FPGA–based slope computation for ELTs adaptive optics wavefront sensors

L.F. Rodríguez Ramos*[a], J.J. Díaz García[a], J.J. Piqueras Meseguer[a], Y. Martin Hernando[a], J.M. Rodríguez Ramos[b]
[a]Instituto de Astrofísica de Canarias, Vía Láctea s/n, 38200 La Laguna, Spain;
[b]University of La Laguna, Spain

ABSTRACT

ELTs laser guide stars wavefront sensors are planned to have specifically developed sensor chips, which will probably include readout logic and D/A conversion, followed by a powerful FPGA slope computer located very close to it, but not inside for flexibility and simplicity reasons. This paper presents the architecture of an FPGA-based wavefront slope computer, capable of handling the sensor output stream in a massively parallel approach. It will feature the ability of performing dark and flat field correction, the flexibility needed for allocating complex processing schemes, the capability of undertaking all computations expected to be performed at maximum speed, even though they were not strictly related to the calculation of the slopes, and the necessary housekeeping controls to properly command it and evaluate its behaviour. Feasibility using today's technology is evaluated, clearly showing its viability, together with an analysis of the amount of external memory, power consumption and printed circuit board space needed.

Keywords: Adaptive optics, wavefront sensor, FPGA, real-time processing

1. INTRODUCTION

Wavefront sensing for the Adaptive Optics of Extremely Large Telescopes (ELTs) is a major challenging task to be accomplished at speeds in excess of 1 KHz, with high resolution detectors required for an adequate phase wavefront resolution and devoted to a number of Laser Guide Stars (LGS) up to seven. Measured slopes will be delivered to the reconstructor engine to perform all tomography computations and finally the control data of the set of deformable mirrors will be calculated, essentially at the mentioned speed and with minimum latency.

This article is specifically addressing the first step of the reconstruction process, i.e. the wavefront slope measurements, provided that a Shack-Hartmann arrangement is likely to be used in the optics design of the wavefront sensor. Early design data from both TMT and E-ELT is put together in order to conceptually design a real-time slope calculator capable of handling worst case of every parameter, showing that today's technology, smartly used, is capable of fulfilling the overwhelming demands of such processing. A big margin is then available to afford new improved algorithms to be implemented with faster future technology.

Planned detectors for giant telescopes LGS wavefront sensors are having a huge number of pixels (10^5-10^6), digitized to 16 bits, to accommodate from 50x50 to 210x210 subapertures. Closed loop periods ranging from 700 to 1000 Hz, with exposure times of 500 microseconds, and latencies below 200 microseconds as a goal. The very important issue of how many simultaneous outputs might be used is not addressed, so reasonable estimations will be assumed in this conceptual design.

The proposed conceptual design described in this article is based on the use of massive parallel processing with FPGAs[3][4], to be used in the critical path of the slopes computation. This very fast and low-latency scheme will be helped by a conventional processor in charge of all "housekeeping" tasks, including initialization, supervision, data logging, and also all other small-speed loops and calculations to be performed.

Next chapter will describe the algorithm to be implemented, followed by the description of the proposed stream processor and a conceptual design of the different modules (cap. 4). Finally, key parameters of the design will be evaluated in order to verify the viability of the concept and extract its main features.

*LRR@iac.es; phone +34 922 605 200; fax +34 922 605 210; www.iac.es

2. SLOPE COMPUTATION ALGORITHM

There are a number of strategies that can be used to determine the slopes in a SH wavefront sensor[1]. Most of them are based on the measurement of lateral image displacements to be directly associated with wavefront slopes, quadrant detector, thresholded/weighted centroids, correlation,…, but also a constrained matched filter can be used to allow for dynamically adapting the computation to a number of telescope/atmosphere relevant parameters. This last case is planned for the TMT LGS wavefront sensor, together with several non essential processes which can be considered not being directly related to the slope computation, but requiring its real time implementation in close relation to the main computation.

Fig. 1 depicts the processing scheme being considered for the TMT LGS wavefront sensor, and will be used here as a guideline of the algorithm to be implemented. The WFS detector is first corrected from bias and flat using the appropriate information previously stored in the system. A direct detector output (not drawn) should be used for this purpose, allowing for bias and flat images to be obtained and pre-processed in advance of any observation by a conventional computer, having them downloaded afterwards to the slope computer. This approach, as previously mentioned, will be used all along the design, in order to keep the massively parallel computing reserved for the fast loops, but allowing conventional (slower) processors to help in doing the low-demanding tasks. We will only address in this work the fast processing needs, deliberately neglecting all time relaxed computations.

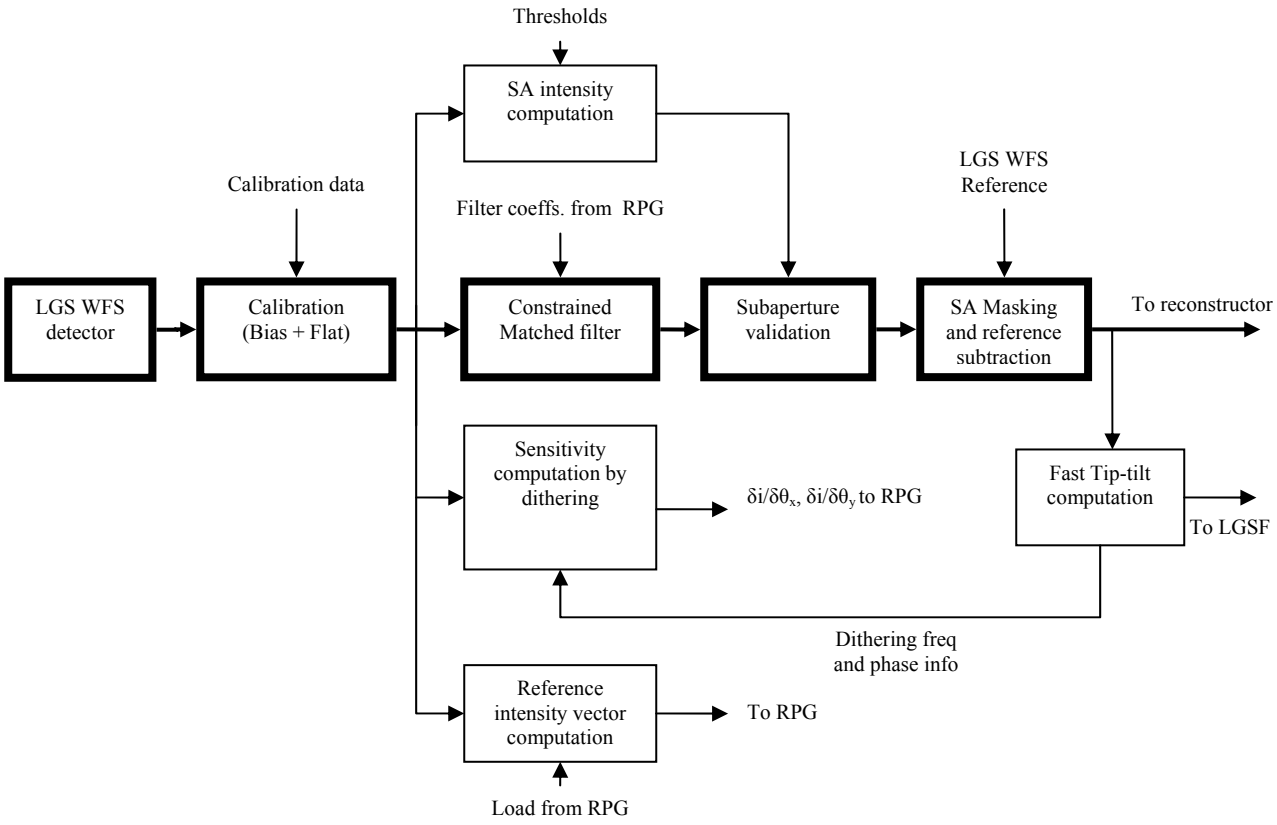

Fig. 1. Graphical layout of the processing pipeline needed for the slope computation of the laser guide star wavefront sensor. Only processing to be performed in real-time at fastest loop speed has been depicted, provided that conventional slower computer will be in charge of all necessary housekeeping and non-fastest loop calculations. Thicker boxes show the main processing pipeline, from detector readout to slope output.

The corrected intensity image is then delivered to four different blocks, for computing the subaperture (SA) intensity, the constrained matched filter itself, the angular sensitivity measurement and the reference intensity vector. Detailed

description of each module falls beyond the scope of this article, but can be found elsewhere [2]. SA intensity will be used to validate or discard the output of a slope depending on the intensity level against a predefined threshold, using the previous valid value as the alternative, and finally may be masked or subtracted from reference depending on the available information regarding the telescope pointing status. There is also a fast tip-tilt computation, which is performed on the complete set of subapertures, to be used when commanding the mirrors to stabilize the pointing of the LGS.

The thicker boxes in Fig. 1 depict the main process pipeline horizontally arranged, from detector readout to slope output, and thinner boxes describe other required processes to be computed also at maximum loop speed for practical reasons.

3. STREAM PROCESSOR ARCHITECTURE

A very important issue not sufficiently considered in the past is the number of outputs of the wavefront detector. This number is crucial in order to properly arrange a fast low-latency processor capable of handling the pixels at the proper speed. Detectors have been manufactured so far as some kind of black boxes having a light input and an electronic output, sometimes standard like Camera-link or similar. This should no longer be the case when high resolution detectors have to be readout in the kilohertz range, but instead a processor engine should be located to each detector output, provided that no subaperture shall lie between two outputs. This limitation, on the other hand, should not be difficult to satisfy provided that the ELTs AO programs include the custom development of the detector itself. However, even in the case of having some standard output interface for practical reasons, detector outputs can be deserialized to provide a different stream processor to each output.

3.1 Detector readout scheme and timing

Fig. 2 depicts the expected readout scheme to be found in a S-H wavefront sensor detector, to be found at each detector output. In order to easily visualize the sequence, a square SA arrangement has been supposed, being the subaperture rows aligned with the horizontal pixel rows, but any other arrangement may also be used, even non-cartesian pixel layouts, with the same readout scheme. Upper row represents the arrival of photons to the detector, and the integration period. Readout starts immediately after the end of the integration period, after the (not drawn) frame transfer time. Detector readout (second row) is not expected to last for all the integration period, conceptually separating the unavoidable task of collecting photons from the technology driven reading-out and digitizing. Rows or any other subsets of subapertures will be readout and made available for processing during fractions of the readout time, as shown in the third line of the picture, and finally, individual subapertures shall be read discontinuously in the depicted way, having identified the instant when all pixels of the first and last subaperture have been readout.

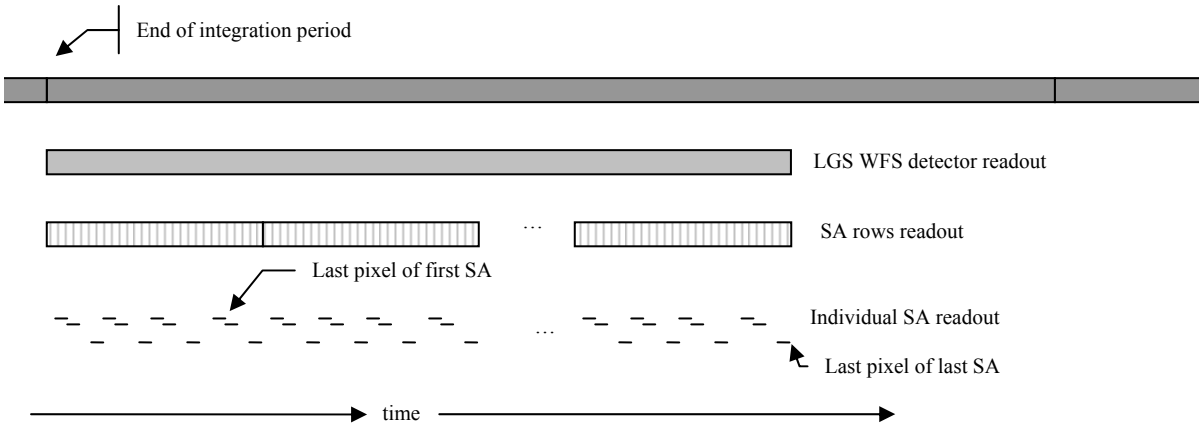

Fig. 2. Conceptual time diagram of the readout scheme considered for any of the outputs of the detector of the LGS wavefront sensor. Each SA pixel is made available to the processor engine in a discontinuous way, until a complete row of subapertures is read.

3.2 Stream processing approach

Fig.2 directly advices on how to arrange the parallel processing of the detector pixel values, in order to implement the computation of the slopes. Combining also the information of Fig.1, processing should be based on having the subaperture as the finest grain, and delivering slopes to the reconstructor as soon as they are available, in order to help to minimize the latency of the correction loop(s). To this end, the overall tip-tilt computation should be considered as part of the reconstruction process, and moved to the second step of the loop, or at least treated separately from the slope computation streams.

The physical arrangement of the processing can be found in Fig.3. Every detector output is treated separately by its dedicated stream processor. Each processor will also handle its work by adapting itself to the interlaced way in which the subapertures are readout. This scheme will also be valid if all the detector outputs are somehow serialized to comply with some standard interface, camera link or similar, provided that the proper deserializer is installed before the stream processors battery. In this case, only a very small extra latency should be accounted for, and will be neglected at this point.

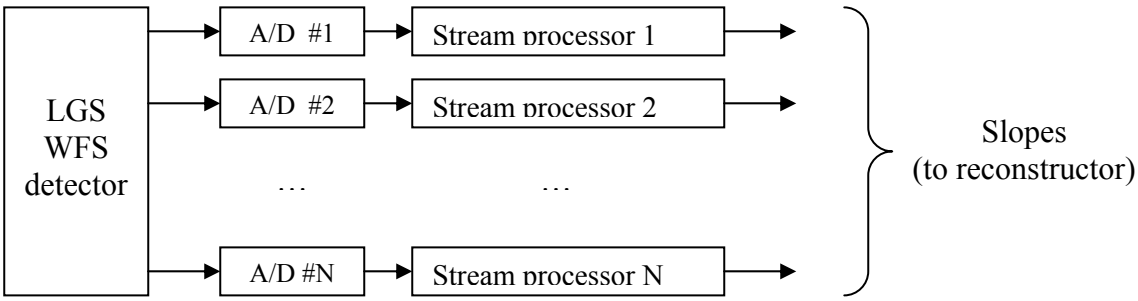

Fig. 3. Physical arrangement for the LGS WFS detector processor. Every detector output is separately handled with a dedicated stream processor.

With this scheme, every stream processor should also have as many subapertures processors as there are in a row, according to Fig. 2, separately implemented and working in parallel. Each of this processors will be reused in every new row of SAs, conceptually being in charge of a column of subapertures.

4. MODULES CONCEPTUAL DESIGN

Having established the overall outline of the slope processor, from now on will concentrate in the finest grain identified, i.e. the SA processor. The tasks depicted in Fig.1 will be conceptually designed, up to a level of detail big enough to allow for an estimation of the required resources from the electronic point of view: PCB space, power consumption and FPGA slices.

4.1 Calibration

Calibration operation will basically consist in a multiplication plus an adding, in order to correct for flat field and subtract bias. The necessary information for the correction will be prepared at much lower speed by the housekeeping processor and downloaded to the dual port memory depicted in Fig. 4. The use of dual ported memory is highly recommended in order to allow for very easy interaction with the module, not requiring bus arbitration. Furthermore, a FIFO memory will be used to serve as buffer between the data memory and the processor, easily guaranteeing the presence of the correct information whenever a pixel is to be processed. This module may also be designed to cope will all subapertures of the particular output, instead of a column-based approach, because only a pixel-based linear transformation is involved, without any storage of the intermediate results.

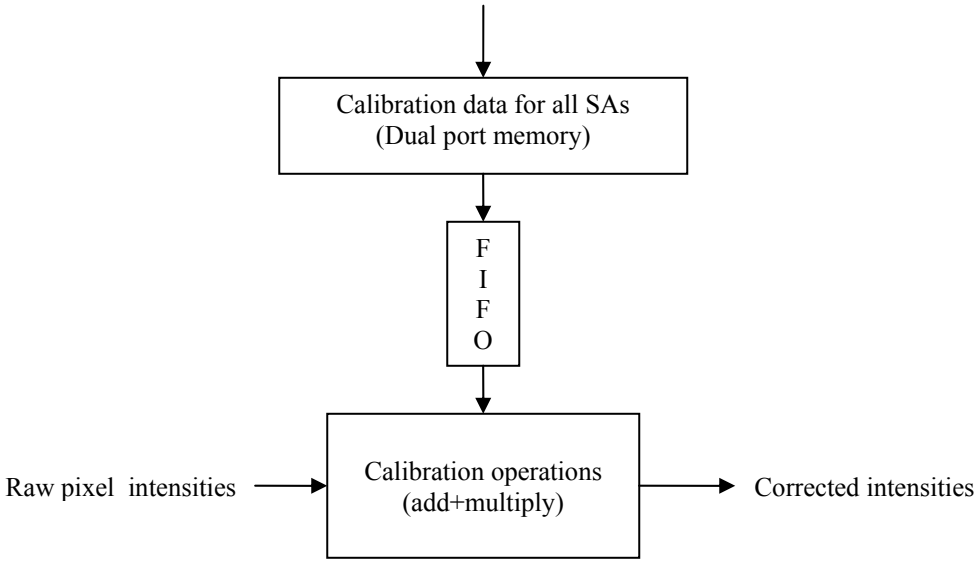

Fig. 4. Calibration module conceptual design.

Memory requirement for every stream processor can be estimated from the amount of pixels expected for the LGS WFS, divided by the number of outputs, times the bias and flat field information per pixel (see below). FIFO memory can be very shallow because is only intended for timing isolation between modules, and is expected to be implemented internally in the FPGA using the minimum block RAM usable size (~1Kbyte).

4.2 Constrained matched filter

The concept proposed for the pixel calibration may also be used for the implementation of the constrained matched filter, provided that the nature of the operations and the information flow is very similar, obviously considering the fact that independent intermediate storage shall be installed for each SA (Fig. 5). In this case, the filter coefficients will be computed and downloaded, at some lower speed, to the dual port memory, and the estimated slopes will be computed and delivered at the end of the SA readout.

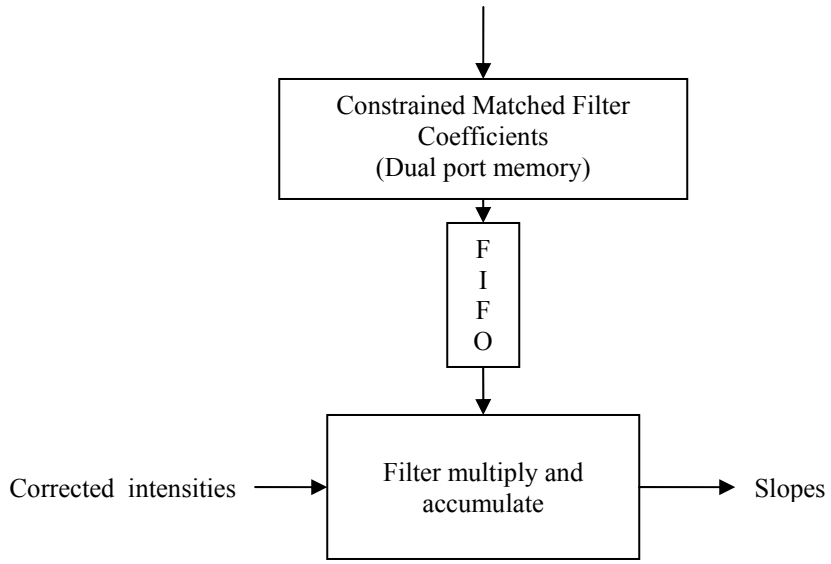

Fig. 5. Constrained matched filter conceptual design.

Depending on the way in which this conceptual design may be implemented, it could be needed some multiplexing between the dual port memory and the FIFOs, in order to make it possible to have only one physical dual-port memory.

In any case, the amount of memory needed here can be estimated accounting for as many coefficients as twice the number of pixels, because two slopes are being calculated. Thus, the figure obtained for the calibration module (1 Mbyte) may be used again, but the FIFO depth should be increased to hold the complete set of coefficients for one SA, approximately 4Kbyte.

4.3 SA intensity computation

This module will compute the sum of all the corrected intensities of a subaperture, and will only need some threshold information to compare with. Even in the case of updating these thresholds at some lower speed, it will be only one value per SA, so no special attention will be paid to it in our conceptual design.

4.4 Sensitivity measurement by dithering

The sensitivity measurement requires the time averaged combination of the instantaneous value of the dithering signal with the corrected pixel intensities, in order to estimate the expected value of the partial derivative of the slope with respect to the x and y coordinates. The magnitude to be calculated is:

$$\mathrm{E}\left(\frac{\partial i}{\partial \theta_x}\right) = \frac{\int i(t)\cos(\omega t + \Delta)dt}{a\int \cos^2(\omega t + \Delta)dt}$$

Where a, w, and Δ are the parameters describing the amplitude, frequency and phase of the dithering signal, and i(t) is the corrected pixel intensity of a particular pixel. An equivalent calculation shall be done for the y slope.

This computation is not due to be done in real time at the fastest speed, because the results will only be used at much lower speeds to update the constrained matched filter coefficients. It could be implemented by storing all data and performing the computation "off-line" by a conventional computer. However, it is also sensible to implement the computation of the integrals as a part of the SA processor, and is included here to show the strength of the FPGA stream processor approach to cope with any processing need.

The proposed scheme for the sensitivity computation is depicted in Fig. 6. The instantaneous value of the dithering signal should be obtained from the dithering generator, in charge of generating both x and y dithering signals and delivering them to the fast steering mirror. The module will accumulate separately both numerators and denominator leaving the division to be done by the housekeeping computer. Only one denominator will be needed whereas a different numerator will be accumulated for each pixel on the subaperture. Some reset signal (not drawn) will also be required for the start of the accumulations. Again, an equivalent module will be needed for the Y-axis sensitivity.

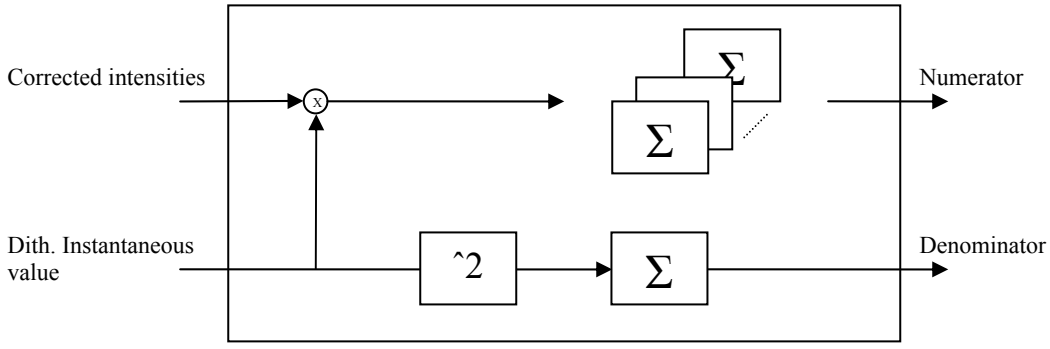

Fig. 6. Concept of the sensitivity computation module.

This module will require the storage of the intermediate accumulators, one per pixel and per sensitivity, estimating the need of 4 Byte per accumulator.

4.5 Reference Intensity Vector computation

The computation reference intensity vector requires a different implementation approach, because it shall be initialized by the housekeeping processor based on the available information regarding sodium profile, point spread function of the SA, laser beam elongation, etc., but should also be updated in real time using every new readout data available. The updating is computed as

$$i_0(k) = \alpha * i_0(k-1) + (1-\alpha) * i(k)$$

Being $i_0(k)$ the reference vector, i(k) the SA corrected intensities, and α the smoothing coefficient. The computation requires then to access to the previous value, the capability of being initialized, and also some way to be read out by the housekeeper computer whenever is to be used. The proposed approach to implement all these features is depicted in Fig. 7, where again a dual ported memory is used to store the main information, with one of its ports devoted to the communication with the higher level processor for initializing and reading out, and the other port being used for reading the previous value and storing the updated value. Some read/write arbitration is required to make compatible both actions and independent of the pixel stream timing, with the help of two FIFO memories.

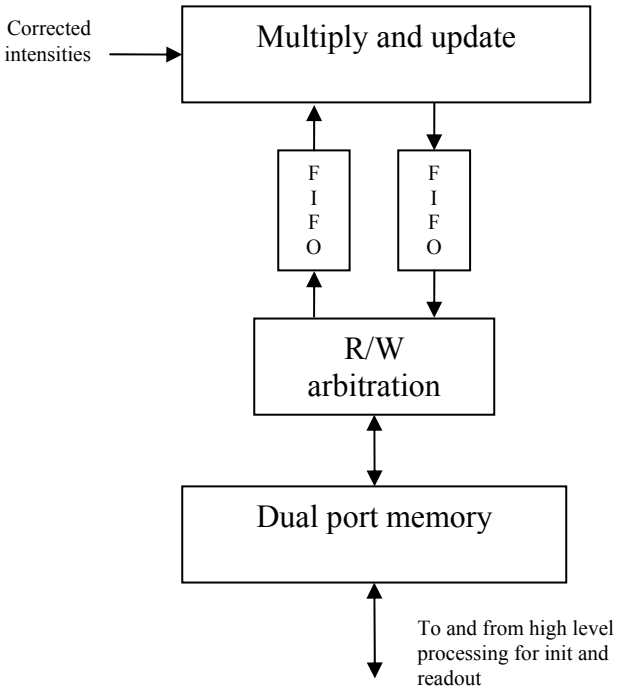

Fig. 7. Reference intensity vector computation module concept.

The storage needs for this module is again related with the overall number of pixels of the LGS WFS detector, being estimated 3 Bytes per pixel in this case, understanding that some intermediate number between the raw detector output (16 bits) and the maximum precision possible obtainable from the calibration module (32 bits).

4.6 Subaperture validation

This module is not requiring any special storage capacity because its only function is to discard the SA slope measurement if the computed intensity does not rise above some threshold, using instead the last valid value. This task can be easy done needing only to store the valid previous value of the SA being analyzed, and using appropriately the information output of the SA intensity computation module.

4.7 Subaperture masking and reference substraction

Masking is intended to discard all subapertures not adequately illuminated due to the central obscuration. This information and reference data will be delivered to this module by the housekeeper computer at a lower speed,

depending on the telescope pointing status, partially illuminated subapertures, etc. Only a few pieces of information per SA will be needed, so no special memory storage needs will be accounted for.

5. ASSESSMENT OF THE RELEVANT FIGURES OF THE DESIGN

The proposed conceptual design will be evaluated in this chapter against a number of relevant parameters, in order to assess its adequacy to the problem to be solved. Three special cases will be analyzed whenever convenient: 64x64, 128x128 and 256x256 subapertures. Only the final conclusions will be depicted, with a very short description of the underlying rationale, because the details of estimation is far beyond the available space for this article.

5.1 Memory

The memory requirements has already been separately evaluated for each module, provided this is a very important issue to be managed when designing an FPGA-based system. Most information related to pixels should be stored in external memory, at least when dealing with today's technology, due to the elevated number of pixels involved. In all cases the resulting estimation comes from the overall number of pixels (10^5 -10^6), divided by the number of detector outputs (8-16) and multiplied by the number of bytes needed for an adequate storage of the information. Some intermediate figure has been chosen as a combination of all possibilities, and only modules needing relevant memory quantities have been analyzed.

Module	Per pixel memory	Estimated memory
Calibration	4 Bytes	1 Mbyte
Matched filter	4 Bytes	1 Mbyte
Dithering	8 Bytes	2 Mbyte
Reference intensity comp.	3 Bytes	0.75 Mbyte

5.2 FPGA resources

Assessment of the number of FPGA slices needed is rather difficult until the exact design and algorithms are fixed and clearly stated, but some estimation follows. Again the proposed concept establishes the SA to be the fine grain to be repeated for parallelism, so as many SA processors as they are in a row should be present. They all will be reused for every new row, because rows are distant in time long enough to allow for this arrangement. A figure of 5%, 10% and 20% extra slices will be added due to the expected managing needs for the cases of 64x64, 128x128 and 256x256 SAs.

Module	slices (per SA)	slices (64x64 SA)	slices (128x128 SA)	slices (256x256 SA)
Calibration	12	807	1690	3687
SA intensity comp.	11	740	1549	3380
Matched filter	25	1680	3520	7680
Dithering	49	3293	6900	15053
Reference intensity comp.	40	2688	5632	12288
SA validation	2	135	282	615
Masking +Ref. subtraction	8	538	1127	2458
TOTAL	**147**	**9881**	**20700**	**45161**

Despite of the required slices to implement the overall logic involved in the system, another important resource is the available multiply/accumulator logic which will be used when doing such operations, and will thus be needed only in selected modules. An estimation of the required amount of them is summarized below.

Module	DSP48 (per SA)	DSP48 (64x64 SA)	DSP48 (128x128 SA)	DSP48 (256x256 SA)
Calibration	1	64	128	256
SA intensity comp.	-	-	-	-
Matched filter	4	256	512	1024
Dithering	8	512	1024	2048
Reference intensity comp.	8	512	1024	2048
SA validation	-	-	-	-
Masking +Ref. subtraction	-	-	-	-
TOTAL	**21**	**1344**	**2688**	**5376**

5.3 Internal FPGA memory

The proposed concept uses FIFO memories to ease the arbitration of the different modules regarding the input and output of information. These memories should be internal to the FPGA and implemented using suitable pre-designed modules like BRAMs, built in the FPGA silicon. The module can only be used as whole, so it is a minimum building block to be accounted for. The estimation of the needed BRAMs will follow, using again today's technology.

The number of subapertures is now relevant because the processing grain has been related to them, so as many modules as SA columns will be needed.

Module	BRAMs (per SA)	BRAMs (64x64 SA)	BRAMs (128x128 SA)	BRAMs (256x256 SA)
Calibration	1	64	128	256
Matched filter	1	64	128	256
Reference intensity comp.	2	128	256	512
TOTAL	4	256	512	1024

5.4 Latency

Latency will be defined here as the time elapsed from the instant when the last pixel is readout to the output of the last computed slope, provided that every detector output will be handled in parallel. An estimation of the number of FPGA clock cycles needed will follow, together with the equivalent time for today's technology. Only the modules directly involved with the slope calculation will be considered.

The latency evaluation has been made from a conservative point of view, using registered logic and allowing two cycles for input/output between modules. Adding and multiplying operations has been evaluated to one cycle minimum with extra cycles depending on the estimated width of the binary number to work with, and an overall margin of 50% has

been added for handling finite state machines intermediate states. Finally a 150 MHz clock has been used as a more than reasonable clock speed to be used even in today's technology, in order to evaluate latency in absolute time units.

Module	Latency (cycles)	Latency (ns)
Calibration	4	27
Matched filter	6	40
SA validation	4	40
Masking +Ref. subtraction	4	40
50% extra margin	9	60
TOTAL	**27**	**180**

The obtained latency figure is extremely small in comparison with the integration period, four orders of magnitude, and is clearly one of the interesting features of this concept.

5.5 PCB physical space

Just in order to provide a rough estimation of the physical size of the resulting slope computer, an estimation of the board space needed to accommodate the required elements is described below. The estimation will again be assessed for the three configurations under study, after the previously estimated data of FPGA resources and memory, and using today's available off-the-shelf technology. A Xilinx VIRTEX-5 SX 240T has been chosen as a good candidate, using from two to six units depending on the configuration. A 25% of the accounted PCB space has been added as a margin to allow for reasonable routing an placing of the components.

	64x64 SA		**128x128 SA**		**256x256 SA**	
component	**#**	**Size (mm^2)**	**#**	**Size (mm^2)**	**#**	**Size (mm^2)**
FPGA Virtex-5 SX 240T (42,5mm x 42,5mm)	2	3612	3	5419	6	10837
Dual port SRAM (17mm x 17 mm)	4	1156		1156		1156
Platform Flash (12mm x 20mm)	6	1440	9	2160	18	4320
Camera output pin (240pin) 12 outputs x 20 pin/output		7680		7680		7680
Power Regulators		4240		6130		11800
Other connectors		1381		1381		1381
TOTAL (mm^2)		**19659**		**24151**		**37624**
TOTAL + 25 % (mm^2)		**24574**		**30189**		**47030**

The estimated figures clearly show that boards of reasonable (standard) sizes could be developed for the slope processor, because even the biggest 256x256 case could easily fit in a 6U VME-like board size (~80.000 mm^2).

5.6 Power consumption

Evaluation of the expected power consumption for the slope processor is based on the FPGA power, identified to be the key element in the overall power expenditure. The methodology followed is to use the available computer tool which provides a rather accurate estimation after accounting for every important aspect of the FPGA programming (slices, DSP48, clocks,I/0,…). Some extra power has been added as a security margin to accommodate the power consumption of the other chips and components.

	Power (W) (64x64 SA)	**Power (W) (128x128 SA)**	**Power (W) (256x256 SA)**
FPGAs	9	14	24
Other components	20	25	30
TOTAL	**29**	**39**	**54**

6. CONCLUSIONS

The conceptual design of a FPGA-based slope processor for the wavefront sensors of laser guide stars of extremely large telscopes has been presented and evaluated in relation with a number of relevant parameters, clearly showing its feasibility even using today's technology. The main concepts involved are the use of the subaperture as the finest grain for the parallel processing, the need of a different stream processor for every detector output, and the use of the row of subapertures (or equivalent subset) for determining the reuse of processing hardware.

In order to show the strength of the FPGA approach, several non-essential processes have been included within the FPGA concept, to maintain the conceptual integrity of the solution, being capable to cope with every task expected to be performed at the real-time faster speed loop.

REFERENCES

[1] Campbell, H.I., Greenaway, A.H., "Wavefront sensing: From historical roots to the state-of-the-art", Astronomy with High Contrast Imaging III, EAS Publication series, 22, (2006) 165-185.

[2] NFIRAOS RTC Algorithm Description. Thirty meter telescope. January 2008.

[3] L.F. Rodríguez Ramos et al. "FPGA adaptive optics system test bench". Proc. SPIE 5903, 120-128, 2005.

[4] L. F. Rodriguez-Ramos, T. Viera, G. Herrera, J. V. Gigante, F. Gago, and A. Alonso, "Testing FPGAs for real-time control of of adaptive optics in giant telescopes," in *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 6272 July 2006.