper enabled the realization of a FPGA based multi-camera synchronization core by actively modulating the supply voltage on 4 NanEye image sensors. Phase-frequency synchronism between the cameras was achieved regardless of: ambient temperature (synchronism is maintained even when the temperature gradient is 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. On the other hand the system's operation is limited by the operating range of the sensor. One of the characteristics of this architecture relates to the limited supply voltage 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] M. Wäny and E. Reis, "NanEye_2D ASIC SPECIFICATION," Awaiba vol. 41 no. 2.00.6, 1–24 (2014).
[9] 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).
[10] 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).
[11] Awaiba, "NanoUSB2.1 Product Specification," no. 1.0.0, 1–8 (2013).
[12] V. Bexiga and F. Piedade, "Evaluation Board Interfaces," Awaiba no. 0.0.1, 1–36 (2013).