# Addressing the verification challenge of SERDES-based FPGAs: The performance/accuracy/efficiency trade-off

Chris Schalick
CTO GateRocket, Inc.
19 Crosby Drive
Suite 100
781-908-0082
cschalick@gaterocket.com

## ABSTRACT
The multi-GHz line rates enabled by SERDES introduce new design challenges in FPGAs, notably signal integrity issues which have given rise to a number of design tools and methodologies. But equally as demanding, if not more so, are the functional verification challenges associated with this complex technology.

FPGA designers find that logic simulation of SERDES-based designs is bogged down by long serial test sequences that can extend simulation times by 1-2 orders of magnitude. In addition, SERDES technology employs complex, hierarchical protocols, which makes it harder to thoroughly exercise internal logic. And, because SERDES are often incorporated via unfamiliar third-party IP blocks, debugging the resulting system is problematic.

This paper will look at various functional verification strategies that can be used to address the SERDES simulation bottleneck. Experiences from multiple commercial FPGA based system designs will be considered to examine the trade-offs of different verification approaches for SERDES in FPGA designs. Each method will be looked at for its impact on verification performance, accuracy, and engineering productivity.

Among the approaches considered are:

- Removing the SERDES from the simulations and verifying the rest of the chip using parallel communication
- Placing a second SERDES in the test bench and connecting them back-to-back
- Verifying the SERDES portion of the design on the board in the lab in-system
- Executing the entire device in native FPGA hardware using an emulation-like approach
- Writing custom behavioral models of SERDES

Some familiarity with logic simulators, FPGA based system design, design verification and SERDES serial I/O technology are assumed in the discussion.

## Categories and Subject Descriptors
Advanced Design and Verification of ASICs and FPGAs.

## General Terms
This paper applies to the following designated general terms: Performance, Design, and Verification.

## Keywords
FPGA, SERDES, simulation, verification.

Some frequently used terms are defined here:

| | |
|---|---|
| core-side: | the signal connections between a SERDES element and logic inside the FPGA design that instantiates it |
| IP: | Intellectual Property in the form of pre-developed FPGA modules. These can be hard-diffused in FPGA silicon or delivered as pre-verified programmable elements. |
| target system: | the physical PCB design that is built to hold the implemented FPGA being designed |

## 1. INTRODUCTION
As FPGAs increase in performance and capacity, they are being more widely used for connectivity in a broad range of media, signal processing and communications applications. At the same time, developers have turned to the use of higher speed serial connections for on-chip and chip-to-chip communications, replacing parallel buses to achieve significantly higher data rates. SERDES (Serializer-Deserializer) technology is the key enabler for this type of interface as protocols based on a SERDES approach allow higher data rates with fewer device pins.

This paper will discuss the challenges involved in SERDES-based FPGA design, as well as the alternatives available to address the verification bottleneck that these large complex devices introduce.

## 2. SERDES MODELS
Modern FPGA devices use configurable, high performance SERDES elements to provide access to SERDES technology for a broad range of applications. These are commonly delivered to end users as hard IP blocks and used in applications ranging from simple pin-reducing chip to chip data transfer protocols to standards-based high performance busses that connect to modern computer motherboards. Commonly available FPGA SERDES technology has advanced to bit rates beyond 10 gbit/s.

## 2.1 Xilinx GTP_DUAL

The Xilinx Virtex5 GTP_DUAL cell is representative of modern SERDES and has the following characteristics in its simulation model:

8 serial I/O signals that operate from 100 mb/s to 3.75 gb/s
342 core side signals, some of which are optionally active
184 configurable parameters
9 input clocks and 5 output clocks

Without any further information, designing or verifying a design with those characteristics suggests a substantial endeavor.

The transceiver documentation[1] for this SERDES lists 17 communication standards supported by the SERDES module that make use of 15 different reference clock frequencies. Each of these communications standards uses a unique selection of the configured parameters. Home grown protocols can make use of any FPGA SERDES settings and clock frequencies, whether overlapping standards based protocols or not.

Of the configurable parameters, 68 have two possible values, 70 are numerical with a total of 730 variable bits and 8 x are non-numerical, multi-value. The span of available configurations by parameter settings alone for this simulation model is greater than 2 ^ 730, an astonishingly large number. The SERDES transceiver can clearly operate in a wide variety of modes and support an additionally wide variety of user designs.

To accurately model the behavior of this SERDES design element, its simulation model is understandably very complex. Though this cannot be determined by inspection for the cited SERDES, the apparent load on a logic simulator of designs that use the simulation model for this SERDES is an indicator of its complexity. As will be shown in section 4.2, FPGA SERDES models can dominate simulation time for designs that use them.

Other FPGA vendors and device families have similarly complex applications interfaces and associated SERDES simulation models. It can be observed that the flexibility required to address a broadly selectable and programmable function makes the simulation models complex.

Standard verification practice to test an interface of any design is to use transactors or models that couple to pins of the interface and deliver and consume the data protocol of the interfaces. These transactors typically abstract cycle accurate, pin level function to less granular, more easily understood and manipulated functions. Developing complete transactors to model FPGA SERDES system connections requires the same level of flexibility and functionality that the SERDES themselves deliver to fully test designs that use them.

Modeling external interfaces for FPGA SERDES has tradeoffs between development time, simulation time and accuracy. These tradeoffs will be compared against example designs in section 5.

The simplest transactor implementation uses another SERDES simulation model in the transactor. There is little development time required for this, but the effective load on the logic simulator of the SERDES simulation model is doubled. Alternatively, a "quick and dirty" behavioral model can be written with fast execution times at the expense of functional completeness and accuracy. An option between the two extremes is to model only the functions in use by the FPGA design and leave other SERDES functions untested.

# 3. VERIFYING SYSTEM FUNCTION

Before discussing techniques for verifying SERDES based designs, some sources of verification escapes and challenges of identifying them will be presented. In other words, why is functional verification of FPGA SERDES necessary?

Verification of an FPGA based system is similar in scope to ASIC based systems of three years past. Readily available FPGA devices are capable of implementing logic designs with 500k flip flops, multiple megabytes of on board RAM, hard and soft microprocessor cores and a host of purpose built communications, data processing and bus interface IP. Verifying a system incorporating these devices requires discipline to validate assumptions of interface behavior, sub-system interaction and logical and implementation correctness. Use of SERDES technology in FPGA designs contributes to complexity in each of these areas.

## 3.1 Interface Behavior

One obvious source of functional escapes is in the immediate connections to the SERDES themselves, either on the serial or parallel sides. The serial side data is typically an encoded form of user data, encapsulated by a stack of data manipulations between the user's core-side logic and electronics outside the device being designed.

The stack of conversions on user data may be shallow or deep. An example of a simple, shallow stack is shown in the design in Figure 1. It is a simple conversion of 8 bit parallel data to 10 bit serial data using built in 8b10b encoding in the FPGA SERDES device. Functional patterns to validate the user path of this example are straightforward; an incrementing pattern for 256 core-side cycles will verify all possible data words can traverse the interface.

Though architecturally this is a simple conversion, in practice there are hundreds of signals to connect and hundreds more parameters to configure to make use of such a simple conversion with the native FPGA SERDES devices. Mistakes made by either misconnection, mis-configuration or misunderstanding of device specifications can appear even with this simple example. Thus even a simple use of SERDES can result in functional escapes without proper verification.

More complex examples include packet based bus protocols like Xaui, PCI Express or RapidIO. These interfaces are commonly crafted using a combination of hardened SERDES IP and soft (programmable logic only) IP inside the FPGA devices. The combined FPGA IP is configured to meet system requirements. The external serial interfaces in these examples form connections to standard busses.

Use of pre-validated IP to implement the bus interface helps prevent basic functional errors on the bus, but shifts the interface verification upstream to the parallel core-side data interfaces of the soft IP. The control and data operations on these interfaces differ from vendor to vendor, leaving room for misinterpretation of specifications and creating a vendor specific design and verification task for a standard bus interface.

Because the core-side interfaces are vendor specific and non-standard, the logic on the core-side of the design that interfaces to it can only be exercised by generating vendor specific unique activity on the standard bus that results in desired events. This testing then is not portable from IP vendor to IP vendor.

Long simulation sequences are sometimes required to activate interface signals on the core side of the IP, extending required simulation time to validate user logic that interconnects. For the Xaui design in Figure 3, of the 59 seconds of simulation time 55 seconds are used to initialize the Xaui link and 4 seconds are used to test data transfer on the link. This initialization sequence must be repeated for any other tests.

Verifying interface behavior is complicated by the simulation time introduced by FPGA SERDES simulation models and the uniqueness of protocol IP core-side interfaces.

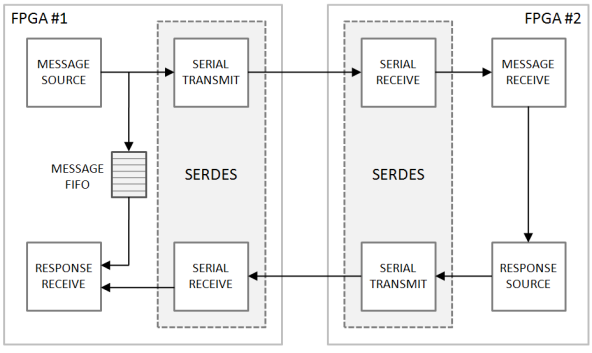
## 3.2 Sub-System Interaction

Another area of functional escapes common in SERDES-based systems is interaction between sub-systems on opposite ends of a serial link. System designs employing multiple FPGAs that use SERDES for chip to chip communication must operate to specifications both independently and in tandem to deliver contemplated system function. Design assumptions for SERDES components contribute to the sub-system verification effort.

For example, the round trip time of data from one FPGA device to another and back across a SERDES based link involves 2 x serialization and 2 x de-serialization times. Given that these times can vary based on the temperature and voltage of the device, systems built around them must be tolerant to these variances. Models must be selectively varied to best or worst case conditions to verify designs will behave correctly under those conditions.

For example, consider the circuit below in Figure 1 which issues a request each cycle to a remote device attached via serial link. Each request when issued is stored in a FIFO until the remote device acknowledges receipt. Variability in serialization and de-serialization times will cause changes in the consumed depth of this FIFO. Verifying the maximum depth occurs before FIFO overflow requires accurate modeling of FPGA SERDES latencies.

**Figure 1. Diagram of Design Susceptible to Latency Modeling Inaccuracies**




As noted previously, accurate simulation models come at the cost of added simulation time. Using inaccurate models increases simulation speed but introduces opportunity for escapes like the FIFO overflow outlined. Such escapes cannot then be detected until designs reach the lab or customer site.

## 3.3 Implementation and Tool Flow Correctness

It is common knowledge that gate simulations show behavior that RTL simulations can not. Test initialization sequences and device simulation models frequently ignore or take for granted initial state of the design in logic simulation. Further, logic simulators can demonstrate behavior with valid HDL code that, in gates, performs differently when fed with high impedance or unknown inputs. These conditions can manifest themselves as functional escapes from RTL simulation, only discoverable in the lab. For these reasons, gate simulations are done to "sanity check" initialization and gate behavior of implemented FPGA designs. SERDES based FPGA designs are no exception.

Working with gates in logic simulation is time consuming. There are typically an order-of-magnitude more events for the simulator to process with a gate level design, relative to the original RTL design. Coupling gate level simulations with the performance impact of SERDES simulation models can result in excessively time-consuming work. Depending on modeling accuracy as noted earlier, this may not even capture tolerance issues.

As an example, the RTL portions of the example design shown in Figure 1 operate 30 times slower in gate simulations than in RTL.

The most common functional error escapes from implementation and tool flow are initialization errors. Undefined values presented to SERDES models may go undetected in RTL where gate simulations may expose defects clearly visible in a lab environment.

# 4. APROACHES TO VERIFICATION

Verifying FPGA SERDES-based designs puts the user in the position of managing the impact of the flexibility and complexity of the SERDES simulation models in the context of common escapes discussed above. Simulation times balloon and cause prolonged initialization sequences before any useful testing can occur. The following sections detail approaches for handling these prolonged sequences.

For the following verification approaches, consider the two designs shown in Figure 2 and Figure 3 below. The design in Figure 2 was described in section 3.1 and is a simple bidirectional 8b10b serial link. The second design is a 10 gbit/s Xaui interface, which converts and transfers data between a core-side XGMII parallel interface and the industry standard serial Xaui interface. This example design was generated by Xilinx Coregen using version 8.1 of the core. The Xaui core documentation is available from Xilinx[2].

The presented functional verification approaches will be compared using the designs shown in Figure 2 and Figure 3. All simulation runs presented were performed on the same machine[3].

**Figure 2. Simple 8b10b FPGA Block Diagram**

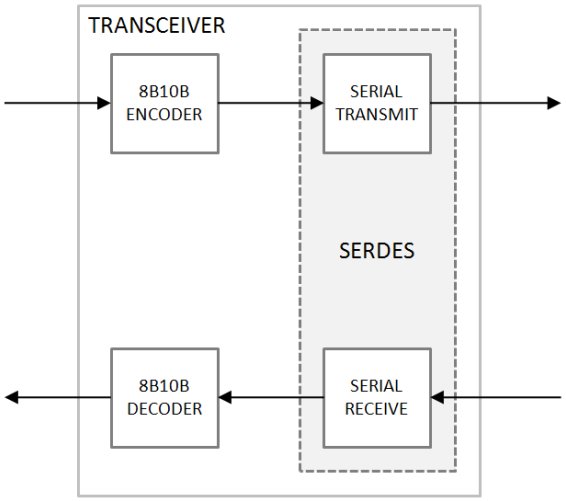
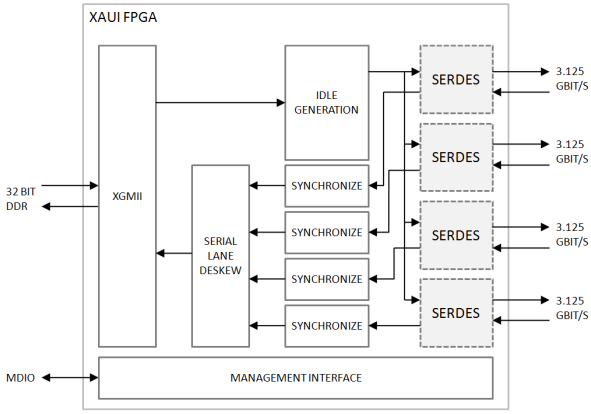



**Figure 3. Xaui FPGA Block Diagram**




## 4.1 Removing the SERDES Models

One common approach to simulating designs with FPGA SERDES models is to eliminate the models altogether. The models are replaced with shells that connect the parallel core-side data directly from source to destination. This is typically done by "scoping in" to the simulation model to drive the SERDES model's parallel outputs directly and monitor the SERDES model's parallel core-side inputs directly from transactors in the testbench.

This has the advantage of avoiding time to develop complex serial transactors at the expense of accuracy of dynamics and function of the serial link. For example, any impact on the core side logic of serialization or de-serialization time or errors occurring during de-serialization will not be properly modeled.

For the simple 8b10b example in Figure 2, this is a straightforward solution because the SERDES elements are used solely to transport parallel data from end to end. There is no control information passed through the serial link.

By contrast, using this approach with the Xaui design in Figure 3 requires substantial knowledge of the configuration and I/O of the core-side of the SERDES. Simply looping tx data and comma controls to the receiver is not sufficient as the Xaui core logic expects transitions through synchronization states on the control outputs of the GTP_DUAL SERDES elements. This highlights the need for accurate SERDES models; core-side circuitry can easily be built that will succeed with simplified models which will fail when connected to the accurate behavior of real physical devices.

It is noteworthy that creating a model that passes logic simulations while scoping around the minimum required set of core-side signals will not guarantee a functional device once the FPGA is built and operating in the lab. Verifying interface operation has two sides: 1) verifying the core-side logic and interface and 2) verifying the SERDES operation itself. Using the presented scoping method does nothing to verify the configuration or implementation of the SERDES itself.

Using this approach with the design in Figure 2 results in very fast simulations; testing completes in less than 1 second. The same approach applied to the Xaui design in Figure 3 results in 18 second simulations. Modeling effort is trivial (minutes to hours) for the 8b10b design and more substantial (days to weeks) for the Xaui design.

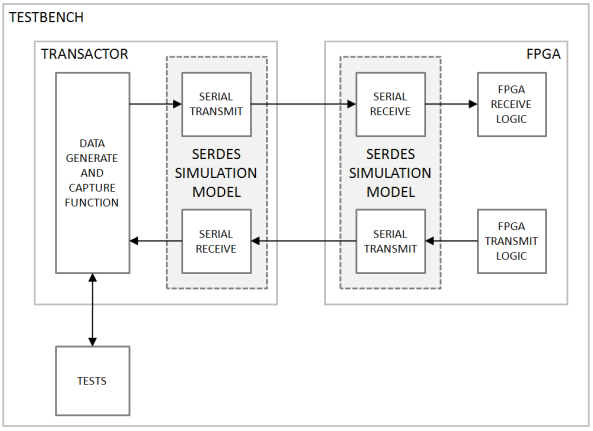
## 4.2 Using a Second SERDES in the Testbench

Another approach to functionally verifying FPGA SERDES designs is to use an FPGA SERDES simulation model in the testbench as a transactor. This has the benefit of rapid development time at the expense of the time to run each simulation since the load on the simulator from the SERDES models doubles.

The basic approach is to instantiate the SERDES model in the transactor exactly as it is instantiated in the design being tested. All configuration options are preserved, and the serial transmit outputs of the transactor are wired to the serial receive inputs of the design being tested and vice versa in the testbench.

This approach is diagrammed in Figure 4 below.

**Figure 4. Connection Diagram of Testbench SERDES**



The data below in Table 1 shows the simulator profile of the design in Figure 3 running in simulation. This testbench uses behavioral transactors for serial side stimulus and captured response.

**Table 1. Simulation profile of Xaui core**

| Entity | Percent of CPU Load | Simulation Time in Seconds |
|---|---|---|
| GTP_DUALs | 71% | 41.8 |
| Xaui core | 11% | 6.5 |
| transactors | 10.6% | 6.3 |
| testbench | 7.4% | 4.4 |

This table shows the simulation time is dominated by the SERDES in the Xaui design during the 59 second simulation.

Converting the testbench to use SERDES simulation models instead of behavioral transactors in the testbench doubles the load on the simulator from the SERDES models. In this case it nearly doubles the entire simulation time for each simulation run; run times with this approach increase from 59 to 101 seconds.

For the 8b10b design in Figure 2, the profile is more dramatic; the simulator reports 100% of the time in the SERDES simulation models. Using a second SERDES in the testbench does double simulation time from 32 seconds to 64 seconds.

## 4.3 Verifying SERDES in the Lab

One solution to verifying FPGA SERDES designs is to skip verification of the SERDES serial and core-side connections in logic simulation and advance directly to the lab for validation of those parts of the design.

One benefit of this approach is the volume of data that can be transferred through the SERDES interfaces in a real-time system. Simulation clock frequencies are commonly 5-7 orders of magnitude slower than free-running physical FPGA devices. Verifying with silicon in the lab thus provides many orders of magnitude more test cycles than software based simulations.

This approach requires a hardware system to perform verification. Waiting for the target system hardware to be available puts the verification of the SERDES paths at the very end of verification, guaranteeing it will be in the critical path to system verification.

Using an off the shelf FPGA evaluation board is an option for lab verification. Unless the design of the evaluation board matches the target system exactly the design being tested must be ported to the evaluation board. The device pinout, synthesis and place and route scripts must be tailored to the evaluation board. If the device on the evaluation board is not the same size as the target device, portions of the design may be required to be taken out to perform this verification.

Regardless of the content of the FPGA design, a separate design effort is necessary to tailor the design being verified to a fixed physical platform. Any changes made to the design or implementation can contribute or mask functional escapes leading to false positive or negative results. The ability to debug the FPGA design using this approach is further limited by lack of visibility of the high frequency serial signals and the opaque contents of the core-side logic.

## 4.4 Verifying With Native Hardware

A different solution to verifying FPGA SERDES designs is to build an emulation platform that incorporates the FPGA design and surrounding electronics to stimulate and respond to it. A purpose built hardware platform with the FPGA being tested can be connected to a logic simulator or software to deliver patterns and check responses of the FPGA design.

This presents a challenge to either: 1) define and produce a seamless interface to the simulator for existing RTL tests or 2) to discard existing RTL testbenches in lieu of decoupling the FPGA design from the tests and transactors.

Circuits operating within a software simulator maintain a strict notion of synchronicity. Rigorously defined rules of HDL language behavior and scheduling make software-only design and simulation predictable and productive. Coupling a logic simulator with a hardware platform and maintaining setup and hold relationships between hardware and simulated HDL is a non-trivial task.

Designing for verification of a decoupled design is also a complex task. Typical FPGA verification flows build up from simple block level functions to test full chip functionality. Decoupling the design from the testbench is a step beyond common FPGA verification environments.

In either case, a custom hardware platform must be constructed to emulate the behavior of the SERDES design with this approach. Such a platform is quickly made obsolete by any of: design changes leading to a different device, device vendor, or changes to the pinout requiring re-wiring of the PCB. A hardware platform built for the design shown in Figure 2 would not be usable for the design shown in Figure 3 since many of the SERDES serial connections in Figure 3 do not exist in Figure 2.

Crafting a flexible, purpose-built hardware emulation solution that is tightly coupled with a logic simulator is a substantial undertaking and is beyond the scope of this discussion. Such an emulation system was used to generate the results below.

Running the Xaui design in Figure 4 in an FPGA emulation system results in silicon accurate SERDES behavior and reduced the 59 second simulation to 18 seconds.

## 4.5 Writing Custom Behavioral Models

As presented in section 4.1, replacing a SERDES simulation model with a simplified model can result in incorrect behavior.

It is common practice to write behavioral models for a specific mode used in a design. Rather than implementing all possible functions of a SERDES in a simulation model, only the selected parameter settings and port connections can be implemented. This reduces the overall scope of model development.

Models produced in this way can be re-used when the parameters and port connections of the SERDES model are used again. For example, the simple 8b10b design in Figure 2 can be modeled with a simple 8b10b encoder / decoder. Any

use of a SERDES with this simple operation can then re-use this simulation model.

The usefulness of these models is limited to designs of the specified function. When new designs or SERDES functions are desired, new models must be written.

A good example of this approach is seen in the testbench written out by the Xilinx Coregen tool when generating the presented Xaui design in Figure 4. Behavioral serial transactors specific to the Xaui protocol are implemented in the testbench. However these cannot be re-used by other designs, including the very simple 8b10b design shown in Figure 2.

This simulation runs in 59 seconds with accurate, vendor provided models for SERDES in the design and purpose built behavioral SERDES models in the transactors in the testbench.

## 5. Comparisons of Approaches

The following tables compare the presented approaches for both the 8b10b and Xaui designs shown in Figures 2 and 3. Table 2 shows results for the simple 8b10b design and Table 3 shows the Xaui results.

**Table 2. Comparison of Approaches for 8b10b**

| Approach | Development Time | Accuracy | Simulation Performance |
|---|---|---|---|
| remove serdes | Trivial | Low | 1 second |
| use serdes in testbench | Short | Medium | 64 seconds |
| Verify in the lab | Medium | High | n/a |
| Native hardware emulation | Long | High | 1 second |
| Write custom behavioral models | Medium | Low | 1 second |

**Table 3. Comparison of Approaches for Xaui**

| Approach | Development Time | Accuracy | Simulation Performance |
|---|---|---|---|
| remove serdes | Medium | Low | 18 seconds |
| use serdes in testbench | Short | Medium | 101 seconds |
| Verify in the lab | Long | High | n/a |
| Native hardware emulation | Long | High | 18 seconds |
| Write custom behavioral models | Medium | Low | 59 seconds |

Using a second SERDES is appropriate only if the SERDES models represent a small portion of the overall simulation time or if there are very few tests. If the simulation profile shows a large percentage of time spent in the SERDES models, the resulting simulation time impact is dramatic.

The presented verification approaches are sometimes used in conjunction. For example, removing SERDES models in simulation and verifying SERDES function in the lab.

The native hardware emulation approach gives the performance of custom behavioral models with the accuracy of the silicon behavior.

Accuracy of software simulation models generally comes at the expense of simulation performance. Hardware solutions provide higher performance and accuracy at the expense of development time.

Extending simulation times creates pressure to reduce the number of tests in verification cycles, which naturally leads to functional defect escapes.

Using evaluation boards or target systems to verify SERDES designs can create schedule risk of waiting for final article and / or throw away hardware and design effort.

## 7. REFERENCES

[1] Xilinx, Inc., Virtex-5 FPGA RocketIO GTP Transceiver, December, 2008,
http://
www.xilinx.com/support/documentation/user_guides/ug196.pdf

[2] Xilinx, Inc. Xaui User Guide, April 2010,
http://www.xilinx.com/support/documentation/ip_documentation/xaui_ug150.pdf

[3] system specifications: Intel(R) Xeon(R) CPU X3363 @ 2.83GHz, 6mB cache, 16 gB DDR3 main memory PC2-5300 / 667 MHz