Server SoC Density Efficient on RAS Validation Activity

Thesis submitted in partial fulfilment of the requirements for the award of the degree of

Master of Technology in VLSI Design & Microelectronics Technology

by

JUI GHOSH

Examination Roll No: M6VLS22009

Class Roll No: 001910703009

Registration No.: 150137 of 2019-2020

Under the Esteemed Guidance of

Prof. Sayan Chatterjee

(Internal Supervisor)

Mrs. Manjula Kumar and Mr. Ashok Kumar Dabbugunta

(External Supervisor)

DEPARTMENT OF ELECTRONICS AND TELE-COMMUNICATION ENGINEERING

JADAVPUR UNIVERSITY, KOLKATA - 700032, WEST BENGAL, INDIA

June 2022

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that the thesis entitled “Server SoC- Density Efficient on RAS Validation activity” submitted by me, to be awarded a Master of Technology degree from VLSI Design and Microelectronics Technology at the University of Jadavpur is a record of honest work done by me under the guidance of an Internal Professor. Sayan Chatterjee, external director Ms. Manjula Kumar and manager Mr. Ashok Kumar Dabbugunta.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, in part or in full, for any other degree or diploma from this institution or any other institution or university.

Place: Kolkata

Date: 28/6/2022

Jui Ghosh

Signature of the candidate

Faculty of Engineering and Technology, Jadavpur University

CERTIFICATE

This is to certify that the work contained in this thesis entitled "**Server SoC Density Efficient on RAS Validation Activity** " is a bonafide work of **Jui Ghosh (Exam Roll No.: M6VLS22009; Class Roll No.: 001910703009; Registration No.: 150137 of 2019-2020),** carried out in the Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata under my supervision and that it has not been submitted elsewhere for a degree.

INTERNAL GUIDE
Prof. Sayan Chatterjee
Jadavpur University, Kol-700032

[illegible]

EXTERNALGUIDE(MANAGER)
Ashok Kumar Dabbugunta
INTEL INDIA PRIVATE LTD.

Prof. Ananda Shankar Chowdhury

Head of the Department

Department of ETCE,
Jadavpur University, Kol-700032

Prof. Chandan Majumdar

Dean

Faculty Council of Engg. & Tech. (FET),
Jadavpur University, Kol-700032

Faculty of Engineering and Technology, Jadavpur University

CERTIFICATE OF APPROVAL*

*This is to ensure that the Master Thesis entitled "**Server SoC Density Efficient on RAS Validation activity**" is recognized as a credible engineering study conducted and presented satisfactorily to validate its acceptance as pre-academic requirements. brought to us. It is understood that with the consent you sign below it does not confirm or accept all the statements made, the opinion expressed, or the conclusions reached but only authorizes the thesis for the purpose for which it was sent.*

Committee on Final Examination

for Evaluation of the Thesis

Examiners

*Only in case the thesis is approved

ABSTRACT

With the expansion and complexity of System on Chip, it is necessary to do verification and validation to meet the industry standards, this is very important and even harder to achieve. Coverage is very much important in domain of verification to verify all the specifications of the protocol and fabric are covered. By getting good coverage number it can be determined if the code is perfect for covering specification and the code is exercised properly or not. Although verification is a vast domain, not only for coverage, but this test environment can also be created for any protocol. Now a day for SOC's Pre-Silicon Verification and Post Silicon Validation both are very much important. In this paper the work will be discussed based on coverage for Intel Specific Protocol especially functional coverage and code coverage. Here for writing configuration System Verilog is used and for simulation and testing scripting language Perl is used.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to my guide Prof. Sayan Chatterjee, of Jadavpur University, for his constant guidance, continual encouragement, understanding, more than all, he taught me patience in my endeavour. My association with him is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of verification. I would like to express my gratitude to Pro Vice Chancellor Prof. Samantak Das, Prof. Chiranjib Bhattacharjee, Dean of Faculty Council of Engg. & Tech. (FET) Dr. Rajat Ray A, project coordinators of Jadavpur University for providing with an environment to work in and for his inspiration during the tenure of the course. I would also like to thank my external guide, Mrs. Manjula Kumar, my manager Mr. Ashok Kumar Dabbugunta and the experts of our team from INTEL who were involved in the validation survey for this research project. Without their passionate participation and input, the validation survey could not have been successfully conducted. In jubilant mood I express ingeniously my whole-hearted thanks to the panel members and all teaching staff & members working as limbs of our university for their not-self-centred enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. It is indeed a pleasure to thank my parents and friends who persuaded and encouraged me to take up and complete this task. At last, but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

I would also like to thank seniors and all the faculty members of the department and all the team members of intel, who directly or indirectly helped me with this work.

I want to thank my family with my whole heart especially my father, my mother and my younger brother for helping me complete this work. Their support was invaluable to me.

Place: Kolkata

Date: 28/6/2022

Signature of the candidate

CONTENT

List of Figure

Abbreviations

Term	Definition
ACPI	Advanced Configuration and Power Interface
ADR	Asynchronous Data Refresh
AER	Advanced Error Reporting
ART	Always Running Timer
AWFCS	Assured Write Frame Checksum Sequence
BIOS	Basic Input/Output System
BSP	Boot Service Processor
CATERR	Catastrophic Error
CCP	Converged Core Perimeter
CIA	Configurable Ip Adapter
CLKI	Clock Interface standard
CPU	Central Processing Unit
CXL	Intel Compute Express Link
DIMM	Dual Inline Memory Module
DSO	DTF Source Observer
DSP	DFX secure plugin
DTF	Debug Trace Fabric
DTS	Digital Thermal Sensor
DVFS	Digital Voltage Frequency Scaling
DVP	Debug Visibility Packetizer
ECC	Error Correction Code
FCS	Frame Checksum Sequence
FIVR	Fully Integrated Voltage Regulator

FPC	**Fuse Protocol Converter**
FQN	**IPX Fully Qualified Name**
FRB	**Fault Resilient Boot**
GPF	**Global Persistant Flush (ADR for CXL)**
GPIO	**General Purpose Input/Output**
HPM	**Hierarchical Power Management**
HWP	**HardWare Performance (states)**
HWRS	**Hardware Reset Sequencer**
ICL	**Instrument Connectivity Language**
IERR	**Processor Internal Error**
IPX	**IP eXchange**
ISA	**Ip Specific Adapter**
ISS	**Intel Speed Step**
ITD	**Inverse Temperature Dependence**
IVT	**Ivy Town server**
LTA	**Line Tracker Algorithm**
LTR	**Latency Tolerance Reporting**
MBVR	**Mother Board Voltage Regulator**
MBVR	**Mother Board Voltage Regulator**
MBVRM	**Mother Board Voltage Regulator Module**
MCA	**Machine Check Abort**
MCERR	**Machine Check Error**
MCTP	**Management Component Transport Protocol**
MSR	**Model Specific Register**

N/A	**Not Applicable**
NMI	**Non-Maskable Interrupt**
OOB	**Out-of-band**
OS	**Operating System**
OS	**Operating System**
PA	**Protocol Aware**
PCH	**Platform Controller Hub**
PCH	**Intel Platform Controller Hub**
PCIe	**PCI Epress**
PDL	**Procedural Description Language**
PECI	**Platform Environmental Control Interface**
PGCB	**Power Gate Control Block**
PMC	**Power management controller**
PMT	**Platform Monitoring Technology**
PState	**Performace State**
PUnit	**Power control Unit**
PWM	**Pulse Width Modulation**
PWM	**Pulse Width Modulation**
RAC	**Read access control**
RAS	**Reliability,Availability,Servicibility**
RC	**Resource Controller**
RA	**Resource Adapter**
SAI	**Security Attributes of the Initiator**
SCU	**Soc Control Unit, synonym for PUnit**

SGX	Software Guard Extensions
SDL	Security Design Lifecycle
SMI	System, Management Interrupt
SMM	System management mode
SRAR	Software Recoverable Action Required
sVID	Serial Voltage IDentification
TDX	Trust Domain Extensions
TRM	Technical Reference Manual
TSC	Time Stamp Counter
TSOD	Temperature Sensor on DIMM
UCNA	UnCorrectable No Action
UFS	Uncore Frequency Scaling, generally mesh DVFS
URT	synonym for ART
VISA	Visualization of Signals Architecture
VR	Voltage Regulator
VRM	Voltage Regulator Module
VRCI	Voltage Regulator Common Interface standard
VSEC	Vender Specific Extended Capabilitites
WAC	Write access control

CHAPTER 1
INTRODUCTION

Introduction

1.1 Introduction

According to Moore's law, as transistor sizes get smaller, the density of integrated circuits will almost double annually. A shift from computation-centric designs to communication-centric architectures with several IP cores has occurred as a result of these developments in Very Large Scale Integration (VLSI) integration density. An integrated circuit known as a "system-on-chip" (SoC) combines all of the parts of a computer system onto a single silicon chip. These include memory, general-purpose input and output ports, a central processing unit (CPU), and auxiliary components like modems and a graphics processing unit (GPU). An SoC's fundamental block diagram is displayed in Figure 1.1. Any SoC will have a memory subsystem, CPU subsystem, multimedia subsystem, and peripheral subsystem. Communication between these subsystems is possible.by means of various interfacing protocols through a Network-on-chip.

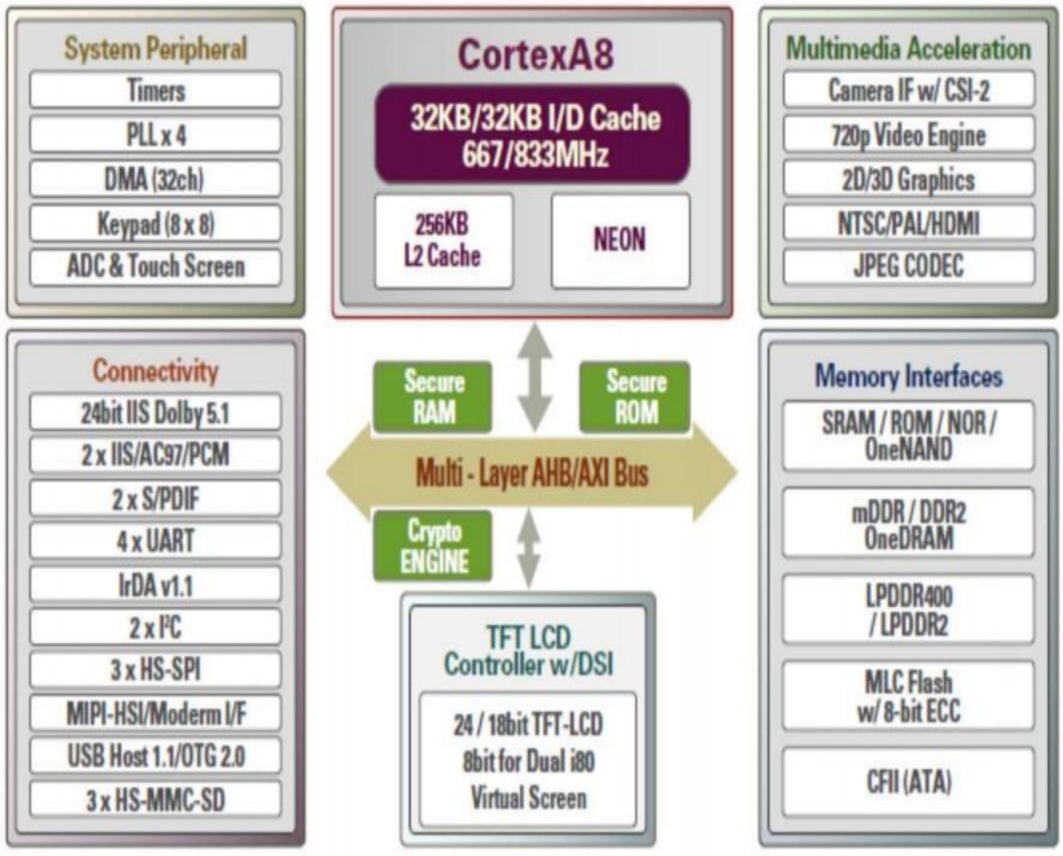


Figure 1.1: General SoC design

a lot of power and space limitations. As a result, a lot of fundamental architectures need a reliable communication infrastructure. These problems are now being solved using Network on Chip (NoC).

NoC is a network-based communications subsystem and router-based packet switching network that lies between SoC components. For SoC integration, it is necessary to comprehend functionality, system requirement requirements, numerous protocols, architecture, and interface details. It takes a lot of time.

Design Verification is a crucial stage of SoC Design. By identifying potential mistakes in both the design and the system's architecture, the design verification stage assesses the design's quality and ensures its accuracy in operation. Both the block/IP level and the SoC level can undertake design verification. The top-level SoC operations are the focus of SoC level verification, together with the accuracy of the intercommunication between the sub-blocks.

The security blocks, which are in charge of ensuring the security and legitimacy of sensitive data used in the SoC, are the sub-block that is being focused on for SoC verification in this case. These security blocks are essential in current chips where data security is of highest importance since they can authenticate and encrypt data as needed.

1.2 Background:

It is no secret that Intel's Enterprise processor has been expanded in recent generations Compared to the competition, Intel is chasing its mass death strategy while relying on a manufacturing facility that has not provided the best in the market. Today's businesses are increasingly relying on Intel Soc servers to run applications that require data and are important for policy. Server reliability, availability, and efficiency (RAS) are key issues in today's business enterprise IT systems that deliver critical applications and services, as failure to deliver applications can cost every hour of system malfunction. In addition, the chances of such failures increase statistically with server size, data, and memory required for this post. The Intel SoC family provides a complete and robust set of RAS features in silicon to provide error detection, correction, content, and detection across all processors, memory, and I/O data methods. This set feature is a powerful platform where software and software vendors can create modern RAS layers to provide complete server reliability across the hardware-software stack from silicon to delivery of applications and services. The Intel SoC family brings all of these features to a competitive and energy-intensive price compared to traditional RISC-based solutions in the market.

Intel's new generation SoC building strategy will have a dual track of product-based (Performance-core (P-core) and Efficient-core (E-core) products, from two advanced platforms to one common site, defining the industry. This new approach will increase per watt performance, component features and Intel's overall competitiveness in the industry. While introducing this new approach, Intel has strengthened its compliance with SoC largest ecosystem platform today and the benefits that customers will receive from a single investment.

1.2.1. Server SoC:

This is a Intel next generation processor with low power at SoC Product Swim Lane. It replaces the ICL-D product and is targeted at a wide range of markets, including comm, storage, and IOTG.

Server SoC can be divided into 2 main components, the reusable previously used SoC Compute die (s) and the Server SoC unique IO die, which includes the 2xNAC 200G Ethernet Subsystem, and various Networking Accelerator. Compute die and IO die are MCP within the BGA package at the Kelseyville platform. Server SoC supports two product line flavours Advance and Ultra. Advance SOC has compute die computer (s), HCC 42 cores and MCC 18 cores, as well as a single IO. During the Ultra SoC, XCC has up to 48 cores and 2 examples of IO deaths.

Server SoC provides a way to prevent the processor from operating above the maximum safe operating temperature. It is useful for a variety of reasons such as thermal throttling to reduce heat to avoid processor damage, and electrical compensation due to Inverse Temperature Dependency (ITD). This is achieved by using a network of various temperature sensors to measure temperatures in a wide range of tropical and subtropical regions. PCcode analyses the temperature to initiate thermal movements and to report data to the platform through various MSRs. If you are not in this throttling-based firmware function, there is a hardware protection called THERMTRIP that is fully hosted by compatible hardware.

Digital Thermal Sensors (DTS) sensors are precise hybrid signal circuits that generate internal electrical energy equal to the total temperature and then convert analog to digital. DTS supports CRI Slave virtual interface that will be connected to the Service Adapter (RA). The App Manager controls the DTS temperature sensor values with the help of the App adapter over the PMSB bus. RA reads DTS temperature values over the CRI interface and provides data back to RC via PMSB bus. The RC reads the amount of DTS periodically almost every 500µs and sends them to Punit as part of the telemetry service.

Server SoC is targeted at the Micro server market in the Kaseyville area. The GNR-D SOC contains Compute die (from previously used SoC) and the IO die and Integrated ETX Server SoC will have a PCH less with integrated boot supported by S3M.Bacon profile. It does not have support for multiple sockets. It has an integrated clock solution for clock production. S3M is used as BACON (Boot Access Control). S3M is in the S5 domain and ETX components.

1.2.2. RAS (Reliability, Availability, Serviceability) Feature:

For the most crucial applications, such as websites, enterprise resource planning (ERP), customer relationship management (CRM), and business intelligence (BI) applications, new technologies in more advantageous price conditions are necessary. Many of these applications must also be accessible locally or worldwide twenty-four hours a day. The largest website, BI, and advanced apps cannot be solved by integrating and leveraging real resources alone, even though they can help fulfil availability needs. The hourly cost of a single business plan failing can easily go from tens of thousands to millions of dollars. All of this evidence necessitates the use of extremely sophisticated and durable servers that are well-suited to large-scale integration and vital business applications. Only those failures that can be handled at the hardware level are allowed by standard RAS server usage. Normally, RAS is an important component of system design. The reliability of the system depends on how well it can detect errors, correct them or reuse them to maximize Availability and provide sufficient error information to service errors.

Larger designs are less flawed that can make parts less reliable, especially in storage structures. Although solid errors are similar to production errors that are corrected by replacement, severe errors due to aging are not. Also, soft errors due to cosmic rays can temporarily affect values. Reliability: The ability to recognize all errors increases the reliability of the correct answer. Acquisitions are therefore increased.

Reliability: Data integrity is related to data protection through the detection and repair of errors, or if it cannot be corrected, the retention of these errors. Error detection confirms that errors are detected in the data and command level. Debugging deals with an error found by returning an error piece of data or fragments with their correct algorithm. Content error ensures that corrupted data is thus marked across all major components and data pathways so that sub-systems other than those that fail can take appropriate action if they encounter such errors.

Availability: Modern methods improve system availability by providing methods that allow for seamless operation even in the face of irreparable errors. These methods include multiple depletion levels (spare processors, DIMM memory, and I / O resources), automatic failover to silicon and Hardware levels, and software-assisted error detection in various layers of the software stack, from OS and virtual. The memory manager (VMM) at the bottom of the website, commercial, or application layers at the top end. All of this provides server stability in the face of irreparable hardware failures. Using methods such as ECC data allows for one-bit errors (per set) and increases the availability of a useful component.

Service: Modern methods improve usability using predictable failure analysis to identify problematic components before creating irreparable errors or real downtime, thus facilitating the replacement of components in the event of severe failure. System partitioning is used to further separate workloads that are affected by irreparable errors from other functional loads that operate on the same server infrastructure and facilitate repair. Errors may be corrected, irreparable, or even fatal. The last two PVC clubs. Reporting these errors in detail, allows you to identify when and where errors occur. This allows for servicing (changing, excluding, etc.) part of the full system.

1.3 Brief Methodology of the project

The project is broadly divided into two phases namely

1. RAS Integration - Different types of IP cores are grouped according to interfacing details, port names, port width and given SoC specifications. The process for compiling is as follows:

• For a list of ports, port width, hole direction, hole and block functions and related details from the design team.

• Look at the information obtained from key design owners and understand what and how the communication should be made.

• Creating connections between different IP Cores according to the specifications provided by updating the files received by the design team. These files contain a list of hole names, directions and a wide range of holes. A compatible port that should be connected to every hole should be added to the files.

• Keep files updated and open an Intel ID connection tool.

• Examine the output list of output files to detect offline inputs, errors, unloaded output and multi-drive holes and correct them.

2. RAS Verification - Verification is done with the help of System Verilog test benches.

The verification process is as follows:

• Adherence to the Universal Validation Methodology (UVM) standard, which allows for the rapid development and reuse of verification facilities and IP verification (VIP).

• The construction phase is performed at the beginning of the UVM testbench simulation, in which parts of the test bench are built, repaired and connected.

• The operating phase consists of the start of the simulation and the running phase. The running phase is time consuming, as this is the stage where the case is examined.

• The final phase of cleaning, which includes extraction, inspection, report and final clauses. Here, the results of the test case are collected and the number of errors, if any, is reported.

1.4. Assumptions made / Constraints of the project

The communication and verification of the various IP Cores is done using Qualcomm proprietary tools. Screenshots of results, captions, internal configurations of various IP cables cannot be disclosed. Caption code, results are adjusted so that company information is not compromised and true to actual results obtained.

1.5. Objective:

- The main purpose of this project is to study the various techniques of Physical Design and Physical Verification of the structure generated.
- The structure generated in the netlist list with various processes, power, and temperature using industry tools, which are useful for verification and compliance details. Intricate designs such as Test-chips.
- The work also aims to create the most common challenges one faces over time by making changes, time, and power over complex chip chips and providing possible solutions.
- The role of physical validation in some of the most useful design processes like ECO is also explained in this work.
- This activity is also a discussion of possible improvements and improvements to existing systems using a specific writing strategy to prepare the verification process for future challenges.

1.6. Motivation

RAS integration is a challenge and a complete process. It requires a good understanding and grasp of the various protocols that come together to make connections between the two modules. You need to first understand the functionality of each block, the function of the anchors in it and the type of visible interface between all the IP cores present in the RAS before merging.

RAS authentication differs from IP / block level verification as it not only ensures the performance of individual IPs / blocks, but rather ensures their performance when integrated as part of the entire system. This is very important in ensuring the production of flawless and flawless chips. Additionally, security blocks are an integral part of the SoC, and their flawless operation is an important step in ensuring secure data transactions.

1.7. Problem statement

One SoC chip has about 90000 to 1 lakh connections to be made between the different IP cores present in it. Every IP core has its own debug, clock, Design for Test (DFT) signals with the exception of the integrated signals that must be present.

connected to other cores. Connecting IP cores requires a good understanding of their performance, integration details, RAS structure, and should be done according to the provided SoC / ASIC data.

The complexity and development of SoC design and architecture has made design validation one of the most important stages in the design cycle. It is a large and complex domain that is constantly evolving. Therefore, it requires a systematic approach to properly verify and correct errors for advanced SoC designs

1.8. Organisation of Thesis:

Chapter 2 discusses a brief discussion of the earlier function of the SoC processor

• Chapter 3 discusses various collaborative agreements such as PUnit, IERR, MONFAIL used in RAS. Apart from this, the principles of design validation and its emergence from System Verilog to UVM are discussed. Additionally, an overview of the various defines protocols is presented.

• Chapter 4 discusses the specification of the UVM test bench design and its customization to ensure SoC quality. Implementation of a test bench designed using UVM classes and methods is described in detail.

• Chapter 5 discusses the results obtained after integrating and validating the various IP Cores. Displays a variety of simulation logs and results generated during design validation, as well as a brief summary of the pass / fail status for advanced testing.

• Chapter 6 discusses the different learning outcomes, the scope of the future of the work done, and the conclusion of the work done.

CHAPTER 2
Literature Review

Literature Review

2.Literature Review

There is a growing need for the successful processing of big data in the education community and industry in order to obtain valuable information. Most of the frameworks are built on commercial machines with multi-core or multi-core processors such as the Intel SoC E-series and Intel SoC Phi-series. These processors always have complex mesh connections, caches, and memory hierarchies. There are many differences in structural features between central memory systems and much more. This chapter will review some of the work done on Intel SoC Processors: Multi-Core, Memory Systems and the Integrated Genome-Scale Networks Information for SoC Processor.

2.1.1 Design and Integration Methods for a Multi-Threaded Dual Core 65nm Processor

The successful construction of a complex multi-billion transistor processor relies heavily on solid, certified silicon and composite methods. Process weight of 65nm, the pursuit of the best performance of the segment, and the aggressive schedule to go to market are strongly emphasized in the new construction and the composite method that allows silicon performance. Here, researchers Raj Varada, Mysore Sriram, Kris Chou, James Guzzo did an excellent job of explaining the design and integration techniques used successfully in the multi-threaded dual processor 65nm SoC.

The Moore Performance Measurement Process The process allows multiple processors to be integrated into the same chip in a 65nm process area. Combined with multi-thread technology [1], the dual-core processor looks like four sensible processors and can run four wires simultaneously on the same processor chip. Designing and assembling a processor with more than 1 billion transistors is an important challenge given its market-time objectives and complexity of the original 65nm processors. Designs are based on solid design principles and methods.

The SoC MP [2] processor described in this white paper contains two 64-bit cores and a 16MB Unified Level 3 (L3) cache. Each theme supports two strands and has a combined 2 MB (L2), which is widely used in the construction of standard buildings. This processor was limited to creating a highly efficient high-performance virtual interface (SDI) that uses core processor cores, integrated L3 repository, and processor front side bus (FSB). Main preparation is required. CBC). This structure reduces the reservoir and delays in external bus access. This, in turn, reflects the visual SDI connector at the conceptual level. CBC archive handles basic solutions, L3 repository access, and external bus applications.

This is a brand-new design for using low latency, high-density chip connectivity fabric, which includes RAS capabilities for robust server performance. CBC logic and chip level integration are performed using the techniques described in this document. The main components of the portable CBC component were control blocks and data exchange blocks.

Apart from these blocks, there was another sensible part of washing, DFx, and crossing the domain. Part of the chip cache was designed using the same method as it is treated as a black composite box. Therefore, the definition of temporary storage strategies is beyond the scope of this document. This reflects the high-level structure of the processor. The processor chip is a 435mm2 chip, and has a transistor of 1.33B. It operates above 3.0GHz from 1.25V core supply

The CBC is designed to limit strength. The CBC block is divided into two categories based on the average function factor (AF) and sensitivity of power or value of time. The most important time block with the highest AF rate was originally designed using transistor-based cells. When the required time interaction level is reached, the cells in the critical timeline are selectively replaced with a cell containing a low leakage transistor (LL cell) without interfering with the design metrics of the design. By blocking the lower aspect of the function, the design is made using LL cells and other self-adhesive cells are used to integrate the last few timing channels. It was noted that using only LL cells would increase the number of cells and the total volume. Therefore, using the self-adhesive cell block of high AF reduces the dynamic performance. The strategy, which incorporates these two features, allows for faster integration of design and application of appropriate workplace reduction strategies.

The die size of this chip is determined primarily by the centralized component and the repository and was not highly dependent on the assembling method. Additionally, CBC performance trading allows for the creation of time and a way to integrate all of the upgraded chip to integrate full chip block chain and CBC block chain independently without further ado. The first parameter to be adjusted was the height of the signal line repetition so that by protecting half all full-chip wires and minimizing the mirror junction using a set of each predefined cable width and space widths wider than the minimum process process and space. It is now designed as full chip path as possible. Also, recurring distances at different levels have been adjusted to have easier relationships with each other. For example, the repetitive metal8 range was twice the repetitive metal6 distance, which is twice as much as the repetitive metal4 range.

A visual duplication approach was used to increase the productivity of the project. Visual repetitions are figurative annotations placed on the phone and contain information about the recurring cells that they represent. The Extrusion and Time tool detects these annotations and can calculate delays and endings as if a real recurring cell was placed in a specified location, without the maximum header listing or termination. Repeaters were kept in visual format until about a quarter of the tape outs were too late for the project. Once the design is stable, a real repeater is added to the network list based on the repeter's location. By performing a virtual repetition, the integration team was able to respond quickly to the latest design changes while maintaining acceptable accuracy in the time model.

The Virtual Repeater Insertion Tool minimizes total delays while completing the target set with each receiver machine and repetitive input. All full chip duplicates are placed in the space between the full chip blocks. The system has a recurring grid. This is a narrow horizontal or vertical block placed in the program space indicating the location of the official repository. These are separated by a repetitive tone of the full chip track path in that area. For example, in a horizontal area, repeating stations were available at about 600u separately. Metal4 wire requirements every time it passes a channel, Metal6 wire needs to repeat on all other channels, and Metal8 wire can skip 3 channels before a replay is required. The Virtual Repeater Installation Tool was restricted to repeat repeaters only in Keep in Region when the repetitive channel location is not set.

Marketing time is an important parameter of this project, so as much automation was needed to use the full chip route and repeat. However, the general reversal of this slow compound is due to the variation in the results produced by the changing instruments. The Auto-based Route specification method and the well-defined control method of the Repeater Insertion Tool have helped to control variability and produce predictable results.

Auto Router allows you to specify route specifications with different levels of granularity inside. In the early stages of the project, clarification was a command to pair a simple layer. However, as the design was more stable and more routes over time, additional obstacles were presented to the route in the form of topology guidelines, a "snow" successfully route. By the time the design was almost integrated, almost every FC network had topology specifications attached to avoid unnecessary diversity.

The visual repetition is converted to visual repetition quarterly just before the tape out, so the repetitive installation result is predictable, and if you set the location of the visual repetition the next time you use the tool. But I had to be able to reproduce it. The control file system is designed for this purpose, allowing the user to specify how many first or last repetitions are included in a particular block PIN. You should use the subset of recurring channels to complete a certain repetition or write more. Automatic gradient setting, such as a set of networks. By maintaining this control file, the visible recurring input tool has been able to produce results that can be very reproducible, albeit from the beginning of each integration cycle.

Block design and integration methods are key factors in bringing more than a billion transistor server processors to market. We have introduced a set of new strategies and methods used to design and integrate server processors to enable industry-leading performance with a rapid development schedule. The powerful use of cell-based design techniques has made the pipelines come together in early space-based space. The performance target was achieved through a block chain-reduction system, which includes a high-density cell design with high voltage. Using more ropes and half shields allows for more repetitive space for each metal layer, allows for larger block sizes, reduces the number of repetitions, and is within working blocks. It is no longer necessary to embed FC repeaters. This well-designed FC metallization technique significantly improves the electrical conductivity of the entire design (FC sound, EM, minimal effort to combine sensible heat) and allows for recycling of hard-core IP. Significantly separate the design integration of the block chain integration into the full chip level. The repetitive approach has reached the near-final quality of time at the beginning of the design cycle, with minimal disruption to the design process.

Using a comprehensive and integrated set of approaches adopted at the beginning of the design cycle, they were able to strategically distinguish a few aspects of IC design. The development of RTL, complete chip integration, time, cell-based composition, context, and repository form are almost identical. Solid disintegration from FC integration, timing, and work block integration expands FC's visual interface with larger working block sizes, pre-defined metal sharing, termination of embedded duplicate channels, and a specification base. This was achieved by reducing the time allocation to make the transaction calculated locally, schedule, performance, and complexity, able to provide a solid design with the best performance in the market while taking a break during development. a field of rice. Most of the new features described in this white paper are now part of the process of designing a processor, which provides the server processor with high performance and reliability in the market.

2.1.2 Dual-Core Intel SoC Processor 65-nm 16-MB Shared On-Die L3 Cache

A 16MB single hole, 16-way associative cache for dual-core Intel SoC processor 7100 series using cell -0.624m2 of 65nm steel technology 8. Low power technology is used in the L3 repository to reduce both leaks and flexible performance. Sleep transistors are used in SRAM systems and peripherals to leak more than double the reservoir. Only 0.8% of the cache is unlocked to access the cache. Intensive Cache Safe Technology deployment using archives protects the archive from potential errors and unauthorized child deaths.

Jonathan Chang, Senior Member, IEEE et al. did an excellent job of demonstrating the use of 65-nm 16-MB Shared On-Die L3 Cache for Dual-Core Intel SoC Processor 7100 Series. The dual-core Intel SoC processor 7100 with LMB integrated storage up to 16MB is powered by 65nm process technology with 8-layer copper interconnect [3], [4]. This shows the dying image of the processor. It contains two cores, each containing 1MB L2. The processor has a total of 1.3 billion transistors, and each context has 100 million transistors.

The processor works at 1.25V and 3.5GHz. Supports 150W and 95W hot design power. The L3 repository and compatible mind set have different power source and context, PLL, and I / O. Here it shows four voltage sources for the processor. The front side bus can operate at 800 or 667 MT / s in 3 load configurations. Both L3 and L2 use the same 0 in the 624 m bit cell. Designed with SRAM arrays and their peripherals, the sleep transistors deliver an average power of 0.75 W / MB while always maintaining the archive content [5]. Total energy savings are more than double, with guaranteed silicon concentrations. Whenever possible, they use long-channel devices to further reduce power consumption. The power reduction option is applied to the same SRAM members to reduce power leaks in the inactive subarray. Aggressive clock entry, good sleep adjustment, and wake counters are used to reduce dynamic performance. Column duplicates are available for data and tag programs. Block redundancy is available in repository size. Intel Cache Safe Technology (formerly known as Pellston Technology) tracks random ECC events for each repository line and disables possible errors and repository lines that often die in children. A variety of test solutions are available to ensure performance.

Sensible repository size is 16MB. It's 19MB with ECC and no need. L3 Cache is an associative repository of 16 sets and 16 set modes. The repository line is 6 4 bytes with two blocks sent to the data bus. Each episode has 256 data bits, 32 ECC bits, and two idle bits. Each local address is 40 bits long. Cache size achieved by reducing the set. The set can be adjusted to 16K, 8K, and 4K. This summarizes the cache configuration of the three main configurations, 16M, 8M, and 4M. The established organization remains 16 in all three settings. Set reduction is used to achieve the target repository size. The data path and control concept associated with the tag list are designed to support the widest tag range from 4M configuration.

Low power consumption is one of the main objectives. Many low-energy technologies are used to achieve design objectives. N sleep design is used in the SRAM system. Sleep bias can be set between 50 and 300 mV. The Psleep design was used in the repository peripherals such as decoders, WL drivers, and I / O repository. Floating bit beams are used in local bit lines. The shutdown method can be used with invalid subarrays in 4M and 8M configurations. The wake-up counter has been used to improve the performance of the sleeping design. Long channel devices are widely used to reduce power consumption. Here shows the n sleep design SRAM of the same members.

The bias circuit controls the temperature change of the visible area. Sleep bias is so structured and manifold in such a way that the variability in the visible area at process angles and temperatures is within the specification. Added nMOS diode and pMOS pulse to control temperature change of visible area use of the diode structure as sleep bias is described in [6] and [7]. In contrast, the visual support of this publication is largely controlled by the flexible electric current, and the diode structure is used to reduce flexibility.

When VSS visible exceeds transistor threshold voltage (Vt), MND is turned on because Vgs is greater than Vt. Since Vds is larger than Vt, the MPB pull function is also very important. As a result, virtual VSS is limited to 1Vt or less. As the temperature rises, the Vt of MND and MPB decreases, the transistor becomes more powerful, and the bit cell leaks more, preventing significant increases in visible ground. Cowardice. This indicates a change in the visual environment due to the temperature and fluctuations of the device. During closure, sleep bias is disconnected from the ground, the pMOS diode is shut off, and the visible ground floats upwards.

PMOS sleep is used in the scanner and I / O cache. This includes the repeating column (MUX), the write driver, and the sensor amplifier. Figure 9 shows the sleeping design of the decoder and the WL driver. Figure 10 shows the I-O Core repository psleep design. When repository I / O is in sleep mode, thin lines float to minimize leakage of the thin line. This also avoids DC throughout the MUX reading / write column when operated by Vcc. The Vcc decoder and Vcc repository for virtual I / O is a fixed setting, limited only to stay within a given Vcc drop point so that reasonable values should be found in the word line during sleep mode. The given drop from Vcc of the word is chosen to achieve reasonable energy savings without producing significant sound. The nMOS diode (MND) is used to limit the decrease in power in visible Vcc.

Slight line chargers are not included in the design of sleep transistors to avoid long charging times and to meet the strict line-of-line measurement requirements that amplify different sensors. The timer has been deliberately removed from sleep construction because time is of the essence. Whenever possible, long Le transistors are used in the timer without sacrificing the accuracy of the signal edge. With the cache peripheral psleep design, it saves up to about 6W. All 16 routes are contained in one block, mainly hiding the delay penalty by opening the sleep transistor and reducing the number of blocks and Subarray unlocked.

This indicates the allocation of unwanted bits to each part of the data repository line. Each component has two inactive columns, and each repository row can fix two random bugs. This repetition program also deals with errors in line-based bit cells. You can adjust up to 8 sets of bit positions. Three bits of local address used to control MUX no longer active. The defunct MUX bit is selected so that the total independent adjustment value remains the same in the 4M and 8M configurations. Particle size of part subarray. The now defunct MUX is used in the global data route. Unnecessary columns are used for each tag. You can fix a random error in any of the entries. Here, unnecessary bit allocation for each tag. A total of 4 possible locations can be adjusted independently of each other. Most SRAM cells are in sleep mode with reduced operating bias, which uses ECC technology and Intel Cache Safe to ensure the integrity of the repository. Complete post-silicon alignment specifications and statistical analysis are performed to ensure that the product meets the reliability specification. For the data system, each of the 8 bytes data is protected by SECDEDECC. Larger tag size is also protected by SECDEDECC. In addition, the database is protected by Intel Cache Safe Technology. If ECC adjustments are received after production, up to 32 repository lines or up to 2 modes per set may be disabled. This has little effect on performance. Cache line will be disabled until reset. Power repository line malfunction has been used to correct Vccmin sensitivity to potential errors. The live history table (Pellston Engine line) traces the random ECC events of each repository line. Unlike previous uses, this flexible cache line functionality allows you to separate random cache holes. If the ECC error occurs for the first time in the cache line, it may be a soft error. If the ECC error occurs twice in the same area, it means it will not be a soft error and may be a physical problem as an error. Potential feature or low VCC sensitivity.

Cache security technology is enabled for both self-testing of power and normal operation of. When a 1-bit ECC error is detected, the ECC intelligence informs the Pellston engine of the error information. This includes a set of Global Bus Queue (GBSQ) and route information. If the specified set and route meet the error first, the input is inserted into the Pellston engine line. Cash method is not disabled or cleared by default. If the error is the second error in the same area, the Pellston engine intervenes and deactivates the corresponding archive lines. Many features of the test are performed to ensure that the 16MB -L3 cache is checked and executed. PBIST is widely used in both error correction and silicon test cover. Low yield analysis, pWWTM, and stability testing methods are used. A series of indie diversity was available to employ indie diversity. To fix the speed limit there are many clock circuits.

The reported processors have a large on-die L3 repository and transistors x86 processors. Low power design allows you to save more than twice as much energy in your repository. Extensive design, reliability, and testing capabilities ensure warranty production and reliability. The processor is marked to meet the power envelope and frequency terms.

2.2 Power Reduction Techniques for an 8-core SoC

Stefan Rusu et al. did an excellent job of introducing power reduction solutions and managing the 45nm, 8-core Nehalem-EX processor. Using multiple clocks and voltage sources to reduce power consumption. Long station equipment and cache sleep mode are used to reduce leakage. Core acquisition and cache improve productivity yield and allow for a wide range of products from the same silicon chip. Clock and Power gating reduce power consumption of invalid blocks. Microcontroller Only controls voltage and frequency, as well as current events and temperature driving points. Load power is reduced by switching off the I / O connection of the power controller and the switch phase to improve the efficiency of power conversion.

Power efficiency is the main design goal of this 45nm, 8-core, 16-thread SoC processor. The total performance of the hot design remains at 130W, in line with previous generation processors, but the number of cores per socket increases by 2 cores each year. The processor has 2.3B transistors, which are used in 45nm CMOS process technology using High-K metal gate dielectric transistors and nine layers of copper wires [8]. IOOO X for PMOS transistors compared to 65nm process production.

The processor is integrated into a 14-layer live grid system package (545), 1567-rand, 40 mil pitches, and an integrated heat exchanger. The processor supports multiple platform configurations, from dual processor options to quads up to 8 options. The image of the processor chip whose main block is marked [9]. The shared L3 repository is divided into eight pieces. Each piece of cache works with the processor core, but the entire L3 repository is considered by all cores as one large, shared repository. The upper side of the floor plan consists of four point-to-point quick path interconnect (QPI) connectors operating at 6.4 GT / s, while the lower side maintains a visible connector. The central channel consists of two memory controls, two harp links in the cache, an 8-hole router, a power control unit, and a visual system interface including the OFT control box. The Uncore Clock generator and fuse box are also located in the center of the channel. Primary and repositioning is a method of crop improvement that has been made possible by placing many of the same contexts and the repository in one place. If one of the cores has a production feature, close the spine and restore that part as a low point. The same applies to large repositories that cannot be repaired by duplicate built-in repository. Key cache and cache fragments are disabled in horizontal alignment, but disabled cache fragments and key fragments do not need to be aligned. Few combinations are possible, which cover most of the chip width and allow for better recovery.

To allow all eight cores to be tested seamlessly, the chip is redesigned and each context will see the aligned cache piece as its dedicated repository. In this way, all eight cores use a test pattern simultaneously, allowing the tester to quickly determine which cores are appropriate and which cores need to be disabled. Ondie, a one-time electrically adjustable fuse used for content acquisition and storage. The initial die triage and merging are performed on the wafer type and re-validated during the final test.

The disabled context has a clock and power control and is logically disabled to avoid using power in a customer system with a function line that does not give value to the user. The deformed cocoon piece is also closed and powered down, reflecting infrared radiation back from the chip, with only six active cores. A light gray shade indicates that the flexibility and leakage are high and that photon emissions are increasing. Two closed cores are dark in the image because they do not emit infrared light. One bright spot for each course is due to the heat sensor, which is powered by a clean PLL voltage field and is not affected by the main power gates.

This shows a block diagram of the power control unit (PCU). The PCU is a small controller with a few circuit breakers that control the output of the main voltage and the main power gate, controls the transition between different operating conditions of the application, and controls the detection and response of thermal events. The PCU detects the output voltage of the main voltage and the sensor of temperature and power required in each context.

The microcontroller calculates the voltage ID bit that adjusts the frequency with the external power controller of the core PLL. The PCU also controls the temperature sensor by lowering the voltage and frequency to maintain the chip temperature within safe reliability limits. If the temperature or operating voltage exceeds the reliability limit, the PCU shuts down the PLL and the external power controller to protect the chip processor from catastrophic failure.

A guideline to reduce idle energy is to reduce energy consumption in unused blocks. An example of reducing the power consumption of an unused I / O connection. There is a platform setting where the I / O connection is not finalized. Dual processor platforms usually use only two or three links (of the 4 available on each chip) and leave the rest unused. The partially implemented quad processor platform also has multiple unconnected links. To detect these conditions, they implemented the connection detection circuit. It detects the presence of the Rx terminator at the other end of the link. At on the left side of the figure, there is an Rx term that pulls the connection to less than half the VCC level (the Rx termination resistor on the receiving chip is much lower than the pull-up resistor on the driver chip).

Stefan Rusu et al. did an excellent job of introducing power reduction solutions and managing a 45nm, 8-core Nehalem-EX processor. Using more watches and power sources to reduce energy consumption. Long channel equipment and cache sleep mode are used to reduce leakage. Critical discovery and repository improves production productivity and allows for a wide range of products from the same silicon chip. Clock and Power gating reduce power consumption of invalid blocks. Microcontroller controls voltage and frequency only, as well as current events and temperature driving points. Loading capacity is reduced by turning off the I / O connection of the power controller and the switch phase to improve the efficiency of the power conversion.

Power efficiency is the main design goal of this 45nm, 8-core, 16-thread SoC processor. The total performance of the hot design remains at 130W, in line with previous generation processors, but the number of cores per socket increases by 2 cores each year. The processor has 2.3B transistors, which are used in 45nm CMOS process technology using High-K metal gate dielectric transistors and nine layers of copper wires [8]. IOOO X for PMOS transistors compared to 65nm process production.

The processor is integrated with a 14-layer grid system package (545), 1567-rand, 40 mil pitches, and an integrated temperature switch. The processor supports multi-platform configurations, from dual processor options to up to 8 options. The image of the processor chip whose main block is marked [9]. The L3 repository is divided into eight pieces. Each piece of cache works with the processor core, but the entire L3 repository is considered by all cores as one large, shared space. The upper side of the floor plan consists of four point-to-point quick path interconnect (QPI) connectors operating at 6.4 GT / s, while the lower side maintains a visible connector. The central channel consists of two memory controls, two harbor link connectors, an 8-hole router, a power control unit, and a visual system interface including the OFT control box. The Uncore Clock generator and fuse box are also available in the center of the channel. Basic and repositioning is a method of plant development that has been made possible by placing many similar themes and storage in one place. If one of the cores has a production feature, close the spine and replace that part as a low point. The same applies to large repositories that can be repaired with a built-in repetitive storage component. Core pieces and cache pieces are locked in a horizontal alignment, but disabled cache and key pieces do not need to be aligned. Few possible combinations, which cover most of the chip width and allow for better recovery.

To allow all eight cores to be easily tested, the chip is redesigned and each context will see a piece of aligned cache as its final dedicated area. In this way, all eight cores use a test pattern simultaneously, allowing the tester to quickly determine which cores are appropriate and which cores need to be disabled. Ondie, a one-time electrically adjustable fuse used for content acquisition and storage. The first triage of death and fusion is performed on a wafer type and re-confirmed during the final examination.

Disabled content has a clock and power control and is logically disabled to avoid using power on a client system with a performance line that does not give value to the user. The crippled cocoon piece is also closed and degraded, showing infrared radiation from the chip, which has only six active coils. A light gray shade indicates that fluctuations and leaks are high and that photon emissions are increasing. Two closed cores are dark in the image because they do not emit infrared light. One bright spot for each study is due to the heat sensor, which is powered by a clean PLL electric field and is not affected by large power gates.

This shows a block diagram of the power control unit (PCU). A PCU is a small controller with a few circuit breakers that control the output of the main voltage and the power gate, controls the transition between different operating conditions of the application, and controls the detection and response of thermal events. The PCU detects the output voltage of the main voltage and the sensor of temperature and power required in each context.

The microcontroller calculates the voltage ID bit that adjusts the frequency with the external power PLL controller. The PCU also controls the temperature sensor by lowering the voltage and frequency to maintain the chip temperature within safe reliability limits. If the temperature or operating voltage exceeds the reliability limit, the PCU shuts down the PLL and external power controller to protect the chip processor from catastrophic failure.

Guide to reducing inefficiency to reduce energy consumption in unused blocks. An example of reducing the power consumption of an unused I / O connection. There is a platform setting where the I / O connection is not completed. Dual processor platforms typically use only two or three links (out of 4 available per chip)

2.3 Parallel Mutual Information Based Construction of Genome-Scale Networks on SoC

Building a whole genetic network from big data genetics is an important issue for system biology. Some technologies have been developed, but most cannot handle genome-wide network reconstruction, and other technologies that can support it, require larger clusters. Sanchit Misra, et al. have done excellent research by proposing a solution to the Intel SoC Phi coprocessor [10], using its similarity involving multiple levels including multiple x86-based cores, multiple cables per core, and vector processing units. They also introduced a solution for Intel SoC processors. Our solution is based on TINGe [11], [12] how to update the same fast network using the same information and authorization tests [13] to assess statistical significance [11]. By building a 15,575 genetic network of the Arabidopsis plant from a 3,137 microarray test in just 22 minutes, we demonstrate the first consideration of a plant genome control network on a single chip. In addition, statistical comparisons of integrated data on Intel SoC Phi coprocessor provide lessons that can be applied to other domains.

Building or reversing genetic network engineering in genetic testing over multiple test conditions is an important issue for systematic biology. Such networks can be used to identify statistically significant relationships between genetics and model control models within a mobile system. Differences between such networks in different sets of conditions can be used to study genetics and speech patterns that are important to the behavior of an organism in all areas of interest. At the heart of allowing such programs is the problem of building networks from data. The input problem for network construction is a rectangle with lines such as lines, columns as shapes, and inputs that show a limited amount of statement after processing the statistics to eliminate the results. of sound and diversity testing, Gene expression matrix. The output is a graph, where the nodes represent the genes and the edges represent the relationships of interest. This may vary depending on the design of such a network. For example, the edge represents co-expression in the co-expression network and reflects the control effects on the genetic control network. In the Bayesian network, the network represents only the integrated feature of the hidden distribution of hidden opportunities.

When the G [i] viewing vector is loaded into the L1 spinal cord, it ensures that all pairs (G [i], G [j]) are processed in the same context. Hyper-threading on the core shares L1cache, so assign hyper-threading to work with the same G [i] and different G [j] viewers. This shows an example of a multi-threaded n¼ 16 method with 5 cores with 4 strands of each spine. loop span line 312 diagram. 1 calculates the upper triangle of the 1616 matrix represented by the cells shown in white. For each line in the matrix, Cz (where z 2 f0; 1; 2; 3; 4g) refers to the nucleus to which the line is assigned. Number in a cell means the corresponding hyper-thread of the core to which that cell is assigned. Assign 1 context for each row in the first combined 5-row block. To spread the load on each course, assign one theme to each row, while releasing the cores distribution plan for each subsequent block in row 5. The MI figure does enough work to hide memory delays and does not require cache blocking. Additionally, the approval test section requires a predefined vector observation Y (row 7) to be in the L2 repository.

Adjustment of Scalar static instructions, as well as co-processor and vector-enabled launch of the CPU. Clearly, the use of vectorised significantly reduces the number of commands required. However, there are a few reasons why vectorization reduction instructions are incorrect. The critical area has instructions for accessing memory with very high density (88 percent or 69 percent, of vector usage). Most of these memory access W and P’XY, which can be found in the L2 repository. In addition, access to memory is rare. Many uncommon access is classified as L2 and requires aggressive downloads from the co-processor. However, adding a download to P'XY affects performance, so you can add a download to W. This is because (i) the total number of commands when you add this download is large, (ii) too many preload instructions that cause the download cache block, and (iii) the download delays for other required commands. It may be due to inability to hide. In addition, the amount of vectorization is limited, as shown in the table. Some commands in both processors are scalar commands. Additionally, for co-processor and CPU, approximately 50% and 35% of vector commands are low vector power distributions, respectively. Under appropriate vectorization conditions, the co-processor only needs, which is part of the CPU cycle calculation. However, because there is no vectoring space in this loop, the number of cycles required to perform the same amount of work when both processors work in the same context is not significantly different. there is no. If both processors use all cores, the higher the number of cores in the co-processor the better the performance.

Measure the impact of various improvements on common integrated loops of opportunities by measuring appropriate performance indicators using a VTune amplifier. Specifically, it measures how different configurations affect the number of commands issued, the number of commands to access the issued memory, and the amount of memory accessed at different repository levels. Performance data was used for this study. The benefits of using a particular setting also depend on the state of the entire code. This is because it affects the level of the repository beat. The results, however, show the accelerated acceleration of a variety of preparations. This section takes a different approach. It reports the calculations after disabling and reports only one setting at a time from the best use of. It takes time for the co-processor to adjust the initial software download, which must be done separately for each code change. To avoid this, it disables complete implementation setting that does not use software pre-downloads. Report performance indicators in Tables 5 and 6. The original code corresponds to the one shown in FIG. The total command number and the number of memory access commands usually calculate the base of the code. L1 cache beat rate is defined as the average amount of memory attained that hits L1 cache in the total amount of memory that goes into L1 cache. L2 and LLC hit ratings are also defined.

The basic L1 beat rate is very high. This is because this implementation is completely scalar. Use of the scale reads 4 name bytes at a time. The vectorised implementation, on the other hand, reads 32 bytes and 64 bytes simultaneously on the CPU and coprocessor, respectively. Clearly, in scalar applications, most memory access commands reach the same repository line compared to vectorised implementation. Compared to basic usage, full implementation uses very small memory access commands. With the location of the data Disabling data setting (Section 3.1) leads to a slight decrease in cache rate and a slight decrease in speed. Disabling the second loop vectorization increases the number of commands rather than disabling the first loop exposure. This is also reflected in the corresponding acceleration. We are launching the first single-chip reconstructed gene control network using the Intel SoC Phi coprocessor. This is an act previously performed on the same computers with a large, distributed memory. An important use of such whole-genome networks is the development of sound ideas for understanding the interaction of genes and novel genes in signature methods with incomplete features. The huge cost of using a single coprocessor-based workplace means that such a system-based approach to system biology can be made more widely available. We present a set of development strategies to the Intel SoC Phi coprocessor related solution for this application. This is useful for moving other applications. In particular, both random shuffling and shared information occur in other applications, including outside the biological field of calculation.

2.4 Optimizing Purdue-Lin Microphysics Scheme for Phi Coprocessor

The demand for more precise weather forecasts is rising as a result of sudden, severe weather changes. The frequency and severity of these catastrophes have both risen due to climate change. The weather model's run-time can be sped up and its accuracy increased by optimising its source code. One such meteorological model is the Weather Research and Forecast (WRF) model, which was created for both atmospheric studies and numerical weather prediction (NWP). Dynamic solvers and physics schemes are just two examples of the various parts that make up the WRF software architecture. A rather complex microphysics system for the WRF model is the Purdue-Lin scheme. Six categories of hydro meteors are included in the plan: water vapour, cloud water, raid, cloud ice, snow, and graupel. This plan is very suitable for massively parallel computations because there is no interaction between the horizontal grid points.

The Purdue-Lin microphysics scheme's optimization results were proposed by bright researchers Jarno Mielikainen et al. To better leverage numerous vector units inside each processor code, these optimizations further improved code vectorization. The changes made have resulted in a 4.7-fold increase in speed over the original, unaltered Purdue Lin microphysics code operating natively on the SoC Phi7120P. Similar to this, the Purdue Lin Microphysics scheme's performance in a dual socket configuration with an 8-core Intel SoC E52670 CPU has increased 1.3 times compared to the original code with the same improvement.

Extreme weather events like the summer heat have become more frequent and more severe as a result of climate change [14]. Consequently, there is a growing need for weather forecasts that are more precise. Shorter execution times or more precise weather forecasts will be obtained by optimising the weather model. The Weather Research and Forecasting Model (WRF) [15], which was created for both numerical weather prediction (NWP) and atmospheric research, is an example of such a weather model. The WRF model is a collaborative NWP model of the next generation created for atmospheric research and operational prediction. WRF is the most commonly used community weather forecast and research model in the world, being utilised in more than 150 nations. Dynamic solvers and physics schemes are just two examples of the various parts that make up the WRF software architecture. To solve huge flows, numerical models are used. The Purdue-Lin microphysics technique, however, was what we opted to enhance.

For each physical component and many dynamic core possibilities, the WRF model offers a variety of schemes. The completely compressible non-hydrostatic Euler solver serves as the foundation for the WRF kernel [16]. Its software architecture also makes use of several CPU cores. WRF also has a 3D variability data assimilation system that may be used to simulate meteorological events at various sizes, from metres to thousands of kilometres. These adaptations allow the WRF to be used in a variety of domains, including tropical storm predictors, veld fire simulations, air quality modelling, regional weather forecasts, and storm scale assessments. Weather forecasters have an insatiable need for computer power. By increasing WRF computing power, you can use more ensembles, make higher decisions, use less time simulation steps, or use more sophisticated models. All of these are good for more accurate weather forecasts. Using these combinations requires additional computing power. Collection forecasting is an interesting aspect of this desire for the power of integration. Weather forecasts tend to fluctuate input and are sometimes surprising. To measure sensitivity, the computer model is running times from the first slightly different positions. Ensemble guessing systems are usually designed in such a way that each member has equal strength and the small initial difference in input is similar to the input uncertainty (visual value). Preparation can lead to more integrated performance. This allows Tuck to more accurately predict predictable uncertainty and potential weather events.

Recently, many research teams have used graphics processing units (GPUs) to speed up the WRF kernel. The advantage of using a multi-core WRF architecture over a GPU is that the full WRF code already works in legacy CPU and Intel MIC architectures. This allows you to focus only on the use of the code and not waste time getting your code working. Conversely, using an existing code in a GPU requires a lot of effort. To get the most out of the WRF acceleration GPU [17] - [18], the entire WRF must be included in the GPU. The advantage of using WRF with GPU is that performance can be improved if all code is embedded in GPU performance. Not all 700,000 lines of WRF code can be transferred to Field Programmable Gate Array (FPGA). The WRF does not have enough hot spots to accelerate the WRF with effective development efforts using FPGAs. FPGAs are better suited for problems with higher levels of calculation and memory than WRF. The arithmetic capacity of the WRF module is relatively low and operand reuse is limited. In addition, the larger WRF module operating module will overflow with the current coprocessor repository. Thus, the WRF optimization process begins with rewriting the code, revealing vectorization, reducing the memory size of the cache, and better utilizing cache memory to show that the removal of temporary variables is the most effective [19] of WSM5. The great advantage of using SoC Phi WRF acceleration is the fact that you can see the full development of WRF after each development step. This is because WRF operates in native mode in SoC Phi.

In arithmetic, the visible part is horizontally independent and has only a direct dependence. Therefore, they are very good at calculating the same data. In contrast, the Dynamics Kernel calculation also shows the horizontal dependence between adjacent grid points. As a result, different optimization strategies are needed for the dynamic kernel. The Purdue cloud model, which is detailed in Chen and Sun, served as the inspiration for the concept. There are six classes of hydrometeors in the Purdue Lin microphysics scheme. Rain, cloud ice, snow, sleet, water vapour, and clouds with water. Lineartal is the foundation for all parameterization production circumstances. Numerous modifications have been made to both Rutledge and Hobbs, including ice deposits and saturation correction by Taoetal. Six classes of hydrometeors are included in the model, and they are all handled using a lot of parameters. Because the water and ice particles in the cloud are so minute, their terminal velocity is not thought to be particularly significant. Hail, snow, and rain all have noticeable terminal speeds.

An overview of the overall time spent executing Purdue Lin calculations on Intel SoC Phi coprocessors and SoC CPUs in 1 and 2 socket setups is shown. The original code failed to utilise the SIMD unit correctly, as seen in Table V. Code vectorization has therefore enhanced the Intel SoC Phi coprocessor's performance relative to the CPU. The secret to good performance on both the CPU and the Intel MIC architectural platform was the utilisation of multithreading and vectoring. It took good code vectoring and multithreading for the Intel SoC Phi coprocessor to ultimately outperform the Intel SoC processor in terms of performance. Additionally, the outcomes demonstrate that the Intel SoC Phi coprocessor performs better than the 1-socket and 2-socket CPU setups after code optimization. Additionally, the improvements brought to the with the Intel SoC Phi coprocessor are considerably more crucial because they result in the larger acceleration benefits of the. Lin Purdue described how the microphysics approach was optimised. As a result of the optimization, the performance of SoC Phi has improved by 4.7 times. Other WRF physical components also benefit from similar code optimizations. Due to the different structure, dynamic cores require different optimization approaches. In addition, the optimization has improved the performance of the Intel SoC E52670 by 1.3 times. Therefore, both the SoC CPU and the SoC Phi coprocessor will benefit from the same optimization work. To make greater use of the vector units on each CPU, the optimization included improved code vectorization. Additionally, by streamlining access to a few intermediary data arrays, memory access has been enhanced.

Knights Landing (KNL), the codename for the next-generation Intel SoC Phi chip, is not binary compatible with the first-generation coprocessor. This is because the original 1st generation many core instruction is not the same way that the AVX512SIMD instruction is encoded (Intel IMCI). But we anticipate that the code optimizations we've made so far will also be successful in KNL. The AVX512 instruction encoding will also be used for vector instructions on next Intel SoC systems. The first step towards a thorough overhaul of the WRF code is to work on optimising the Purdue-Lin schema. Code optimization is crucial for effective performance on multi-core processors and many-core co-processors, as you can see from the speedup number.

2.5 Optimization of EULAG Kernel on Intel SOC Phi Through Load Imbalancing

A well-known method for improving the performance of scientific applications in a parallel architecture is load balancing. In fact, balanced apps maximise processor usage because they don't waste time waiting at synchronisation and data exchange points. The load balancing strategy, which enhances the performance of parallel applications, is called into question by this white paper. Create first the requirements that your application's performance profile needs to meet in order for load balancing to provide it the optimal performance.

The performance profile on the most recent parallel architecture, Intel SoC Phi, deviates dramatically from these requirements, according to the studies by Alexey Lastovetsky et al. that provide us with a means of employing the EULAG [20] MP DATA kernel, a Real-Life Scientific application. On the basis of this finding, we suggest approaches for utilising load imbalances to enhance the performance of scientific applications. This method finds partitioning that doesn't always balance the processor load while reducing computation time for parallel data applications by using the functional performance model of the application. Apply this strategy to the optimization of MPDATA for Intel SoC Phi [21], [23]. According to experimental findings, this highly tailored load balancing application's performance is further enhanced by an additional 15% with the proposed load balancing technique.

A well-known method for increasing the computational capability of parallel scientific applications is load balancing. In fact, Intuition contends that, in contrast to unbalanced programmes [26] - [30], balanced applications maximise processor efficiency by avoiding wasting time waiting at locations for synchronisation and data interchange. The study questions the applicability of load balancing techniques for maximising the processing capability of parallel applications. Let's start with comprehending the drawbacks of the load balancing strategy. A well-known method for increasing the computational capability of parallel scientific applications is load balancing. In fact, Intuition contends that, in contrast to unbalanced programmes [26] - [30], balanced applications maximise processor efficiency by avoiding wasting time waiting at locations for synchronisation and data interchange. The study questions the applicability of load balancing techniques for maximising the processing capability of parallel applications. Let's start with comprehending the drawbacks of the load balancing strategy. It can be applied, for example, to ocean currents, turbulent areas, urban current simulation, and numerical weather prediction (NWP). This solver, which was first created for conventional HPC computers, is currently being reworked for the most recent HPC platform. To run better on the Intel SoC Phi coprocessor, MPDATA in particular has recently been rebuilt and improved. In our experiment, we found that the MPDATA performance profile significantly deviated from the criteria for the load balancing technique's applicability. Based on this discovery, a generic approach to improving the performance of scientific applications by load imbalance and the best (potentially imbalanced) setup of data parallel applications with a set of related data processing has been developed. Offer a finding algorithm. As an alternative, this approach finds partitioning that optimises computing time without necessarily balancing the load on the processor by using the functional performance models of the programme. Last but not least, use this approach to improve MPDATA on Intel SoC Phi.

an approach to load balancing that enhances the performance of concurrent scientific applications on both homogeneous and heterogeneous platforms. Additionally, we use these methods to define the circumstances in which computational performance is maximised. A well-liked and frequently applied method for improving the efficiency of scientific applications on parallel systems is load balancing. You can categorise a load balancing algorithm as static or dynamic. Static techniques (such those based on data partitioning) call for prior knowledge of parallel platforms and applications. Both compile time and run time data collection are options. Because they rely on an accurate performance model as an input to forecast future application execution, static algorithms are also known as future predictions. Static algorithms are particularly helpful because they don't call for data redistribution. However, these algorithms cannot fit in a dedicated environment where the load changes over time. Powerful algorithms (such as job scheduling and task theft) distribute the load by moving the work done smoothly between processors during calculation. Powerful algorithms do not require prior knowledge about usage but can include important connections due to data transfer. Powerful algorithms often use standalone partitions as a first step because of their proven close-to-right communication costs, limited load loads, and high-level editing.

The performance profiles of real science applications on modern social media platforms show that load balancing can deviate significantly from the situation, which ensures that computer power is always maintained. Based on this recognition, we propose an upgrade method that uses the power profile to optimize the application for its performance. This function creates an MPDATA performance profile on Intel Phi. MPDATA is a major component of EULAG (Eulerian / Semi-Lagrangian Fluid Solver), a standard calculated model developed to mimic the flow of hot liquid over a wide range of scales and optical conditions. A carefully crafted compact data setup on the Intel Phi 61-core divides 3D nl line segments into four n2n2l sub-domains, each modified in a group of 15 cores. This application configuration is a complete configuration of the specified load rating.

MPDATA is commonly used in long-term estimates that require thousands of time measures, such as numerical weather forecasts. Therefore, in the context of actual imitation, MPDATA will be used thousands of times with the same domain size. Importantly, in dividing the space provided, the action speed of the action step is the same for each step of time, whether the first step, second, or 1234. Using these facts, you can use a few first-step steps to build up speed, get the right workload division, and apply that complete separation to everything else you do. In this way, MPDATA can adapt to it. Our design does not build the full speed of the work during the run, but a small part of it. This is sufficient to determine the optimal distribution of a particular domain. Specifically, for the size of a particular problem, which is n the focus of the value of the equation solution, only m is divided, and m is the approximate distance of the values around the equation solution. Build speed as a task.

The proposed method has been successfully used as a time-consuming method for the implementation of MPDATA practice. In the first step, the machine creates the required part of the speed function and then uses algorithm 1 to get the best partition. After obtaining the best partition, MPDATA will continue to operate normally on this fixed partition. The dynamic surface of this process may be due to three additional functions, (i) the scaling of specific distributions, (ii) the use of a data classification algorithm, and (iii) redistribution. Generally, in order to accurately measure the speed of any distribution, you need to repeat the same step several times to ensure an acceptable accuracy of the measurement. In the case of MPDATA, however, the calculated stiffness of each MPDTA step is stable, proving that only one duplicate is sufficient. In the second pass, the test field speed ratings did not improve as the resolution returned by the first Algorithm changed. This MPDATA property has been able to significantly reduce related overhead. Lastly, if all data in the primary memory and archive needs to be reloaded regardless that the re-equation takes place, the redistribution will take place within successive steps. Balance does not include additional overhead associated with data location. In all experiments, MPDATA practice was able to obtain appropriate variations in steps of less than 20 times. In addition, the total value associated with the proposed operating time has always been less than 0.05 seconds. Given that the number of first-time steps involved in building a model has never exceeded 20, this has never exceeded 0.0025 seconds more than each step of the time involved in finding the best separation. It means. If the size of the domain is 240 x 240 x 128, this counts less than 2% of the action time, and this% continues to decrease as the size of the domain increases. For numerical weather forecasts using MPDATA, simulation runs in thousands of steps over time (over 120 x 112 x 128 two-day weather forecast). Therefore, the potential cost of this development method is less than 0.005 percent of the total application time. Finally, the cost of the above is not payable. Computer distribution in these early stages is incorrect, but the overall performance gain can reach up to 15%. This is especially evident in long-term comparisons. Assuming that good rot occurs in the first 20 steps, the cost of the proposed adaptation process is generally stable regardless of the size of the problem or the number of steps in time. If the number of time steps is too low and / or the size of the domain is too small, the complexity of the process has a small effect on performance, but in all other cases it has no adverse effect.

Modern computer nodes are characterized by both a growing amount of processing material (possibly diverse) and a high level of complexity in their integration. Various resources such as repositories and data links are shared in a seamless and consistent manner. This makes building efficient applications of such a platform a very difficult and difficult task. It is unrealistic to expect that the performance profile of real science applications on these platforms will always be comfortable and smooth to match the standard load balancing techniques used to reduce calculation time. it is. Therefore, a new way of doing things that is not based on such growing ideas is necessary. This function introduces such a method and demonstrates its usefulness in preparing real applications on modern HPC platforms.

2.6 Multi-Core and Many-Core Memory Systems SoC

The sheer amount of data has exploded at an unimaginably high level over the past decade. Therefore, extensive data analysis has focused on recent times. Many frameworks are designed for data analysis, including Hadoop [34], Spark [35], etc. Most frameworks are based on important or multiple memory systems, so developers and users need an in-depth understanding of properties in order to find them. much on their hardware.

Researcher Yuxuan Xing has et al. has done an excellent job of providing an in-depth study of the multidisciplinary and multimedia memory system and describes various aspects such as context, cache, memory, and on-chip networks. In addition, they have developed a simple and effective way to share lies over the archive that can reduce the large number of LLC loading / store instructions and LLC cache-misses.

To take full advantage of hardware resources, developers and programmers should make full use of the features of the memory system. However, users who want to analyse the performance of their system often lack the same detailed documentation. Therefore, it is suggested that you use the Micro Benchmark 4304 system to capture the properties of many basic and multiple memory systems. This program can be used in conjunction with the application requirements model to analyse performance in detail because it can guide by analysing the performance of properties. To show how it works, they built a high-memory model for two memory systems. SoC-server is based on Intel SoC E52692v2 multi-core processor and Knl-server is based on Intel Phi7210 many-core processor. SoC E52692v2 is also used in the Tianhe2 supercomputer [36], Trinity Supercomputer is based on the SoC Phi 7250 processor (advanced version 7210) [37].

Tests are performed specifically on the SoC server as well as the knl server and bench-tagged with respect to the data sets of the physical and real-world graph. As a result, the native Ligra (Na-Ligra) runs faster on the SoC server than the knl server while not appearing as a major difference to the well-designed Ligra (Op-Ligra). Overall, Op-Ligra behaves better than Na-Ligra on both SoC-server and knl-server. The configuration effect is more obvious on the knl server than on your SoC server for both PR and TC, and TC has achieved significant PR-related performance enhancement for both.

servers.

Tests are performed directly on the SOC server and knl server and are tested in relation to real-world graphical data sets. Results show that native Ligra (NaLigra) works faster on SoC server than knl server, but with advanced Ligra (OpLigra), no significant differences were observed. Overall, OpLigra works better than NaLigra on both SoC server and knl server. The effect of configuration is more pronounced on both PR and TC on knl servers than on SoC servers, and TC has experienced significant performance improvements compared to PR on two servers.

Both CPU capacity and memory access are known to have a significant impact on performance. SoC Server has two SoC E5 sockets, each socket with a maximum performance of 211.2 G. flops. The knl server has a SoC Phi 7250 processor, but

offers a maximum performance of 5.3 T flops. Although the knl server is more computer-friendly, NaLigra spends more time on the knl server than on the SoC server in both PR and TC statistics. Therefore, for further analysis, calculate the maximum amount of archive / store orders (LLC loads and LLC stores) as well as the maximum amount of archive (archive loss) generated during the calculation. Typically, the number of LLC loaded, LLC stores, and cache miss increases by graph size. Statistics on knl server generate more upload / store commands compared to SoC server, resulting in more repositories. Additional commands may be triggered by various cache categories. The SoC E5 has three repository levels, the KNL has only two repository levels, and the L2 repository has, which has been instrumental in reducing the final level of access to the repository. Loss of storage increases as the load and store orders increase. KNL also has a total of 64 MB of final level cache (64 tiles and 1 MB LLC per tile), and each LLC is very secretive on its own tile. That is, one series needs to convert the required data to another. Save to LLC or local LLC. In the worst case, each LLC contains the same 64 tile data, so it could be 1MB LLC. Therefore, SoC E5 fibers can directly access the entire 30MB L3 repository. This greatly helps to reduce the number of warehouse losses. The cache misses the results of accessing additional memory. Memory delays are several times longer than the cache in the processor, so it takes hours. In particular, differences in cache hierarchies and links cause differences in LLC load / store value and warehouse loss, which in turn affects SoC server and knl server performance.

Multi-core and many key structures are very different. Understanding buildings is very important for engineers to build efficient systems. The study used SoC E5 and KNL models to make clear comparisons. It also deals with the issues of archive release that many programmers often overlook and provides detailed performance analysis with image details. Many basic memory systems can provide a lot of low cores and seem to prefer high memory bandwidth, but they seem to prefer longer memory delays compared to multi-core memory systems. Missing cache sharing is present in many large data analytics. By fixing this problem, you can fully improve compliance and performance improvements. Many basic memory systems are more sensitive to data dependence due to the complex interactions between the context and the two-phase repository category. Engineers need to pay close attention to potential problems: B. Data volatility, particle size, task scheduling, and contextual data structure.

2.7 Package Design Optimization for Intel SoC

The SoC system-on-chip combines great density and low power with the high performance of the Server SoC. the significance of the analysis of the cost-performance trade-off for SoC packages. Find out how to calculate the low-cost package components—size, footprint, pin map, and number of layers—without having an impact on system performance. explains the high-speed differential cost performance optimization and 10 GbE signal integrity architecture. The package's low power architecture and power supply capabilities, including a fully integrated voltage regulator, are also demonstrated in this document.

By introducing Intel's first 14nm Broadwell core to the server industry, the SoC D Processor [38] completely revolutionises the low-performance area of the X86 server market. This system-on-chip (SoC) architecture, also known by the codename Broadwell DE [38], incorporates many of the features present in both high-end SoC chips as well as the most recent manufacturing techniques and high-speed networking. Make a product with an integrated low power package. 20-45W design points now have access to SoC-class RAS virtualization features thanks to Broadwell DE. This design includes important servers, network I/O, communications, and storage applications in addition to focusing on ground-breaking performance per watt and optimising the high-density form factor. also incorporates (see Figure 1). The power supply architecture of the Broadwell DE uses a combination of switches, a linear and switching voltage regulator (VR) to extend five times with a power envelope, and a fully integrated voltage regulator (FIVR) [39].

Qi Zhu et al. had done a great work in focusing on the topics of Node, overall rack performance/watt and Compute density. In the topic of Node, overall rack performance/watt they have discussed on A balanced node that optimizes compute performance vs. memory bandwidth vs. I / O bandwidth, operating points near the knee on the power frequency curve to provide optimal power / watts and At this operating position, a sizable percentage of the leak current cake is supplied by the silicon process designed for low leakage. Similar to how they explored SoC integration for platform components, BGA package selection, thermal design power, and platform thermal co-optimization for the compute design, a reference design with 16 nodes in a 3 U module is likewise viable, as are up to 6 nodes per 1 U.

When dealing with unknowns, the launch of the new segment entails some co-optimization. This is accomplished by carefully co-optimizing platforms, packaging, and SoCs to meet the density and performance objectives. You must design to the lowest common denominator of each optimization vector in order to satisfy the requirements of the hyper-segmented roadmap in a single design.

This can satisfy the performance requirements of the top performing SKU, the cost and margin needs of the product with the lowest average selling price, and the packaging form factor of the platform design with the maximum density. included. While compressing packet sizes to fulfil form factor criteria helps save money, it may put more pressure on the number of layers because it reduces the amount of space available for optimization. Cost and form factor appear to be strongly tied to co-optimization (cost). (Again) The BGA ball court is more significant.

For performance, debugging, and a shorter time to market, it's critical to match the increasing pin count requirements. Optimization of BGA ball pitch and pattern is one of the most crucial design factors to take into account when trying to fulfil board cost objectives and reliability standards for various market segments. The introduction of a new segment where many of these targets are unknown presents another difficulty. Flexibility must be built into the design phase to account for Uncertainty Related to Performance Requirements and Delayed I/O Counts, Performance and Custom Requirements. The cases of how these optimizations were handled are the main subject of this document.

The specifications for the top (die side) and bottom (land / pin side) of the Broadwell DE 37.5mm x 37.5mm package are used to determine the package size. According to preliminary study, both the top and bottom have size restrictions. The location of decoupling capacitors is the biggest obstacle for the to accomplish its power supply performance targets after decreasing the dining space, the integrated heat spreader footprint, and the epoxy barrier zone. Some huge capacitors are relocated from the lid, and more than half of the core power capacitors are positioned in the lower cavity, preventing the package's size from increasing.

Although cavities result in pin loss, they are cost effective overall because they require fewer capacitors when in optimal position. The cavity is directly below the chip. The biggest advantage of the FCBGA package is the ability to reduce the footprint of the and increase the number of pins. On the other hand, if the footprint and pin outs are not optimized for the, extra layers on the board, deep knockouts, and non-optimized routing on the package can occur. The footprint of the case should be designed with the circuit board.

With its small form factor, efficient power architecture, and rich SoC feature set, SoC is ideal for applications that require high computational density. Broadwell DE package design demonstrates that high performance design can be achieved at a reasonably low cost by performing performance trade-off analysis of key cost factors such as size, footprint, number of layers, pin map, etc.

2.8 Integration of Performance Improvement and Stencil Power Application Intel to Intel Scalable Processors

Large data sets produced by contemporary apps are challenging to process and analyse. Because they can represent the entities involved in the development of enormous datasets from a node perspective and their relationships from an edge perspective, graph algorithms have become a viable option for the analysis of such data. did. In order to extract data for additional research, graph analysis techniques are employed to identify patterns within these interactions. A variety of parallel computers have been utilised to do considerable research on the optimization of breadth-first search (BFS), one of the top graph search algorithms for graph analysis. BFS parallelization has proven challenging, nonetheless, because to its limitations on scalability and unstable memory access patterns, data dependencies, and workload imbalances.

A unique development framework coupled with the Graph500 [40] benchmark was used by Mireya Paredes et al. to investigate BFS optimization on the SoC Phi (Knights Corner), a contemporary parallel architecture with an advanced vector processor that supports the AVX512 instruction set. The classic top-down BFS method formed the foundation for a hybrid BFS algorithm that was created employing vectorization in an optimised parallel version of the two high-level BFS algorithms [41]. For 1-million-vertex graphs with optimal implementation, the KNC offers 1.37x (top-down) and 1.37x (hybrid) speedups compared to the most recent. Performance on KNL and Skylake is better than on KNC. The top hybrid algorithm solutions also appear in the real plots from the SNAP dataset, delivering speedups of up to 1.3x. using KNC. The robustness and portability of the algorithm are shown by the improved performance on KNL and Skylake. Other graph analysis methods can be accelerated by hybrid BFS algorithms, and other SoC Phi and advanced vector architectures can benefit from the vectorization lessons that were learned. The algorithm can benefit from it.

Large volumes of data are processed by modern applications. Since graphs may represent entities as vertices and their interactions as edges, graph analysis has become a crucial tool for analysing this data. In order to get data that may be further analysed, it is usual to search for patterns within these associations. One of the most popular graph search algorithms for graph analysis is breadth-first search (BFS), whose optimization has been thoroughly researched utilising a variety of parallel and distributed systems.

It shows that parallelization of BFS is difficult due to the inherent characteristics, such as unstable memory access patterns, data dependencies, and workload imbalances that limit the scalability.

However, only six publications deal with the new parallel architecture using advanced vector units. Example: B. SIMD Intel AVX512. This article describes BFS optimization and vectorization in SoC Phi (Knights Corner (KNC)), an experimental framework for the Graph 500 benchmark, a parallel architecture with enhanced vector capabilities.

They also display the outcomes of a direct port to a more recent Phi model (Knights Landing (KNL)) as well as those of a Skylake CPU, which supports the majority of the AVX512 instructions used by the technique. Paredes et alhybrid's BFS is a parallel approach. It is presented via vectorization and is based on the top-down vectorized BFS developed by Paredes et al. By combining top-down and bottom-up BFS algorithms, the hybrid BFS method was first proposed by Beamer et al. For 1 million vertex graphs, our new hybrid BFS outperforms KNC's cutting-edge solution by 33%.

This study's main contribution is based on research by Gao et al. The top-down BFS algorithm is vectorized in the first study by Paredesetal utilising the available vector built-in routines. It exceeded, shows the effects of prefetch, thread affinity, and vector unit utilization. Second study by Gao et al, refers to the vectorization of the hybrid BFS algorithm. However, few details of those implementations are provided, so this white paper provides top-down and bottom-up vectorization details for both approaches included in the hybrid BFS algorithm. The performance of the hybrid BFS algorithm is superior to that of Gaoetal. In Furthermore, the work of Golobina et al. The vectorization [43] of The Graph 500 List for November 2017 includes the hybrid BFS algorithm. The experiment produced 1.80 GTEPS on a platform with an Intel Phi (5110P, Knights Corner) processor comparable to the one used in this publication.

The main creation and analysis of the algorithm in this paper were performed on the Intel Phi KNC 5110P coprocessor, which has a 60 4-way SMT Pentium-based core running at 1.05 GHz and 8 GB. A potent 512-bit vector processing unit, L1 (32 KB) and L2 (512 KB) of cache memory, and a globally distributed tag directory (TD) tuned by cache coherency are all present in each core. maintains its entire coherence. MESI procedure. As depicted in FIG., a high-speed bidirectional ring bus links the cores together. The 320GB/s maximum memory bandwidth is stated. A vector register and a 16-bit mask register make up the Vector Process Unit (VPU). 16 (32-bit) or 8 (64-bit) operations can be processed simultaneously by each vector register. The vector mask, which has 16 bits, manages how the vector element is updated. The vector register only updates items with the bit set to 1, leaving unaltered elements with a mask value of 0. The Phi has both software (SWP) and hardware (HWP) prefetches, which could lower memory latency.

Vector units and multithreading are features of the Phi programming environment. The Phi processor supports shared memory multithreaded systems using the flexible programming interface known as Open MP. There are two programming options for the vector unit: automated and manual. With automated vectorization, any programmes that can be optimised for vectorization are found and optimised by the compiler without the need for programmer interaction. The use of vector units, however, may be constrained by certain flaws such data dependencies and sporadic memory access.

You can employ manual vectorization in these circumstances. By doing this, the user can instruct the compiler to vectorise particular sections of the code. The SIMD pragma instruction, which is also supported by the Open MP library, can be used to specify manual vectorization. Additionally, the compiler offers a broad variety of built-in functions that give programmers vector-by-vector low-level control. Optimization of programme performance is a challenging undertaking. Tools for retrieving real-time hardware performance data, such as the PAPI- Library, are available to aid in this process

(Performance Application Programming Interface). By monitoring different events while the programme is executing, this library gives users access to multiple hardware performance counters.

Traditional top-down layer sync BFS is an implementation of the serial BFS algorithm for processing vertices over layers. The idea of layer is constructed by this approach using two lists. The input list, also known as the borders, is the first list that comprises all the vertices processed in the current layer. The second list is the output list, though it is occasionally referred to as the output queue to be consistent with the literature. It comprises a series of vertices that are processed before being switched with the front layer to process the next layer. is marked as visited when the vertices are processed. The vertices won't be visited if this doesn't happen. Each vertex has a set of neighbouring vertices that are related to it by an edge known as the adjacency list. The output queue only contains vertices that have been identified as unvisited vertices. The algorithm's output is a BFS tree that is represented by a list of the parents (P) or ancestors (A) of the passed vertices.

This work's main contribution is a thorough evaluation of KNC's performance in a vectorized version of the bottom-up BFS algorithm based on hardware performance counters and the PAPI library, which boosts the graph's top speed by 33%. 1 million vertices in size (scale 1/40 and edge factor 1/46) Gaoetal. They also used bottom-up algorithm vectorization, however the results were not completely understood because bottom-up BFS algorithm vectorization had uncertain consequences. The second contribution is the creation of an independent vectorization of the Gao et alproposed's hybrid BFS algorithm, which combines SIMD top-down and bottom-up SIMD. However, it outperforms Golovina's creation of the BFS algorithm, who is placed 151st. on the list of 500 graphs. In addition to improving speed by 12 percent to over 30 percent on KNC's Graph500 benchmarks, our techniques also perform well on a variety of real-world graphs from KNC, KNL, and Skylake. To do. dataset for SNAP. This demonstrates portability and durability. Finally, they discussed problems with algorithm portability. Future research on the hybrid BFS algorithm can be pointed in a few different directions.

First, the bottom-up BFS algorithm's vectorized version's performance examination offers various points for thought. For instance, the programme asks you to determine whether the Phi architecture's CPI (cycles per instruction) criterion is more than 4.0. Our findings take this into account. You will comprehend the Phi architecture and how to exploit characteristics like vector units and cache memory, which are crucial for grafting rubber monkeys, more fully after further examination. The analysis of variability from the run-by-execution tests provided helps to understand the impact of variability on workload sharing between threads because BFS is non-deterministic. Finally, for algorithms that need to generate multiple BFS trees from different starting nodes in the same graph. B. Proximity Centrality Calculation [45], information from the initially generated BFS tree may be used to speed up later tree calculations.

2.9 Correlation of Performance Optimizations and Energy Consumption for Stencil-Based Application on Intel Scalable Processors

Details of the impact of improving the efficiency of real CFD applications called MPDATA, as well as the efficiency and power interaction between these configurations and basic computer systems representing the first generation of Intel SoC expandable processors. Full of analysis. Taking the repetitive MPDATA application as a matter of usage, the memory-bound application contains a series of development steps that improve the performance of the code and improve the performance of the application through additional resources. Examine the basics of energy and performance analysis. Works well on memory-bound applications, high performance optimization has been shown to be a powerful strategy for improving application performance efficiency. In fact, in the proposed performance improvement, energy benefits are associated with performance benefits, but to a different degree. As a result, this setting greatly improves both performance and power consumption, reaching approximately x10.9 and 8.8x, respectively. The impact of the Intel AVX512 SIMD extension instructions on power consumption and performance is shown. We also found that there is a limit to the ease of use of CPU frequency measurements as a tool to measure energy savings and acceptable performance losses.

The power consumption of modern and new computers is still growing, despite improvements in energy efficiency. This practice is one of the five major challenges you must overcome in the path of exascale computing [46]. Most of the total increase in energy consumption is due to technology [47], [48], which provides efficiency and energy efficiency by providing efficient energy and computer hardware. Develop consistency. A notable example is the family of Intel scalable processors [49]. This reduces the power consumption of HPC applications and offers promising opportunities to explore new trade-offs between power and performance.

Authors Lukasz Szustak, Roman Wyrzykowski, Tomasz Olas, and Valeria Mele contributed to various aspects such as a real CFD application called MPDATA [50], providing an in-depth study of the impact of improved system efficiency and collaboration. performance and power between this setup and the basic hardware. Provides an intuitive analysis of the results. Represents first-generation Intel processor; MPDATA Iterative. The application is considered an application, the capabilities of the application are tied to the memory and the basis for performance analysis is investigated, and the application is executed. a series of development steps to improve code power. Improves performance. Effective use of computer resources; For applications bound in memory, optimizing high performance can be a powerful strategy to improve energy efficiency. In fact, in the proposed energy development measures, energy gain is related to energy gain, but to a different degree. In the first step, which enhances both repositioning of the repository and data center, the energy gain is approximately 10% higher than the processing time, but the other two reduce the syncing and the interaction. In the process, energy benefits are diminished. performance. As a result, this setting [51] significantly improves both performance and power consumption, reaching approximately 10.9x and 8.8x, respectively; Intel AVX512 Displays the effect of SIMD extension instructions on improving application power and performance. A small advantage to vectorization is the original unprocessed version of MPDATA, which consumes less power despite having the same performance time as the corresponding scale due to the reduced power requirement of the competent SIMD code. When it is reduced by about 20%. The highest benefit comes from the highly optimized MPDATA version. This can improve both power consumption and operation of 2.8 times.

Measure the benefits of CPU frequency measurement as a tool to measure energy saving and acceptable performance loss. Using MPDATA code that is not optimized for very low frequency can significantly reduce power consumption without disrupting performance, but in the optimized version, reducing the frequency only improves performance [52]. No progress has been made. Powerful, provides the first accurate comparison of power / power measurement software using an on-chip power sensor and RAPL as well as hardware accuracy using an external wattmeter (Yogogawa WT310). Simple and effective methods to correct the results obtained with RAPL have been proposed to avoid significant differences in energy levels in the long-term estimates. This method is based on obtaining the final result as the sum of the measurements taken for each packet.

As major computer infrastructure becomes increasingly difficult to operate / power constraints, it is understandable why the efficient operation of the corresponding HPC operating load has been the focus of more recently. In addition to the energy-saving trends achieved by hardware redesign, you can see another powerful power-saving process gained through the complete modification and rethinking of algorithms and software for the HPC platform. increase. Following this guide, the development of the latest HPC computer science software is now focused on expanding the scientific code by carefully analysing the transaction between time and performance / power consumption. As per our research, many of today's research efforts are aimed at developing modern Power Aware Computing technology, focusing on various aspects of the domain.

The most common method is DVFS (Dynamic Voltage and Frequency Scaling), which allows hardware to reduce operating voltage and frequency but which can lead to longer operating times. Such an approach provides an opportunity to improve energy efficiency when the cycle is often wasted because it relies on memory resources. In the future, study the use of hardware hardware to reduce power consumption when using applications in line with the various operating and memory usage. This work explored the features of Intel RAPL infrastructure and how to measure the frequency that limits the power allocated to computer devices while using HPC applications.

Wlotzka et al. addresses the main problem of a highly efficient computer system by providing a way to save energy on calculations by using the power-enabled time on a shared memory platform. he did. This work has identified some of the most commonly used scientific methods applications and introduces power profiles and tracking methods, which are appropriate to analyze the power consumption of applications. The project has revealed a number of ways in which software can be used to reduce power consumption without changing the hardware. This study described an application that performs automated machine configuration that does not require knowledge of the underlying hardware structure. Hassan et al. investigates the impact of the use of different styles of coding to achieve a balance between efficiency and efficiency of Jakobsetal power. By considering both automatic and manual vectorization techniques, we have shown the possibilities and limitations of vectorization in terms of energy efficiency. The effect of vectorization 4344 combined with multithreading on the efficiency of system performance has been investigated. In, Rojek proposed a way to reduce the power consumption of applications running in the Supercomputing Center through mixed-use functions. Tasks have introduced a biological approach that aims to distribute workloads across all processing platforms of various similar platforms. The goal is to exploit the opportunity to trade between working time and energy consumption and to reduce energy without increasing working time.

The MPDATA code has a data structure suitable for vectors, making it easy to convert any version of the code into AVX 2.0, AVX512, or non-AVX (scalar) instructions set by carefully selecting the producer argument. It can be modified ([53]). This allows you to evaluate the impact of SIMD processing on power consumption and performance of all MP DATA versions. Testing uses four metrics: (i) performance time, (ii) total power consumption measured by the Yokogawa WT310 wattmeter, (iii) average processor frequency, and (iv) waiter. The central power of. This shows a collection of all four metrics restored to MPDATA domains of size 1024x512x128 and 1000 step steps. Vectorization advantages are also included to reflect the benefits of AVX512 over various MP DATA versions. In addition, this demonstrates the impact of using the AVX512 extension to improve performance and power consumption. As expected, the greatest vectorization benefits are achieved in version D.

In this case, the SIMD-enabled code can improve both performance and power by about 2.8 times compared to its scalar counterpart with vectorization closed. In versions B and C, vectorization reduces both the processing time and the energy consumption of, but their benefits decrease. In contrast, version A with or without vectoring function is almost identical. This is because the performance of these releases is limited to the main memory bandwidth, as described in paragraph 5.2. Therefore, using AVX512 to increase available computing power does not improve performance. At the same time, enabling vectoring can reduce energy consumption by about 20%.

The key to understanding this behaviour is to analyze the frequency of the processor. When the executes the AVX512 instruction, it clocks down significantly [49]. In our tests, the clock speed of the dropped from 3.19GHz to 2.42GHz. The average performance of version A with vectorization enabled during running is about 105 watts compared to the scalar code, because version A cannot use vector units efficiently and reduces the CPU frequency. It will decrease (18.5%) (Table 4). Finally, due to the same run time and reduced power requirements, MPDATA-based version with vectorization enabled consumes less power than the counterpart.

This work also investigates how to modify the CPU frequency to maximise the power efficiency of MPDATA using the DVFS approach. When the CPU cycle is wasted because it is stuck in memory, this method is known as an effective way to conserve energy for memory-reliant programmes like MPDATA. To alter the CPU frequency, use the CPU freq framework. The minimum clock frequency is set to 1.0 GHz, sampled every 0.1 GHz, and the maximum turbo boost speed is 2.3 GHz in the BD version and 2.42 GHz in progress due to the heat and power restrictions of the test platform. is reachable. A version. This displays the results achieved using the least efficient version D and the non-optimized version A. (Vectorization is enabled in both versions.) The Yokogawa WT310, which monitors the entire platform, measured the total energy consumption value that was given. The 10 displays execution time monitoring and overall energy usage as a function of CPU frequency.

Even at the lowest CPU frequency of 1.0GHz, version A's execution time is essentially constant. This version's performance is primarily determined on the memory speed. As a result, CPU frequency scaling has no impact on MP DATA's execution time. However, the non-optimized variant provides the most possibility for CPU frequency scaling power consumption reduction. In fact, operating this version at the slowest possible clock speed can increase power usage. The maximum clock speed is 28% lower than this. Reduced power requirements are mostly to blame for this.

Although the lowers the clock frequency from 2.42GHz to 1.0GHz, the average power of the is nevertheless decreased by 105 watts to 360 watts, and the power loss is only 1.5 percent. However, while using the optimised version D, reducing the CPU frequency has no beneficial impact on either performance or energy. The maximum CPU frequency offers the best performance and power trade-off for this. This compromise optimises this performance while also reducing the device's power usage.

In version D execution time decreases as frequency increases therefore setting an inversely proportional relation between execution time and frequency from the lowest clock speed to the highest clock speed. This approximately doubles the execution of MPDATA and reduces the power consumption of the as follows: About 16 percent. The reason for the small energy gain of the compared to the performance benefits of the is due to the increased power demand and the increased CPU frequency of the, resulting in more intensive use of computing resources.

Energy / power consumption is one of the most important limiting factors in exascale systems [38]. Currently, some supercomputers consume more than 15 MW. The scaling factor for large cluster installations can improve the power efficiency of on a single node while still significantly saving power / performance for the entire system. This white paper uses both hardware and software power analysis techniques to show single-node power / power consumption based on the Intel SoC Platinum 8180 dual socket processor.

Reliable measurements of energy efficiency per nodule are made possible by the Yokogawa WT310 Digital Power Meter and RAPL. This work focuses on the verification of a set of parametric optimizations recently proposed in for a real CFD [39] application named MPDATA in terms of energy and power efficiency. Let's examine how this memory-bound iteration code's power consumption is impacted by the combination of the proposed optimization steps. The results demonstrated show that these modifications significantly increase the effectiveness of MPDATA simulations while consuming less power. These gains in performance and power consumption are in the range of 10.3910.94 and 8.278.76 for the size of the MPDATA domain that is being taken into consideration.

Next, learn how activating vectoring affects the CPU clock speed and power requirements, as well as how SIMD vectoring affects the application's performance and power usage. As this study has demonstrated, the Intel AVX512 expansion has a significant opportunity to enhance MPDATA applications' performance as well as their power consumption. Tests specifically demonstrate that allowing vectorization can reduce power consumption even when memory performance constraints hinder the efficiency of parallel programming.

For the highly optimized version of MPDATA, they found that lowering the frequency did not improve performance or power consumption. Finally, carefully evaluate the measurements of RAPL and Yokogawa's WT310. These two approaches are consistent in the short-term simulation, but the long-term simulation shows a high inconsistency due to the overflow of the 32-bit MSR register that collects the measurements made in the 4484RAPL.

However, these discrepancies can be successfully overcome by developing aggregate methods for RAPL measurements. Our further work involves extending the experimental comparisons proposed in this paper to the current and new architectures of and to other applications.

CHAPTER 3

PHYSICAL ARCHITECTURE

3.1 3.1 IMPLEMENTATION OF SOC INTEGRATION

Different IP cores make up a SoC. Each core has a distinct design and serves a particular purpose. An SoC typically consists of a processing unit, memory blocks, peripheral blocks such as camera and audio modules, a debug subsystem, a clock controller block, multiple pipelines and throttle blocks between IP cores, and other peripheral blocks that are all connected to one another through a NoC. A brief description of a few IP cores found in the chip is provided in this chapter. It describes the internal **organization**, purposes, purposes of ports, port types, and information about interfaces with other blocks. The chapter also discusses the specifics of how the full integration process was implemented.

- **Pre analysis work for performing integration**

The integration process calls for a thorough understanding of the SoC's architecture, the design of all IP blocks, their capabilities, pin details, and interface information. It should also be carried out in accordance with the provided SoC/ASIC requirements. Only after having a solid understanding of the various interfacing protocols can connectivity be carried out. Every protocol has a distinct read and write cycle, with a particular set of pins designated to carry out those cycles, as was discussed in chapter 2. Two cores should have corresponding master and slave pins on their boundaries that need to be attached to one another if they need to be interfaced using a specific protocol. In addition to the common protocols like PLL and BGR, Intel also has a few extra proprietary interface protocols with advanced functions. Understanding the proprietary protocols is also necessary in order to join two blocks. An exclusive connect tool from Intel is used for the connectivity process. To utilise the tool, a thorough understanding of the commands is necessary.

- **Design Methodology of SoC integration**

The design team sends the IP Cores' layout, along with information on their functions and interfaces. It is important to comprehend the specifications provided by the owners of the core designs and to connect devices appropriately. By updating the.csv files that the design team receives, the connectivity between various IP Cores is carried out in accordance with the provided standards. The names, directions, and port width of each port are listed in these.csv files.

In Server SoC I am working on RAS Validation in this RAS Validation I am basically working on PUnit IERR Validation

3.2 RAS

SoC supports the RAS feature set including Standard and Advanced RAS. New features include: support for DDR5 X8 full ECC scheme. This is a new ECC scheme which provides half device correction for soft errors and full device correction for hard errors. In addition to LZ features, support has been added for RAS and error reporting changes due to Hierarchical PM, adding an error logger for error reporting of non-os visbile Ips

Feature:

- Support for DDR5 X8 full ECC scheme
- Error Logger for compute die
- I/O MCA moved to global IEH
- allows merging upto 128 Bank Indices per MCA BankID
- error reporting via FULLBANK_MCA_MSG to ubox for IPs that don't have a local MCA Bank
- Hierarchical PM MCA changes - every Punit will send MCA_MSG to UBox in merged format
- root Punit (i.e. one Punit) will own the RAS Pins (CATERR#, rMCA# and NMI#)

3.2.1 RAS Security

The safety of Granite Rapids -D (GNR-D) consists of two main areas:

• Safety Certificate

• Certain security features

Security assurance identifies threats and potential mitigation across the Server SoC. Certain safety features are new capabilities in the SoC which aims to provide new capabilities to address security needs / concerns for our customers.

Server SoC is integrated with the compute die from previously used SoC and will use both security and features from the previously used SoC product line

3.2.1.a. Security Assurance

The process to ensure SoC security includes compliance with Security Design Lifecycle Essentials (SDLe). SoC Compute Die is registered and completes SDLe with the previously used SoC program and will be used again in Server SoC. Server SoC has a separate IO die registered and will include IO dye directed at Server SoC and other Server SoC deltas from previously used SoC.

IP-level SDLs are made into a business unit. Any Server SoC-specific IP, such as HWRS, must register with SDLe.

3.2.1.b. Security Feature

Server Soc inherits features from the previously used SoC product line. The SoC family defense structures inherit all the infrastructure and security features from the SoC wave line / wave2 / wave3. (ICX, ICX-D and SPR). This includes chassis infrastructure and legacy security features. The SoC family generation security features are add-ons and deltas listed below. Key changes to the SoC include increasing the number of MK-TME keys, link encryption and providing TDX IO support.

3.3 PUnit IERR and MONFAIL:

RAS architecture at Server SoC needs to support cross-die networks which fall under IERR/Global Fatal, MON_FAIL and FSB Error categories.

3.3.1 IERR Network:

IERR is one of the signals of the RAS. At Server SoC IO die, it has 2 donors: local and international. The IERR location signal ranges from the integration of Service Controls (RC) and Service Adapters (RA) to PUNIT. Global IERR signals from both IO Dies say OR'ed and are sent to PUNIT. The PUNIT-driven global IERR is integrated with IERR signals from the death of one IO and transferred to another IO die.

This network is further subdivided into the local IERR network and the global IERR network.

• IERR Local Network: IERR local network network is required to provide an independent asynchronous and sideband to display the accumulated error from all separate PM resources to local PUNIT. These applications usually include errors from Resource Controllers, app adapters and pmusvid. IO Die SoC needs to build an integration network to collect all of these errors and distribute them to PUNIT.

• IERR Global Network: The IERR global network is used to notify all PUNITs that a system error has occurred so that PUNIT / PUNIT code can take appropriate steps such as loading / activating crash log.

PUnit Error:

PUnit logging is accompanied by PUnit signature errors. Signed PUnit errors will cause log entry to PUnit. Errors installed on Punit will have a signature error from Punit. The exception to this statement of consistency may be due to the following reasons:

1. The severity of the new error is less than the previous error installed

2. If PUnit error mode is changed and a new error is found or signed in PUnit close at the time of this change of error mode

Note: It is recommended that the presence of the error log be checked before and after changing the SoC error mode. There are racing situations due to SoC mode changes for different IPs in different die and same die as PUnit is integrated into it. These race conditions mean that part of the SoC is in the new MCA mode while other parts of the SoC are in the old MCA mode. While it is in this state of instability the errors found can be signed and entered inconsistently

3.3.2 MON_FAIL Network:

This network is primarily a Functional Safety (FuSa) requirement. Like this feature, the SoC needs to use the GPIO PIN to point out strong errors from internal security guards from death. This PIN is monitored by an external security feature to detect an internal error reported in clock / power monitors. This PIN is useful in situations where the error message cannot be transmitted reliably as the error may disable the reporting method.

Examples

• IP contains a PLL that automatically or indirectly stores a set of status registers containing IP status information. When PLL merges, PLL loss error, status registers cannot be read by software.

• IP contains a FIVR that enables logically bound to form IOSF-SB packets that can normally report an IP error. If FIVR fails, IP is not accessible via the sideband network.

3.4 MDFCI

MDFIC (Modular Die Functional Interconnect for Control) is a block that logically connects infrastructure between deaths over EMIB. Infrastructure includes:

• Sideband, includes Power Management Sideband (PMSB) and General-Purpose Sideband (GPSB)

• Fabric Testing Fabric (DTF)

• Scanner Fabric (STF)

• Miscellany Async Signals from multiple domains

MDFIC IP GNR HAS is ongoing and will be connected when available.

MDFIC uses a forwarded clock / enhanced GNR supply. See the MDFIS section for more details.

Features

The key features of MDFIC are:

• All signals and assumptions use the pre-VINF provision

• RX and TX share the same VINF (from motherboard VR)

• MDFC is available after downloading the fuse (enable repair)

• With DTF and Sideband, the RX and TX operate at the same frequency based on a standard reference clock, but with a different PLL.

o MDI has an RX disked FIFO to align data with the RX clock, but NO CONTROL LEFT as it works with the same frequency.

• In STF, a forwarded clock is expected. MDFC handles this with each HSD

• Unconnected cables pass through MDFIC without washing. The maximum conversion speed of these signals is 400Mhz.

3.5.1 PUnit Overview

While PUnit primarily is intended to be a power management IP it also performs boot, configuration, RAS, and miscellaneous functions for the SoC. PUnit does not provide reset functions for the SoC.

PUnit has the following high-level interfaces:

- Power good, reset, and clock
- idle power management
  - Q channel (SoC facing side) from Punit for idle power management
  - Q channels (interdie side) from Punit for MBVR idle power management
  - adhoc idle and wake pins
- Private IOSF sideband fabric (PMSB)
  - 2 instances of these
    - FSM/register accesses
    - sVID/PECI

- Public IOSF sideband fabric (GPSB)
  - 3 instances of these
    - register accesses (fsms interface)
    - FSM accesses (ioregs interface)
    - SB_BRIDGE

- Registers
  - GPSB registers
  - PMSB registers
  - IO registers
  - RTDR registers

- RAM/ROM
    - Tensilica RAM interfaces
    - Tensilica ROM interfaces
    - converged telemetry (including TPMI) array RAM/RF
- PECI
- sVID
- PMSYNC (CPU side)
- TSC download (from PUnit to SoC receivers)
- throttling
- Debug
    - RTDR for TAP access
        - several RTDR exist
    - JTAG interface for Tensilica
    - VISA
    - DTF
    - Debug triggers and trigger events
- HVM
    - Scan controls
- Error signalling and logging
- Configuration straps

PUnit functions are organized into service stacks, each service stack having its own functional description, its own programming model and its own initialization sequence. These functions share PUnit IP access points of GP IOSF SB, PM IOSF SB, or TAP.

3.5.2 Punit Services

Punit is organized into various services. Each service may have any of the following resources it interacts with.

- IO Registers
- GPSB registers
- PMSB registers
- RTDR registers
- PUnit pins

PUnit supports the following high-level features:

1. reset
   Cold reset, warm reset, surprise reset, HVM reset, S3, S5 all supported
2. fuse pull
   Fuse pull for hardware as well as fuse pull for firmware usage
3. Power management control
   - dispatcher
     Complicated FSM to handle voltage, frequency, license level, voltage phases, cstate values for cores and uncore
   - sequencer
     Ancillary FSM working with dispatcher to accomplish changes in operating points
   - Idle power management flow
     Idle (aka PkgC) entry/exit handling for this die and coordination between die within a socket. This includes inter-die communication and inter-socket communication.
4. Firmware processor: Tensilica
   Firmware runs on Tensilica working together with PUnit to deliver Punit/firmware feature set
5. IO Registers
   Firmware/Hardware interface
6. Hierarchical Power Management (HPM)
   Provides communication and coordination capability for multiple die between various firmware instances. Used for reset, boot, and functional communication between the various firmware instances. Additionally, used to communicate with CXL devices for the same purposes.
7. TPMI
   Hardware MMIO/register range to expose extensible feature set communication between OS and firmware.
8. Crash log
   Hardware set of registers to expose dispatcher state and firmware state after MCA for BMC or BIOS harvesting.
9. RMID
   Hardware set of registers to expose specific power telemetry from firmware to OS. Legacy and still supported even though converged telemetry supported.
10. telemetry: converged
    Hardware MMIO/register range to expose power telemetry to OS from firmware. More generic than RMID.
11. telemetry: internal
    Hardware register range receiving telemetry writes from SoC, providing telemetry mapping to firmware in IO Register space.
12. ADR
    Asynchronous **D**ata **R**efresh functionality support for platform power supply failure handling where SoC flushes accesses to memory and puts memory into self-refresh.
13. sVID
    Link and protocol layer support for **s**erial **V**oltage **ID** interface to Mother Board Voltage Regulators (MBVR). Supports requests from dispatcher as well as firmware, supports

telemetry pull and push. Supports error detection, signalling, logging. Additionally, supports up to 2 *sVID* interfaces and up to 16 independent MBVRs.

14. PECI Comms
    Platform **E**nvironmental **C**ontrol **I**nterface for manageability access from BMC to CPU. PUnit contains the link and transaction layer layer of this interface to the platform. OOB contains the protocol layer of this interface. PUnit PECI Comms module communicates only with OOB for these transactions.
15. pmsync/sblink/sbbridge
    Slow serial wire interface *pmsync* containing both legacy pmsync capabilities for register read/write capability as well as SB_LINK functionality transporting IOSF SB over SB_LINK through the SB_BRIDGE inside PUnit. *PMSync* used for reset and runtime usages. Runtime usages communicate ADR and idle power management wake indications.
16. mailboxes
    Hardware mailboxes to provide communication conduits between various other SoC hardware and software entities and firmware. These mailboxes are at different permission levels and have different semantics depending upon the scaling of the mailbox. Firmware controls what commands are supported by each mailbox.
17. Fastpath
    Hardware logic to notify firmware of events needing firmware action. Used instead of interrupts for firmware to accomplish low latency response to events.
18. BulkCR
    Hardware DMA engine supporting register reads and writes on GPSB.
19. [DMA]
    Would be new for gen4. Hardware DMA engine supporting more than just register reads and writes on GPSB. Would support more than the 24 accesses the bulkCR supports.
20. Timers
    Firmware needs timer notification of events. Hardware provides 100MHz timers and crystal clock timers for pcode. Fastpaths can be signaled upon a timer reaching a timer trigger value set up in advance by firmware.
21. Time
    Hardware timer receiver from PMSync interface starts TSC usage in PUnit. This TSC in PUnit is the root for the SoC and is distributed using this hardware to the SoC via
    - single serial wire download
    - Hammock Harbor timer provider within PUnit
22. Throttling capabilities
    Hardware throttling generation for SoC usage from various sources. Sources include package PROC_HOT pin, PMax, temperature threshold compares, firmware direction. Gen4 PUnit would include SIRP as a source for throttling.
23. Catastrophic temperature handling
    Hardware which allows mapping MEMTRIP signals into CAT_TRIP and THERM_TRIP signals.
24. LTR
    Latency **T**olerance **R**eporting VDM receiving logic in PUnit to give to firmware.
25. Perfmon
    Performance monitors including both hardware and firwmare performance monitor capabilties.

26. governor
    Hardware feature to provide indication to firmware if xxBCLK is being overclocked or not relative to xxPCIE_BCLK.
27. Error handling
    Hardware detection of errors signalled from SoC via wires or PMSB. Hardware error signalling to platform and global IEH. Hardware logging of errors.
28. Core support
    Various core support features in PUnit include
    - core cstate handling
    - core GV handling
    - A Code supported autonomous P States
    - PSMI support
        - gen3 supported, gen4 attempting to deprecate
    - Probe Mode support
        - gen3 supported, gen4 simplifying
29. GPSB
    3 endpoints
    - FSM
    - Registers
    - SB_BRIDGE
30. PMSB
    2 endpoints
    - PUnit
    - sVID/PECI (shared)
31. RAS features
    Hardware error detection, logging, signalling for various errors. For gen4 expect simplifications in this area.
32. DFX
    Many various DFX features including
    - JTAG for Tensilica
    - Several RTDR
    - VISA from various hardware blocks
    - DTF providing packets for IO Register tracing and for PCode GDXC message tracing
    - event outputs from PUnit
    - trigger inputs to Punit
33. Miscellaneous
    Various registers for supporting miscellanous features of SoC. For gen4 some of these will be investigated for deprecation.
    - CPUID
    - INFO_CARD
    - softSKU
    - Stepping ID
    - CAPID

34. Security
    Each of the various features has its own security support level. However, security as a built in capability for each feature needs to be considered as a top level feature. PUnit supports the following security capabilities
    - SAI on GPSB
    - gen4 SAI on PMSB
    - dfxsecure_feature_en0 control from SoC to indicate security level for internal capabilities
    - dfxsecure_feature_en1 control from SoC to indicate security level for internal capabilities

3.6 GPIO

The GPIO Chassis is intended to support the common use of GPIOs in SoC based on Intel process across all types of products. The GPIO Chassis begins with a well-defined category of Pad, Family, and Community. The pad contains pin multiplexing logic, a single I / O controller, edge recognition method, deglitching logic, and other pin-specific logs. Family group pads with standard RCMP, power, and kill rate settings. Families are usually grouped together based on duplicate protocols on Pads such as I2C, SDIO, SPI, and UART. Families are also grouped together on the basis of their performance and similar electrical and bath features. The Family Group is called the community. The community represents a real Family Group on a particular part of the estate and consists of GPIO SIP Controller, RCOMP Regional Machine, fuse puller and IOSF-SB connection.

3.7 RAS Features in PUnit:

• Punishment to initiate the detection of its errors, logging, signing, and actions of errors that occur within the PUnit (similar to an asset).

• PUnit to use the signature mind (similar to the death) error of the death level and the socket level pins.

• PUnit to use the mind to sign an error (similar to an asset) business death level error.

• PUnit to use new intelligence to log in and show errors from other sources.

• PUnit to use integrated MCA bank.

• PUnit to start signing MCA message using the new integrated Gen3 format.

• PUnit to use IERR_in_System daisy-chain support: IERR_in_System input should cause IERR_in_System to be vaccinated.

• PUnit to use Global Fatal daisy-chain support: Global Fatal inputs should result in Global Fatal outflow of immunity.

• PUnit to use CTF Crash dump set of crash log automated hardware, but only if it has enough information to determine if there is an IERR in the local socket.

• Punishment for using the best attempt to sign and enter RAS errors before config_done (but does not guarantee any behavior before config_done).

• PUnit to approve any wiring error order and CrWr with error login information.

• Punit login overflows when multiple CrWr arrows and overflows required.

• PUnit to store legacy logic dial CATERR # PIN.

• PUnit legacy logic retains PIN CATERR #.

• PUnit to keep the logical asset signature received IERR # in ubox.

• PUnit to maintain Recoverable SMI logic.

• PUnit to store NMI logic.

• PUnit to save the PM_EVENT () error log detected but only from the heart

• PUnit to store Global Fatal signaling output from PUnit, signal as an asset

• PUnit to save Global Fatal signature input to PUnit, use as virus recovery indicator as a filter

• PC Uncoder PUnit installs an error loop on all PUnits in the socket

General RAS Architectural Requirements:

1. All Punit links must be muted during de-assertion reset and must be silent until config_done is accepted (rset = idle requirement).

2. Proper processing, signing, and logging of any RAS event is only guaranteed to happen after config_done is vaccinated; RAS signing, and signing are not verified prior to config_done confirmation.

3. Expectations: All Prime Code firmware settings need to install their error loop whenever there is an IERR in the system. The best effort is to achieve this where possible.

4. PUnit to send MCA MSG if an error occurs and is prepared to do so, even if the destination ID is not ready to receive a message e.g. External IP is reset.

5. All Punit cases are able to sign the outgoing Global Fatal (Viral event)

6. Each Punit has a copy of the RAS registers to log a local error.

3.8 Design Verification

Design authentication is one of the most important aspects of the digital design process, and with the development of VLSI and logical design, validation has become a major task, often consuming a large portion of the product development time. The goal is to ensure the design capacity to meet the various operational and operational requirements. In general, functional models that ensure the specificity of a project performance are developed to evaluate the correct performance of a project in terms of logic and function.

As is well known in the VLSI architecture, transistor sizes decrease rapidly, leading to high density of ICs in any digital system. These size requirements conflict with low power requirements and high performance and lead to the formation of complex structures. Continued advances such as multi-core processing concepts and low power requirements make it difficult to design complex that directly affects design validation cycles. As a result, the onus is in the design phase to ensure that the design meets the system requirements.

The design confirmation phase consists of four main steps:

1. Development: This involves developing a system of assessment based on the construction of buildings and

design requirements, as well as the design features of the test bench

2. Imitation: This is the stage at which advanced tests are performed against the design and are modeled to check for the presence of any errors.

3. Debugging: Done after the simulation phase to identify and resolve reported errors during simulation.

4. Cover: The installation of advanced tests at the performance level and code is analyzed, and this any lack of expected installation needs to look back at the development stage.

At the design validation level, each of these 4 stages has major obstacles and challenges. With the introduction of the system on the chip (SoC), the open risk associated with the verification and testing of SoCs has become more severe and challenging compared to that of a single intellectual property (IP) backbone. There is an urgent need to ensure that the time taken to perform each of the above 4 steps is greatly reduced at the SoC level. This requires the use of more efficient verification methods and the allocation of additional resources to ensure the product development process.

In this regard, many such verification languages and methods are developed. The most effective and widely used however, is the language of System Verilog and Universal Verification Methodology.

3.9 System Verilog for Verification

System Verilog is a definition of computer hardware and hardware authentication language that can be used to design and validate digital systems. In fact, System Verilog is an extension of Verilog. However, it incorporates a number of verification features that allow developers to verify design using advanced test bench structures and the production of motives.

In any design, authentication is required to include three key features - random dynamic generators, response testers and cover parts. Stimulus generators provide design updates, feedback testers test the design response in the test, and coverage is required to evaluate whether all required design details have been included.

Random motivation is important in ensuring complex designs. When the motive is corrected and corrected, the expected errors are found, however, random testing can reveal unexpected errors by the designer. The next important step when using a random stimulus is that functional coverage is required to ensure a successful level of random motivation in combining all design elements. Additionally, if the prompt is generated automatically, it requires the development of a pre-programmed method to test the results again, which is the above-mentioned feedback checker. As these requirements have to be integrated into a single language and operating system, the development of a test bench becomes more difficult. This is where the concept of a ordered or horizontal test bench comes into play. This is possible with the help of System Verilog and build functions.

The key features of the System Verilog that include these requirements are listed below:

• Object-oriented language: This is the main feature used by the verification method such as UVM. Asset-based OOP concepts are widely accepted in System Verilog to allow for a systematic definition of class components, following sequence.

• Immunization: The use of vaccines to control the flow of testing, based on previously described conditions or data requirements.

• Functional Installation: Installation is a method of analyzing and verifying whether compliance with the motives generated by the test can cover a large part of the design features and specifications that need to be verified.

• These listed System Verilog features are able to cover these key components of design validation, making it an appropriate language and standard for verification purposes.

3.10 Universal Verification Methodology

UVM (Universal Verification Methodology) is a well-known and widely accepted method of validating digital designs. Built in System Verilog language and presents advanced processes to ensure efficient design. The recycling concept, where UVM components are reused in all types of designs to produce an efficient structural test bench, is one of its main strengths.
In fact, UVM is developed at a practical level, which means that different components are designed to transmit and receive each other's transactions, each performing a specific function. It contains a classroom library, using which code can be written by extending the classes defined in the library. This classroom library contains sections that can be used to build a test bench. Large parts driver, sequencer and monitor. Additionally, a scoreboard and specific content classes are defined. These sections are explained in more detail in Chapter 4. These sections are inherited from the classroom library and customized according to user specifications, depending on the design being tested. The difference between the simple System Verilog (SV) method and the UVM method is that SV has no imaginary standard or code coding. Therefore, it varies depending on the design being tested, and cannot be reused for other designs. UVM, on the other hand, follows the same structure for any experimental design, with custom components. Interaction between components using transactions. Therefore,

communication between classes is not linked to the actual implementation of these components of the classroom, which leads to re-use on a large scale.

UVM follows 3 main phases, and briefly explained The main phases along with their sub-phases are listedbelow.

1.Build Phase

- Build
- Connect
- End of elaboration

2.Run Phase

Start of simulation

Run (reset, configure, main,shutdown)

3.Cleanup Phase

- Extract
- Check
- Report
- Final

Contains sub-sections for building, connecting and ending specifications. The small construction section forms parts of the test bench while the small connecting section connects the different parts of the test bench. The end of the data shows the UVM sequence and other functions required after connection.

Under the running phase, a real imitation of the Design Under Test and the performance of the test bench took place.

Contains an issue, a check, a report, and the subheadings. Expected data from the scoreboard is extracted in a small section. It checks errors between expected and actual values in the sub-test section. The sub-section of the report reflects the outcome of the test and any last-minute activities performed in the sub-final phase.

3.11 Security in Mobile device based SoCs

Advances in mobile technology have resulted in a large amount of sensitive data contained in mobile devices, which are used in many different applications. Therefore, it is absolutely necessary to introduce security measures at the hardware level across all cores in the SOC. These security measures require the use of certain secret keys and cryptographic algorithms.

The security requirement requires a secure location in the SoC, commonly referred to as the root of the trust, to be included in the chip construction where security-related basic functions can be performed. These security requirements in the SoC are composed of specific security blocks in the SoC, which can provide encryption and encryption for sensitive data.

3.11.1 Cryptography and Encryption

Encryption is the process of encoding certain communications or files so that only authorized users can view them. Data is compiled or encoded using an algorithm. The information is then deleted or recorded by the recipient group using the key. Plaintext is a message that will be encrypted or encrypted. It is known as the cipher text once it is encrypted.
A cryptographic key is a randomly generated series of bits used for encryption and encryption. Each key is of one type and is produced using encryption, which ensures that it cannot be predicted. There are two types of keys:
Public or Symmetric Key: Public keys refer to these key pairs, where the key is used
encryption can be viewed by anyone (the general public). However, the key used to decrypt is only known to the recipient of the message.

• Private or Symmetric Key: These pairs of keys, in which both the encryption and encryption keys are the same, and are only available to groups involved in communication, are known as private keys.
Standard key length 128 bits of symmetric key algorithms and 2048 bits of asymmetric key algorithms.
Some common encryption algorithms are:

3.11.2 Data Encryption Standard (DES) *:*

This is one of the oldest algorithms used for encryption. Developed over the years, by IBM, mainly for the protection of sensitive messages and confidential information. The key used to encrypt DES is 56 bits long and uses a cipher structure called the Feistel structure.

The DES algorithm is represented in Figure 3.5. The algorithm generates two 32-bit blocks by separating the 64nit input data it receives as plain text. It then applies the encryption method to all of them separately to convert the cipher text. This contains 16 rounds of various processes, such as approval, expansion, modification, and XOR functionality. Lastly, the output contains 64-bit blocks of encrypted text.

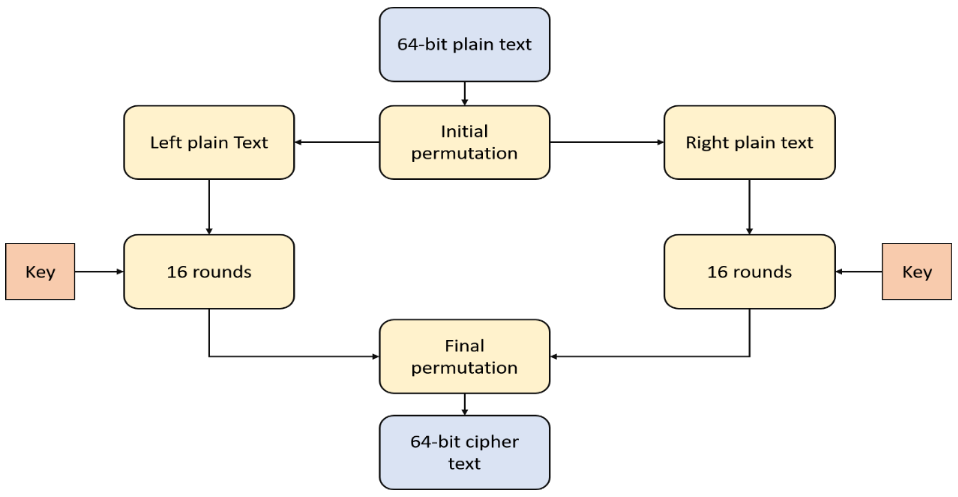


Fig 3.5: The DES algorithm representation

However, due to its small key length, the use of DES was not as secure as expected. Brute force attack was able to reveal the original empty text data. TO overcome this, DES and AES were developed three times.

3.11.3 Triple DES (3DES):

To overcome the downsides of the DES algorithm, the next algorithm that emerged as the development of existing standards was DES three times. As is clear from the name, DES triple combines the use of the DES algorithm three times over in a data block or blank text. This ensures that the success of the brute force attack is greatly reduced. Because of its greater security than the DES standard, it has even been used in large financial and technical systems.

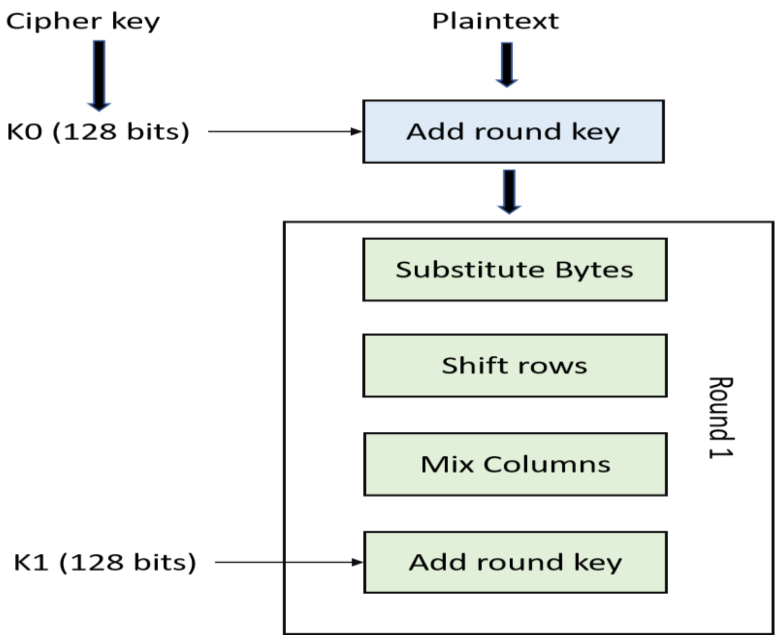


Figure 3.6: Round of AES Encryption

3.11.3.a Advanced Encryption Standard (AES):

This is a widely used encryption method created as an upgrade to the DES algorithm. Unlike DES, it is made of a variety of block ciphers with flexible block sizes and main lengths.

The substitution and approval functions are explicitly used in the AES. As a first step, the input data block or blank text data is separated into smaller blocks, and the encryption key is used to create encryption for each sub-block. Encryption rounds consist of various subtypes, namely, changing bytes, changing lines, adding columns, and adding round keys. The number of rounds performed is not adjusted, instead it is a variable depending on the size of the encryption key. The typical AES round is shown in Figure 3.6. The AES Encryption Algorithm has the advantage of being secure, fast, and flexible while also extremely fast. The fact that there are a few other key lengths makes it difficult to break. Today, it is used in a variety of applications, including wireless applications, encryption on mobile devices, virtual private networks, and so on.

The use of various integration protocols is essential for the basic communication of any SoC. The specificity of the various meeting protocols was discussed. Additionally, the set of Universal Verification Methodology standards for design validation was described in detail. A brief set of details of the security algorithms used in the highlighted SoC defense blocks. The next chapter covers the design approach followed based on the ideas of the tea discussed.

CHAPTER 4
Design and Implementation of UVM based Test benches in RAS

4. Design and Implementation of UVM based Test benches in RAS

UVM structures introduce the use of different components on the test bench to perform certain functions. These components are customized to fit the needs of the experimental design. The specificity of this UVM structure for validation at SoC level is discussed in more detail in this chapter. The use of a UVM class library to implement this design has been discussed. The custom requirements for the implementation of the SoC ban are also highlighted.

4.1 UVM Test Bench Design Specifications

A UVM test bench consists of various components, which are implemented using the base classes of the UVM Library. The architecture of the UVM test bench is shown in the Figure. 4.1

• UVM Test:

UVM testing is the most advanced feature of imitation based UVM tracking. It is the part that sets up a test bench call and ensures low-level components during the construction phase. It is responsible for the following key activities

1. Arrange the bench

2. Build test bench parts and implement them according to the level of governance.

3. Launch the sequencers and pass the inspiration to the experimental design

• UVM Environment

The surrounding area is a container class containing one or more agents, as well as other items such as a scoreboard, monitor, and drivers. It contains various reusable verification components and launches them with their default specifications and settings. The site may also have sub-areas that have been used at block / IP level, which are then merged to form a sub-program.

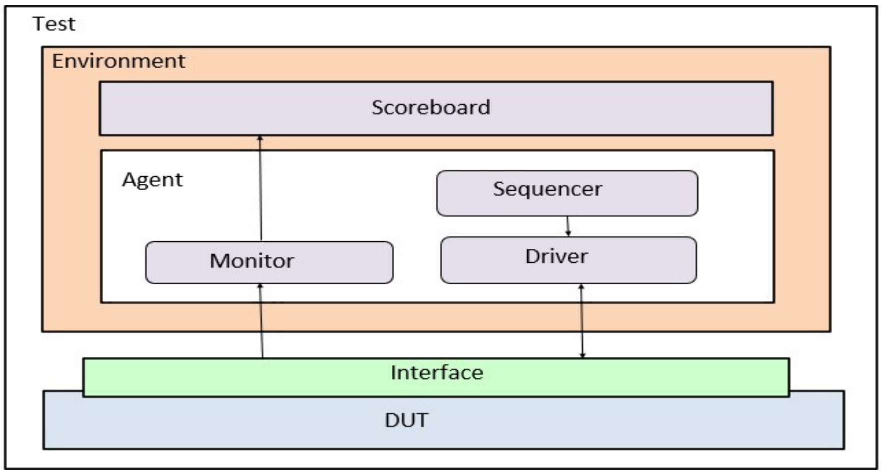


Figure 4.1: UVM Testbench Architecture

• UVM Agent

Agent is also a container component that is associated with a core interface / design interface. It includes three components, namely driver, monitor and quencer. Communication between agent elements is achieved through synthetic materials or visual components of the synthetic level. Agents can be classified as active or inactive. Active agents are used when data is required for transmission to DUT. Passive agents, on the other hand, are used when there is no need to drive data, and only testing or integration tasks are required. In the case of inactive agents, only the monitoring component enters the image.

• UVM Scoreboard

The scoreboard, as the name implies, is a way to keep track of the outcome of test cases. It compares the outputs of the test output with the output of DUT with the monitors for it. In general, it can be labeled as a feedback checker and is required in the final step to ensure the effectiveness of the project. Automatically performs the final output test process, compared to the standard Verilog test benches where the user needs to personally check the accuracy of the output output.

• UVM Sequencer

The sequencer is responsible for producing motives that are included in the experimental design. The sequencer produces a series of sequences, which are naturally random. This is done with the help of a limited randomization system of System Verilog. Consecutive items are usually in the form of class objects and are transferred to the driver, who will drive them to DUT. Some test benches also include a virtual tracker, where multiple sequences are located on a single test bench. This virtual tracker contains references to all other trackers and ensures that they provide synced sequential objects.

• UVM Driver

The driver is part of the UVM where the signals from the verification area are driven to the DUT interface. It picks up sequences and converts them from class objects into portable signals to DUT and drives them to construction using virtual connectors. The output of the DUT after receiving the installation sequence from the driver is given the next component, i.e., the monitor. Therefore, the driver's job involves sending the required data to DUT, not checking it. It is also involved in the definition of signal time, so that the signals operate according to the design of the design clock under test.

- **UVM Monitor**

In simple terms, the UVM Monitor performs the retrieval function compared to the driver. It receives signals from the design and is required to convert these signals into classroom objects understood by the verification environment. These classroom lessons are then sent to the test board to evaluate the output. To perform this function, the monitor is designed to have an analytical hole, from which it sends the transaction to the points board. The analysis hole is a phase formed by the interoperability level of the task, and which requires the inclusion of a writing task only, to be written on the score board. The purpose of the monitor, as the name suggests, is limited to caution. Regular monitoring and evaluation is done without a monitor, with a score board.

4.2 Design methodology of SoC Design Verification

Any System-on-chip contains several IP cores - CPU, memory and multiple peripherals. Each IP domain forms a verification component as shown in Figure. 4.2. Additionally, the virtual bus connector forms a separate authentication component as well. Each verification component contains the features described in Section 4.1

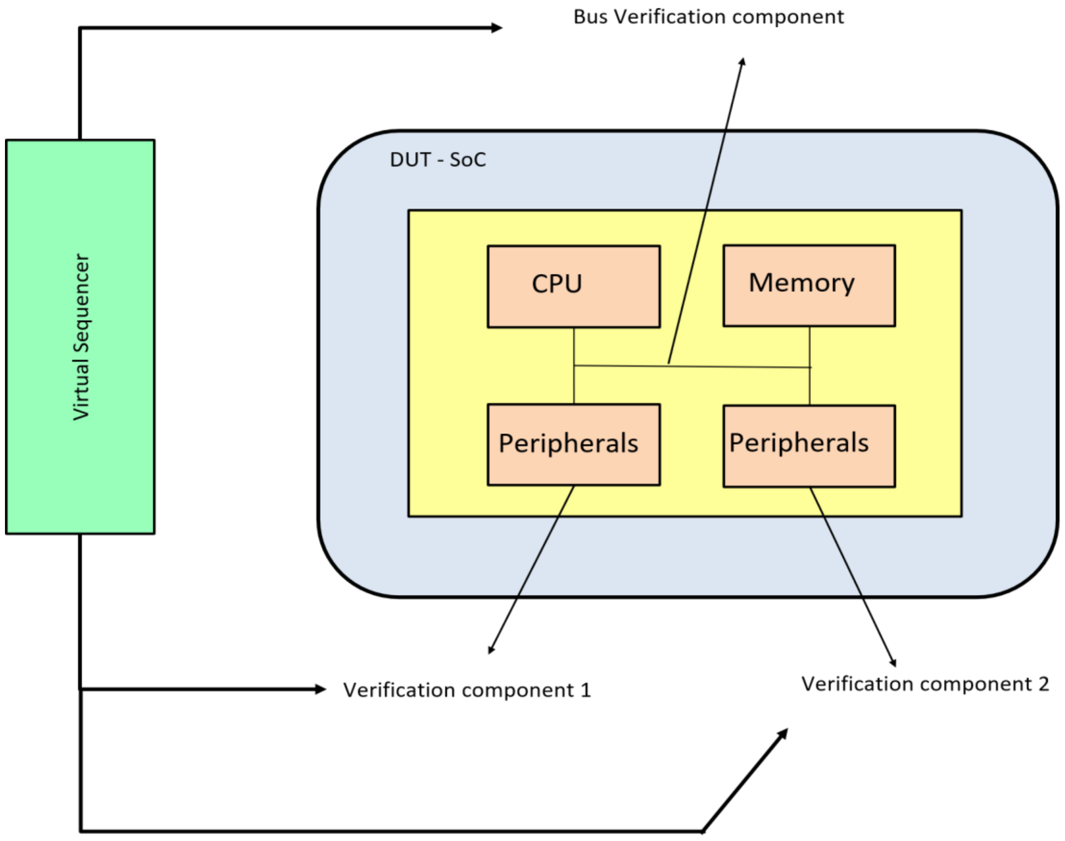


Figure 4.2: UVM Testbench design for SoC

However, there are a few additional features that ensure consistent performance of all components. Virtual tracker added, as shown in Figure. 4.2. The purpose of adding a virtual tracker is to ensure that the sequences of all verification components produce consistent motives. In fact, the material tracker consists of sequential identifiers for each of the verification components. Additionally, primary level verification will involve the installation of only one authentication components. SoC level verification requires the implementation of all components with a specific test incentive used in the test block. To clarify, the following section contains details of the design of the testbench site with AHB protocol.

4.3. Implementation using UVM Class Library

The UVM classroom library is used to build test benches in accordance with the UVM method. Contains basic classes, in which user-defined testbench components can be built, using asset concepts and class objects. This is where UVM reuse comes into play. The library helps to achieve this repetition on a large scale with the pre-defined basic classes.

Different basic classes are available

1. uvm void - this is the most basic base component of all other UVM classes

2. uvm root - root is automatically created when the testbench is executed. A variable pointer uvm top is used to access this base phase.

3. The uvm object - Describes all important functions based on class objects, such as creation, copy of the class, method of printing output, etc. It is also capable of producing random seeds for the production of motives.

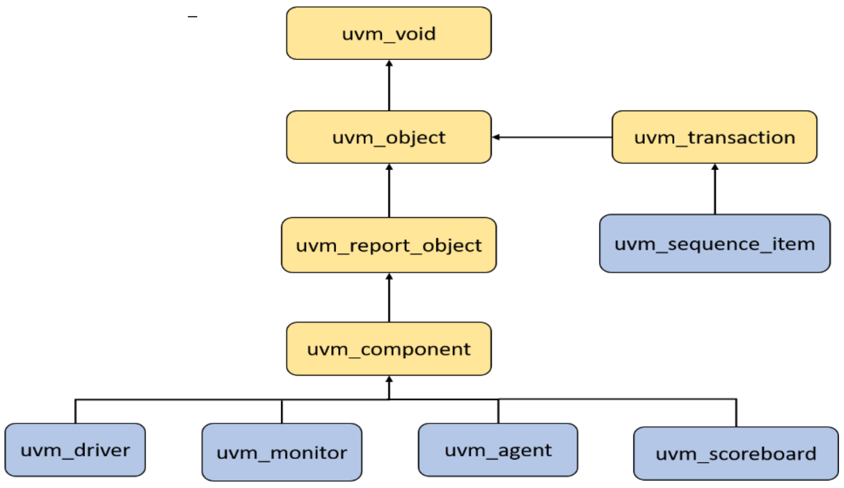


Figure 4.3: UVM Class Heirarchy

4. Uvm Reporting Object - UVM requires a virtual connector where any error messages or alerts should be reported to the user. This item has an obligation to act as a visual connector suitable for this purpose. All uvm transaction classes and part of uvm are taken from the uvm object.

5. uvm _component - This is one of the major categories. It has a wide variety of sub-categories taken from it, each of which is a class describing a specific part of the test bench, such as driver, monitoring, etc. The various categories are also defined in the UVM component.

6. Uvm transaction - Used mainly for renewable energy production

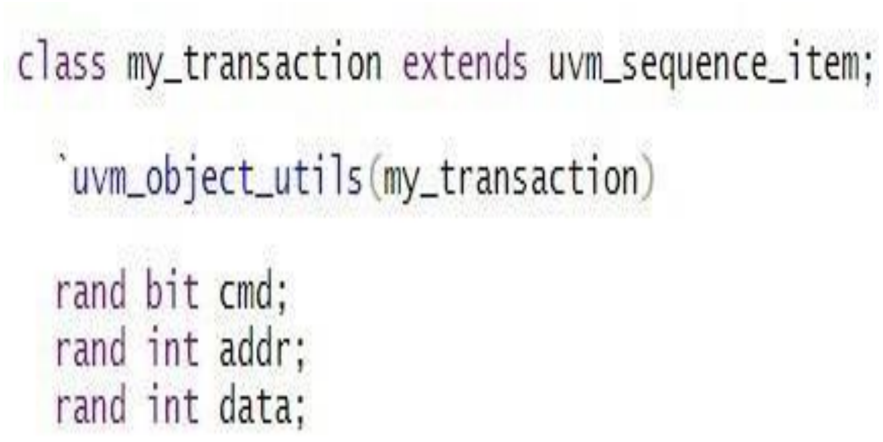
To use these basic classes in a UVM-designed test bench, they are expanded to

create a child class where specifications are mentioned. The following paragraphs indicate the implementation of these sections of the children's classes.

4.3.1 Modelling a sequence-item

Sequence is basically an activity that is responsible for making sequences. As the series builds inputs into the design under test, the sequence object acts as a catalyst for the structure. About the use of the base section of uvm sequence material comes in the picture. This is contained in the UVM class library and this base category was inherited as a defining custom sequence object.
There are also built-in procedures defined for use in acquired classes. For example, a print function () is available to extract data, create () to create data content and more.

The code snippet used to model a sequence item is shown in Figure 4.4:



```
class my_transaction extends uvm_sequence_item;

  `uvm_object_utils(my_transaction)

  rand bit cmd;
  rand int addr;
  rand int data;
```

Figure 4.4: Definition of a sequence item

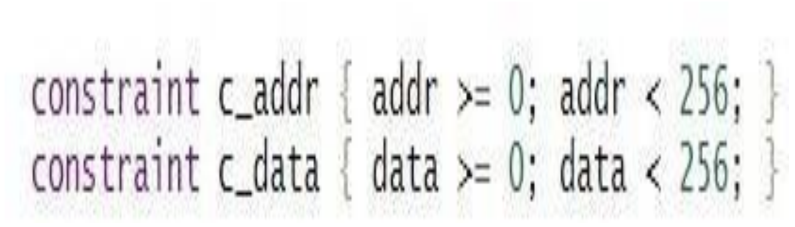
As shown, the class my work is explained by extending the class of uvm sequence object present in the UVM library. All classes taken from these basic classes are required to be registered using the uvm utils macro object. Data members include basic memory access address and data variables, as well as command bit.

Randomization

The data members of the sequence object class are defined using the keyword 'rand'. This is a feature of System Verilog, which allows the user to perform random dynamics using a randomize function. This is a much-needed feature in any production sequence, as sequences must be random in order to provide high-performance tests.

Constrained Randomization

Random performance is always below the limits of all data variables. Limits to be added to random variables are set using the System Verilog blocking feature. To limit dynamic address and data into 8-bits or 1 byte, barriers are written as shown in Figure 4.5.



```
constraint c_addr { addr >= 0; addr < 256; }
constraint c_data { data >= 0; data < 256; }
```

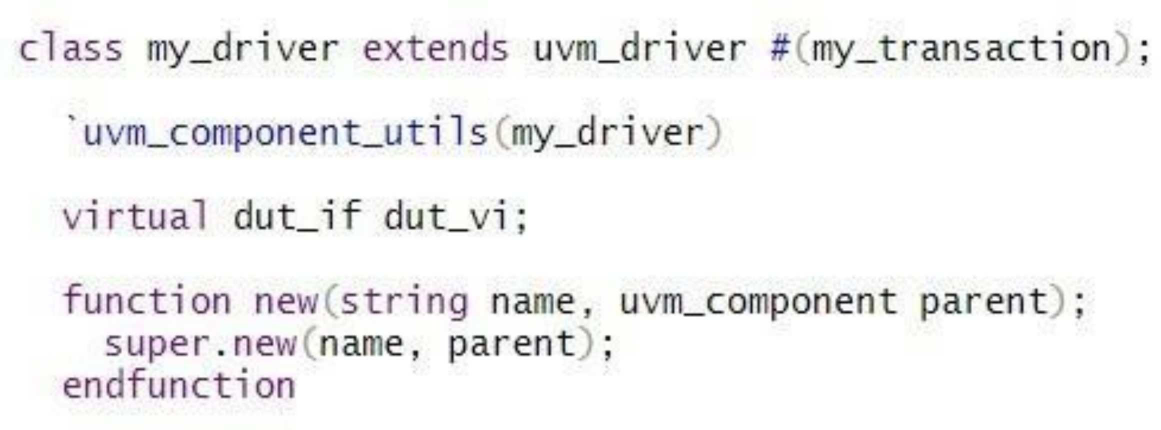
Figure 4.5: Constraints on sequence item data

4.3.2 Building UVM Driver

When stimuli are produced, they should be carried on in the formulation. This is possible by using the driver component. It works as a link between the sequencer and the design using the interface protocol. Any contact between the driver and the tracker creates the need for a hole level model of the action on the driver's side. The stimu from the sequencer comes in the form of a class object, however, these require flexible conversion / design-sensitive signals. This modification is the main function of the driver. The implementation of this driver in the test bench code is possible using the uvm driver base section installed in the UVM library. The defined driver categories are inherited from this base category. However, as can be seen in the building test bench in Section 4.1, DUT signals cannot be driven directly to the system based in the System Verilog class. The interface was defined for that purpose, in order to transmit DUT signals.

The description of the driver component is shown in Figure 4.6

As shown, my driver class is defined by extending the uvm driver category available in the UVM library. The sequence object, my work produced by the predefined tracker, is sent to the driver as a parameter. Same as sequence.

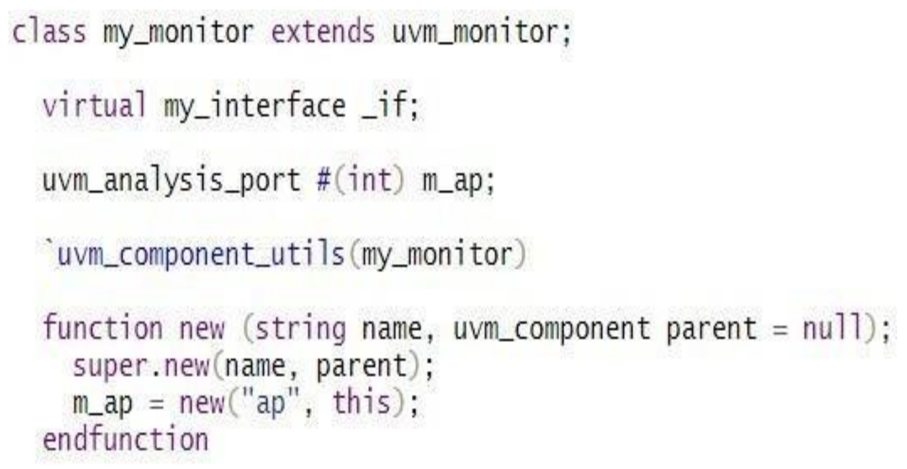


```
class my_driver extends uvm_driver #(my_transaction);

  `uvm_component_utils(my_driver)

  virtual dut_if dut_vi;

  function new(string name, uvm_component parent);
    super.new(name, parent);
  endfunction
```

Figure 4.6: Driver definition

The thing is, all the classes taken from the basic component classes are required to be registered using the uvm utils macro component.

Additionally, a description of the previously defined visual interface is displayed. Finally, the classroom was built using the architect of its parents' section, here, which is part of the uvm section.



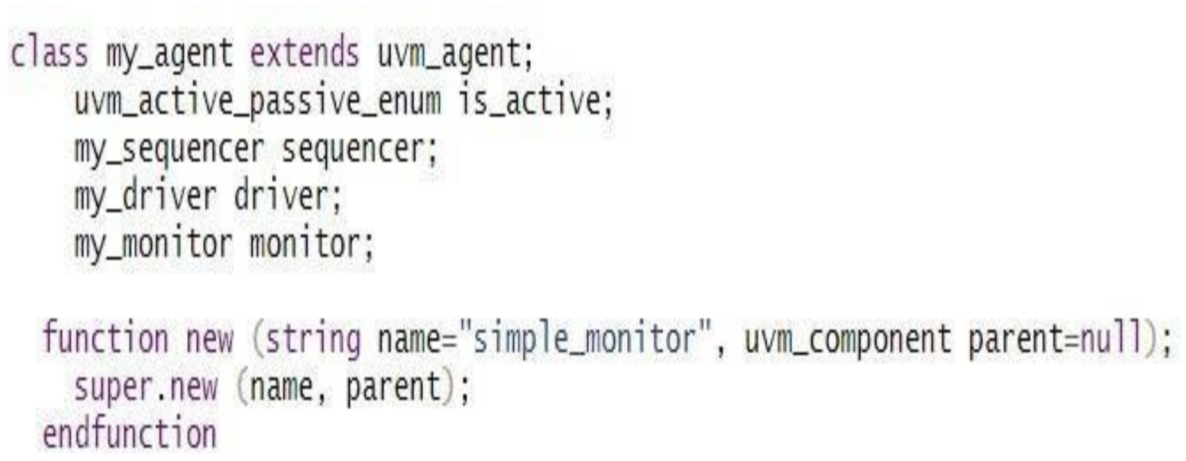
```
class my_monitor extends uvm_monitor;

  virtual my_interface _if;

  uvm_analysis_port #(int) m_ap;

  `uvm_component_utils(my_monitor)

  function new (string name, uvm_component parent = null);
    super.new(name, parent);
    m_ap = new("ap", this);
  endfunction
```

Figure 4.7: Monitor definition

Additionally, virtual interface handles are announced for the monitor to communicate with DUT. An analysis port is announced. The purpose of the analysis port is to send the collected data to the points board, for review and reporting.

4.3.3 Building Agent

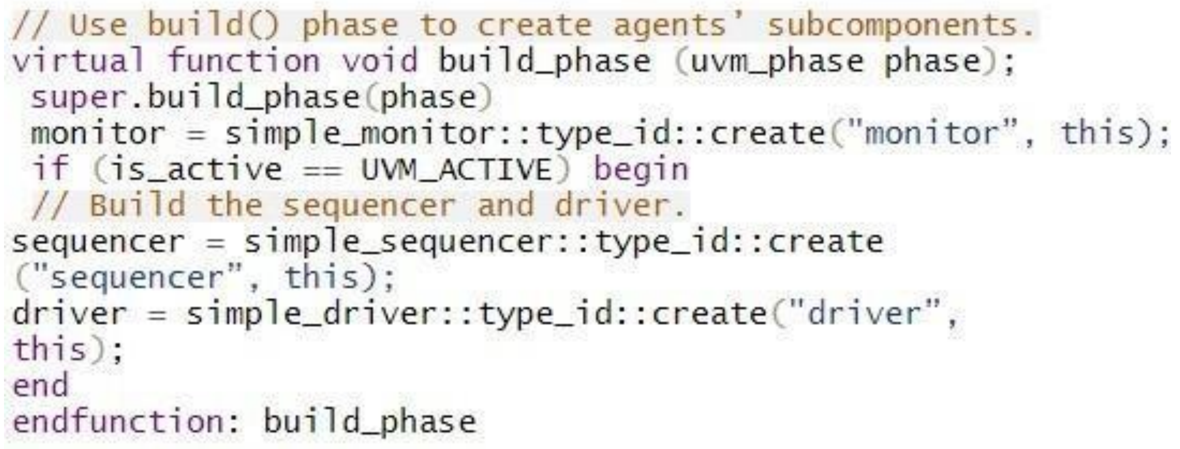
The agent is created by expanding the base phase, the uvm agent. It was built using a parent class builder and registered using utils macro. The sub-components of the agent class i.e., driver, monitor and object follower are declared as objects of their classrooms as described earlier. This definition is shown in Figure 4.8 The active uvm type of passive enum is used to indicate whether the agent is active or not. In the event of an inactive agent, only a monitor should be built. On the other hand, an active agent requires a driver, sequencer and monitoring to be built.



```
class my_agent extends uvm_agent;
   uvm_active_passive_enum is_active;
   my_sequencer sequencer;
   my_driver driver;
   my_monitor monitor;

  function new (string name="simple_monitor", uvm_component parent=null);
    super.new (name, parent);
  endfunction
```

Figure 4.8: Agent definition

The lower parts were then created or constructed using construction work (), which is part of the construction phase of the UVM flow. A monitor is built first, then a sequencer and driver and then built depending on whether the agent is working or not. This construction method can be seen in Figure 4.9



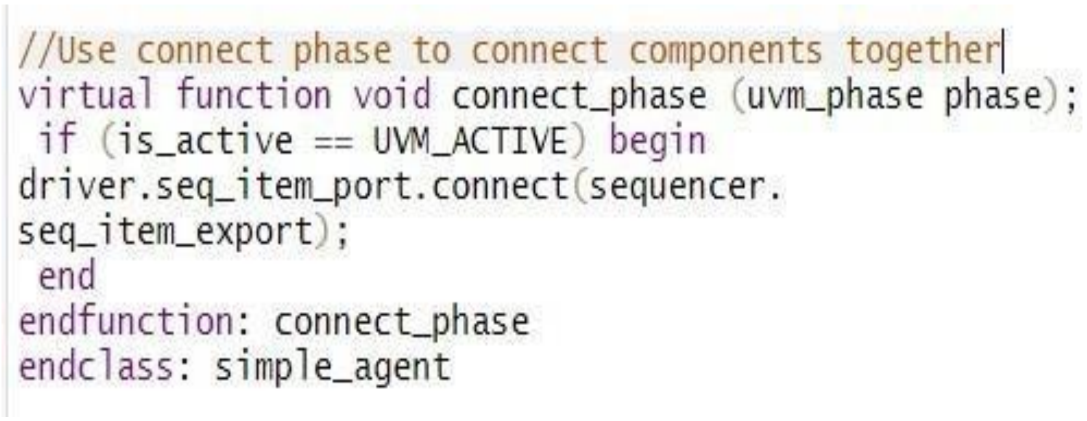
```
// Use build() phase to create agents' subcomponents.
virtual function void build_phase (uvm_phase phase);
 super.build_phase(phase)
 monitor = simple_monitor::type_id::create("monitor", this);
 if (is_active == UVM_ACTIVE) begin
 // Build the sequencer and driver.
sequencer = simple_sequencer::type_id::create
("sequencer", this);
driver = simple_driver::type_id::create("driver",
this);
end
endfunction: build_phase
```

Figure 4.9: Building the sub-components of the agent

The components are then connected using a connection method, as shown in Figure 4.10. This involves connecting the driver to the tracker, so that the scanner-generated object is available to the driver.

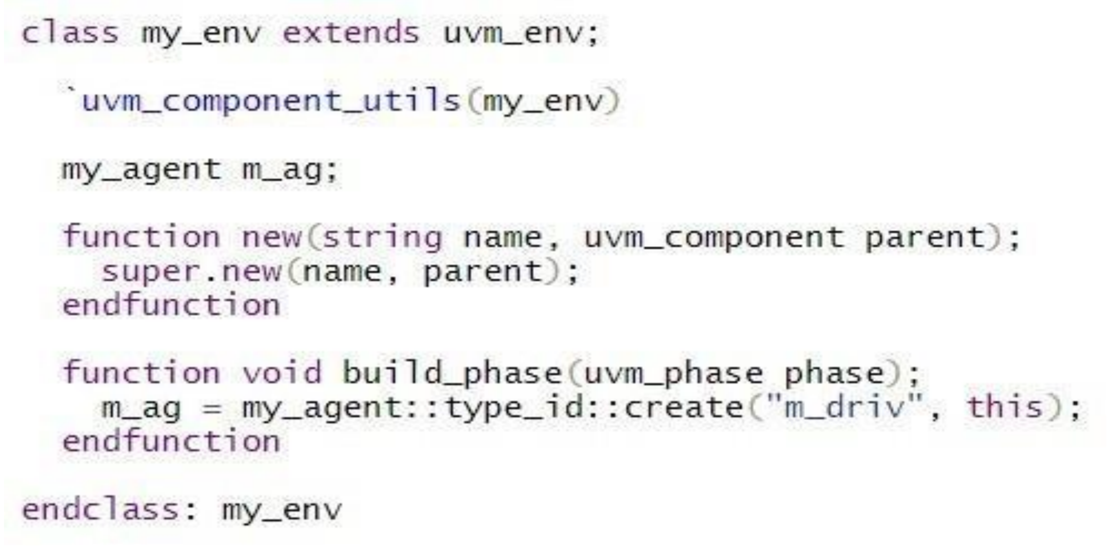
4.3.4 Building the environment

The surrounding area is the highest level of the system and consists of the components described in the paragraphs above. All the components described above are reinforced in the surrounding area. The reusable theme is seen in this category of high-volume containers, where the developer creates a local category and builds it with small components for specific verification tests. A description of the natural class is shown in Figure 4.11



```
//Use connect phase to connect components together
virtual function void connect_phase (uvm_phase phase);
 if (is_active == UVM_ACTIVE) begin
driver.seq_item_port.connect(sequencer.
seq_item_export);
 end
endfunction: connect_phase
endclass: simple_agent
```

Figure 4.10: Connecting the sub-components of the agent



```
class my_env extends uvm_env;

  `uvm_component_utils(my_env)

  my_agent m_ag;

  function new(string name, uvm_component parent);
    super.new(name, parent);
  endfunction

  function void build_phase(uvm_phase phase);
    m_ag = my_agent::type_id::create("m_driv", this);
  endfunction

endclass: my_env
```

Figure 4.11: Definition of the environment class

As can be seen in Figure 4.11, the natural phase declares the agent class object, m ag. At the SoC-level site, many such agents are announced depending on the number of agent certification components. The natural class is built by calling the builder part of the parent category. In the construction phase of the natural phase, the declared agents are built or strengthened.

In this way, the UVM components are defined and reinforced to form a test bench. The next section will explain in more detail the direct use of test benches in relation to SoC protection blocks.

4.4 SoC level Verification of security blocks

The use of UVM test benches as discussed in Section 4.4 was applied to certain SoC protection blocks, to ensure their effectiveness. SoC quality verification requires the following factors to be verified:

- Verification of the functions of a particular SoC IP block
- The interaction of the block with other IP cores is present in the SoC
- Performance verification and SoC power-related features

To achieve this, the test program is performed according to the design of the IP security cables. The test bench is then designed according to the requirements of the test plan

4.4.1 Test plan

The first phase of the verification flow was the design of the test system from the IP-based design specification. An evaluation plan was needed to determine the specific tasks to be assessed, the location to be assessed, and indicators to announce results such as passing or failing. The following factors have been determined to be validated in the evaluation process:

• Power encryption capability with different cryptographic algorithms.

• Read-write activities between processor and security environment.

• Memory and registration access via security cores.

4.4.2 Test development

The test is built by adding additional features to the usable UVM environment. Depending on the test site, the test system is not only written using UVM or System Verilog, but also includes advanced languages at the compression level, such as C / C ++.

There were 2 types of tests designed to verify performance, namely, performance testing and gate level simulation

Functional tests

Functional tests, as the name implies, were written to ensure the performance specificity of the test only, without the inclusion of time or operational requirements. This is the most comprehensive and most common test performed and the fastest in time tp run these tests.

Gate-level simulation

Gate-level simulation is also performed to validate the design by exchanging a high-quality RTL design with a gate-level net list. This is because the actual net list may change after a specific installation, such as a clock. These changes must be verified and evaluated, which is a function of gate level simulation

4.4.3 Test debug

Improved testing is performed on SoC, and any test failure requires error correction to determine the cause of the failure. Failures may be due to errors in SoC formation or errors in the test bench area. Test bench bugs were corrected, and design errors were reported and suggestions were changed to achieve a 100% pass rate in all test conditions.

The benchmark of the UVM test bench and its customization level of SoC authentication is highlighted. The implementation of the architectural design was discussed, as well as the inspection details of the SoC protection blocks. A detailed flow of experimental implementation has been described. The next chapter contains the results of the test used, as well as the error-correcting procedure to remove any failures in test testing.

CHAPTER 5

RESULT AND DISCURSSION

RESULTS AND DISCUSSIONS

The concepts of SoC integration and design validation discussed so far require careful and detailed implementation in the RAS. The results obtained from the application are necessary to ensure the accuracy of the design and to make further integration and verification cycles, if necessary. The results obtained after integrating and validating the various IP Cores are discussed in this chapter. It also presents a variety of simulation logs and results generated during integration and design validation, as well as a brief summary of the pass / fail status for advanced testing.

5.1 Simulation results

5.1.1 The integration of RAS and validation seems to be done successfully. The simulation results are detailed in the sections below.

5.1.2 Simulation of various interfacing protocols

Various interfacing protocols were written using Verilog and copied using the Xilinx Vivado suite. The simulation results of using the BGR protocol for PUNIT IERR authentication are shown in Figures 5.1 and 5.2

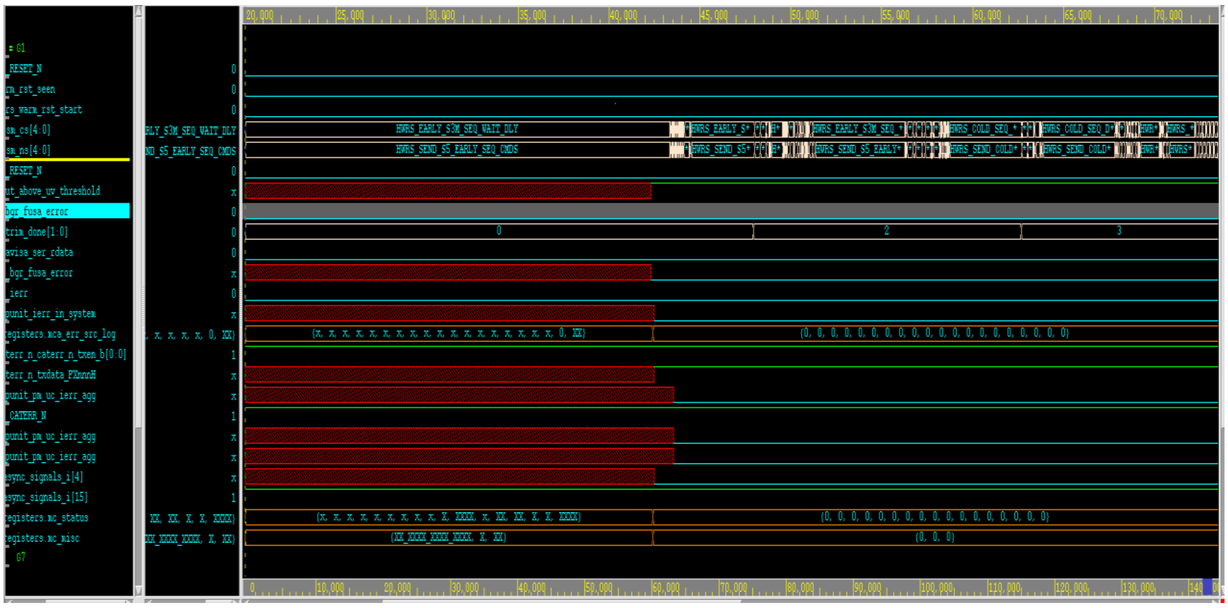


Figure 5.1: Simulation of BGR protocol

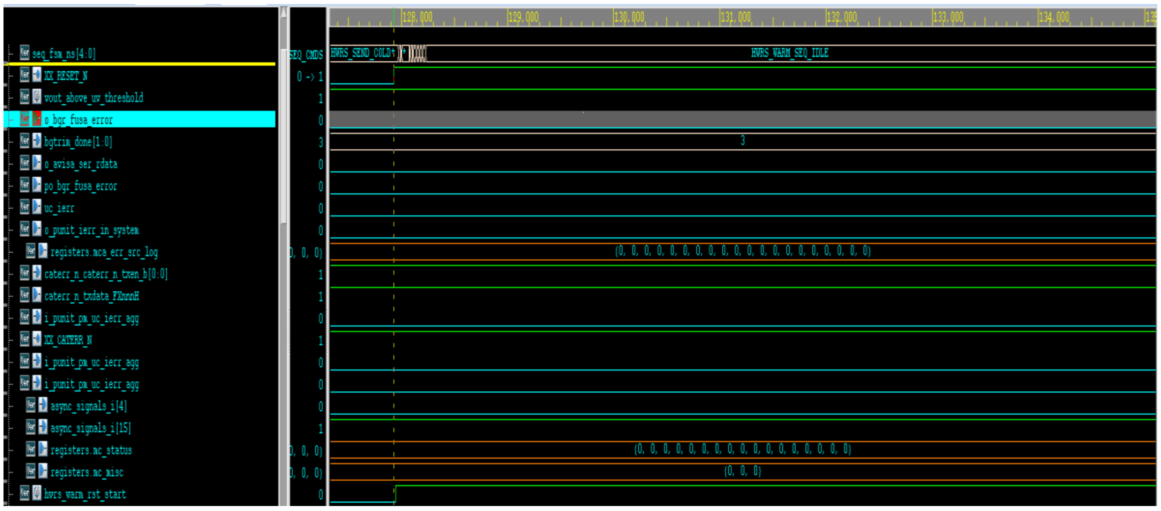


Figure 5.2: Simulation of BGR protocol in PUNIT IERR

The simulation results for implementing PLL protocol in PUNIT IERR Validation are shown in figure 5.3 and 5.4

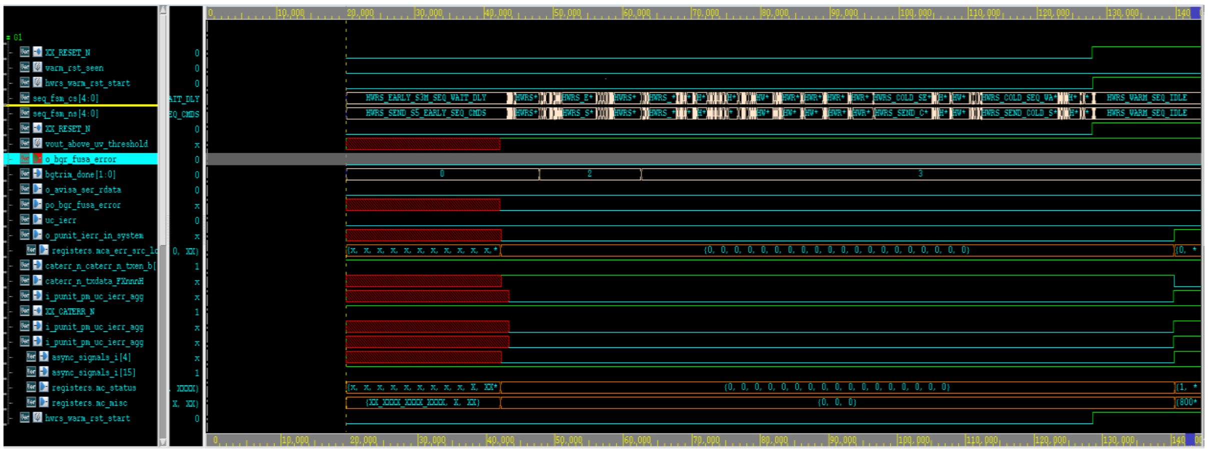


Figure 5.3: Simulation of BGR protocol

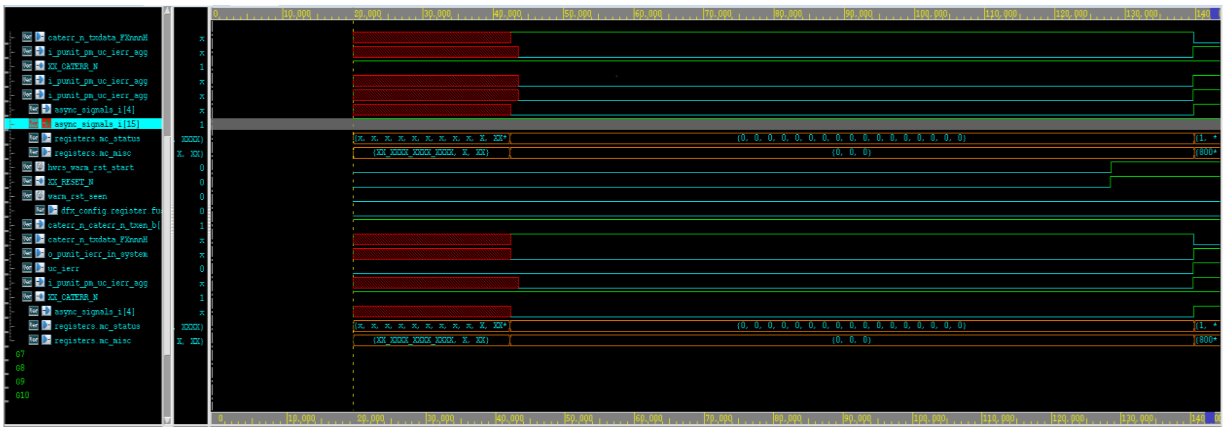


Figure 5.4: Simulation of PLL protocol

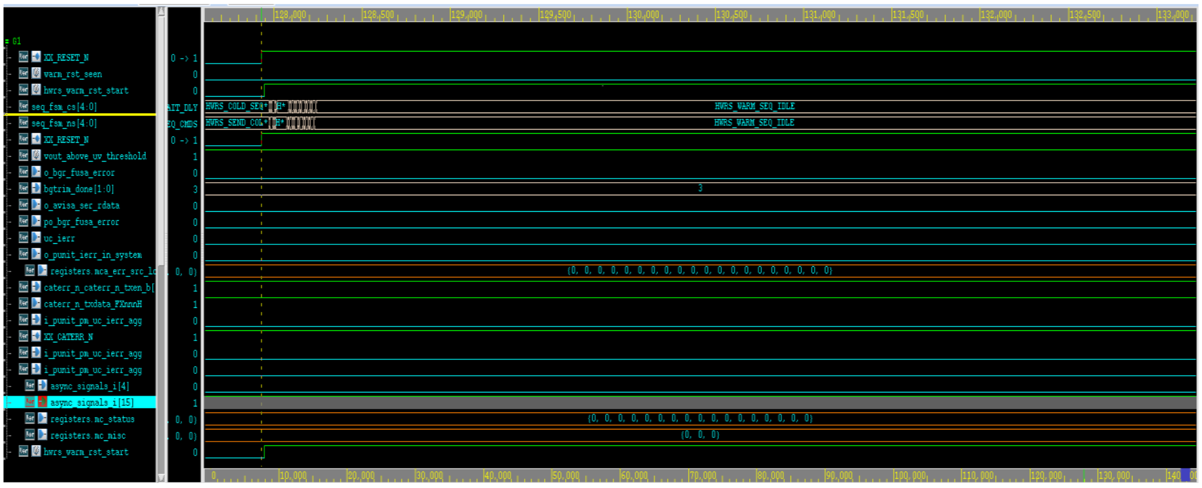


Figure 5.5: Simulation of PLL protocol in PUNIT IERR

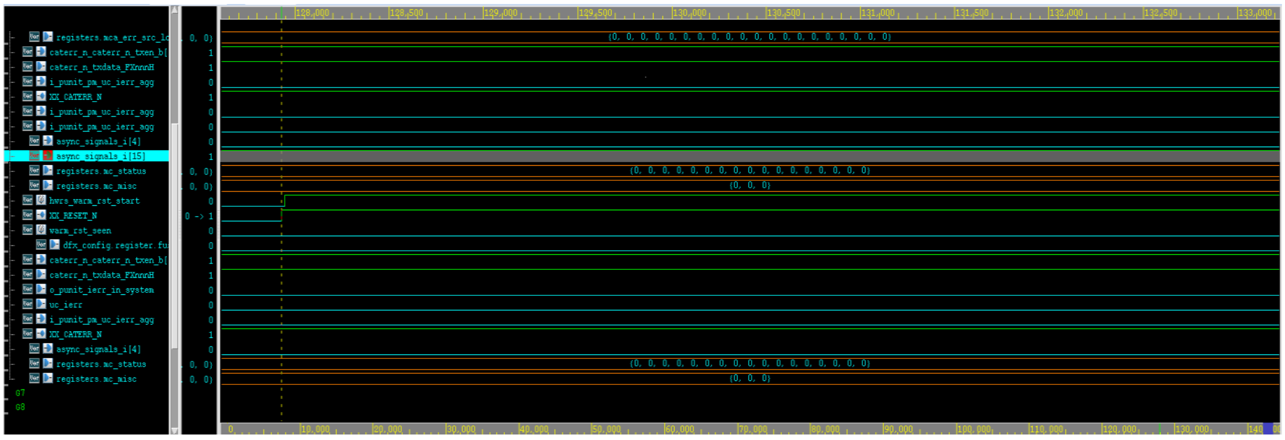


Figure 5.6: Simulation of PLL protocol in PUNIT IERR

5.1.2 Simulation logs of the connectivity tool

Figure 5.7 shows a screenshot of the log file that is got after the completion of the connectivity run tool. Figure 5.8 and 5.9 shows the resource consumed during the run and figure

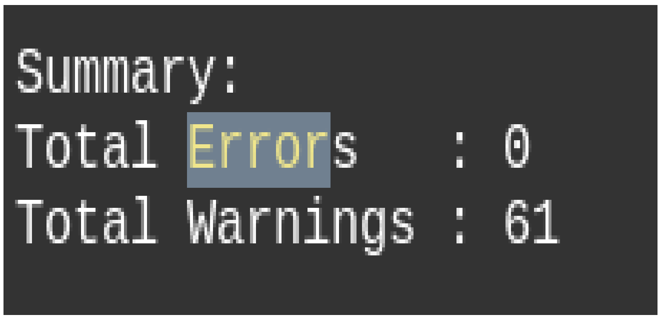
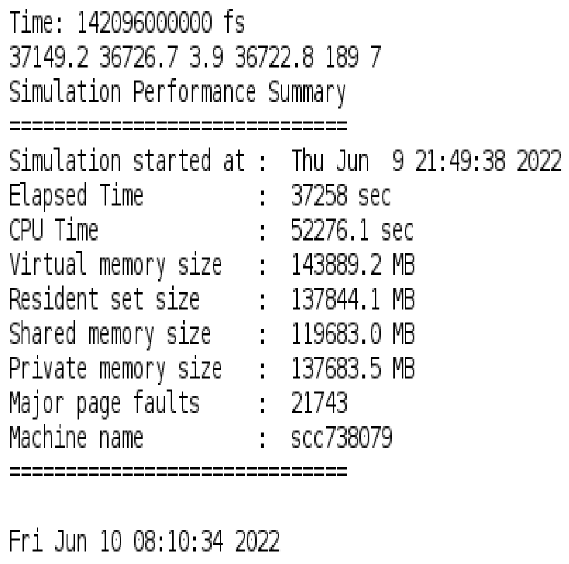

```
Summary:
Total Errors   : 0
Total Warnings : 61
```

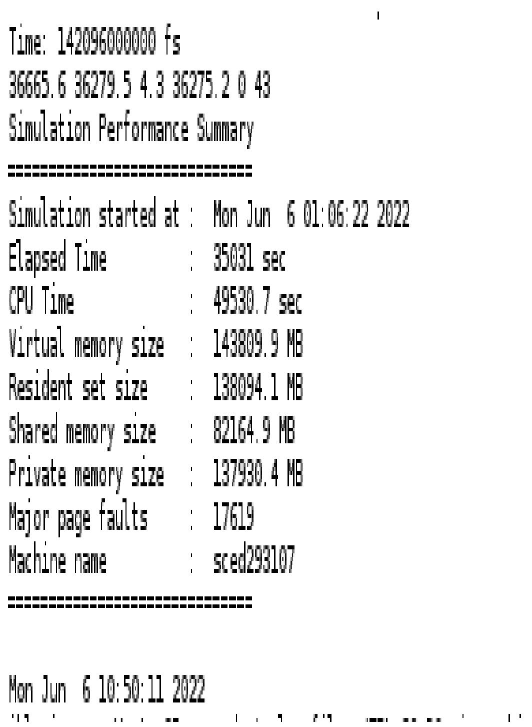
Figure 5.7: Screenshot of the log file generated

The list of errors during the entire timeline of the project. These are caused mainly due to the addition or deletion of IP core wrappers as per the ASIC/SoC requirements.



```
Time: 142096000000 fs
37149.2 36726.7 3.9 36722.8 189 7
Simulation Performance Summary
==============================
Simulation started at :  Thu Jun  9 21:49:38 2022
Elapsed Time          :  37258 sec
CPU Time              :  52276.1 sec
Virtual memory size   :  143889.2 MB
Resident set size     :  137844.1 MB
Shared memory size    :  119683.0 MB
Private memory size   :  137683.5 MB
Major page faults     :  21743
Machine name          :  scc738079
==============================

Fri Jun 10 08:10:34 2022
```

Figure 5.8: Resource Usage for PLL Sequence



```
Time: 142096000000 fs
36665.6 36279.5 4.3 36275.2 0 43
Simulation Performance Summary
==============================
Simulation started at :  Mon Jun  6 01:06:22 2022
Elapsed Time          :  35031 sec
CPU Time              :  49530.7 sec
Virtual memory size   :  143809.9 MB
Resident set size     :  138094.1 MB
Shared memory size    :  82164.9 MB
Private memory size   :  137930.4 MB
Major page faults     :  17619
Machine name          :  sced293107
==============================

Mon Jun  6 10:50:11 2022
```

Figure 5.9: Resource Usage for BGR Sequence

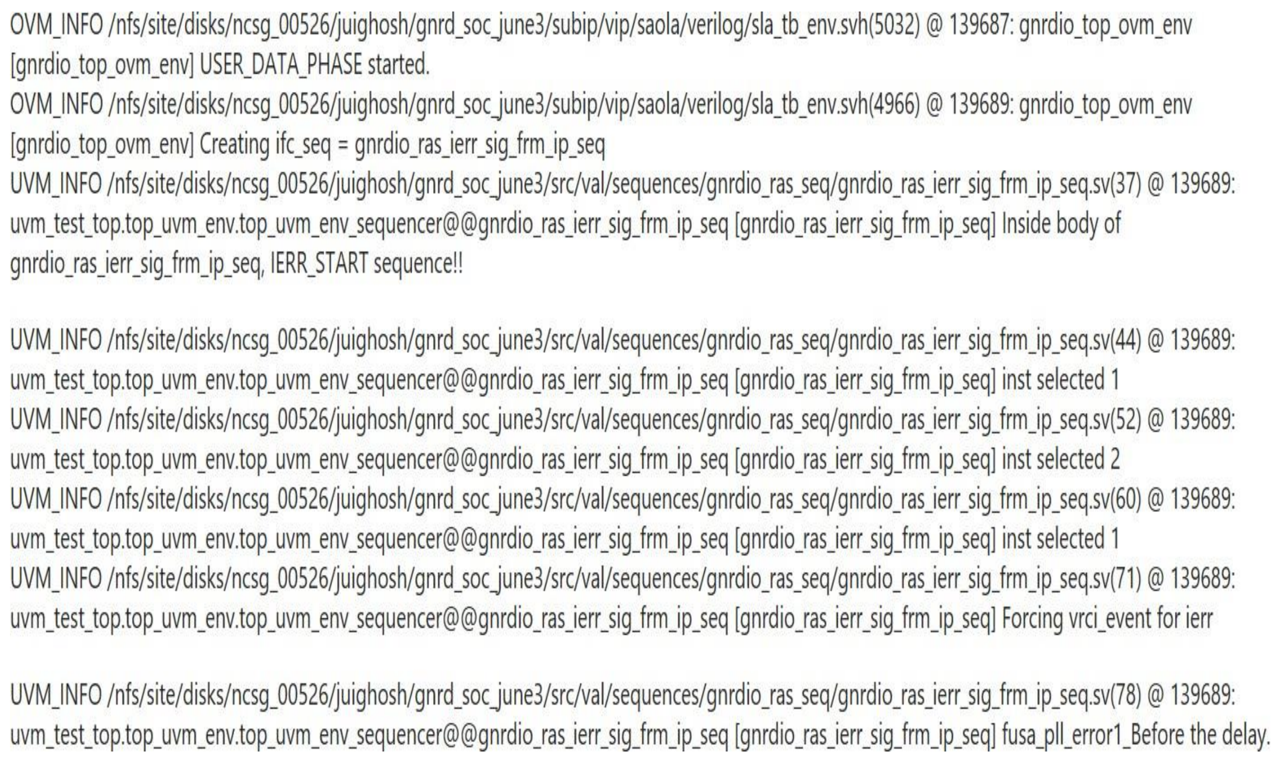
5.1.3 Simulation logs of UVM testbenches

The simulation of any UVM test bench produces a log of the components called and the order followed by the UVM flow. It starts with the construction phase, where the components are built from the upper level. Then it is followed by the connecting section, where the parts are connected at the top and bottom. eliminates imitation runs from the lowest level of the class.

This generated log shows that the UVM phases are successfully run, with the required components being strengthened and connected. The simulation results, however, are produced by a scoreboard, which tells the user, whether the design has passed the test or not.

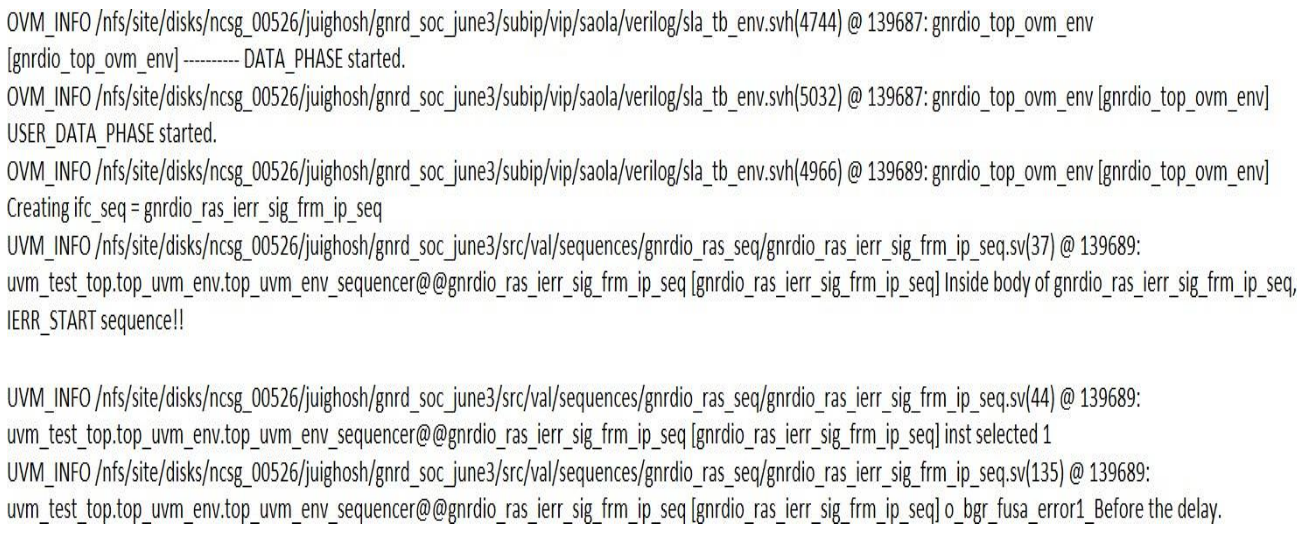
5.1.4 Simulation result of memory access tests:

The simulation results for a memory access and read verification test is shown in Figure 5.10 and Figure 5.11 The data fields are address, write data read data and write signal, which indicates whether a read or write is taking place.



```
OVM_INFO /nfs/site/disks/ncsg_00526/juighosh/gnrd_soc_june3/subip/vip/saola/verilog/sla_tb_env.svh(5032) @ 139687: gnrdio_top_ovm_env
[gnrdio_top_ovm_env] USER_DATA_PHASE started.
OVM_INFO /nfs/site/disks/ncsg_00526/juighosh/gnrd_soc_june3/subip/vip/saola/verilog/sla_tb_env.svh(4966) @ 139689: gnrdio_top_ovm_env
[gnrdio_top_ovm_env] Creating ifc_seq = gnrdio_ras_ierr_sig_frm_ip_seq
UVM_INFO /nfs/site/disks/ncsg_00526/juighosh/gnrd_soc_june3/src/val/sequences/gnrdio_ras_seq/gnrdio_ras_ierr_sig_frm_ip_seq.sv(37) @ 139689:
uvm_test_top.top_uvm_env.top_uvm_env_sequencer@@gnrdio_ras_ierr_sig_frm_ip_seq [gnrdio_ras_ierr_sig_frm_ip_seq] Inside body of
gnrdio_ras_ierr_sig_frm_ip_seq, IERR_START sequence!!

UVM_INFO /nfs/site/disks/ncsg_00526/juighosh/gnrd_soc_june3/src/val/sequences/gnrdio_ras_seq/gnrdio_ras_ierr_sig_frm_ip_seq.sv(44) @ 139689:
uvm_test_top.top_uvm_env.top_uvm_env_sequencer@@gnrdio_ras_ierr_sig_frm_ip_seq [gnrdio_ras_ierr_sig_frm_ip_seq] inst selected 1
UVM_INFO /nfs/site/disks/ncsg_00526/juighosh/gnrd_soc_june3/src/val/sequences/gnrdio_ras_seq/gnrdio_ras_ierr_sig_frm_ip_seq.sv(52) @ 139689:
uvm_test_top.top_uvm_env.top_uvm_env_sequencer@@gnrdio_ras_ierr_sig_frm_ip_seq [gnrdio_ras_ierr_sig_frm_ip_seq] inst selected 2
UVM_INFO /nfs/site/disks/ncsg_00526/juighosh/gnrd_soc_june3/src/val/sequences/gnrdio_ras_seq/gnrdio_ras_ierr_sig_frm_ip_seq.sv(60) @ 139689:
uvm_test_top.top_uvm_env.top_uvm_env_sequencer@@gnrdio_ras_ierr_sig_frm_ip_seq [gnrdio_ras_ierr_sig_frm_ip_seq] inst selected 1
UVM_INFO /nfs/site/disks/ncsg_00526/juighosh/gnrd_soc_june3/src/val/sequences/gnrdio_ras_seq/gnrdio_ras_ierr_sig_frm_ip_seq.sv(71) @ 139689:
uvm_test_top.top_uvm_env.top_uvm_env_sequencer@@gnrdio_ras_ierr_sig_frm_ip_seq [gnrdio_ras_ierr_sig_frm_ip_seq] Forcing vrci_event for ierr

UVM_INFO /nfs/site/disks/ncsg_00526/juighosh/gnrd_soc_june3/src/val/sequences/gnrdio_ras_seq/gnrdio_ras_ierr_sig_frm_ip_seq.sv(78) @ 139689:
uvm_test_top.top_uvm_env.top_uvm_env_sequencer@@gnrdio_ras_ierr_sig_frm_ip_seq [gnrdio_ras_ierr_sig_frm_ip_seq] fusa_pll_error1_Before the delay.
```

Figure 5.10: Simulation result of a memory access test for ierr_pll sequence

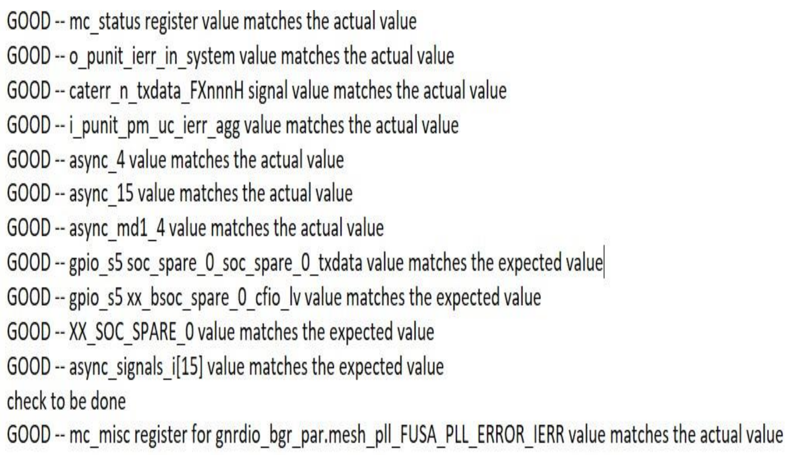


```
OVM_INFO /nfs/site/disks/ncsg_00526/juighosh/gnrd_soc_june3/subip/vip/saola/verilog/sla_tb_env.svh(4744) @ 139687: gnrdio_top_ovm_env
[gnrdio_top_ovm_env] ---------- DATA_PHASE started.
OVM_INFO /nfs/site/disks/ncsg_00526/juighosh/gnrd_soc_june3/subip/vip/saola/verilog/sla_tb_env.svh(5032) @ 139687: gnrdio_top_ovm_env [gnrdio_top_ovm_env]
USER_DATA_PHASE started.
OVM_INFO /nfs/site/disks/ncsg_00526/juighosh/gnrd_soc_june3/subip/vip/saola/verilog/sla_tb_env.svh(4966) @ 139689: gnrdio_top_ovm_env [gnrdio_top_ovm_env]
Creating ifc_seq = gnrdio_ras_ierr_sig_frm_ip_seq
UVM_INFO /nfs/site/disks/ncsg_00526/juighosh/gnrd_soc_june3/src/val/sequences/gnrdio_ras_seq/gnrdio_ras_ierr_sig_frm_ip_seq.sv(37) @ 139689:
uvm_test_top.top_uvm_env.top_uvm_env_sequencer@@gnrdio_ras_ierr_sig_frm_ip_seq [gnrdio_ras_ierr_sig_frm_ip_seq] Inside body of gnrdio_ras_ierr_sig_frm_ip_seq,
IERR_START sequence!!

UVM_INFO /nfs/site/disks/ncsg_00526/juighosh/gnrd_soc_june3/src/val/sequences/gnrdio_ras_seq/gnrdio_ras_ierr_sig_frm_ip_seq.sv(44) @ 139689:
uvm_test_top.top_uvm_env.top_uvm_env_sequencer@@gnrdio_ras_ierr_sig_frm_ip_seq [gnrdio_ras_ierr_sig_frm_ip_seq] inst selected 1
UVM_INFO /nfs/site/disks/ncsg_00526/juighosh/gnrd_soc_june3/src/val/sequences/gnrdio_ras_seq/gnrdio_ras_ierr_sig_frm_ip_seq.sv(135) @ 139689:
uvm_test_top.top_uvm_env.top_uvm_env_sequencer@@gnrdio_ras_ierr_sig_frm_ip_seq [gnrdio_ras_ierr_sig_frm_ip_seq] o_bgr_fusa_error1_Before the delay.
```

Figure 5.11: Simulation result of a memory access test for ierr_bgr sequence

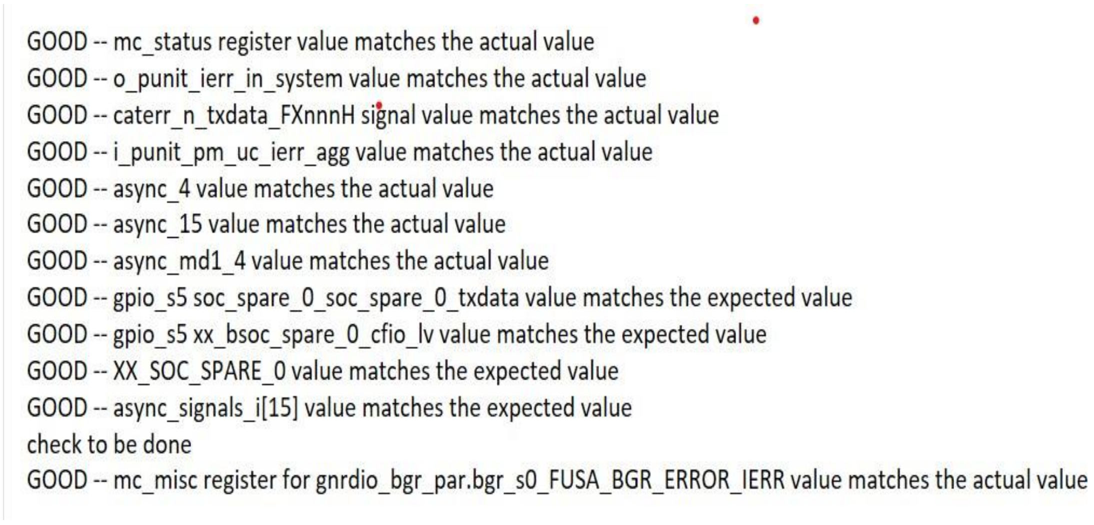
UVM_INFO shows the sequence of events taking place during simulation at sequence of events taking place during simulation at specified time interval, which are as follows:

- In simulation, first we create the data phase for that particular test case otherwise UVM does not understand for which test case we are checking the value as well as the read operation.
- This data phase has been created after that we create ierr file from the test list file we then commented all the sequence files except the bgr and pll sequence file.
- This pll and bgr sequence file has been created in this file and we are forcing the error injecting value '1' in one bit for bgr and pll sequence.
- In the sequence file's self-check part, we can check the sequence, if the sequence is matching, we get the actual value, else an error is generating below which seems like this



```
GOOD -- mc_status register value matches the actual value
GOOD -- o_punit_ierr_in_system value matches the actual value
GOOD -- caterr_n_txdata_FXnnnH signal value matches the actual value
GOOD -- i_punit_pm_uc_ierr_agg value matches the actual value
GOOD -- async_4 value matches the actual value
GOOD -- async_15 value matches the actual value
GOOD -- async_md1_4 value matches the actual value
GOOD -- gpio_s5 soc_spare_0_soc_spare_0_txdata value matches the expected value
GOOD -- gpio_s5 xx_bsoc_spare_0_cfio_lv value matches the expected value
GOOD -- XX_SOC_SPARE_0 value matches the expected value
GOOD -- async_signals_i[15] value matches the expected value
check to be done
GOOD -- mc_misc register for gnrdio_bgr_par.mesh_pll_FUSA_PLL_ERROR_IERR value matches the actual value
```

Figure 5.12: Check the input and output of a PLL Sequence



```
GOOD -- mc_status register value matches the actual value
GOOD -- o_punit_ierr_in_system value matches the actual value
GOOD -- caterr_n_txdata_FXnnnH signal value matches the actual value
GOOD -- i_punit_pm_uc_ierr_agg value matches the actual value
GOOD -- async_4 value matches the actual value
GOOD -- async_15 value matches the actual value
GOOD -- async_md1_4 value matches the actual value
GOOD -- gpio_s5 soc_spare_0_soc_spare_0_txdata value matches the expected value
GOOD -- gpio_s5 xx_bsoc_spare_0_cfio_lv value matches the expected value
GOOD -- XX_SOC_SPARE_0 value matches the expected value
GOOD -- async_signals_i[15] value matches the expected value
check to be done
GOOD -- mc_misc register for gnrdio_bgr_par.bgr_s0_FUSA_BGR_ERROR_IERR value matches the actual value
```

Figure 5.13: Check the input and output of a BGR Sequence

- If error has generated, we can see this message in our sequence’s output part of that log file
- This UVM INFO shows that error has generate in output of the bgr and pll file.
- The PLL file works as clock monitoring and BGR file works as voltage regulator When the PLL error occurs we encounter PLL lock loss error, and the status registers cannot be read by the software. When the BGR fails, IP becomes unreachable via sideband network.
- Then we debug that pll and bgr sequence, we recieve figure 5.1- 5.6 waveform
- This bgr and pll error sequence goes to Global IERR and MONFAIL.
- Global IERR basically works on soft faults in IO die network to accumulate all these errors and propagate it to PUNITs, Now PUNITs can take appropriate action such as error loop/triggering crash log.
- MONFAIL basically works on hard faults in C die network this is monitored by external safety element to detect an internal error reported by clock/voltage monitors.

5.1.5 Details of Tests for Sequences:

These types of tests are built by running against the RAS design under test. The initial state of passing / failure of the defense blocks of these tests as shown in Table 5.2

Debug fixes are then performed on failed tests. The reason for the few failures is found to be an error in the UVM verification test area. Processor requests do not reach the security blocks as expected. The error has been fixed in the following

Table 5.1: Initial pass/fail status of security blocks

	Total No. of tests	No. of tests passed	No. of tests failed
PLL	18	10	8
BGR	18	7	11

Table 5.2: Subsequent Pass/fail status of security blocks

	Total No. of tests	No. of tests passed	No. of tests failed
PLL	18	14	4
BGR	18	10	8

test cycle. The pass / failure rate for defense blocks for the next round test is shown in Table 5.1

Debugging is then performed on the remaining failed tests. This means that the block does not receive any signal / instructions from the processor. The address recording has been edited, and the tests have been performed, and the pass / failure status is as shown in Table 5.3. A 100% pass is achieved in the final round of testing.

Table 5.3: Final Pass/fail status of security blocks

	Total No. of tests	No. of tests passed	No. of tests failed
Security Block 1	20	20	0
Security Block 2	12	12	0

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

A system-on-chip (SoC) is a single substrate or microchip that contains all or part of a computer, including a central processing unit (CPU), memory, inlet / outlet holes, and secondary storage, and auxiliary components. such as radio modems and graphics processing unit (GPU). Due to the reduction in transistor sizes, Moore's law predicts that the density of connected circuits on a chip will almost double twice a year. As a result, advances in VLSI architecture have seen a shift from computer architecture to interconnected projects with a large number of IP cores to be connected to the internet and fully tested.

The main objectives of the project are to integrate IP cores and perform block authentication at SoC level, with a focus on SoC-related security blocks. Before performing integration and verification, all SoC architecture, IP-based design, integration details, port details, port functionality, integration agreements must be understood. Verification is done by building test benches using a universal verification method, and continuing to correct the error in the event of a failed test. Among the many IP Cores, Ethernet block, PCIE block, fewer pipes, throttle blocks and interface controller block are integrated and crypto-related and security blocks are guaranteed.

The design of IP Cores is sent by the design team and its functions which include details, structures, port details, direction and functions in the form of csv files. The specifications obtained from the main project owners should be understood and the communication should be done properly by updating .csv files. After updating the csv, it was launched to run the Qualcomm proprietary connection tool. The output directory is checked for offline inputs, errors, unloaded exits and multi-drilled holes. The number of unintended openings and bindings is reduced by getting the contact updated by the main owners. Design verification is done with the help of System Verilog test benches, in accordance with the Universal Verification Methodology (UVM) standard. This enables faster development and reuse of verification facilities and IP verification (VIP). The testbench has a number of different components each performing a specific function to validate the design under test. Testbench is also customized and customized to ensure blocks at SoC level, which requires the installation of these components in each IP context separately.

SoC integration is performed, and the number of errors and offline inputs is reduced to zero. Once the reports are clean and accurate, a high-quality module is submitted for testing across the clock background and compiled using a sandbox tool. The number of ports frequently operated and reduced to a minimum. Various testbenches were developed to test the performance of existing block blocks in the SoC block. The results show that blocks more than 70% of test cases in the first phase and in subsequent stages, design errors are removed to first reach 85% and finally 100% success rate.

6.2 Future Scope

SoC usually has more than 500 reinforced cases. However, due to time constraints all cores cannot be assembled and verified by one person. Work becomes hard and tedious. So a few large blocks like Ethernet and security blocks are considered and integrated and validated.

The work can be expanded by moving on to other different blocks in the SoC and applying the discussed concepts of integration and validation across different projects.

6.3 Learning Outcomes of the Project

- Understand the basics of RAS and SoC, focusing on the design and importance of RAS.
- Read various meeting agreements. Error detection and debugging using protocol signature waveform.
- Understand the functions of the various IP cores present in the RAS. Gain insight into the various ports, IP Cores port operations.
- Structural and flow model design of the universal verification system.

- Build test benches using UVM to test the performance of various IP cores at the RAS level.

REFERENCES

1. Hyper-Threading Technology, Intel Technology Journal, Vol. 6 Issue 1, Feb 2002

2. S. Rusu, S. Tam, H. Muljono, D. Ayers, J. Chang, "A DualCore Multi-Threaded SoC*) Processor with 16MB L3 Cache", International Solid State Circuits Conference, pp 102-104, January 2006.

3 S. Rusu et al., “A dual core multi-threaded SoC processor with 16MB L3 cache,” in IEEE ISSCC 2006 Dig. Tech. Papers, pp. 315–324

4. J. Chang et al., “The 65-nm 16-MB on-die L3 cache for a dual core multi-threaded SoC processor,” in Symp. VLSI Circuits 2006 Dig. Tech. Papers, pp. 158–159.

5. K. Zhang et al., “A SRAM design on 65-nm CMOS technology with integrated leakage reduction scheme,” in Symp. VLSI Circuits 2006 Dig. Tech. Papers, pp. 294–295.

6. K. Nii et al., “A 90-nm low power 32 K-byte embedded sram with gate leakage suppression circuit for mobile applications,” in Symp. VLSI Circuits 2003 Dig. Tech. Papers, pp. 247–250

7. A. J. Bhavnagarwala et al., “A pico-joule class, 1 GHz, 32 KByte 64 b DSP SRAM with self reverse bias,” in Symp. VLSI Circuits 2003 Dig. Tech. Papers, pp. 251–252

8. K. Mistry, et aI., "A 45nm Logic Technolog y with High-k+ Metal Gate Transistors, Strained Silicon, 9 Cu Interconnect Layers, 193nm Dry Patterning, and 100% Pb-free Packaging", IEDM Tech. Digest, December 2007

9. S. Rusu, et aI., "A 45nm 8-Core Enterprise SoC" Processor", ISSCC Tech. Digest, February 2009

10. Intel SoC Phi coprocessor. [Online]. Available: http://www.intel.com/content/www/us/en/processors/SoC/SoCphi-detail.html

11. J. Zola, M. Aluru, A. Sarje, and S. Aluru, “Parallel informationtheory-based construction of genome-wide gene regulatory networks,” IEEE Trans. Parallel Distrib. Syst., vol. 21, no. 12, pp. 1721–1733, Dec. 2010

12. M. Aluru, J. Zola, D. Nettleton, and S. Aluru, “Reverse engineering and analysis of large genome-scale gene networks,” Nucleic Acids Res., vol. 41, no. 1, e24, 2013.

13. C. Daub, R. Steuer, J. Selbig, and S. Kloska, “Estimating mutual information using B-spline functions—An improved similarity measure for analysing gene expression data,” BMC Bioinformatics, vol. 5, article 118, 2004.

14. T. Qiuhong, X. Zhang, and J. A. Francis, “Extreme summer weather in northern mid-latitudes linked to a vanishing cryosphere,” Nat. Clim. Change, vol. 4, no. 1, pp. 45–50, 2014

15. W. C. Skamarock et al., “A description of the advanced research WRF version 3,” NCAR, Boulder, CO, USA, Tech. Note TN-4475+STR, 2008

16. J. B. Klemp, W. C. Skamarock, and J. Dudhia, “Conservative splitexplicit time integration methods for the compressible nonhydrostatic equations,” Mon. Weather Rev., vol. 135, no. 8, pp. 2897–2913.

17. G. Ruetsch, M. Fatica, E. Phillips, and N. Juffa, “GPU acceleration of the long-wave rapid radiative transfer model in WRF using CUDA Fortran,” in Proc. Many-Core Reconfigurable Super comput. Conf.,2010, pp. 1–11 [Online]. Available: https://www.spec.orgwww.pgroup. com/lit/articles/nvidia_paper_rrtm.pdf

18. E. Price, J. Mielikainen, M. Huang, B. Huang, H.-L. A. Huang, and T. Lee, “GPU-accelerated longwave radiation scheme of the rapid radiative transfer model for general circulation models (RRTMG),” IEEE J. Sel. Topics Appl. Earth Observ., vol. 7, no. 8, pp. 3660–3667, Aug. 2014

19. J. Michalakes, “Optimizing weather models for Intel SoC Phi,” Intel Theater Presentation SC, Nov. 20, 2013

20. P. Smolarkiewicz, “Multidimensional positive definite advection transport algorithm: An overview,” Int. J. Numerical Methods Fluids, vol. 50, pp. 1123–1144, 2006

21. L. Szustak, K. Rojek, and P. Gepner, “Using Intel SoC Phi coprocessor to accelerate computations in MPDATA algorithm,” in Parallel Processing and Applied Mathematics, vol. 8384, R,. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Wasniewski, Eds. Berlin, Germany: Springer, pp. 582–592, 2014.

22. L. Szustak, K. Rojek, T. Olas, L. Kuczynski, K. Halbiniak, and P. Gepner, “Adaptation of MPDATA heterogeneous stencil computation to Intel SoC Phi coprocessor,” Scientific Programming, 2015. [Online].

23.Available: http://dx.doi.org/10.1155/2015/642705

24. L. Szustak, K. Rojek, R. Wyrzykowski, and P. Gepner, “Toward efficient distribution of MPDATA stencil computation on Intel MIC architecture,” in Proc. 1st Int. Workshop High-Perform. Stencil Comput., 2014, pp. 51–56

25. A. Lastovetsky and J. Twamley, “Towards a realistic performance model for networks of heterogeneous computers,” in High Performance Computational Science and Engineering. Berlin: Germany: Springer, 2005, pp. 39–57.

26 A. Lastovetsky and R. Reddy, “Data partitioning with a functional performance model of heterogeneous processors,” Int. J. High Perform. Comput. Appl., vol. 21, no. 1, pp. 76–90, 2007

27. M. Fatica, “Accelerating linpack with CUDA on heterogenous clusters,” in Proc. 2nd Workshop General Purpose Process. Graph. Process. Units, 2009, pp. 46–51.

28. C. Yang, et al., “Adaptive optimization for petascale heterogeneous CPU/GPU computing,” in Proc. IEEE Int. Conf. Cluster Comput., 2010, pp. 19–28.

29. Y. Ogata, T. Endo, N. Maruyama, and S. Matsuoka, “An efficient, model-based CPU-GPU heterogeneous FFT library,” in Proc. IEEEInt. Symp. Parallel Distrib. Process., 2008, pp. 1–10

30. K. Rojek and R. Wyrzykowski, "Parallelization of 3D MPDATA algorithm using many graphics processors," in Proc. 13th Int. Conf. Parallel Comput. Technol., 2015, pp. 445–457.

31. M. D. Linderman, J. D. Collins, H. Wang, and T. H. Meng, "Merge: A programming model for heterogeneous multi-core systems," ACM SIGPLAN Notices, vol. 43, pp. 287–296, 2008.

32. C. Augonnet, S. Thibault, and R. Namyst, "Automatic calibration of performance models on heterogeneous multicore architectures," in Proc. Parallel Process. Workshops, 2009, pp. 56–65

33. G. Quintana-Ortı, F. D. Igual, E. S. Quintana-Ortı, and R. A. van de Geijn, "Solving dense linear systems on platforms with multiple hardware accelerators," ACM SIGPLAN Notices, vol. 44, pp. 121–130, 2009

34. K. Shvachko, H. Kuang, S. Radia, and R. Chansler, ''The Hadoop distributed file system,'' in Proc. MSST, Incline Village, NV, USA, 2010, pp. 1–10.

35. M. Zaharia, M. Chowdhury, M. J. Franklin, and S. Shenker, and I. Stoica, ''Spark: Cluster computing with working sets,'' in Proc. Hot Cloud, Boston, MA, USA, 2010, pp. 1-10

36. X. Liao, L. Xiao, C. Yang, and Y. Lu, ''MilkyWay-2 supercomputer: System and application,'' Frontiers Comput. Sci., vol. 8, no. 3, pp. 345–356,2014

37. (2017). Trinity: Advanced Technology System. [Online]. Available: https:// www.lanl.gov/projects/trinity/index.php

38. D. Nagaraj and C. Gianos, "Intel SoC processor D: The first SoC processor optimized for dense solutions," in Proc. IEEE Hot Chips 27 Symp. (HCS), Aug. 2015, pp. 1–22.

39. E. A. Burton et al., "FIVR—Fully integrated voltage regulators on 4th generation Intel Core SoCs," in Proc. IEEE Appl. Power Electron. Conf, Mar. 2014, pp. 432–439.

40. J. Willcock, "Graph 500 Benchmark," Salt Lake City, Utah,Nov. 2012, presentation at Super Computing 2012.

41 M. Paredes, G. Riley, and M. Lujan, "Vectorization of hybrid breadth first search on the Intel SoC Phi," in Proc. ACM Int. Conf. Comput. Frontiers, 2017, pp. 127–135, doi: https://doi.org/ 10.1145/3075564.3075573.

42. E. A. Golovina, A. S. Semenov, and A. S. Frolov, "Performance evaluation of breadth-first search on Intel SoC Phi," Vychislitel'nye Metody i Programmirovanie, vol. 15, no. 1, pp. 49–58, 2014.

43. G. Tao, L. Yutong, and S. Guang, "Using mic to accelerate a typical data-intensive application: The breadth-first search," in Proc. IEEE 27th Int. Parallel Distrib. Process. Symp. Workshops PhD Forum, 2013, pp. 1117–1125

44. Intel, Optimization and Performance Tuning for Intel SoC Phi Coprocessors, Part 2: Understanding and Using Hardware Events, 2012. [Online]. Available: https://software.intel.com/en-us/articles/optimization-and-performance-tuning-for-intel-SoC-phicoprocessors-part-2-understanding

45. A. E. Sariyuce, E. Saule, K. Kaya, and U. V. Catalyurek, “Hardware/ software vectorization for closeness centrality on multi-/many-core architectures,” Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops, 2014, pp. 1386–1395.

46. K. Bergman et al., “Exascale computing study: Technology challenges in achieving exascale systems,” Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Tech. Rep, vol. 15, 2008.

47. J. V. Ferreira Lima, I. Raı¨s, L. Lefevre, and T. Gautier, “Performance and energy analysis of OpenMP runtime systems with dense linear algebra algorithms,” Int. J. High Perform. Comput. Appl., vol. 33, pp. 431–443, 2018

48. R. Montella et al., “Accelerating Linux and Android applications on low-power devices through remote GPGPU offloading,” Concurrency Comp, Practice Experience, vol. 29, no. 24, 2017, Art. no. e428

49. “Intel SoC Processor Scalable Family Specification,” Feb. 2018. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/specification-updates/SoC-scalablespec-update.pdf, Intel

50. P. Smolarkiewicz, “Multidimensional positive definite advection transport algorithm: An overview,” Int. J. Numer. Meth. Fluids, vol. 50, no. 10, pp. 1123–1144, 2006

51. L. Szustak and P. Bratek, “Performance portable parallel programming of heterogeneous stencils across shared-memory platforms with modern Intel processors,” Int. J. High Perform. Comput. Appl.,vol. 33, no. 3, pp. 507–526, 2019.

52. K. Rojek, A. Ilic, R. Wyrzykowski, and L. Sousa, “Energy-aware mechanism for stencil-based MPDATA algorithm with constraints,” Concurrency Comput, Practice Experience, vol. 29, no. 8,2017, Art. no. e4016

53. J. Jeffers, J. Reinders, and A. Sodani, Intel SoC Phi Processor High Performance Programming: Knights Landing Edition. San Mateo, CA,USA: Morgan Kaufmann, 2016