Architecture of the MAD Real Time Computer

Enrico Fedrigo[*a], Robert Donaldson[a]
[a]European Southern Observatory

ABSTRACT

The ESO-MAD (MCAO Technological Demonstrator) is a fast track project aimed at demonstrating the maturity of the Multi-Conjugate Adaptive Optics concept through a prototype MCAO instrument that uses only natural guide stars. This prototype features two different wavefront sensing architectures (Shack-Hartmann and Layer-Oriented), two deformable mirrors and one tip/tilt stage.

One of the objectives of MAD is also to explore computing architectures different from the ones adopted so far. MAD-RTC is based on the latest generation general-purpose processors in a parallel architecture that can easily (even though not unexpensively) scale up to accomodate large or very large MCAO systems.

MAD-RTC is a multi-wavefront-sensor multi-algorithm real time computer implemented in a Quad-G4 PPC computing board. It is designed to be a test-bed to study different solutions for the future MCAO systems: it can accept data from multiple CCDs in different configurations, use different reconstruction and control algorithms and drive multiple mirrors simultaneously.

Keywords: AO, MCAO, RTC, Real-time, Control

1. INTRODUCTION

The ESO Multi-Conjugate Adaptive Optics Demonstrator (MAD) has been identified as an essential component to demonstrate on sky MCAO techniques, for both the OWL 100-m telescope concept and the ESO VLT 2nd Generation instrumentation. MAD is intended to be a Multi-Conjugate Adaptive Optics (MCAO) prototype to be tested at the VLT with the aim of testing various MCAO techniques.

As every AO control system, the MCAO demonstrator comprises a sensing part (wavefront sensor), a controlling part (the real time computer), and an actuating part (the deformable mirrors). The actuating part is almost identical in all the different MCAO techniques. The MCAO demonstrator has 2 deformable mirrors with 60 actuators each, conjugated at different altitudes (0, ground, and 9.2 Km, high-altitude layer). The other components can vary, so that different combinations are possible, depending on the architecture of the sensor and the technique used for reconstruction.

1.1. Multiple Star Wave front Sensing Architectures

MAD uses data coming from several stars. Multiple-Star wave front sensing can be performed in two main ways:

- One can point a sensor at a single target and record the atmospheric disturbance seen through the column of atmosphere around that target. Using many of these sensors at once creates a multiple star wavefront sensor. We call this architecture **Star-Oriented Wavefront Sensing**.
- Alternatively, one can conjugate detectors to a specific set of altitudes, by imaging the turbulence present at a certain altitude. We call this architecture **Layer-Oriented Wavefront Sensing**.

Star-Oriented Wave Front Sensors (WFS) can be implemented with any AO sensor. In the MCAO demonstrator this is implemented using Shack-Hartman WFSs. The operation of the Layer-Oriented Wave front sensing needs the co-addition of the information coming from multiple sources in order to emphasize the information related to the desired altitude. This can be implemented in two different ways:

- **Optically**: with an optical setup one can add-up pupils coming from different targets creating a bright ground-conjugated pupil and meta-pupils at altitudes different from the ground. Placing a single detector at each altitude of interest completes the sensor.
- **Numerically**: taking the data coming from each target in the Star-Oriented WFS and arranging the data in order to numerically co-add the light at the different altitudes. This can be seen as a Virtual Layer-Oriented wave front sensing.

[*] enrico.fedrigo@eso.org; phone +49.89.32006324; fax +49.89.3202362; http://www.eso.org; European Southern Observatory; Karl-Schwarzschild-Str. 2; D-85748 Garching bei München; Germany

Optical Layer-Oriented can be implemented with any sensor that has the detector conjugated with the pupil, to take advantage from the light co-addition, like the pyramid or curvature WFS. In the MAD this is implemented using the Pyramid WFS because of its advantages due to the simplicity of the implementation.

1.2. Reconstruction Techniques

Given the data arranged either in Layer-Oriented or Star-Oriented wave front sensing architecture it is possible to reconstruct the wave front in two different ways:

- **Globally**: taking all the data coming from all the WFSs at once and reconstructing globally the wave front at the altitude of interest. We call this technique **Global Reconstruction**.
- **Locally**: taking the data coming from a single altitude and driving directly the mirror conjugated to that altitude. We call this technique **Local Reconstruction**.

1.3. Control Strategies

It is possible to combine the reconstruction techniques and the wave front sensing architectures in several control structures. The importance of the MCAO Demonstrator Real Time Computer is that it enables all these modes in order to explore all the possibilities and to test them on the sky.
MCAO is achieved by coupling one wave front sensing architecture with one reconstruction technique leading to the four different control strategies represented with the four arrows in Figure 1.
In this article we present the conceptual design of the MAD Real Time Computer (RTC) that aims at implementing all four techniques, though with more emphasis to the pure Star-Oriented MCAO and pure Layer-Oriented MCAO.

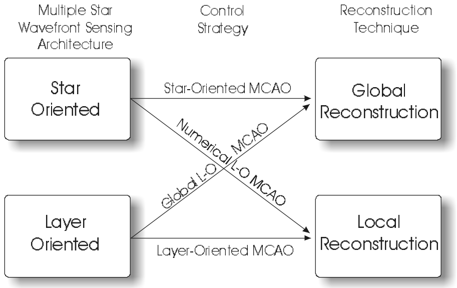

Figure 1: Control Strategies for MCAO

2. ARCHITECTURE

2.1. Hardware Architecture

The diagram in Figure 3 gives an overview of the hardware components included in the MAD RTC subsystem:

- **Control Workstation**: a standard ESO instrument workstation (WS) is used to interface to the Local Control Unit (LCU) and to host the WS components of the MAD RTC SW.
- **Supervisory Computer (LCU):** the LCU is a VME rack-mountable PowerPC (PPC) 604, running at 400 MHz. It has a RS232 serial link going to the High Voltage Amplifier for housekeeping operations. As part of the standard configuration it has a 10FL network port to connect to the VLT/LAN.
- **Real Time Computer (RTC):** the Real-Time Computer is the Dy4 CHAMP-AV board, a Quad-G4 board that mounts 4 PowerPC 7410 running at 500 MHz. Its architecture is shown in **Error! Reference source not found.**. The board is made of 2 independent units each of which includes 2 G4 CPUs, a 128 MB bank of memory and one PMC site for I/O expansion

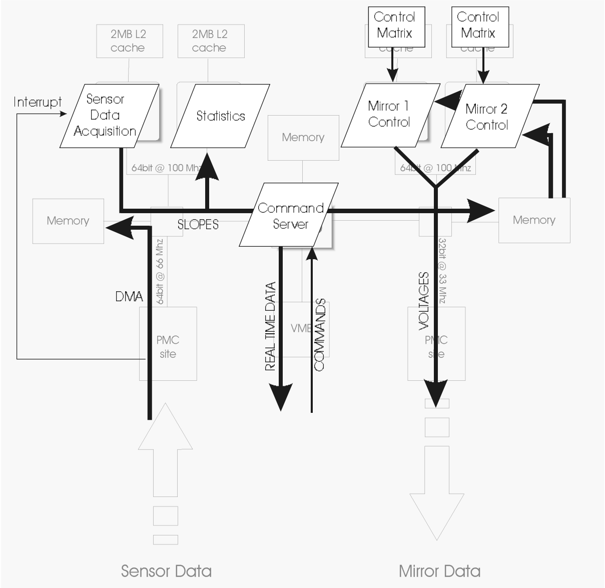

Figure 2: Quad-G4 board architecture

boards. A third section of the board includes a fifth processor with its independent bank of memory that manages the VME bus. A pair of bus controllers connects the first two sections to the third one. The board is mountable on a VME rack. The fastest PMC site is equipped with a digital I/O board (HPDI32 by General Standards) that reads the sensor data. The other PMC site is with a fibre link board (FibreXtreme SL100 by Systran) that sends voltages to the HVA.

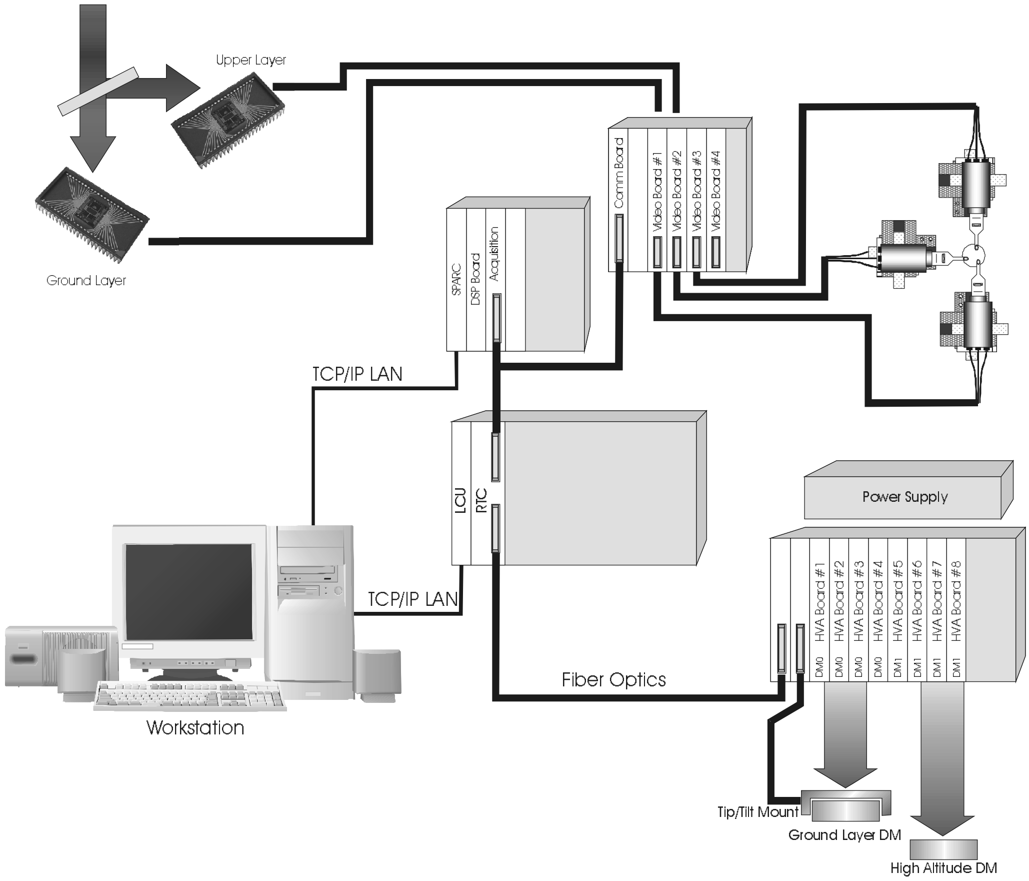

Figure 3: MAD RTC Hardware Architecture

This architecture is particularly suitable to our application which is consists of two main computational blocks plus a data management block: one half of the board is dedicated to acquisition and statistics while the other half is dedicated to control. One processor on the first side is dedicated only to acquisition: the I/O board on the fast PMC site #1 will read data from the CCD controller and write the pixels directly into memory through DMA. After a programmable number of bytes (function of the current mode) the board will send an interrupt to the first processor and then one chunk of the acquisition will start. At the end, Processor #1 will send the resulting slopes (see later for their definition) to the other side of the board waking up the other 2 processors (#3 and #4) for the control part. These two will read the slopes from their memory, get a copy of the control matrix from the cache and produce the final voltages. One of the two processors (#4) will gather the voltages and send them through the fibre card on the PCM site #2. Meanwhile processor #2 is monitoring the data produced by the other processors to produce statistics on them.

The central processor will run the command server that will wait for commands coming from the LCU through the VME bus. It is also the gateway for the processor #3 and #4 to send the real time data to the LCU for display.

The design of MAD-RTC is based on the RTC developed for VLTI, SINFONI and CRIRES projects. Some modules from MACAO-RTC will be reused and some others will be adapted to run in the MAD-RTC environment.

One important new development is the module that manages the real time communications within the processors in the Quad-G4 board (synchronisation and data transfer). New developments and module re-engineering will also have the target of creating the foundation for the Adaptive Optics Real Time Computer Platform, a platform we intend to develop to support AO real time computers for the future VLT instruments.

2.2. Software Architecture

The MAD RTC software is divided into three blocks as shown in Figure 4, each running on a different board. The connection between MACAO-RTC and MACAO-LCU is VME based. The connection between MACAO-WS and MACAO-LCU is TCP/IP based.

Real Time Computer: the Real Time Computer is designed and implemented using an object oriented approach as a collection of interacting classes. Some classes model the devices that are attached to the RTC. It includes the 'Sensor' object responsible for reading data from the CCDs through the digital link and producing the wave front slopes. It supports two sensing architectures. The 'Mirror' object, in charge of applying the control values to the mirrors and checking for over-voltages. The 'Control Loop' object, which is the glue of the system that links the sensor data to the mirror commands through the controller. The 'Data stream' client is responsible for sending the real-time data produced by the control loop to the LCU where they can be distributed to other clients. Figure 5 shows a simplified diagram of the RTC. The white boxes indicate new modules that have to be developed specifically for MAD-RTC,

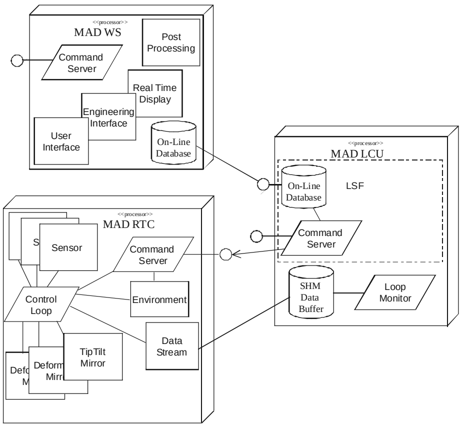

Figure 4: MAD RTC SW Architecture

Local Control Unit: the Local Control Unit is based on LSF (LCU Server Framework), a module of the VLT software. It is designed and implemented using an object oriented approach. It also includes non Object Oriented software which is part of the standard VLT software distribution. It includes a 'Command server' responsible for exchanging commands and data with the RTC over the VME bus and with the Workstation over the network; an 'on-line database' where all the system configuration parameters (both static and dynamic) are stored; a 'loop monitor' in charge of computing statistical data and monitoring the operations of the RTC and the 'real-time data flow server' that distribute data produced by the RTC to network clients.

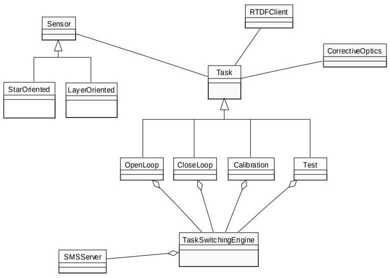

Figure 5: Core System Architecture

Workstation: the Workstation software is implemented using VLT standard software. It provides a programmatic interface (the command server) and a UI interface for maintenance and operations. Final data post processing takes place in the Workstation. It includes a 'Command server' responsible for exchanging commands and data with the MCAO OS over the network; an On-line database that mirrors the relevant parameters of the LCU database; a 'real-time data flow client' in charge of retrieving data from the LCU for display or post-processing.

3. FEATURES

3.1. Generic Real Time Computer Structure

MAD-RTC shares its structure with any AO system. It is made of two blocks: the acquisition and the control module.

- The **Acquisition Module** is in charge of reading the pixels coming from the wave front sensors, flat-fielding them, subtracting the background, descrambling the pixels to distinguish pixels coming from different sensors and the individual subapertures and, finally, computing the wavefront slopes.
- The **Control Module** takes the slopes and turns them into mirror commands, applying the control matrix, the controller and the truncation to ensure the voltage limits are respected. The command computation is usually the most demanding operation. Local Reconstruction operates with a matrix of 60 by 120 in dimension, twice per cycle, while Global Reconstruction operates with a matrix of 120 by 360, once per cycle. Taking the matrix multiplication as a measure of required computing power, Local Reconstruction requires 14400 operations while Global Reconstruction requires 43200 operations, a factor of 3 more. The driver to dimension the Real Time Computer is then the Global Reconstruction technique.

There are two important optimisations in the Real-Time Computer design: pipelining and parallelism. Pipelining is required to exploit the long CCD read-out time and perform part of the computation while the data are not fully acquired yet. This is accomplished partitioning the control matrix in vertical blocks. The resulting control vector is then accumulated at the end of the pipeline and, once ready, the serial part of the computation can start. Another degree of optimisation can be achieved parallelizing this computation using multiple processors. This means partitioning the control matrix in horizontal blocks and distributing each block to a single processor. The data vector must be distributed to all the processors, but the outputs can be computed independently. We dedicate one CPU to control each mirror following the hardware architecture of the demonstrator.

3.2. Star-Oriented Architecture: Acquisition Module

In the case of Star-Oriented wave front sensor Architecture, the Demonstrator will use 3 Shack-Hartmann sensors pointed to three different stars. In Figure 6 the sensor geometry is shown. The chosen sensor geometry is a grid of 8x8 subapertures, for a total of 52 illuminated sub-apertures. Advanced control algorithms might be able to consider also the 8 partially illuminated sub-apertures. With this subdivision, each sub-aperture is sampled by an 8x8 pixel array when the binning factor is 1x1 or by a 4x4 pixel array when the binning factor is 2x2. With this configuration there are no guard rows neither around each sub-aperture nor around the sensor.

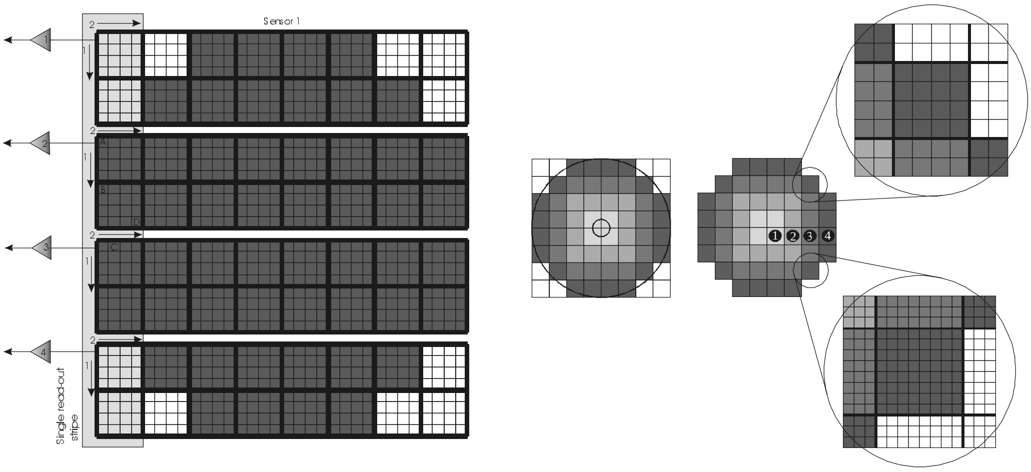

Figure 6: Shack-Hartmann sensor geometry

The two magnified corners of Figure 6 show the pixel arrays under one subaperture and portions of the surrounding ones for both binning cases. In the two cases, the RTC will read:

- 64x64 pixels = 4096 pixels, 1x1 binning, 52 subapertures of 8x8 pixels each, 3328 used
- 32x32 pixels = 1024 pixels, 2x2 binning, 52 subapertures of 4x4 pixels each , 836 used

The 1x1 binning mode includes two different configurations with different read-out speed and, consequently, different read-out noise. Data are read-out from 4 amplifiers located at the end of 4 stripes that segment the CCD. The first operation that the Real-Time Computer has to perform is the data descrambling or pixel reordering. Before the control algorithm can start any sort of computation, at least a single subaperture must be reconstructed, i.e.: all the pixels belonging to that subaperture must be acquired and reordered.

The CCD is read-out through 4 amplifiers each reading 8 or 16 columns of pixels (depending on the binning factor in use, see also Figure 6). Pixels coming from the 4 amplifiers of each of the 3 sensors are multiplexed in one stream and read by the RTC, who reorders the received pixels reconstructing the sub-apertures. To complete one subaperture the RTC has to read an adequate number of rows: this happens when the RTC completes the acquisition of 8 subapertures per sensor. Thus the RTC can compute the slopes for the first 8*3 subapertures and send the results to the control algorithm that can operate in pipeline mode with a factor of 8. Some other operations have to be pipelined: flat fielding and background subtraction have to be performed during pixel acquisition. Moreover, the RTC can accumulate the intensity and the quarters of subaperture that will be used for the final slope computation. We can evaluate the amount of time required for the various operations considering performance figures derived from the board and CPU data sheet. The operations are as follows:

Sensor read-out: The three sensors are read using a digital I/O card that can write data directly into RAM using DMA. Interrupts can be programmed after a selectable number of bytes have been acquired (i.e.: the minimum set of complete sub-apertures equivalent to 3840 bytes or 1152 bytes).

Flat fielding and background subtraction: Flat fielding is done on each useful pixel, i.e.: each pixel under an active sub-aperture. It includes an integer to float conversion and two float operations: *p(x,y) = p(x,y)*flat(x,y)-bg(x,y)* where *p(x,y)* is one pixel, *flat(x,y)* is the flat map and *bg(x,y)* is the background. Floating point single-precision operations (32 bits) in the G4 PPC needs 3 cycles to be completed. We consider here the time required to complete the operations using the standard mathematical co-processor. Better results can be achieved using the AltiVec engine.

Slope Computation: Each subaperture produces two values which are the X and Y slopes of the wavefront. At this point the data type used for the pixels is a floating point number, so all the subsequent operations are done in the same domain. Several strategies are possible but the most convenient one is to accumulate four variables named A, B, C and D, representing each one quarter of the sensor. Then slopes are simply:

$$s_x = \frac{A+C-B-D}{I} - R_x \text{ and } s_y = \frac{A+B-C-D}{I} - R_y \text{ where } I = A+B+C+D \qquad (1)$$

With 1x1 binning, accumulating each quarter takes 16 operations while the final computation takes 6 operations per slope direction. Computing I takes 4 operations. Assuming that a floating point operation is equivalent to 3 integer operations that are done in one CPU cycle, so this strategy takes 240 CPU cycles (96 for the 2x2 case). These operations are distributed in 192 (48) during pixel acquisition and the remaining 48 (also for 2x2 binning) when the last pixel has been acquired. Since each pixel belongs only to one quadrant, only one accumulation operation has to be done with each pixel (6ns). While the RTC reads the pixels, it equalises them and at the same time it accumulates the values A,B,C,D for the slope computation. To accomplish that the RTC has also to read the flat field map, the background map and the pixel map, that tells the RTC which pixel has to be considered and to which quadrant of a certain subaperture it belongs to. Those maps could be placed in cache memory to get better performance, but here we consider storing them in normal memory. In summary, to process the data coming from one entire CCD in case of 1x1 binning:

Operation	Type	Time	Data set	B
Sensor read-out	DMA write	54 µs	4096+1024 pixels	2
Memory to CPU: read pixels from memory for processing	RAM read	34 µs	4096+1024 pixels	2
Memory to CPU: read flat field map	RAM read	56 µs	3328 pixels	4
Memory to CPU: read background map	RAM read	56 µs	3328 pixels	4
Memory to CPU: read pixel map	RAM read	34 µs	4096+1024 pixels	2
A,B,C,D read	RAM read	56 µs	3328 pixels	4
Equalisation, A,B,C,D accumulation	CPU	80 µs		
A,B,C,D write	RAM write	52 µs	3328 pixels	4
(pixel write, for debugging purpose)	RAM write	52 µs	3328 pixels	4
Final ABCD read	RAM read	4 µs	52*4 values	4
Final Slope computation	CPU	5 µs		
Slope write	RAM write	2 µs	52*2 values	4
Slope transfer to other CPU	BUS write	5 µs	52*2 values	4
Total		**490 µs**		

The acquisition time for all three sensors is then 1.47ms, to be compared with the CCD read-out time which is 4.6ms and 2.3ms in the two available modes with 1x1 binning. To exploit the parallelism of the board, the acquisition procedure is organized with the aim of delivering data as soon as possible to the control module, i.e.: as soon as the first sub aperture slopes are computed. When the first is complete, other 7 are completed as well (one stripe) and the same happens to the other 2 sensors. The acquisition module delivers 24*2 slopes to the control algorithm 8 times per frame.

When the RTC is reading one stripe from the sensor, it can perform only flat fielding, background subtraction and A,B,C,D accumulation. Once the stripe is entirely loaded, it can compute I and the slopes and then send the slopes to another CPU. These operations can be performed while the RTC is acquiring the second stripe.

Since the RTC time is smaller that the CCD read-out time, the RTC can complete the computation of the slopes during the acquisition of the following stripe: the acquisition can be fully pipelined. So approximately after 184 (133) µs since the last pixel of the first stripe has been acquired, another CPU can start computing the control vectors and complete half of the control algorithm during the acquisition of the second stripe. The time spent computing slopes for the last stripe will, of course, add up to the total time lag.

3.3. Layer-Oriented Architecture: Acquisition Module

In the case of the Layer-Oriented Wavefront Sensor Architecture the Demonstrator will use a sensor equipped with a variable number of pyramids (up to 8) that will send the light coming from different stars into two detectors conjugated at different altitudes, ground and 9.2 Km.

In this architecture each sensor is conjugated to a single mirror, each with the same geometry, same number of actuators (60), but different size. The sensor configurations (ground layer and upper layer) also change to match the different characteristics of the atmosphere at the two conjugation altitudes:

- Ground Layer: 2x2 binning, 32x32 pixels resolution, 1024 pixels, 1.5ms frame time, 8x8 sub-apertures
- Upper Layer: 4x4 binning, 16x16 pixels resolution, 256 pixels, 2.5ms frame time, 7x7 sub-apertures

The two detectors can be read-out at different frame frequencies but, roughly, multiple by 1, 2 or 4. The detector conjugated with the upper layer is the one that can be read-out at a lower frame frequency. Pixel read-out rate remains in any case constant: the frequency reduction is obtained skipping some frames on one detector to achieve a longer integration time. As in the previous case, data are read-out from 4 amplifiers located at the end of 4 stripes that segment the CCD. In the two cases the stripes have 8x32 pixels or 4x16 pixels. The amplifiers read the pixels from the smaller dimension first (vertically in the following pictures) assembling lines of pixels. Each line is pre-pended with 2 pre-scan pixels and 1 or 2 pipeline pixels, 2 in 2x2 binning and 1 in the 4x4 binning. Data descrambling or pixel reordering for this case is different from the previous case and also different for the two detectors. Before the control algorithm can start any sort of computation, at least a single subaperture must be reconstructed, i.e.: all the pixels belonging to that subaperture must be acquired and reordered. The pyramid sensor creates four pupils each containing information of one quarter of the total wavefront. To give an example, to compute the global tilt one has to compute the total flux present on each of the 4 pupils that we can name {A, B, C, D} and then use the well known equation:

$$tilt = \frac{A + B - (C + D)}{A + B + C + D} \qquad (2)$$

The same applies to the local tip and tilt across the pupil. Taking 4 pixels each from one pupil in the same relative position one can compute the local tip and tilt. These 4 pixels form a *virtual* quad-cell configuration.

The RTC has also to subtract the background and to flatten the field, operations performed during the pixel acquisition. We can evaluate the time required for the various operations using the same assumptions used in the previous section. The operations are as follows:

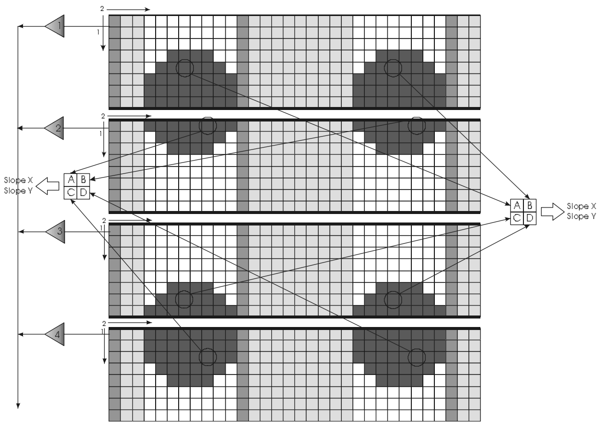

Figure 7: Layer-Oriented wavefront sensor read-out; sub-apertures on the ground layer

Sensor Read-out: pixels are read in DMA transfer mode and stored into local RAM. Since to reconstruct a single virtual quad-cell at least half sensor must be acquired (see example on Figure 7), it is not convenient to organize the computation in pipeline. Instead, the whole frame is acquired and then the computation starts. Binning 2x2 correspond to the ground layer detector while binning 4x4 correspond to the upper layer detector.

Flat fielding and background subtraction: same operation described in the previous architecture, involving a different number of pixels.

Slope Computation for the Ground layer: There are four pupils on the detector with the contributions of the various pyramids all overlapping perfectly, so only four illuminated circles are present. The central obstruction does not obscure any entire pixel. Comparing the specifications of the optical design with the size of the detector, which is a square of 1.536 mm for each dimension, with a 2x2 binning we have a 8x8 geometry with 52 active pixels. Figure 7 defines the active sub-apertures and splits the CCD in four stripes that are read using the four independent channels. The number of active pixels is small compared to the number of pixels on the detector: only 52*4 = 208 out of 1024, so the 20% of the detector. The CCD read-out can be optimized to skip blocks of non used pixels reducing the read-out time by 60%. So the frame time can be reduced to 920 µs.

Figure 7 shows also how to assemble two virtual quad-cell, as an example. To form the right one the RTC must read 4 lines from the amplifier 1 and 4 lines from amplifier 3 plus additional 8 lines from amplifier 1 and another 8 lines from amplifier 3. During the acquisition of the first 8 lines (i.e.: half of the detector) no subaperture is assembled. The first complete virtual quad cell will be available when the RTC read the 9th line and then each additional line completes more subapertures. Once the pixels are acquired and equalized, the slope computation can proceed directly computing I and the two slopes as in Equation 1 where A, B, C and D belong each to one pupil. This corresponds to 48 CPU cycles.

Slope Computation for the Upper layer: The situation is slightly different for the high altitude layer. Figure 8 shows the geometry of the pupils in the high altitude layer in a configuration with 5 stars, one located at the centre and the other 4 located at the edge of the 2’ field-of-view (in diameter) and equally spaced. The light coming from different directions will not completely overlap as in the previous case. It is possible to define an envelope that contains all possible star footprints and this is the so-called meta-pupil. In Figure 8 the dimensions derived from the optical design are reported and it is easy to see that a 4x4 binning produces 45 useful channels with a 7x7 geometry. Depending on the star configuration, some subapertures may not be well illuminated (the ones close to the corners in Figure 8). The RTC needs to know the number and location of the guide stars in order to be able to identify the dark subapertures and avoid using them. The corresponding values must be evaluated extrapolating from the available data. Virtual quad-cells are assembled in the same way as in the previous case. The RTC has to read half of the detector in order to complete the first quad cell and then every new line completes more quad-cells. Still numbers are small and it is not convenient to organize the computation in a pipelined fashion: the RTC can acquire the entire frame and then start operating on it serially. The slopes are computed using the same strategy, so the same number of operations per virtual quad-cell. In summary:

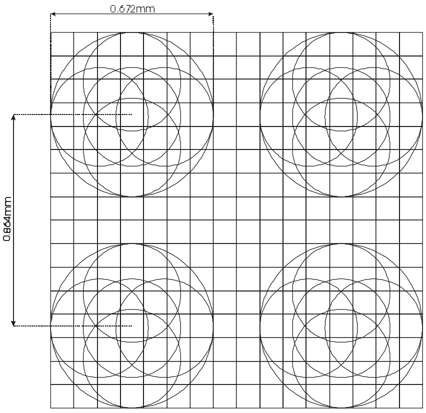

Figure 8: Layer-Oriented wavefront sensor; high altitude layer

Operation	Type	Time	Data set	B
Sensor read-out ground/upper	DMA write	16 µs / 5 µs	1024+512 pixels / 512+192 pixels	2
Memory to CPU: read pixels from memory ground/upper	RAM read	5 µs / 3 µs	628/256 pixels	2
Memory to CPU: read flat field map ground/upper	RAM read	9 µs / 5 µs	628/256 pixels	4
Memory to CPU: read background map ground/upper	RAM read	9 µs / 5 µs	628/256 pixels	4
Memory to CPU: read pixel map ground/upper	RAM read	5 µs / 3 µs	628/256 pixels	2
Equalisation	CPU	4 µs / 3 µs		
(pixel write, for debugging purpose)	RAM write	9 µs / 5 µs	628/256 pixels	4
Slope computation	CPU	5 µs / 5 µs		
Slope write	RAM write	2 µs / 2 µs	52*2 values	4
Slope transfer to other CPU	BUS write	5 µs / 4 µs	52*2 values	4
Total		**69 µs / 40 µs**		

This time budget is structured as a sequence of operations. Since operations are done sequentially they will add up to the total delay. Given the small numbers, it is not convenient to pipeline the acquisition, although it would be an interesting experiment. Moreover the detector is not configured to favour the Layer-Oriented WFS since the virtual quad-cells are completed almost in random order. However it is possible to structure, as an additional experiment, the acquisition as two parallel tasks using two processors and reduce the time delay to, roughly, the biggest of the two (69 µs). Anyway, serializing the two processes we obtain a total delay of 110 µs which is less than the delay of the Star Oriented case. It is worth noting that the two sensors are read synchronously and thus the frame frequency is dictated by the slowest of the two (2.5 ms): the RTC is mainly idle.

3.4. Control Structures

The control architecture is very simple. The system is limited in its ability to control the incoming wave front by the accuracy of the DMs in generating the proper shape to compensate for distortions and the accuracy of the sensors to measure which distortion to compensate for. Thus the best the system can achieve is the intersection of these two spaces. This intersection is determined by the interaction matrix, a linear relation that links the mirror commands with the corresponding wave front measurements.

Ideally, inverting this relation it is possible to apply instantaneously the best correction possible given the constraints mentioned above. Generally it is not possible to invert this matrix. Instead, a Least Square Estimate (LSE) is used in order to compute the best approximation for the mirror configuration given a certain measurement.

The control system we are building is called a tracking system, that is we want that the system follows the input generating a signal equal in module but opposite in sign in order to have a perfectly compensated wavefront. The simplest controller that achieves this goal is the pure integrator. The core of the system is the interaction matrix that we will now analyze for the main two strategies.

3.5. Star-Oriented Architecture: Control Module

Star-Oriented MCAO uses different sensors looking at different directions to get measurements of the turbulence from different *points of view*. These measurements are used to reconstruct the turbulence at the altitudes where the deformable mirrors are conjugated. The interaction matrix is computed using the reverse procedure: the sensors are looking at artificial targets and the mirror actuators are moved one by one while the corresponding measurements are placed in the columns of the matrix. The result is a function of the geometry of the guide stars. This means that the interaction matrix, or at least one half of it, must be computed for every configuration of MCAO guide stars, i.e.: every observation must be carried out with a different interaction matrix. Finding a way to synthesize the interaction matrix for any guide star configuration starting from an initial fixed set of configuration is an interesting topic for investigation. The control loop is implemented multiplying the sensor measures with the pseudo-inverse of the interaction matrix (i.e.: the control matrix) computing the LSE of the best fit for the mirror configuration based on the sensor measurements.

Looking at the structure of the interaction matrix one can see that in the partition of the matrix that represent the upper layer many zeros will be present, corresponding to the fact that actuators far from the pupil on a certain direction will not see any signal. Actuators that are not covered by any pupil of the guide stars in a particular configuration will not produce signal on all the sensors, thus the interaction matrix will also have columns with only zeros in the upper layer partition leaving certain actuators without control. This can be improved using modal control.

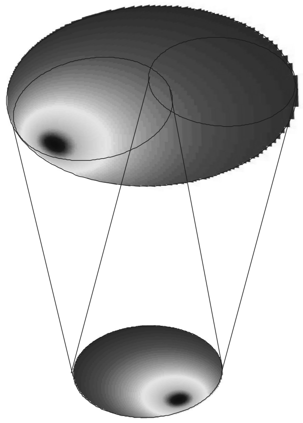

Figure 9: Mirror influence functions seen from different directions

Modal Control

The turbulence can be decomposed in Zernike polynomials on the full meta-pupil and what the sensors are doing is only measuring a portion of this area. The system should then try to estimate the Zernike polynomials on the whole meta-pupil using only the data coming from the illuminated area. This can be done in several steps: first, a set of voltages that generates the Zernike polynomials on the mirror surface we want to control must be computed. Typically, the 60-channel bimorph mirror used for the demonstrator will be able to reproduce around 35-40 modes. Then we apply the corresponding voltages to each mirror, individually. Once the sensors are properly oriented, we record the vector of sensor measurements and we assemble the interaction matrix putting these vectors in the matrix columns. For the demonstrator we will obtain a 312*80 matrix, where 312 = 52*2*3 are the sensor measurements and 80=40*2 are the Zernike modes we want to correct for, 40 for each mirror. Since Zernike modes change the shape of the mirror globally, each mode will produce effects on each sensor, so this interaction matrix is full. Inverting it by means

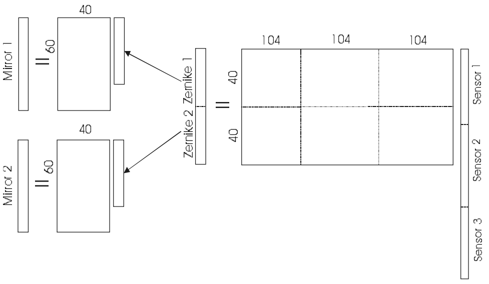

Figure 10: Operations for modal control

of the pseudo-inverse we obtain the control matrix. This matrix estimates the Zernike decomposition of the turbulence at the two altitudes of the deformable mirrors based on the sensor measurements using LSE. Then the voltage must be computed through the reconstruction matrix that maps voltages to Zernikes. There will be one matrix per mirror and each matrix will be 40*60 in size, where 40 is the number of modes and 60 the number of actuators.

Of course it is possible to combine the two operations in only one single matrix multiplication pre-multiplying the two halves of the control matrix by the two reconstruction matrices obtaining a unique large control matrix (120*312). The convenience between the two is dictated by the number of Zernike polynomials to use for the reconstruction. The unit operation in a matrix-vector multiplication is the multiply-and-add. Since there are as many of those operations as the number of elements in the matrix, in the first case we will have 29760 operations and in the second case simply 37440.

Mixed algorithms are possible: the ground layer can be controlled in the zonal approach and the high-altitude layer in the modal approach simply computing the interaction matrix for the corresponding layer in the proper way. The size of the different partitions in the interaction matrix will change accordingly. In case the direct modal projection is computationally convenient (less than 45 modes) we have the opportunity to time-filter the modes for free.

We now evaluate the total time required to complete the computation of the voltages when the slopes are all available, without any optimization or pipelining. We can use an AltiVec-optimized mathematical library. The basic routine is **`vdot`** that computes the dot (inner) product of two vectors. To compute a matrix-vector multiplication with **`vdot`** we have to invoke it with each row of the matrix and the sensor data for every mirror electrode. We then apply a simple integrator that multiply a constant value (the gain) against all the elements in the resulting vector and then accumulate those values in another vector. The basic routines are **`vadd`**, which adds together two vectors to produce a third one, and **`vmultk`**, which multiply a vector by a scalar. Sensor data are 52*2 slopes times 3 detectors. Results are summarized on Table 1. Since this estimation is optimistic, there is the need to optimize this part splitting the computation between two processors and pipelining it. We saw in the previous sections that slopes can be computed in an 8-stage pipeline. Ideally this time could be reduced to 239/2/8 = 15 µs, but of course this does not take into account the additional complexity of the pipeline logic and the required synchronization when using true parallel computing.

Operations	Time
Read data	5.3 µs
Matrix read	139 µs
Matrix computation	89 µs
Controller	0.3 µs
Convert to short[1]	1.5 µs
Write data	3.6 µs
Total	**238.7 µs**

Table 1: Star-Oriented Time Budget

3.6. Layer Oriented Architecture: Control Module

Layer Oriented MCAO with local reconstruction uses different sensors each looking at a different altitude. Measurements taken at a certain altitude are used locally to drive the mirror that compensates the turbulence at that altitude. This means that in Layer Oriented MCAO there are several loops that correct different regions of the turbulence. These loops are not independent since they are optically coupled, but as far as the control is concerned, they are independent and modular. Each loop has an interaction matrix which is recorded similarly to the previous case.

The ground layer interaction matrix is recorded illuminating the ground layer pupil from any direction since any direction produce the same illumination on the pupil. The interaction matrix is assembled moving the actuators one by one and recording the sensor measurements by column. The control loop is implemented multiplying the sensor measures with the control matrix (pseudo-inverse of the interaction matrix) computing the LSE of the best fit for the mirror configuration based on the sensor measurements.

On the high altitude layer the picture changes slightly. As shown in Figure 11, a single source can not illuminate the whole meta-pupil, but only a smaller portion. It is then necessary either to use multiple sources in order to reach the full coverage of the meta-pupil and use the same strategy used for the ground layer, or to point a single source towards a single sensor sub-aperture and record the signal coming from this sub-aperture moving all the mirror actuators one by one. Then the source is moved to point to another sensor sub-aperture and the procedure is repeated. The interaction matrix is built by columns in the former case, by rows in the latter. It is then inverted as usual. As shown in Figure 11 there are guide star configurations that do not guarantee that the upper layer is completely illuminated and sensed. To avoid running the control loop on the sensor noise, a modal approach should be considered.

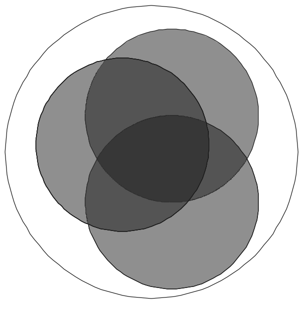

Figure 11: Pupils overlapping on the upper layer

Modal Control

We proceed using the same approach as described in the previous sections selecting the number of Zernike polynomials we want to correct for (around 35-40 for our bimorph mirror) and then we apply the corresponding voltages to each mirror. As in the previous case, the interaction matrix can be computed by columns or by rows, depending on the number of calibration sources and the recording strategy. The last step is the computation of the control matrix that has to be performed for each individual observation since this step requires some additional knowledge: based on the geometry of the guide stars, the non-illuminated sensor sub-apertures have to be determined. The corresponding rows on the interaction matrix should be zeroed in order to cancel the contribution of these sensor sub-apertures.

The modified interaction matrix is turned into a control matrix by means of a pseudo-inverse and the columns corresponding to the non-illuminated sub-apertures will contain zero. The control loop now works in the following way: the measures are multiplied against the control matrix and the Zernike estimation is then computed. Then the resulting vector of 35-40 modal coefficients is multiplied with the Zernike reconstruction matrix to obtain the voltages to apply to the whole mirror. This can be optimized removing from the interaction matrix the rows associated to non illuminated sub-apertures, at the price of an increased complexity in rearranging the sensor measure vector to remove the channels that must not be multiplied with the control matrix.

Then the two matrix multiplications can be combined in one, unless some time filtering is required on the modes. The number of unit operations multiply-and-add is 45*2*60 = 5400 or 52*2*60 = 6240 operations for the two layers. Leaving the two stages separate is not very convenient but given the small number of operations, in absolute terms, it could be acceptable to loose 10 to 25% in computing speed to have the flexibility to temporally filter the modes.

Even if modal control is not required for the ground layer, it is possible to apply the same approach also there. All the sub-apertures will always be illuminated and thus the interaction matrix will not shrink. However, given the modularity of the system, it is possible to drive the high-altitude deformable mirror with modal control and the ground layer deformable mirror with zonal control.

We now evaluate the total time required to complete the computation of the voltages when the slopes are all available, without any optimization or pipelining. We use the same assumptions as in the previous sections, using a simple integrator with gain as controller. For the ground layer detector we have 52 active subapertures thus 104 slopes and we compute 60 control values. For the upper layer we have 45 active subapertures thus 90 slopes, same number of control values. The results are summarised on Table 2. Executing both parts on a single processor will cause a total delay of 95 µs.

This is will be the baseline implementation, but a parallel design will be considered as well. This design will reduce the total time delay to the biggest of the two (45.1 µs), without taking into account the overhead due to the synchronization of the two processors. Pipelining with a factor of 8 can reduce ideally the time delay to 7 µs, but with numbers so small, the overhead of the pipeline organization will be significant.

Operation	Ground Layer	Upper Layer
Read data[1]	1.7 µs	1.5 µs
Matrix read	23 µs	20 µs
Matrix computation	15 µs	13 µs
Controller	0.3 µs	0.3 µs
Convert to short	1.5 µs	1.5 µs
Write data	3.6 µs	3.6 µs
Total	**45.1 µs**	**39.9 µs**

Table 2: Layer Oriented Time Budget

3.7. Global Layer-Oriented MCAO

Global Layer-Oriented MCAO uses the Layer-Oriented wavefront sensor but a different reconstruction strategy. Instead of driving the mirrors using only local information, this strategy uses all the data coming from both detectors at the same time to drive both mirrors. The main difference with Layer-Oriented MCAO is that the interaction matrix is bigger in order to include the contribution of both detectors and their cross-terms. The calibration of the interaction matrix is slightly different: while the RTC is scanning one by one the mirror electrodes (or modes) it will record the signal coming from both detectors instead of recording only the signal from the detector corresponding to the activated mirror. Computationally this mode is closer to the Star Oriented MCAO. We give on Table 3 an estimation of the expected performance, using the same assumption made in the previous sections. Parallelization and/or pipelining may be required.

Operations	Both Layers
Read data	3.3 µs
Matrix read	86 µs
Matrix computation	56 µs
Controller	0.3 µs
Convert to short	1.5 µs
Write data	3.6 µs
Total	**151 µs**

Table 3: Global Layer-Oriented Time Budget

3.8. Numerical Layer-Oriented MCAO

Numerical Layer-Oriented MCAO uses the Star-Oriented Wavefront Sensor to numerically compute what would be the output of a Layer-Oriented Wavefront Sensor. The implementation will reuse the Star-Oriented Wavefront Sensor and the Layer-Oriented Control Strategy entirely, adding a new module in the middle with the task of computing an output compatible with the Layer-Oriented Control Strategy using only Star-Oriented data. At the present time this module is not defined and it is under research as part of the RTN work.

4. VALIDATION TESTS

We recently received the CHAMP-AV board and we are currently installing and configuring it to prepare it to run under our environment. As part of this work we ran some tests to verify if the performance we expect matches the real performance that the hardware delivers. In the following table we report the results: what we found is that the board performs better that expected and our estimates are confirmed.

Test case			Size	Result	Expected
Matrix-vector multiply, Star Oriented case		with CPU	312*120	640 µs	-
		with AltiVec	312*120	192 µs	228 µs
Matrix-vector multiply, Layer Oriented case	Ground layer	with CPU	104*60	97 µs	
		with AltiVec	104*60	32 µs	38µs
	Upper layer	with CPU	90*60	77 µs	-
		with AltiVec	90*60	29 µs	33 µs
Matrix-vector multiply, Global Layer Oriented,		with CPU	194*120	396 µs	-
		with AltiVec	194*120	132 µs	142µs
Pixel acquisition: flat field, background subtraction, ABCD accumulation, with CPU			4096	230 µs	368 µs
			1024	51 µs	92 µs
			256	15 µs	23 µs
Slope computation, subapertures			52 / 45	5.04 / 4.35 µs	10 / 8.9 µs

5. CONCLUSIONS

In this paper we presented the architecture of the MAD Real Time Computer. It is a complete system aimed at supporting different MCAO sensors and control strategies. We provided an estimate of the mainstream computation that the RTC is in charge of, and we proved, through benchmarks, that our estimates were accurate.

We thus proved that this architecture can be implemented within the specifications of the MAD project. The final test on the sky will also prove that this architecture can evolve to the AO-RTC platform we intend to develop.

On the other hand, while the advantages of using the latest generation of multi CPU computing platforms are clear, the costs of using such new technology should not be overlooked. None of the PCI devices used in MAD are currently supported by VxWorks drivers for the CHAMP-AV and the difficulty in porting the drivers from existing single CPU single PCI bus systems to a board hosting 5 CPUs and multiple PCI busses should not be overlooked.

ACKNOWLEDGEMENTS

The authors wish to acknowledge that this research is supported by the European Commission RTC program: “Adaptive Optics for Extremely Large Telescopes”, under contract #HPRN-CT-2000-00147.

REFERENCES

1. Marchetti E, “*Multi Conjugate Adaptive Optics Demonstrator Technical Specifications*”, OWL-SPE-ESO-60000-0024, ESO internal document, 2002
2. Marchetti E, “*Multi Conjugate Adaptive Optics Demonstrator System Overview”*, OWL-SPE-ESO-60000-0027, ESO internal document, 2002
3. Ragazzoni R et al., “*Multi Conjugate Adaptive Optics Demonstrator Layer-Oriented Wave Front Sensor Conceptual Design”*, OWL-SPE-INA-60000-0037, ESO internal document, 2002
4. Rabaud D., “*NAOS Real Time Computer Software Functional Specifications*”, VLT-SPE-NAO-116750-0002, ESO internal document, 1998
5. GEMINI, “*MCAO for Gemini-South*”, Conceptual Design Review Document, GEMINI, 2000