Debug Utility for Validation of OASIS Parser

Mushtaq Ahmed, Kiran V, Subrata Nandy

Abstract: *In EDA industry, GDSII format is the IC industry de facto standard for IC layout data exchange. The designs developed may require up to 1 Billion Byte (1TByte) of disk data. This huge amount of Big Data not only slows down the runtimes from design to physical verification but also increases the time to get a design to market. On the other hand, OASIS stream format which is replacement to GDSII is relatively new and is emerging in the industry. The OASIS stream format significantly reduces the data and thereby the tool is much likely to run faster. There hasn't been significant development in enhancing the robustness of its usage due to lack of in-house test cases. This paper presents an approach to develop a debug utility for OASIS parser validation to increase its robustness. The debug utility is implemented using a Singleton design pattern. The utility essentially enables us to compare the data associated with both the stream formats and highlights the differences. The effective memory utilization of the proposed design is zero since all the structure are dynamically created and destroyed after its use. Iterative and unit testing were performed on the utility and the proposed design was tested with real time test cases to verify its robustness.*

Keywords: *EDA -Engineering Design Automation, GDSII – Graphic Data System II, OASIS- Open Artwork System Interchange Standard, Parser, Debug Utility.*

I. INTRODUCTION

Open Artwork System Interchange Standard (OASIS) is a specification for hierarchical integrated circuit mask layout data format for interchange between EDA (Electronic Design Automation) software, IC mask writing tools and mask inspection tools [1-2]. This stream format when used for EDA exchange is more prone to errors due to its complex nature and often at times leads to misinterpretation of data. Therefore, parsing of these large and complex files has always been a challenge in commercial EDA tools [3-4]. A parser is a computer program which takes input within the kind of a sequence of tokens or program directions associated and builds an information structure in the form tree or an abstract syntax tree [5]. This paper focuses on the validation of OASIS Parser by developing a Debug Utility. A debugging utility eases the task of identifying and eliminating bugs in the software [6-7]. The utility is developed with the assumption that data associated with GDSII stream is always golden. The utility is invoked separately for both the stream format for comparison. This identifies any kind of loss of data or misinterpretation of data with OASIS stream and thereby increases the robustness of the tool. The paper also introduces the support of compressed Oasis stream format along with the utility. Fig. 1 compares the ratio of the OASIS stream format with the existing GDSII format. It is evident that the compressed OASIS stream utilizes lesser memory thereby addressing the big data problem. The utility is invoked using a keyword in configuration file of the design.

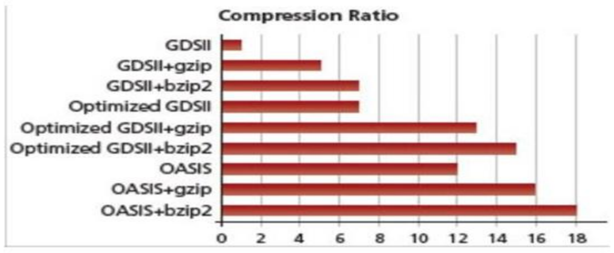


Fig. 1.Data Compression ratios of various stream formats.[12]

II. SYSTEM DEVELOPMENT

A. Methodology Adopted

The current EDA exchange format i.e. GDSII format and OASIS stream format were studied to identify their key differences [8-11]. The block diagram shown in Fig. 2, shows the general methodology adopted by the proposed work.

- It begins with reading the files of both formats and parsing it. Then individually the data is populated for LayoutDB. Finally, the debug utility is invoked to identify differences.
- A C++ program is developed by following Singleton Class Design Pattern as shown in Fig. 3.
- This design pattern enables to create only one instance of the class throughout the program. As the utility is invoked only once, the approach is hence adopted.
- A block keyword is developed which is used in the configuration file of the design. The default value of the keyword is set to 1 which dumps all the data associated with the file and exits from the tool.
- On setting the keyword to 0, the data is dumped, and the tool continues to execute.

B. Implementation Details of Debug Utility

The block diagram, shown in the Fig. 4, depicts the major components of a cell in a semiconductor design. Each design has a top cell which may also contain several instances of other cells.

Revised Manuscript Received on June 15, 2020.
* Correspondence Author

Mushtaq Ahmed*, Final Year Student, Department of Electronics and Communication Engineering, RV College of Engineering, Bengaluru, India. Email: mushtaqas.ec16@rvce.edu.in
Dr. Kiran V, Associate Professor, Department of Electronics and Communication Engineering, RV College of Engineering, Bengaluru, India. Email: kiranv@rvce.edu.in
Subrata Nandy, Director, Software Development, Ansys Inc, Bengaluru, India. Email: Subrat.Nandy@ansys.com

The cell also contains geometries which are created during chip design. These geometries are rectangles, polygons and paths. To specify the nets and certain other features, the cell also has texts.

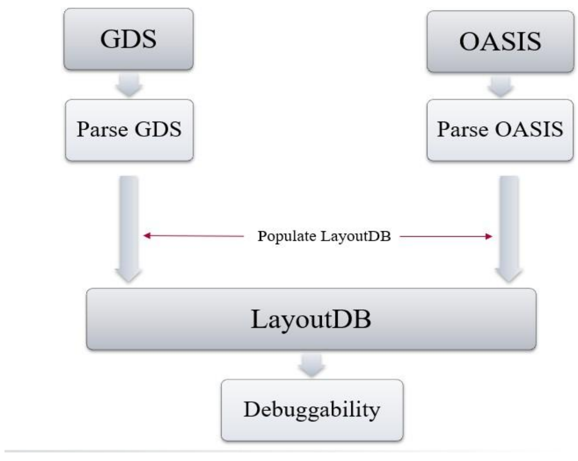
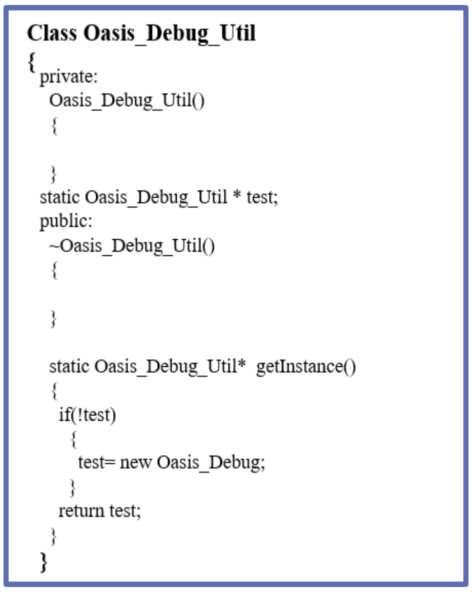


Fig. 2.Block Diagram of Proposed Methodology



```
Class Oasis_Debug_Util
{
 private:
   Oasis_Debug_Util()
   {

   }
 static Oasis_Debug_Util * test;
 public:
   ~Oasis_Debug_Util()
   {

   }

   static Oasis_Debug_Util*  getInstance()
   {
     if(!test)
       {
         test= new Oasis_Debug;
       }
     return test;
   }
}
```

Fig. 3.Singleton Design Pattern

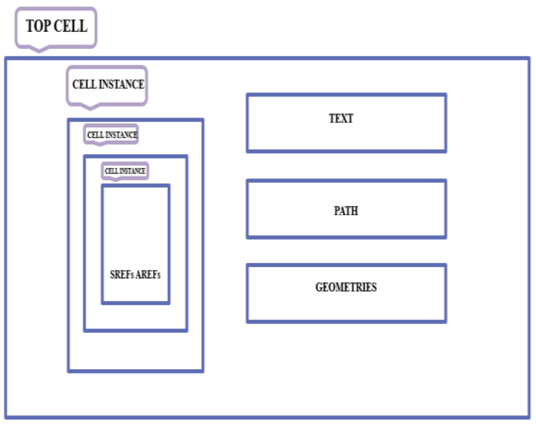


Fig. 4.Contents of a Cell in a Chip Design

In order to validate the parser, the entire data associated with the cell needs to be dumped in a unique format for comparison. Fig. 5 describes the methodology adopted to implement the unique format of dumping the data. The list of cells is obtained from the cell list.

Order the cells within the cell list based on their ASCII values. For each of cell within the cell list sort each of its contents. If the cell contains the texts, then sort the texts based on their ASCII value. The cell also contains geometries which are Rectangles, Polygons and Paths. In order to dump the data in a common format, the entire geometries associated with the cells are converted into Polygons. The Polygons are described using their coordinates. Each of the Polygons are then sorted based on their coordinates and then are dumped into file for their comparison.

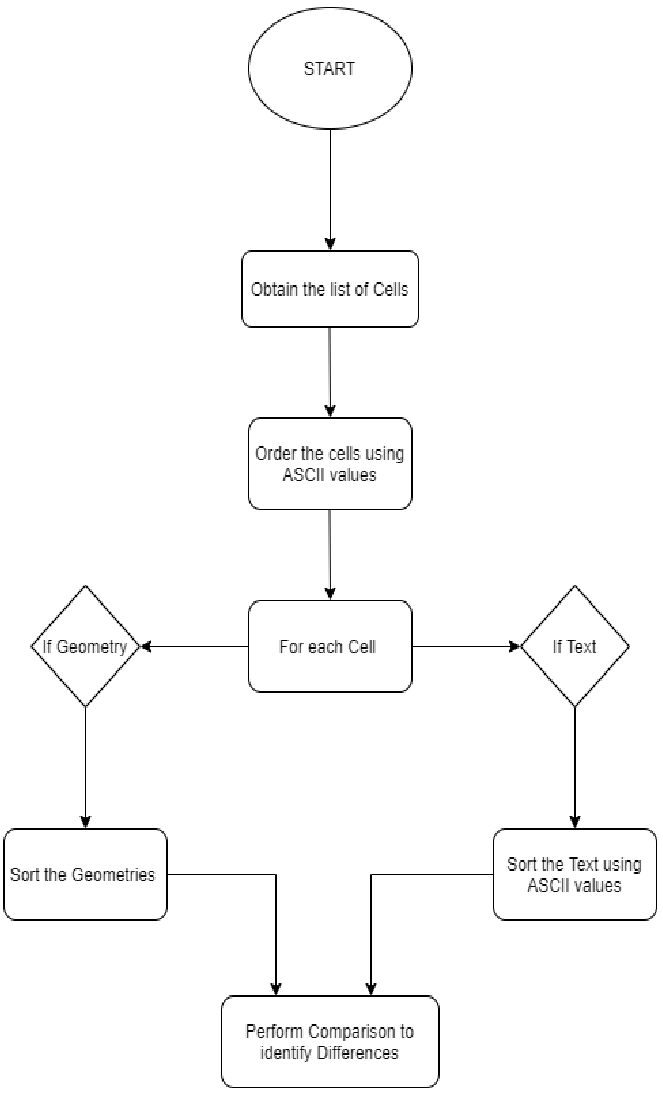

START
Obtain the list of Cells
Order the cells using ASCII values
For each Cell
If Geometry
If Text
Sort the Geometries
Sort the Text using ASCII values
Perform Comparison to identify Differences

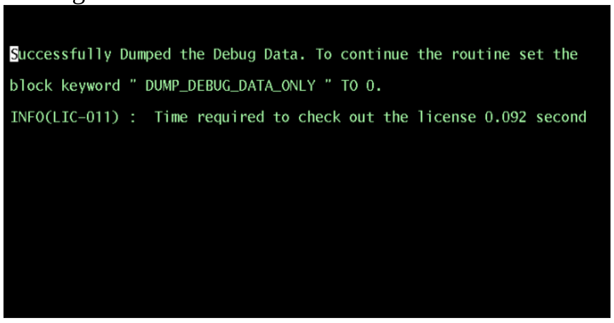
Fig. 5.Block Diagram of Implementation of Debug Utility

The data from both the stream formats are dumped into two separate files. The command "tkdiff" is invoked from the Linux terminal command line which highlights the differences between the two files.

III. RESULTS AND DISCUSSIONS

The C++ code developed using Singleton Design Pattern is compiled on g++ 4.8.2 compiler. The Debug Utility is called using the keyword "Debug_Physical_Layout_Data "in the configuration file of the chip design. The tool is run on the design to obtain the following results:

1. Using the default value of keyword i.e. 1, the Debug Utility dumps the entire cell data associated with the design in a file names "Debug_Data.txt" and exits from its operation. A message is displayed in the Log File of the tool as described in the Fig. 6.
2. When the keyword is set to 0, the Debug Utility dumps the entire cell data associated with the design in a file names "Debug_Data.txt" and the tool continues its operation. The successful run of the tool is displayed in the Log File.



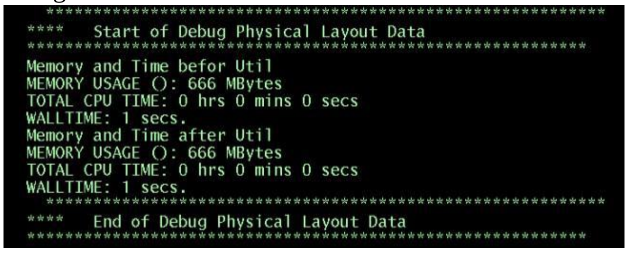
```
Successfully Dumped the Debug Data. To continue the routine set the

block keyword " DUMP_DEBUG_DATA_ONLY " TO 0.

INFO(LIC-011) :  Time required to check out the license 0.092 second
```

Fig. 6. Log File message with default value of keyword i.e. 1

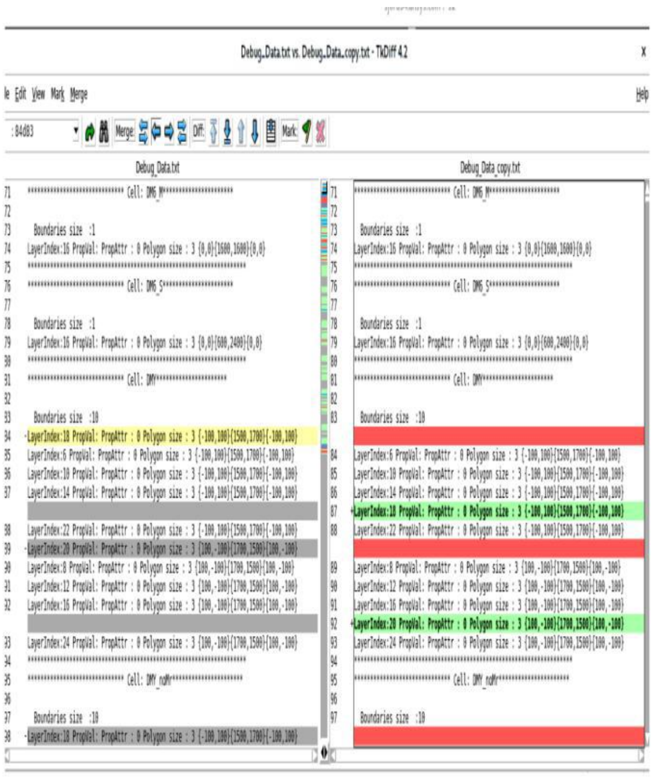
3. The effective memory utilization of the utility along with the CPU wall time is highlighted in the Log File as shown in Fig. 7.



```
 *****************************************************************
 ****    Start of Debug Physical Layout Data
 *****************************************************************
 Memory and Time befor Util
 MEMORY USAGE (): 666 MBytes
 TOTAL CPU TIME: 0 hrs 0 mins 0 secs
 WALLTIME: 1 secs.
 Memory and Time after Util
 MEMORY USAGE (): 666 MBytes
 TOTAL CPU TIME: 0 hrs 0 mins 0 secs
 WALLTIME: 1 secs.
 *****************************************************************
 ****    End of Debug Physical Layout Data
 *****************************************************************
```

Fig. 7. Memory and CPU Utilization

4. The above steps are performed individually for both the GDSII stream and OASIS stream format on the same Chip Design. The command "tkdiff" is called from the Linux Terminal which invokes a GUI to highlight the differences between the two streams as depicted in Fig. 8.
5. An enhancement to support the compressed file format i.e. gzip is added for the OASIS stream. This enhancement enables to use the compressed version of the file which in- turn reduces the data associated with the EDA exchange file, thereby further reducing the memory associated with the EDA exchange file and in-turn solving the Big Data problem. A message is displayed in the terminal.



Debug_Data.txt

```
**************************** Cell: DM6_M********************

  Boundaries size  :1
LayerIndex:16 PropVal: PropAttr : 0 Polygon size : 3 {0,0}{1600,1600}{0,0}
**************************************************************
**************************** Cell: DM6_S********************

  Boundaries size  :1
LayerIndex:16 PropVal: PropAttr : 0 Polygon size : 3 {0,0}{600,2400}{0,0}
**************************************************************
**************************** Cell: DMY********************

  Boundaries size  :10
-LayerIndex:18 PropVal: PropAttr : 0 Polygon size : 3 {-100,100}{1500,1700}{-100,100}
LayerIndex:6 PropVal: PropAttr : 0 Polygon size : 3 {-100,100}{1500,1700}{-100,100}
LayerIndex:10 PropVal: PropAttr : 0 Polygon size : 3 {-100,100}{1500,1700}{-100,100}
LayerIndex:14 PropVal: PropAttr : 0 Polygon size : 3 {-100,100}{1500,1700}{-100,100}

LayerIndex:22 PropVal: PropAttr : 0 Polygon size : 3 {-100,100}{1500,1700}{-100,100}
-LayerIndex:20 PropVal: PropAttr : 0 Polygon size : 3 {100,-100}{1700,1500}{100,-100}
LayerIndex:8 PropVal: PropAttr : 0 Polygon size : 3 {100,-100}{1700,1500}{100,-100}
LayerIndex:12 PropVal: PropAttr : 0 Polygon size : 3 {100,-100}{1700,1500}{100,-100}
LayerIndex:16 PropVal: PropAttr : 0 Polygon size : 3 {100,-100}{1700,1500}{100,-100}

LayerIndex:24 PropVal: PropAttr : 0 Polygon size : 3 {100,-100}{1700,1500}{100,-100}
**************************************************************
**************************** Cell: DMY_noMr********************

  Boundaries size  :10
-LayerIndex:18 PropVal: PropAttr : 0 Polygon size : 3 {-100,100}{1500,1700}{-100,100}
```

Debug_Data_copy.txt

```
**************************** Cell: DM6_M********************

  Boundaries size  :1
LayerIndex:16 PropVal: PropAttr : 0 Polygon size : 3 {0,0}{1600,1600}{0,0}
**************************************************************
**************************** Cell: DM6_S********************

  Boundaries size  :1
LayerIndex:16 PropVal: PropAttr : 0 Polygon size : 3 {0,0}{600,2400}{0,0}
**************************************************************
**************************** Cell: DMY********************

  Boundaries size  :10

LayerIndex:6 PropVal: PropAttr : 0 Polygon size : 3 {-100,100}{1500,1700}{-100,100}
LayerIndex:10 PropVal: PropAttr : 0 Polygon size : 3 {-100,100}{1500,1700}{-100,100}
LayerIndex:14 PropVal: PropAttr : 0 Polygon size : 3 {-100,100}{1500,1700}{-100,100}
+LayerIndex:18 PropVal: PropAttr : 0 Polygon size : 3 {-100,100}{1500,1700}{-100,100}
LayerIndex:22 PropVal: PropAttr : 0 Polygon size : 3 {-100,100}{1500,1700}{-100,100}

LayerIndex:8 PropVal: PropAttr : 0 Polygon size : 3 {100,-100}{1700,1500}{100,-100}
LayerIndex:12 PropVal: PropAttr : 0 Polygon size : 3 {100,-100}{1700,1500}{100,-100}
LayerIndex:16 PropVal: PropAttr : 0 Polygon size : 3 {100,-100}{1700,1500}{100,-100}
+LayerIndex:20 PropVal: PropAttr : 0 Polygon size : 3 {100,-100}{1700,1500}{100,-100}
LayerIndex:24 PropVal: PropAttr : 0 Polygon size : 3 {100,-100}{1700,1500}{100,-100}
**************************************************************
**************************** Cell: DMY_noMr********************

  Boundaries size  :10

```

Fig. 8."Tkdiff" highlighting the differences

IV. CONCLUSION

The Debug Utility for OASIS Parser Validation is successfully developed using a singleton design pattern. The program is developed using C++ computer language. The proposed work is an efficient implementation to increase the robustness of the OASIS parser. The Debug Utility developed can be used with both the stream formats. The subsequent conclusions are arrived at based on the results:

The design development begins with parsing the files of both the stream formats and populating the LayoutDB. A unique format is developed to dump the data associated with both the file formats. After the data is dumped, the differences are identified which highlights any errors or misinterpretation in OASIS stream. A block keyword is developed to call the utility from the configuration file. The default value of the keyword is 1. Further, the effective memory and CPU utilization is highlighted to the user. An enhancement to support the compressed version of the OASIS file format is developed which addresses the issue of memory and Big Data. The messages associated with each feature is displayed in the Log File for the user.

FUTURE SCOPE

The proposed work is an efficient way of developing a debug utility to increase the robustness of OASIS Parser. However, the debug utility identifies the differences by invoking the built- in command "tkdiff" available in the Linux OS. The debugger then needs to manually locate the differences in the GUI to have a clear understanding of the issue.

This manual task can further be extended to incorporate automation. The differences can be directly loaded up in the GUI which will then avoid the manual task. This would also to a greater extent reduce the human error incurred during debugging.

REFERENCES

1. New Standards Specification for Open Artwork System Interchange Standard, December11,2002. [Online]. Available: http://www.semi.org.
2. Y. Chen, A. B. Kahng, G. Robins, A. Zelikovsky and Y. Zheng, "Evaluation of the new OASIS format for layout fill compression," *Proceedings of the 2004 11th IEEE International Conference on Electronics, Circuits and Systems, 2004. ICECS 2004.*, Tel Aviv, Israel, 2004, pp. 377-382.
3. Jin Lin and ZhaoTong, "EDA technology and its implementation in modern electronic technology," *2011 International Conference on Business Management and Electronic Information*, Guangzhou, 2011, pp. 816-819.
4. P. Shanbhag, C. Gopalakrishnan and S. Ghosh, "Methodology for Efficient Multi- threading of Parsers in EDA Tools," *2012 IEEE Computer Society Annual Symposium on VLSI*, Amherst, MA, 2012, pp. 291-296.
5. P. Shanbhag, C. Gopalakrishnan and S. Ghosh, "A Case Study in Developing an Efficient Multi-threaded EDA Parser: Synopsys SDF Parser," *2012 IEEE Computer Society Annual Symposium on VLSI*, Amherst, MA, 2012, pp. 297-301.
6. M. Böhme, E. O. Soremekun, S. Chattopadhyay, E. J. Ugherughe and A. Zeller, "How Developers Debug Software — The DBGBENCH Dataset," *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, Buenos Aires, 2017, pp. 244-246.
7. F. Petrillo, Z. Soh, F. Khomh, M. Pimenta, C. Freitas and Y. Guéhéneuc, "Towards Understanding Interactive Debugging," *2016 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, Vienna, 2016, pp. 152-163.
8. Pinhong Chen, D. A. Kirkpatrick and K. Keutzer, "Scripting for EDA tools: a case study," *Proceedings of the IEEE 2001. 2nd International Symposium on Quality Electronic Design*, San Jose, CA, USA, 2001, pp. 87-93.
9. Bowu Yan, Xicai Cheng, Fei Yang and Li Yao, "Research on EDA technology and its related issues," *2010 International Conference On Computer Design and Applications*, Qinhuangdao, 2010, pp. V4-26-V4-29.
10. A. Singla, B. Lippmann and H. Graeb, "Verification of Physical Chip Layouts Using GDSII Design Data," *2019 IEEE 4th International Verification and Security Workshop (IVSW)*, Rhodes Island, Greece, 2019, pp. 55-60.
11. Calma 1987. [Online]. Available: http://bitsavers.informatik.uni-stuttgart.de/pdf/calma/GDS_II_Stream_Format_Manual_6.0_Feb87.pdf.
12. Going from GDSII to OASIS. [Online]. Available: http://www.eetimes.com/going-from-gdsii-to-oasis/

AUTHORS PROFILE

Mushtaq Ahmed is a final year student currently pursuing his Bachelor of Engineering Degree from RV College of Engineering. He is majoring in Electronics and Communication and will graduate in the year 2020. He is also interning at Ansys Inc, a tech-giant in multi-physics simulation software. His keen interest revolves around programming and developing solutions for modern day problems. He has worked on projects related to memory design, chip design in the VLSI domain. He has also worked on computer science projects involving object-oriented design and has developed it using languages such as C++ and python.

Dr. Kiran V received his BE in Electronics and Communications from Bangalore University, his MTech in Digital Electronics and Communication from Visvesvaraya Technological university and PhD from Visvesvaraya Technological university. He is a life member of ISTE. He has more than 17 years of experience in teaching. He has published more than 35 papers in national, international conferences and Journals. He has filed patent. He has guided many projects. His research interests are communication, networking and signal processing.

Subrata Nandy is currently working as R&D Director, in Ansys Inc., a market leading multi-national company building multi-physics simulation software. Subrata graduated with Bachelor of Engineering in Information Technology, in the year 2004, from Jadavpur University, West Bengal, India. Subrata has 16 years of experience in building Electronic Design Automation (EDA) software, with interest and expertise in logic synthesis from Hardware Description Languages (Verilog, System Verilog, VHDL), Semiconductor Layout physical modeling, Geometric algorithms and physical design database creation and optimization.