SPARTA for the VLT: status and plans

Marcos Suárez Valles*[a], Enrico Fedrigo[a], Robert H. Donaldson[a], Christian Soenke[a], Stefano Zampieri[a], Reynald Bourtembourg[a], Helmut Tischer[a]
[a]European Southern Observatory, Karl-Schwarzschild-str. 2, 85748, Garching b. München, Germany

ABSTRACT

SPARTA, the ESO Standard Platform for Adaptive optics Real Time Applications, is the high-performance, real-time computing platform serving three

major 2nd generation instruments at the VLT (SPHERE, GALACSI and GRAAL) and possibly a fourth one (ERIS).

SPARTA offers a very modular and fine-grained architecture, which is generic enough to serve a variety of AO systems. It includes the definitions of all the interfaces between those modules and provides libraries and tools for their implementation and testing, as well as a mapping to technologies capable of delivering the required performance. These comprise, amongst others, VXS communication, FPGA-aided wavefront processing, command time filtering and I/O, DSP-based wavefront reconstruction, DDS data distribution and multi-CPU number crunching, most of them innovative with respect to ESO standards in use. A scaled-down version of the platform, namely SPARTA-Light, will employ a subset of the SPARTA technologies to implement the AO modules for the VLT auxiliary telescopes (NAOMI) and is the baseline for a new VLTI instrument (GRAVITY).

For the above instrument portfolio, SPARTA provides also a complete implementation of the AO application, with features customised to each instrument's needs and specific algorithms. In this paper we describe the architecture of SPARTA, its technology choices, functional units and test tools. End-to-end as well as individual module performance data is provided for the XAO system delivered to SPHERE. Initial performance results are presented for the GALACSI and GRAAL systems under development.

Keywords: Adaptive Optics, Real-time Computer, Control Systems

1. INTRODUCTION

A comprehensive presentation of the ESO VLT projects, technological drivers and development/maintenance use cases behind the inception of the SPARTA RTC platform was provided in [1], together with some initial performance results for the only platform instance under development at that time -i.e. the VLT SPHERE XAO system (SAXO) [5,6,7]. Today a SPARTA Real Time Computer (RTC) operating at 1.2kHz with end-to-end latency of ~80µs is the core of SAXO, already delivered to the SPHERE consortium and under regular use for instrument AIT at IPAG [8,9]. In addition, two SPARTA platform instances involving 1kHz, high-order control with 4 laser guide-stars (LGS) for the VLT AOF [10] instruments GRAAL [11] and GALACSI [12] have been deployed to a high degree of completion and are starting to be used for instrument AIT at ESO. Initial measurements performed on the AOF SPARTA RTC configuration suggest an estimated overall latency below the one of SPHERE.

Both the SPHERE and AOF AO systems make use of high-order, Shack-Hartmann wavefront sensors (WFS) containing 1.240 used sub-apertures in a 240x240 pixel frame. Neglecting the contribution from the auxiliary/low-order control loops, the above results in a sustained input pixel throughput ranging from ~66Mpixel/s (SPHERE) to ~ 230Mpixels/s (AOF) with corresponding complexities of respectively ~5GMAC and ~12GMAC -i.e. driven by the matrix-vector-multiply (MVM) operation required for wavefront reconstruction. The ability of SPARTA of delivering low latency for such high-throughput systems is the result of a number of design decisions and technological choices (some of them improved since the publication of [1,2]) which will be reviewed in detail here and presented in connection with performance figures derived from actual SPARTA implementations.

*msuarez@eso.org; phone +49 89 32006 0; fax +49 89 3202632; eso.org

Hard real-time performance is however not the only figure of merit applicable to AO systems with the level of complexity of the ones described here. Soft real-time functionalities such as continuous optimization of RTC control parameters, low-frequency offload control loops, modal measurements, loop data recording, atmospheric statistics and system performance monitoring require the propagation of closed-loop telemetry data to number-crunching processes. In most cases, data is required at the loop rate (i.e. respecting time contiguity) and has to be broadcasted to several processes in the same and/or different computing units. This poses stringent requirements on computing power, the clustering of computational units, network design and data propagation middleware. The main SPARTA data distribution design decisions and bottlenecks will be discussed here.

The initial overhead in posing SPARTA as a platform is starting to pay off now: the GRAAL and GALACSI RTC instantiation is intensively reusing platform components which were developed and tested with the focus set on SPHERE. This is true for both software (SW) components (reuse of actual code but also patterns and strategies/interactions) as well as hardware (HW) ones (replication of board setups, firmware blocks and data path layout designs). As a result, it is currently being considered adding a fourth VLT instrument (ERIS [13]) to the SPARTA portfolio with a reduced effort. In addition, the option of implementing a modal control for AOF with little impact in the existing RTC by reusing available platform components is also being evaluated.

Moreover, a careful design of the SPARTA platform SW components used for auxiliary/low-order control loops -i.e. by decoupling the (CPU-based) processing algorithm from the actual data I/O, has enabled the creation of the SPARTA-Light platform targeted to smaller systems with more cost-effective RTC HW. Whereas the SPARTA SW components exchange data by means of high-performance fabrics between computing units, their SPARTA-Light counterparts reuse most of the available code but employ shared-memory I/O within a single CPU. This is the platform of choice for the VLT Auxiliary Telescopes AO system (NAOMI) and the baseline for the VLTI instrument (GRAVITY). A first realization of a SPARTA-Light RTC (NAOMI) WFS with 32 used WFS sub-apertures and ~3,4Mpixel/s input throughput has recently become available. The SPARTA-Light design, performance and limitations will be discussed here in connection with this initial RTC instance.

The above demonstrates the scalability and reusability of the SPARTA platform. However, as the number of instruments in the portfolio grows, the spotlight is shifted towards system maintainability –i.e. a diversity of instruments at different points in their lifecycle making use of the same platform at different versions and not always willing/able to receive SW/HW updates in the short term. Proper tools/procedures for configuration control and SW merging are being introduced in the SPARTA development cycle to cope with this.

As a general rule when tackling the points described above, rather than repeating an exhaustive SPARTA platform description (available in [1,2]), we will focus on key-to-performance architectural aspects and design constraints, as well as on the potential technology roadmap for upgrading the platform design.

2. OVERVIEW

The main component blocks of a SPARTA RTC are shown in Figure 1a. The RTC Box encapsulates a hard real-time, low-latency processing pipeline which implements all the operations required at the loop rate in order to close the inner AO loops. This involves *Wavefront Processing* -i.e. computation of local wavefront gradients from WFS pixel frames, *Reconstruction* -i.e. computation of delta actuator commands from local WFS gradients and *Control* –i.e. time filtering of delta actuator commands, typically involving IIR control (at least integral) plus some form of saturation management. A fully digital AO processing pipeline is provided by the SPARTA platform: well-defined, standard, high-throughput interfaces are specified for the WFS readout electronics and actuators HW, but these functional blocks are intentionally left outside the RTC Box for maximum platform reusability.

A Co-processing Cluster based on multi-CPU, multi-core Linux computers continuously receives loop telemetry data from the RTC Box (e.g. local wavefront gradients and actuator commands at loop rate, sub-sampled WFS pixel frames, etc.) and further processes them to implement soft real-time, outer AO control loop closure (e.g. RTC Box parameters optimization, offloading to external actuators, etc.) and auxiliary tasks (e.g. atmospheric statistics, modal measurements, data recording, etc.). In addition, it also provides RTC-wide coordination. The SPARTA platform makes use of a distributed SW component model (CORBA) for coordination plus a data publisher/subscriber paradigm for high-performance data distribution (DDS) within the cluster. These choices provide the required scalability: the co-processing cluster can be easily dimensioned/optimized for each particular AO problem size.

The RTC Box loop telemetry and command interfaces are also standardized by the SPARTA platform thus allowing different HW realizations of the RTC Box to remain compatible with the Co-processing Cluster –i.e. the complexity/cost of the real-time pipeline may be adjusted to the particular AO problem. In fact, two RTC Box HW mappings are available for the VLT: multiple, FPGA-aided processing boards are used by the SPARTA platform, whereas single board computer (SBC) architecture is the choice for SPARTA-Light.

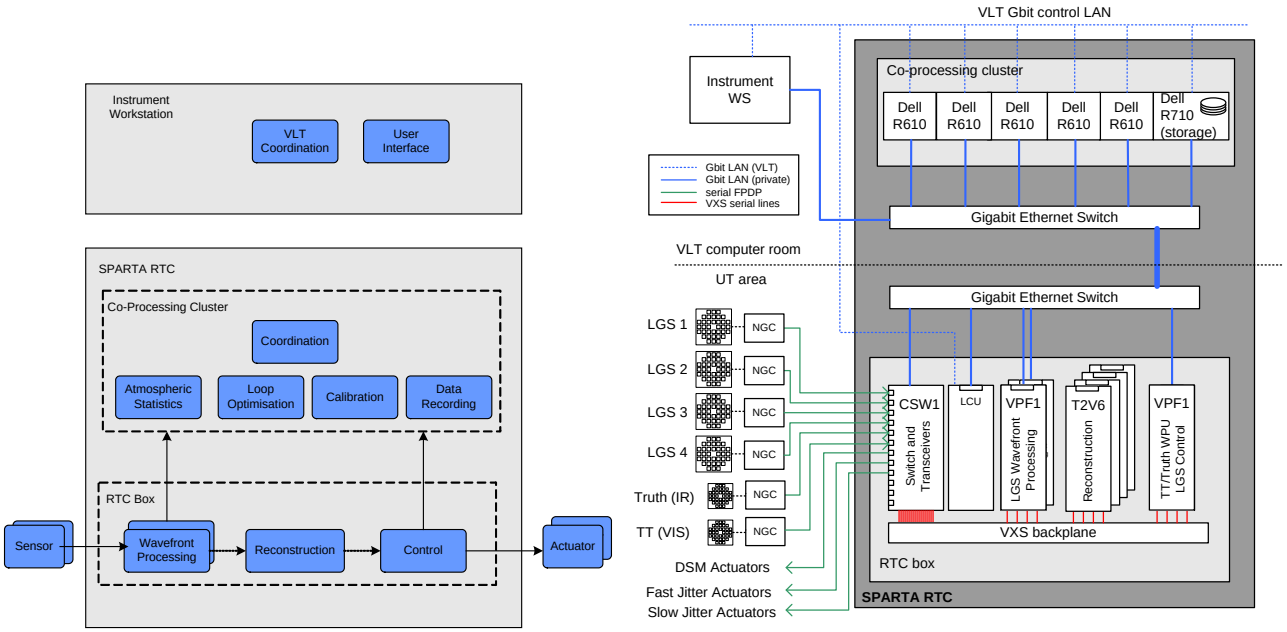

Figure 1. a) Main component blocks and AO real-time pipeline processing stages in a SPARTA RTC (left). b) Example hardware deployment of a SPARTA RTC for the VLT: AOF-like configuration shown (right).

3. SPARTA RTC BOX

The SPARTA platform standardizes a set of hybrid FPGA+CPU/DSP processing boards to be deployed and combined as required using 19" VXS backplane architecture in order to build the RTC Box. The main board features are summarized in Table 1. Their precise number and interconnections will vary depending on the particular AO system. Figure 1b (bottom) shows an example HW deployment for a VLT AOF configuration (GALACSI).

Real-time data is distributed amongst the RTC Box processing boards by means of a switched fabric implemented using point-to-point VXS links running at 2.5Gbps. A zero-latency (i.e. electrical layer) VXS switch (CSW1) connects to the external optic fiber input/output links from/to the WFS/actuators and route them to the corresponding VXS slots in the backplane, where the processing boards (VPF1, T2V6) are located. The serial Front Panel Data Port (sFPDP) protocol is used everywhere across the VXS fabric for minimum overhead and latency.

All RTC Box processing boards integrate Virtex-II Pro FPGA nodes, which implement the real-time data I/O for the particular AO processing stage. They are interfaced to the VXS backplane through RocketI/O transceivers driven by standard sFPDP IP cores and coupled to a PPC CPU running vxWorks RTOS (VPF1 board) or DSP cluster (T2V6 board) by high-throughput links. In the case of the VPF1 boards used by the high-order wavefront processing and control stages, each on-board FPGA node is also responsible for performing all latency-critical computations as well as for pushing telemetry data onto its companion CPU node over the PCI bus -i.e. DMA Transcomm framework. In turn, the FPGA nodes in the T2V6 boards used by the high-order reconstruction stage are in charge of feeding their companion, high memory bandwidth DSP clusters with input data directly into internal memory –i.e. DMA TS Link Port protocol. In both cases, loop computations are performed and final/partial results produced as input data blocks become available, using local fast memory interfaces and requiring no CPU intervention.

VPF1 boards are also used for the implementation of low-order AO processing stages (e.g. tip-tilt, Truth and pupil-centering loops). In this case, several stages may be collapsed in a single board and the FPGA nodes serve as real-time

I/O data router between the VXS backplane and the CPU nodes, where the actual computation takes place using the Altivec 128-bit vector unit. Data exchange between co-located stages is achieved by means of shared memory architecture. This setup limits system complexity and increases control law flexibility at the expense of performance.

Table 1. Standard RTC Box processing HW for the SPARTA VLT platform

Board Model	CPU capabilities	FPGA capabilites
Curtiss-Wright CSW1 - 12 x front panel FO I/O - 14 x VXS links (x4)	N/A	N/A
Curtiss-Wright VPF1 - Dual FPGA+PPC nodes - 2 x VXS links (x4)	- 2 x 744X 1GHz PowerPC nodes - 512MB 125MHz DDR SDRAM/node - 1 x 1000 Base-T Ethernet port/node	- 2 x Xilinx Virtex-II Pro nodes - 4 x 4MB QDR SRAM/node - 8 x 3.1Mbps RocketI/O pairs/node
Bittware T2V6 - Dual FPGA+DSP clusters - 2 x VXS links (x4)	- 4 x TS201 600MHz DSPs/cluster - 3.6 GFLOPS/DSP, 24Mbit RAM/DSP - 4 x 1GB/s TS link ports/DSP	- 2 x Xilinx Virtex-II Pro nodes - 8 x RocketI/O pairs/node - 8 x 250MB/s TS link ports/node

3.1 High-order Wavefront Processing

The Wavefront Processing Unit (WPU) is a critical component which brings major latency reduction by enabling the RTC Box processing stages to proceed in parallel with WFS readout. It was initially specified to deliver negligible latency when coupled to the CCD220 WFS [4] (in use by both SPHERE and AOF) operating at frames rates up to 1.5kHz. A full FPGA-based implementation has proven essential in order to cope with this requirement: the SPARTA platform WPU is based on the realization described in [3], which has undergone further modification by ESO.

The WPU is deployed on a VPF1 FPGA node and processes the full pixel frames from a single WFS. Fixed-point arithmetic is intensively used (i.e. faster and less resource-consuming than its floating-point counterpart), taking advantage of the discrete nature of ADC-converted pixel values. In addition, thoroughly optimized packing of pixel calibration maps has been required so that data access operations can make use of the faster (but scarce) QDR memory. Incoming pixels on sFPDP are calibrated upon arrival (i.e. dark-subtraction, flat-fielding and background-subtraction), individual used sub-apertures are processed as soon as they become fully populated (i.e. pixel thresholding, Weighted Center of Gravity computation and reference gradient subtraction) and resulting wavefront gradients are sent over sFPDP to the subsequent processing stages as they are computed. Pixel frames from a selectable tap point in the pipeline (sub-sampled) together with wavefront gradient (at loop rate) are propagated to the CPU node through Transcomm in unscrambled from (using configurable re-ordering maps), therefore allowing their direct publication as telemetry data.

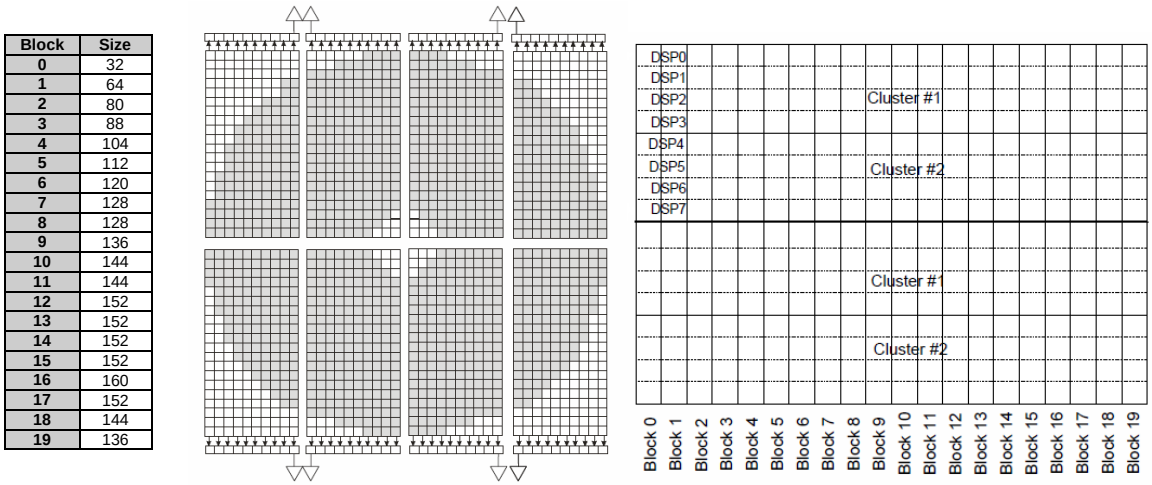

Figure 2. a) CCD220 WFS layout and gradient block sizes (left): 8 channels, 240x240 pixel frame, 40x40 sub-apertures, 6x6 pixel sub-aperture. b) SPHERE control matrix structure (right): size 1377x2480, distributed between two T2V6 boards (upper/bottom halves), each DSP stores a ~86x2480 sub-matrix, each MVM block operation is ~86xsize[block#i] FMAC.

The dimensions, number of channels and readout sequence of each particular WFS coupled to the WPU determine the order and block sizes of the computed gradient stream, which in turn defines the ordering of all subsequent processing

stage results -i.e. the scrambled pipeline order. As an example, Figure 2a shows the sub-aperture grid on the CCD220 WFS and the resulting block sizes. The device is readout simultaneously on all channels and corresponding sub-aperture rows from the top and bottom halves get completely populated within a short interval (i.e. the readout time of two pixel lines), hence triggering a burst of WPU-computed gradients of size twice the used sub-aperture count in the two rows. The main WPU performance figures for this configuration (FPGA nodes clocked at 125MHz) are presented in Table 2.

Table 2. Performance figures of the VPF1-based wavefront processing stage for the CCD220 WFS.

	CCD220 WFS
Input sFPDP latency	< 0.5μs
Pipeline latency - From last sFPDP input received to last sFPDP result sent	< 2μs
Biggest gradients block transfer to next stage	2.56μs (160 values)

3.2 High-order Reconstruction

Reconstruction is performed through direct, single-precision, floating point MVM, the main limitation being the available memory bandwidth -i.e. guaranteeing that new input arguments for the arithmetical units are available every clock cycle to initiate a multiply-and-add operation without pipeline stall. FPGAs embed a limited number of memory interfaces and floating-point units, whilst fast block RAM is expensive in terms of logic resources. General purpose CPUs integrate faster, bigger memory blocks but provide a single bus interface and suffer from cache coherency issues. At the time the SPARTA platform was devised, superscalar DSPs were best suited for MVM implementation: in particular the ADSP-TS201 features four internal 128-bit data buses delivering 33.6GB/s peak memory throughput as long as data remains in DSP internal memory (limited to 3MB) and provides dual computational units with dedicated multiplication and addition blocks.

MVM is performed block-wise in scrambled order by one or more T2V6 boards driven by the arrival of gradient blocks produced by the WPU. Each DSP stores a subset of the (scrambled) control matrix (CM) rows in internal memory, therefore computing only a portion of the output vector. Upon arrival of a gradients block over the sFPDP link, the FPGA nodes make it available to all DSP through TS Link Port channels. Each DSP multiplies the incoming gradients block by the corresponding section of the CM rows assigned to it and accumulates the results. After the last block is processed, the FPGA nodes collect and concatenate the partial output of all DSPs and send the resulting delta command vector over sFPDP. Figure 2b shows an example CM distribution amongst DSPs.

The number of T2V6 boards is dimensioned such that the CM fits in internal DSP memory and the processing of any gradients block is complete before the next one arrives. As an example, four T2V6 boards are required in the AOF configuration (one board per WFS), whereas a single one would be enough for SPHERE -however, two of them are used in order to allow zero frame-delay control matrix update, which doubles the DSP memory requirement. Table 3 presents the main performance figures for both configurations, where FPGA nodes are clocked at 125MHz, TS Link Ports yield an effective 100MBs/s throughput and DSPs deliver 0.7GFMAC/s.

Table 3. Performance figures of the T2V6-based reconstruction stage for the SPHERE and AOF configurations.

	SPHERE	**AOF**
DSP memory use	0.81MB (86 CM rows)	1,4MB (147 CM rows)
Biggest gradients block transfer FPGA→DSP	6.9μs (160 values)	6.9μs (160 values)
Biggest gradients block MVM computation	19.7μs (13,760 FMAC)	33.6μs (23,520 FMAC)
Accumulation of partial MVM results	0.3μs (20 blocks)	0.3μs (20 blocks)
Partial result transfer DSP→FPGA	3.82μs (86 values)	6.38μs (147 values)
Final result transfer to next stage	11.53μs (1379/2 values)	19.22μs (1170 values)

Publication of telemetry data is not possible for this processing stage since the T2V6 board lacks a network interface.

3.3 High-order Control

Time domain filtering of delta actuator commands is performed by means of an IIR regulator including anti-windup control. The initial SPARTA implementation reported in [1,2] allocated this functionality to a CPU node in a VPF1 board, while its companion FPGA was used as an I/O data router between sFPDP and CPU memory using Transcomm. This approach resulted in additional latency (in excess of 200μs) mostly due to data propagation delays through the CPU

Host Bridge and PCI bus. As a recovery path, the current SPARTA implementation splits the time filtering functionality between the FPGA (IIR output update and clipping) and CPU (IIR history update and anti-windup) as follows:

FPGA computation: $w_n = clip\left(b_0 x_n + K_n\right)$ (1)

CPU computation: $K_n = \sum_1^N b_i u_{n-i} - \sum_1^N a_i y_{n-i} + b_0[aw]_n + b_0[gc]_n + [dist]_n + [ref]_n$ (2)

An IIR filter of order N with the same transfer function for all actuator channels is provided –i.e. the numerator b_i and denominator a_i coefficients (normalized such that $a_0=1$) are configurable scalar constants. At loop cycle#n, the FPGA node computes the absolute actuator command vector w_n from the incoming delta actuator command vector x_n according to expression (1). This comes down to a multiply-and-add operation on each incoming value with clipping of the result to a configurable range -typically normalized to [-1,+1]. On-chip single-precision, floating-point FPGA blocks are used which allow full pipelining for minimum latency. Computation is triggered upon arrival of the first input data on sFPDP and proceeds in parallel with data reception, with results being transmitted to the actuators over sFPDP as soon as they are produced. Both the input delta command vector x_n and the non-clipped result $z_n=b_0x_n+K_n$ are propagated to the CPU node using Transcomm. Based on these values plus the actuator bias point $[ref]_n$, the CPU node estimates the IIR block input u_n and output y_n and computes an anti-windup term for the next cycle $[aw]_{n+1}$. If active, actuator garbage collection $[gc]_{n+1}$ and disturbance $[dist]_{n+1}$ received from the co-processing at loop rate are used together with $[aw]_{n+1}$ as per expression (2) to form the overall correction term K_{n+1} for the next cycle which is downloaded to the FPGA node using Transcomm. The equivalent block diagram for the control law is shown in Figure 3.

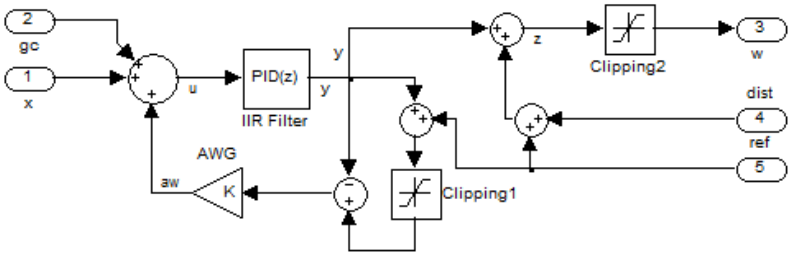

Figure 3. Equivalent control block diagram for the SPARTA high-order controller implementation.

The above approach relaxes the real-time constraints on the CPU node (i.e. one full loop cycle available for correction term computation) while minimizing latency (i.e. FPGA output produced in parallel with data reception) and allowing control law flexibility (i.e. FPGA implementation unaware of the algorithm around the IIR filter, encapsulated in the correction term). Table 4 presents the main performance figures for the SPHERE and AOF configurations, where the FPGA nodes are clocked at 125MHz. The values provided for AOF are estimates, since firmware design constraint prevent direct their measurement on the board.

Table 4. Performance figures of the VPF1-based control stage for the SPHERE and AOF configurations.

	SPHERE	**AOF**
Input sFPDP latency	< 0.5µs	< 0.5µs
Pipeline computation time - From first sFPDP input received to last IIR result computed (not including high-order +low-order stream combining)	17.3µs (1377 values)	14.7µs (1170 values)
Transfer to next stage	11.06µs (1383 values)	9.37µs (1170 values)

The SPARTA FPGA control blocks are fully reusable but must be associated to instance-specific, input/output stream combiner firmware modules to build the so-called RTC Box back-end. As an example, the SPHERE configuration receives two T2V6 reconstructed stream, which are concatenated to produce the x_n input vector for the controller. The FPGA-computed, high-order actuator command w_n is then combined with the tip/tilt correction from one of the CPU nodes and sent synchronously to the actuator HW. In the AOF case, four T2V6 reconstructed streams are averaged to produce the x_n, whereas w_n is dispatched asynchronously with respect to the low-order control channels (i.e. jitter).

Data is processed by the FPGA node in scrambled order but propagated to the CPU node unscrambled (i.e. using configurable re-ordering maps), therefore allowing its direct publication as telemetry data.

3.4 End-to-end RTC Box Latency

Most sFPDP transfers in the SPARTA RTC Box proceed in parallel with the computation of the subsequent processing stage –i.e. T2V6 FPGA-to-DSP transfer starts with the first incoming WPU gradients on sFPDP, delta actuator commands sFPDP transfer starts before T2V6 DSP-to-FPGA transfer is complete, IIR computation starts with the first incoming delta actuator commands. Hence, the time elapsed between the last WFS pixel reception and the last actuator command being computed is roughly given by the addition of the WPU pipeline latency, the T2V6 FPGA-to-DSP transfer time, the T2V6 MVM computation time and the VPF1 IIR computation time, plus number of sub-microsecond sFPDP delays. This yields results below ~50µs and ~60µs respectively for the SPHERE and AOF configurations –an small additional delay (currently not measured) has to be accounted for FPGA-based input stream averaging in AOF.

The AOF high-order and low-order actuators are operated asynchronously. Therefore, high-order commands are sent over sFPDP as they are computed and the above result is the expected end-to-end RTC Box latency -i.e. from the last sFPDP pixel received to the last sFPDP command sent. In the SPHERE case, the back-end FPGA combiner logic waits for both the high-order and tip/tilt commands to become available before sending them synchronously over sFPDP. Tip/tilt correction is computed using a VPF1 CPU node and requires ~6µs CPU time plus ~34µs Transcomm I/O round trip time. Even though both computations are started simultaneously, the tip/tilt correction becomes available ~23µs later than the high-order command, thus causing the subsequent sFPDP transfer to get serialized. As shown in Figure 4, the sFPDP transfer to the actuator HW starts 68.4µs after reception of the last sFPDP pixel -i.e. coincident with the arrival of the tip/tilt command over Transcomm and elapses 11.6µs, thus resulting in a still outstanding 80µs end-to-end latency. An equivalent measurement for the AOF configuration is not available at the time of writing this paper.

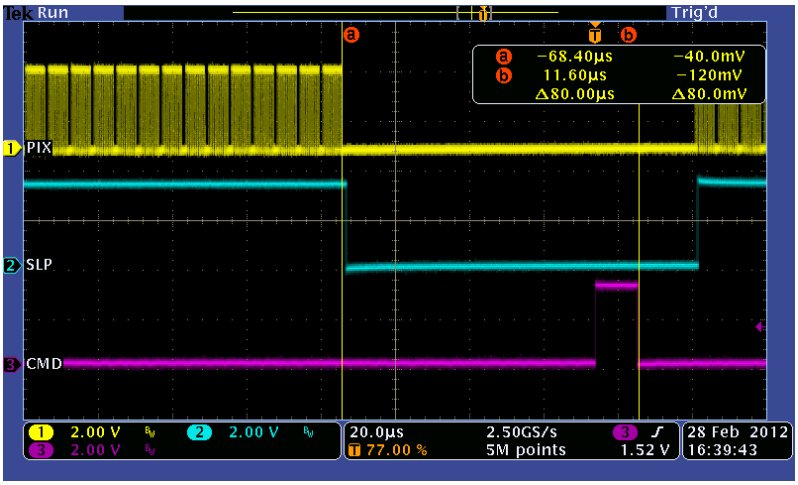

Figure 4. Oscilloscope plot of an end-to-end SPARTA RTC Box latency measurement for the SPHERE configuration using instrumented digital outputs in the FPGA nodes. Cursor *a* marks the last pixel (PIX channel) received by the WPU, whereas cursor *b* marks the last command (CMD channel) sent ~80µs later.

4. SPARTA-LIGHT RTC BOX

The SPARTA-Light platform is targeted to smaller order AO systems of limited complexity and therefore drops the higher performance, FPGA-aided processing modules on behalf of a more cost-effective, full CPU-based solution. An ESO standard MVME6100 SBC running vxWorks RTOS is deployed using 19" VME64 backplane architecture , whereas 2.5Gbps sFPDP WFS and actuator interfaces are implemented by means of dedicated SL240 cards. The main board features are summarized in Table 4.

Table 4. Standard RTC Box processing HW for the SPARTA-Light VLT platform

Board Model	Main features
Motorola MVME6100	- 1 x 7457 1.267GHz PPC node (512k L2, 1MB L3) - 512MB DDR266 SDRAM - 2 x 100/1000 Base-T Ethernet ports
Curtiss-Wright SL240 FiberXtreme	- 2.5Gbps sFPDP (247MB/s, 8B/10B encoding) - 1MB/2kB Rx/Tx buffers (64MB max. DMA size)

The SPARTA processing stages in Figure 1a are mapped to separate SW components in the SPARTA-Light platform as per Figure 5a –the back-end stream combining FPGA functionality is encapsulated in a new combiner processing stage. Being a single-CPU environment, low AO latency can only be achieved by high-duty usage of the MVME6100 Altivec unit (requiring strong algorithm vectorization), optimized L2 cache usage (i.e. minimization/pre-combination of algorithm quasi-static input data) and careful allocation of operations to the on-line/idle phases of the loop cycle. Moreover, low jitter requires a repeatable execution pattern through correct real-time thread priority allocation and similar memory footprint for all loop cycles. The SPARTA VPF1-based, low-order SW components were already devised with these constraints in mind and are mostly reused by SPARTA-Light taking advantage of the fully-compatible CPU/operating system architecture. The major modification brought by the latter is the replacement of the Transcomm I/O SW with an optimized, shared memory Inter-process Communication (IPC) SW layer allowing data propagation between processing stages with minimum overhead. In addition, the control law for the two platform instances currently foreseen (i.e. NAOMI and GRAVITY) provides higher-complexity by implementing advanced Saturation Management and Non-linear Piston Removal.

Figure 5b shows the end-to-end latency measurement results for the NAOMI RTC Box configuration –i.e. 84x84 pixel frame, 6x6 sub-apertures SH WFS, 47-channel high-order actuator, tip/tilt control, 500Hz loop rate. Average ~285µs latency with ~4µs standard deviation is achieved. Fourier domain data analysis reveals jitter excursions up to ~60µs with 1Hz baseline frequency –i.e. coincident with the publication of sub-sampled pixel frame telemetry data in this particular example. This is a general problem: pixel telemetry data requires unscrambling by the CPU (i.e. using pixel frame size reordering maps) as well as network publication in the background (i.e. elapsing several loop cycles). Due to the limited L2 cache size in the MVME6100 board, the memory footprint varies periodically resulting in eviction of useful cached data and Altivec pipeline stall for the subsequent computations. Increasing the pixel frame sub-sampling rate in the NAOMI configuration to the maximum 500Hz loop rate results in still acceptable ~407µs average latency.

L2 cache size is a limiting factor inherent to the CPU architecture that may be partially mitigated by pre-fetching input data directly into cache and implementing non-cached, write-through of results, both of which may be controlled using the Altivec unit. Tests are currently being conducted in this direction for further end-to-end latency reduction.

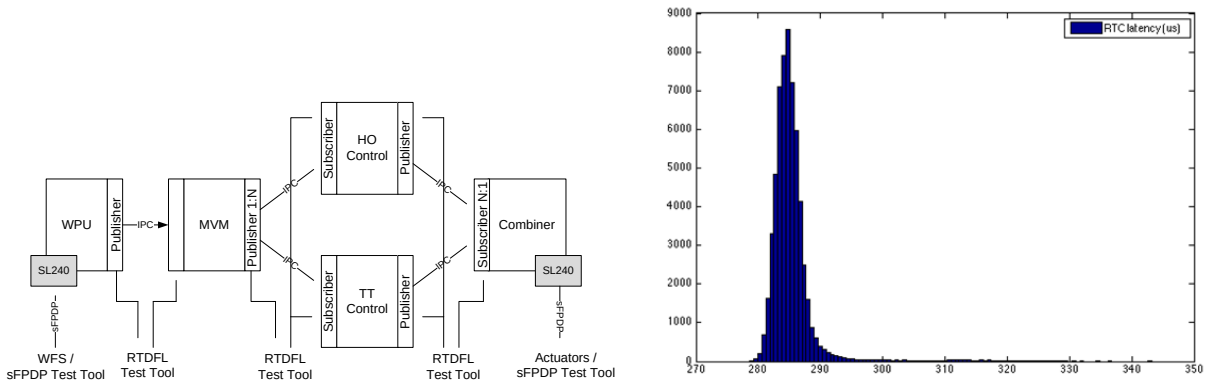

Figure 5. a) SPARTA-Light RTC Box SW component deployment for the NAOMI configuration (left). b) SPARTA-Light RTC Box end-to-end latency histogram (30,000 samples) for the NAOMI configuration (right): average 285µs.

The introduction of a shared-memory IPC in the SPARTA-Light platform poses the problem of testing the RTC Box by input signal injection and/or output signal measurement at intermediate points between processing stages: the original SPARTA Test Tool described in [1] is sFPDP-based and may only be applied at the WPU input and combiner output. The SPARTA-Light IPC design provides additional input/output "test point" interfaces in between processing stages (see Figure 5a), fully compliant with the SPARTA telemetry/disturbance injection protocol (sections 5.1 and 5.4) and accessible through the MVME6100 GbE port. In turn, the SPARTA Test Tool stream generator/reader SW components have been extended to support this protocol, therefore allowing any combination of input/output to be applied to the SPARTA-Light RTC Box internal stages for testing/verification purposes.

5. SPARTA CO-PROCESSING CLUSTER

The SPARTA Co-processing Cluster is comprised of a variable number (depending on the particular AO system complexity) of Linux multi-CPU, multi-core computing nodes interconnected by means of a private, real-time LAN, which is implemented using two distributed network switches (see Figure 1b) coupled through a high-bandwidth uplink (hereafter referred to as a single unit, i.e. the real-time network switch). This accounts for potentially distant RTC Box and Co-processing Cluster installation locations. An additional, physically separate network infrastructure (i.e. the VLT supervisory LAN) is used for CORBA-based coordination of the SW components distributed amongst the cluster nodes.

An overall view of the SPARTA Co-processing Cluster architecture and SW services is provided in [1]. We will focus here on a more detailed description of the main data distribution constraints and design decisions.

Data distribution within the cluster is based on RTI DDS and employs a confirmed reliability quality-of-service (QoS): data reception is acknowledged by the subscribers using ACK/NACK indications, while the publishers keep all the data samples history and retransmit them until delivered -i.e. no data package losses within certain performance limits. All DDS traffic is confined to the SPARTA real-time network by proper switch configuration. A data concentrator node (section 5.1) connects to the RTC Box and publishes telemetry data over DDS for further processing by the data crunching nodes (section 5.2), thus creating a number of DDS data clouds. The pixel frames from the WFS detector(s) used as the input for one or more AO control loops are encoded in a "DDS pixel topic" and form a separate data cloud. Similarly, the wavefront measurements (i.e. sub-aperture gradients and intensities) and actuator commands (i.e. both high- and low-order) derived from the above WFS give rise to a "DDS loop topic" and its corresponding data cloud. All the data in a DDS cloud are sampled at the same frequency and may be easily time-correlated. Additional data clouds are generated by the actuator disturbance DDS topics used for system calibration (section 5.3). Figure 6 shows an example data distribution logical architecture corresponding to a partial view of the SPHERE configuration.

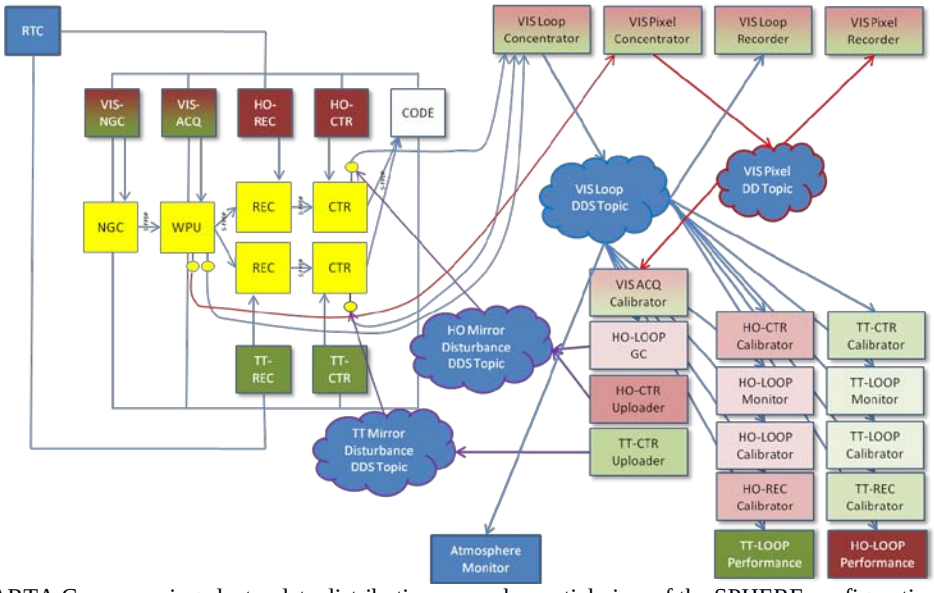

Figure 6. SPARTA Co-processing cluster data distribution example: partial view of the SPHERE configuration (right).

The following discussion applies also to the SPARTA-Light platform, even if the baseline for the smaller AO system is to collapse all cluster nodes into a single machine -including the one serving as the VLT instrument workstation. The scalability and distributed architecture inherent to CORBA and DDS may allow, for instance, the four RTC boxes in the VLTI GRAVITY instruments to share a reduced cluster –i.e. with less than four nodes.

5.1 Data Concentration

Telemetry data are continuously published by the RTC Box processing stages using a custom, TCP-based, point-to-point protocol (RTDFL) over several independent 1GbE segments connected to the real-time network switch. Packets are re-transmitted on a single switch port to the data concentrator node in the Co-processing Cluster, which performs data correlation based on the frame counter, collects all data channels belonging to the same data cloud in a DDS topic and publishes it to the SPARTA real-time network. In addition, the possibility of accumulating telemetry data over a

configurable interval and publishing an averaged DDS topic is also provided. Several data concentrator SW components (i.e. one per data cloud) operate in parallel in a multi-CPU, multi-core node.

In the worst case (i.e. non-averaged DDS loop topics), new DDS content must be published for each loop cycle. For an AOF configuration operating at 1kHz loop rate, this amounts to ~66MB/s sustained throughput. When the sub-sampled (i.e. typically 10Hz pixel frame data) traffic is added, the above value is increased to ~71MB/s. In the SPHERE case throughput is reduced to a still significant ~24MB/s figure. At these data rates, the use of IP multicast over an IGMP-enabled network switch becomes essential to limit both the required network bandwidth and CPU usage. DDS content is published once by each data concentrator SW component over a single, shared network interface and multicasted by the real-time network switch (see Figure 1b) on several ports -i.e. one per data crunching machine. Table 5 shows the main data concentration performance figures for the SPHERE and AOF configurations, using Dell R610 nodes equipped with two Xeon 5670 CPUs -6 cores, 2.93GHz, 12GB total RAM. It can be noticed that CPU usage does not scale with data size -i.e. a significant effort is spent in DDS reliability handling and not only in data copying.

Table 5. Main performance figures of the SPARTA data concentrator node for the SPHERE and AOF configurations.

	SPHERE	**AOF**
HO loop concentrator CPU usage	~45% (1.2kHz rate, 20kB topic size)	~60% (1kHz rate, 67kB topic size)
WFS pixel concentrator CPU usage	~1.2% (1.2kHz rate, 113kB topic size)	~5% (10Hz rate, 450kB topic size)

5.2 Data Crunching

Data crunching nodes are connected to the SPARTA real-time network by means of dedicated switch ports (see Figure 1b) on which DDS content published by the data concentrator node is multicasted. DDS subscribers are typically deployed on them forming a single IP multicast domain -optionally further partitioning is supported for tighter control of network usage in complex systems. They incarnate data-intensive auxiliary processes requiring mathematical computation of variable complexity to be performed on large data buffers in order to produce an optimized loop parameter, perform a measurement, monitor system performance/atmospheric parameters, etc.

A typical SPARTA data crunching DDS subscriber operates periodically by accumulating either loop (i.e. produced at loop rate) or pixel (i.e. sub-sampled) DDS topic data for several tens of seconds and then triggering a mathematical processing on them while the next data set starts to be buffered without losing any sample. Regular computations include simple pixel frame averaging and normalization (e.g. detector maps calibration), projection of loop gradients and/or actuator commands onto modal sub-spaces followed by simple statistics (e.g. modal measurements/performance estimation) as well as higher complexity algorithms involving complex statistics, dichotomy searching, Fourier analysis or iterative processing amongst others (e.g. atmospheric estimation, modal gain optimization and vibration tracking). Alternative business logics for the data crunching components include single-shot (as opposed to periodic) measurements as well as process-N-samples-skip-M-samples schemes for those periodic computations which take longer to execute than the time encompassed by the required data buffer length.

The SPARTA platform provides a DDS-based, templatized data task framework, which encapsulates data subscription, double-buffering and processing callback logic. This allows SW development to focus mainly on the processing algorithm thus enabling simpler outsourcing setups. The use of the Intel Math Kernel Library (MKL) is also standardized by the platform thus additional shielding the developer from the underlying multi-threading and vectorization issues. Several orders of magnitude improvement has been measured in modal projections using MKL MVM and additional enhancement is expected from a full-buffer projection strategy using MKL matrix-matrix-multiplication.

5.3 Real-time Display (RTD)

Some DDS subscribers in the data crunching nodes (i.e. the pixel, loop gradients and actuator commands RTDs) require data at a much slower rate (i.e. 1Hz - 2Hz) than they are actually being produced by the concentrator nodes. They do not perform data buffering but rather sample decimation –i.e. using DDS time-based filtering. However, since a DDS reliable QoS is being used, filtering can only be performed by DDS at the subscriber side -i.e. the subscribers must receive all samples even though only a small subset will be used, thus increasing CPU usage and maybe network traffic -if no other subscriber for that same topic is present in the node. This may be circumvented in the future by having the data concentrator node publish RTD content as a separate topic using a best-effort QoS, in which case filtering can be performed by DDS at the publisher side.

5.4 Calibration

The calibration components are responsible for injecting open-loop actuator disturbance and correlating it with the wavefront gradients measurements in order to compute the corresponding interaction matrix (IM). Disturbance samples are buffered and applied at loop rate by the controller processing stage in the RTC Box. Proper synchronization is required for neither the process to stall nor the RTC buffers to overrun: an initial disturbance batch is published on DDS and, from then on, *N* additional samples are published when *N* new DDS loop topic samples have been received. This turns the calibration components into DDS subscribers and publishers at the same time.

Disturbance injection is performed through the 1GbE RTC Box interfaces using RTDFL. For this purpose, disturbance uploading SW components are deployed in the data concentrator node -i.e. where the single real-time switch port connected to the RTC Box is available. These are DDS subscribers (one per actuator) listening to disturbance frames from the corresponding calibration component and dumping their content onto the relevant RTDFL link.

6. CONCLUSIONS AND FUTURE PLANS

The existing SPARTA platform instances demonstrate that unmatched AO throughput and end-to-end latency performance can be delivered using components off-the-shelf (COTS) HW and standard interfaces by delegating the hard real-time computation to FPGA-based processing nodes, while CPU nodes perform telemetry data forwarding and idle-time computation. Flexible system partitioning in self-contained processing stages can be retained without significant performance penalty as long as data transfers are run in parallel with computations and very fine grain block processing is applied –i.e. again requiring the communication fabrics to be FPGA-managed and ideally point-to-point for maximum parallelization. Wherever the implementation diverts from the above baseline (e.g. processing is CPU-based, thus requiring coarser grain data block propagation through an intermediate Host Bridge) an impact in performance is introduced which has to be traded off for overall system flexibility. Critical, low-level design constraints/drivers are the attainable peak memory bandwidth (i.e. number of memory interfaces and amount of RAM) and, when CPUs come into place, memory cache coherency issues -i.e. L2 cache size and architecture. Potential drawbacks of extensive offload to FPGA nodes are the increased development cycle time and the need for outsourcing major FPGA implementations.

The use of switched network architecture in combination with standard, wide-spread middleware solutions has proven essential to achieve the desired scalability at the coordination and data processing levels. CORBA and DDS are used in their native, non-overlapping domains (i.e. respectively command/reply coordination and publisher/subscriber data flow) exploiting their specific strengths and the available knowledge-base. Additional scalability is brought by the use of MKL, which shields the data processing algorithms from the computing node capabilities, thus allowing the latter to evolve with limited impact in SW.

Several viability studies/design activities [14] are being conducted to bring the SPARTA platform forward for its use with the first generation of E-ELT instruments, which pose orders of magnitude more stringent requirements in terms of data throughput and AO algorithm complexity. The VXS-based, point-to-point, sFPDP communication fabric with multicast capability used by the SPARTA RTC Box may be replaced by a UDP-based protocol running over point-to-point 10GbE links, with multicast being performed by high-performance, cut-through, Layer 2 network switches. It has been shown that such network design can perform deterministically without packet losses. The RTC Box itself may be replaced by an Intel architecture, multi-CPU, multi-core computer running Linux/vxWorks, the FPGA processing nodes taking the form of interconnected COTS PCIe boards with high I/O capability. The SPARTA approach to MVM has been re-evaluated in view of the last generation FPGA capabilities and an implementation based on them has proven feasible, thus allowing dropping the current DSP-based approach, for which no clear roadmap is available. Additional enhancements might include basic DDS content publication by the FPGA nodes –i.e. to avoid CPU cache coherency issues. A competing, higher latency scheme for RTC Box evolution would limit FPGA use to the implementation of optimized, block-wise data transfer to/from the CPU nodes and would actuate on the AO algorithms themselves to achieve the required performance –i.e. make use of recent wavefront reconstruction advances [15,16]. Complementary solutions employing FPGA-driven GPUs for certain processing stages are also being explored.

REFERENCES

[1] Fedrigo, E., Bourtembourg, “SPARTA for the VLT: status and plans”, Proc. SPIE 7736 (2010)
[2] Fedrigo, E., Donaldson, R., "SPARTA: the ESO standard platform for adaptive optics real time applications", Proc. SPIE 6272, (2006)
[3] Goodsell, SJ., Geng, D., "FPGA developments for the SPARTA project: Part 2", Proc. SPIE 6272, (2006)
[4] Reyes Moreno, J., Downing, M., “An overview of the ESO adaptive optics wavefront sensing camera”, Proc. SPIE 8447, (2012)
[5] Beuzit, JL., Feldt, M., "SPHERE: a planet finder instrument for the VLT", Proc. SPIE 7014, (2008)
[6] Fusco, T., Petit, C., "Design of the extreme AO system for SPHERE, the planet finder instrument of the VLT", Proc. SPIE 6272 (2006)
[7] Sauvage, JF., Fusco, T., “SAXO, the eXtreme Adaptive Optics System of SPHERE”, Proc. SPIE 7736, (2010)
[8] Fusco, T., Petit, C., “Control and calibration strategies for SPHERE eXtreme AO system: concepts, implementation and experimental validations”, Proc. SPIE 8447, (2012)
[9] Petit, C., Fusco, T., “The SPHERE XAO system SAXO: integration, tests and laboratory final performance”, Proc. SPIE 8447, (2012)
[10] Arsenault, R., Madec, PY., “ESO adaptive optics facility", Proc. SPIE 7015, (2006)
[11] Paufique, J., Arsenault, R., “Status of the GRAAL system development: very wide-field correction with 4 laser guide-stars”, Proc. of SPIE 8447, (2012)
[12] Stroebele, S., La Penna, P., “GALACSI system design”, Proc. SPIE 8447, (2012)
[13] Marchetti, E., Le Louarn, M., “ERIS adaptive optics system design”, Proc. SPIE 8447, (2012)
[14] Fedrigo, E., Donaldson, R., “SPARTA roadmap and future challenges”, Proc. SPIE 7736, (2010)
[15] Zhariy, M., Neubauer, A., “Cumulative wavefront reconstructor for the Shack-Hartmann sensor”, AIMS IPI 5, 893-913 (2011)
[16] Ramlau, R., Rosensteiner, Obereder, A. M., “Efficient iterative atmospheric tomography reconstruction from LGS and additional tip/tilt measurements”, Proc. SPIE Proc. 8447, (2012)