2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Netlist Enabled Emulation Platform for Accelerated Gate Level Verification

Samhith Kumar Pottem, Vasudeva Reddy Ambati, Rahul S S, Sarang Kalbande, Garima Srivastava, Hyundon Kim
Samsung Semiconductor India Research (SSIR)
Bagmane Goldstone Building, Mahadevapura
Bangalore - 560037

***Abstract*- Conventional Simulation environment follows Register Transfer Level (RTL) based design approach for the verification of SoC chipsets as it offers more granularity and profound insight of the design. But this type of verification approach is vulnerable in identifying the delays, switching activity for power estimation. Hence, in order to tackle these kind of potential issues, Gate Level Simulation is employed to evaluate certain Intellectual Property (IP) features of a System on Chip (SoC). However, it is tedious and time-consuming as the verification of a single scenario takes weeks to complete in Gate Level Simulation (GLS) environment. Gate Level Emulation (GLE) approach is efficient and effective as emulator runs on the actual hardware, driver clock frequency is in the range of MHz as opposed to the simulation which is in Hz range, thus making the execution and validation time faster. The verification results obtained is more accurate and reliable as the emulator is close to the actual silicon. In this paper, we will delineate the techniques of integrating the Netlist Design files in emulation environment, validate a set of use cases, and compare the results with simulation based verification. The verification time is significantly improved by an order of 70-90X in Gate Level Emulation. The emulator on which these scenarios are executed is Synopsys ZeBu [5].**

***Keywords*— Netlist, Gate Level Emulation, Complex System on Chip, SoC SCANDUMP, Gate Level Simulation, Field Programmable Gate Arrays, Register Transfer Level, Design Under Test**

I. INTRODUCTION

As the complexity of design is increasing rapidly across various SoCs, the time to market of the chip is also getting delayed, and the validation of SoC features after the silicon tape-out takes a lot of time. Even if some bugs are identified at this stage, the re-spin of the SoC results in a huge time loss and would incur a significant financial loss to the Company. Therefore, Pre-Silicon Verification is crucial as it allows for the identification of defects and their resolution prior to chip production. Simulation and Emulation environments are widely used in Pre-Silicon verification to verify the SoC functionality before chip fabrication.

Verification is usually performed in different levels of design abstraction. The Register Transfer Logic (RTL) based verification represents the behavioural aspects and functionality check of an initial design. As shown in the Fig. 1, it takes less time to execute the test scenarios and provide the design with better understanding and debug support. However, the coverage is limited and this level of abstraction affect the accuracy of the results obtained as it omits some intricacies of the underlying circuit and non-idealities, physical effects present in real circuits are not taken into consideration. Another approach is Transistor Level Verification in which most of the logics are designed at gate level or higher, using well-tested standard cells. However, custom logic arrays are designed at the transistor level rather than the gate level which delivers more precise output of a design. Though this type of verification approach offers high accuracy and detailed analysis, but requires significant effort, expertise and computational resources. It also adds complexity to the verification process. Hence, the Gate Level Verification approach is used to overcome the trade-off between the aforementioned levels of design abstraction. As the circuit connections of an electronic chip are in the form of gates, this verification is henceforth called Gate Level Verification which represents the logical aspects of design and comes after the synthesis stage. It plays a crucial role in validating the correctness, timing behaviour and physical compatibility of the design. It also helps uncover the low-level bugs, optimize the design and ensure the proper functioning of complex SoCs.

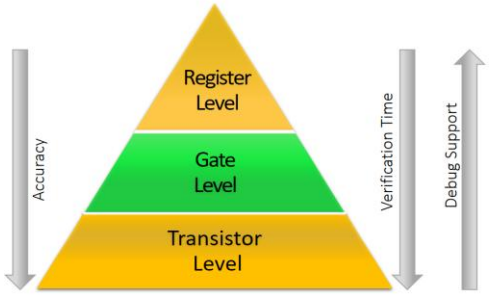

Figure 1: Different Abstraction Levels of Verification

In this guide, we propose a methodology to integrate the full SoC netlist on Emulation environment for complex SoCs having billions of gates. We will demonstrate a set of application scenarios executing on GLE and compare their execution time between GLE and GLS environment.

II. GATE LEVEL NETLIST

A network or a net in short form is a grouping of two or more connected elements. A netlist in an electronic design describes individual logic gates and their interconnections. It specifies the gates used in the circuit, their input and output connections, and the logical functions performed by each gate. A Gate Level Netlist is a representation of a design at the lowest level of abstraction.

The connections between each component in the design are listed in a machine-readable file called a netlist. The netlist representation of a circuit, which is typically obtained using logic synthesis, is referred to as being at the gate level. Thus, RTL emulation is pre-synthesis, whereas Gate Level Emulation is post-synthesis. A complete connection list of gates and IP models with full functional and timing behaviour can be found in the netlist view and a typical Netlist representation of a circuit is as shown in the Fig. 2.

```
1
2
3
4   //Full Adder
5   module full_adder( A, B, Cin, S, Cout);
6
7    input wire A, B, Cin;
8    output reg S, Cout;
9
10   always @(A or B or Cin)
11    begin
12     S = A ^ B ^ Cin;
13     Cout = A&B | ((A^B) & Cin);
14    end
15  endmodule
16
17
18
19
```

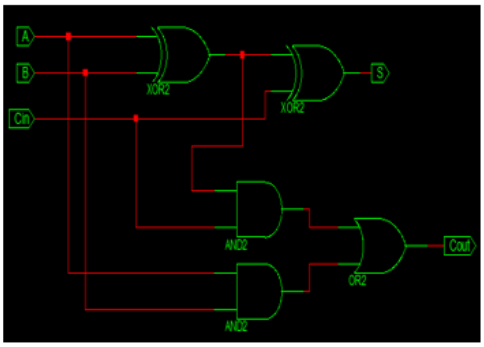

Figure 2: Typical Netlist Representation of a Circuit

III. RELATED WORK

Kriti Sundar Das, Padmini Prakash, Abhimanyusinh Zala [1] from Intel India Technology Pvt. Ltd., Bangalore, India have proposed a vector-mode based GLE for Design-for-Testability (DFT) like Memory Built-In Self Test (MBIST) and Automatic Test Pattern Generator (ATPG) pattern sets in 2021 IEEE International Test Conference India (ITC India). But the authors have confined and limited to validate the DFT features only that too on smaller SoCs with a gate count of around 50M.

IV. GLE IMPLEMENTATION

A. Implementation Pre-requisites

1. Standard Cell Libraries:

Standard Cell Libraries are represented as collections of predefined cells with their corresponding connectivity information that are used to design complex circuits. These elements typically include Cell instantiations and of different cell types such as logic gates, flip-flops and latches. Some Standard Cell

Library may also contain specialized cells like multiplexers, decoders, adders and more. These libraries must be passed along with the design netlist files failing which the compilation would fail in the front-end parsing stage.

2. Memory Models:

Gate Level memory models describe the behavior and functionality of memory elements using a combination of logic gates and their interconnections. These models are usually optimized for specific technology node. The technology node affects the characteristics and properties of memory models including their performance, power consumption and reliability. Hence, the memory models are to be selected based on the target technology node to ensure it meets the performance and power requirements.

Also, the memory models have to be compatible and synthesizable with an emulation compiler.

3. Memory and Physical Layer (PHY) I/O Models:

RTL Memory and PHY I/O Models provide a higher level description of the behavior and data flow across memory elements, interfaces (Double Data Rate Physical Layer DDR PHY) while Gate Level Memory and PHY I/O Models provide a low level representation capturing the actual logic gates and their characteristics. As these models are different from RTL to Gate Level, Netlist compatibility I/O Models must be considered for our compilation.

Actual cell delays are to be maintained in the Netlist I/O Models for proper read and write operations. These models must be updated with synthesizable logic and compatible to our Emulation environment.

To support Scan Chain testing, these synthesized Netlist models should contain the Scan logics.

4. RTL to Netlist (R2N) Mapping Report:

This report deals with the signal polarity and gives the information on how the signals are connected in RTL and Netlist. It is essential to have this report for the validation of any scenario, failing which leads to the undesired results.

B. Implementation Flow

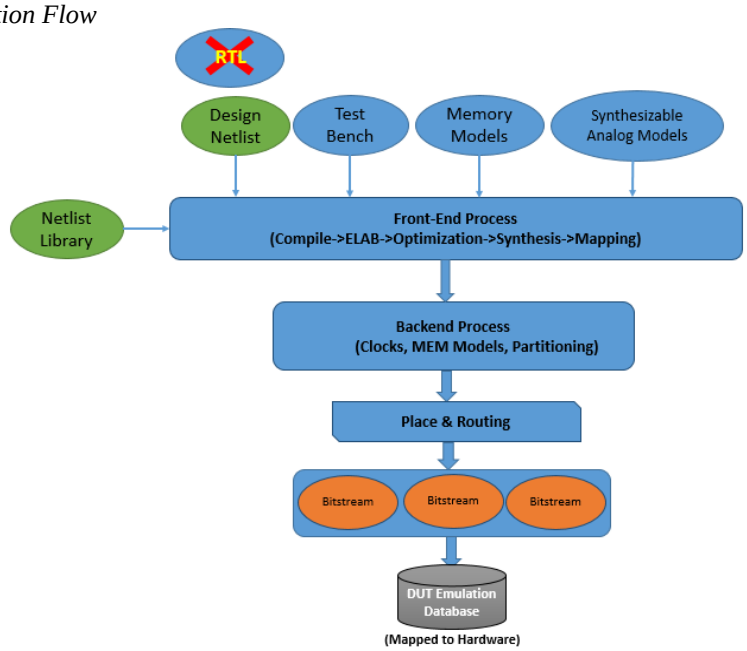

Figure 3. Netlist Database Generation Flow

As shown in the Fig. 3, the GLE based compilation is implemented as follows:

1. Design Preparation:

Design Netlist Files, Synthesizable Memory models, Standard Cell Libraries, Synthesizable Analog Models are the inputs required to generate a Design Under Test (DUT) Database. Memory Models contain synthesizable equivalent of Application Specific Integrated Circuit (ASIC) memory cells. All the Analog

models of the SoC are converted into synthesizable models. In the Test Bench we instantiate DUT and all requisite Transactors in the SoC. There are design primitive library files which contains cell information of GLE Netlist. This must be given as Verilog input to the emulator front end process.

2. Front-End Process:

The goal of the Front-End process is to extract the necessary information from the design source code, perform basic checks, optimizes and mapped to utilise the Field Programmable Gate Array (FPGA) resources effectively.

Design Netlist is elaborated and synthesized to Electronic Design Interchange Format (EDIF), and the relevant gates are optimized during Front-End compilation process. It is then converted to FPGA primitives. The detailed steps that are involved in the Front-End process stage is as shown in the Fig. 4

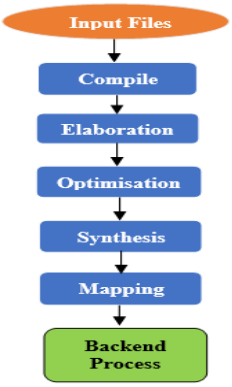

Figure 4: VCS Front End Flow

i. Compile:

Compilation is the process of the converting a Hardware Description Language (HDL) such as Verilog Hardware Description Language (VHDL) into a configuration file that can be used to program in the FPGA devices. This process is crucial as it transforms the high level design description into a specific configuration that can be programmed onto the FPGA.

ii. Elaboration:

The Elaboration stage improves performance even further. It converts the RTL code into modules. This technique maps the logic with General Technology (GTECH), a technology independent cell. It determines whether the design is compatible and if it is not, the compilation flow is terminated.

iii. Optimisation:

Optimisation improves the efficiency and speed of emulation. This process eliminates the unnecessary circuits which are not been used and helps in reducing the gates occupancy as shown in Fig. 5. Thus, this process aims in minimizing the hardware resources utilization whilst maintaining the emulated design accurately.

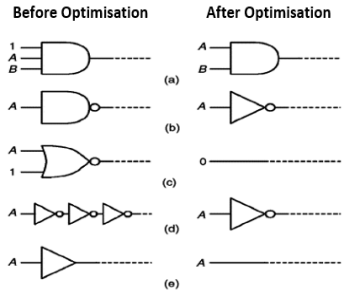

Figure 5: Optimisation Techniques

iv. Synthesis:

In the emulation flow, synthesis refers to the process of converting a high level hardware

description language to lower level representation that can be programmed and fit into a FPGA device. During synthesis the HDL code is analysed and optimised to generate a netlist representation.

v. Mapping:

In this Mapping stage, the HDL code is mapped onto the specific resources available in the target FPGA device. This step determines which logic elements and interconnections in the FPGA will be used to implement the desired functionality.

3. Back End Process:

In Back-End compilation process, the tool inserts all the logic related to clocking and memory mapping. The EDIF Netlist from output of the Front-End process is partitioned to multiple FPGA modules.

i. Clocks:

The emulation compiler identifies and analyses the different clock domains present in the design for handling the clocks in back-end process. To ensure the proper synchronization, the compiler inserts appropriate synchronization mechanisms. It also handles the timing of events or interrupts within the design to ensure that events occur at the expected time relative to the system clocks.

ii. Memory Models:

The compiler analyses the memory architecture of a target system and this process ensures that memory accesses in the design are handled correctly and accurately reflect the behaviour of original hardware.

iii. Partitioning:

The design Netlist file is divided into separate partitions or modules. Each partition represents a distinct unit of functionality or any component of an input file that can be executed independently.

4. Place and Routing:

Place and Route refers to the combined process of placing the logical components of a design onto specific locations on the FPGA chip and routing the interconnections between these components.

i. Placement (Place):

Placement involves determining the physical location of each logical component (Gates, Flip-Flops, memories, etc.) on the FPGA chip. The placement step aims to assign the components in a way that optimizes for factors like signal routing, resource utilization, and performance considerations.

ii. Routing (Route):

After the placement step, the routing process establishes the physical connections (wiring) between the placed components. It determines the routing tracks, which are the paths through which signals travel to connect the inputs and outputs of the various components.

The generated bit stream files are finally loaded onto the emulator hardware at run time. This involves configuring the emulator hardware with the appropriate settings, initializing the design and establishing the communication with any test benches or some external debuggers.

V. GLE ACCELERATION

A. Compile Time Acceleration

Compilation of Netlist based designs takes longer than the conventional RTL design compilation. This is because of the following reasons:

1. The complexity increase in Gate Level designs requires more computational resources and time for synthesis.
2. The finer granularity of Gate Level designs results in a large number of components to analyse, optimize which leads to higher compile time.
3. The Gate Level designs have larger netlist sizes compared to RTL designs. Therefore, processing larger netlists requires more computational resources and time during compilation.

Resolution:

A Multi-core and Multi-Thread support is used which helps in saving the compile time in an emulator. By leveraging the capabilities of multiple cores and threads, emulator can distribute the workload across them, enabling parallel processing and reducing the overall compilation time.

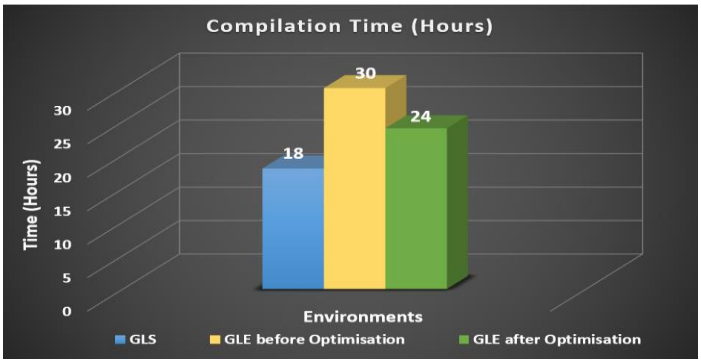

Figure 6: Netlist Compilation Time Comparison

B. Run Time Acceleration

The insertion of Scan Cells in a design can potentially affect the performance of an emulator due to the several listed factors:

1. Insertion of Scan cells adds additional circuitry which increases the complexity and size of the design. This can lead to the slower Emulation speeds.
2. Scan cells can introduce additional flip-flops and longer signal paths that impacts the critical paths in the design. Critical paths are the longest paths in a circuit that determine the maximum operating frequency. Hence, it limits the achievable operating frequency.
3. Insertion of Scan cells can introduce additional clock domains and clocking requirements in the design. Ensuring proper synchronization and timing between these clocks in Emulation can be challenging which results in degradation of Performance.

Resolution:

In order to overcome this and enhance the driver clock frequency, an approach wherein D Flip Flops for a set of test signals in different blocks of SoC are added to be sampled by the fastest design clock. This helps the compiler to improve the compile time and performance as it splits the timing paths.

TABLE I: COMPILE TIME AND DRIVER CLOCK FREQUENCY COMPARISON

Environment	Compile Time (Hours)	Driver Clock Frequency (KHz)
GLS	18	0.45
GLE before Optimisation	30	350
GLE after Optimisation	24	2000

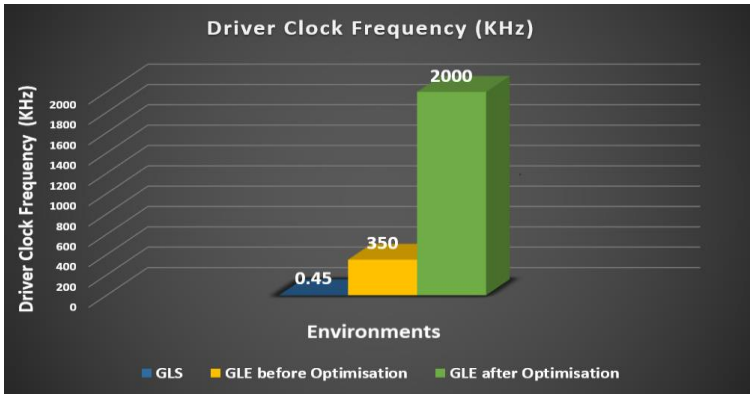

Figure 7: Driver Clock Frequency Comparison

The increase in the driver clock frequency and less compile time after optimization techniques can be observed from the Table I and Fig. 1.

VI. GLE APPLICATION SCENARIOS

1. Logging the Scan Information (SCANDUMP) from any block or entire SoC directly from an external debugger, which is more reliable. (Scenario 1)
2. Switching Activity Interchange Format (SAIF) based Power Estimation. (Scenario 2)
3. To determine whether a design is functional. (Scenario 3)

A. Scenario 1

Another use case is SCANDUMP of a SoC from an external debugger. This mechanism allows us to log the real scan data of an individual block or multiple blocks through debug port in Joint Test Action Group (JTAG) protocol. Each block (Camera, CPU, and USB) has its own DFT circuit and all these blocks are inter-connected with TOP JTAG. This TOP_JTAG has an eternal debug interface through which the scan data can be captured from an external debugger which is essential in case of a SoC hung to analyse and detect the malfunction of a SoC.

To implement the above application scenario, the methodology as shown in the Fig. 8 has to be followed.

B. Scenario 2

SAIF file formats are used to represent the switching activities in SoCs. It provides information about the toggle count of each signal which is valuable for estimating power consumption. It is vital in assessing the power consumption at different stages of verification thereby making the decisions for Power Optimization.

C. Scenario 3

Ensuring that the primary processor boots up successfully without any uninitialized values. This is accomplished by asserting a Power-On Reset signal, after which the Central Processing Unit (CPU) proceeds to Reset Vector locations to fetch and decode the instructions. The bootloader then executes and initializes the processor internal registers, memory controllers and cache.

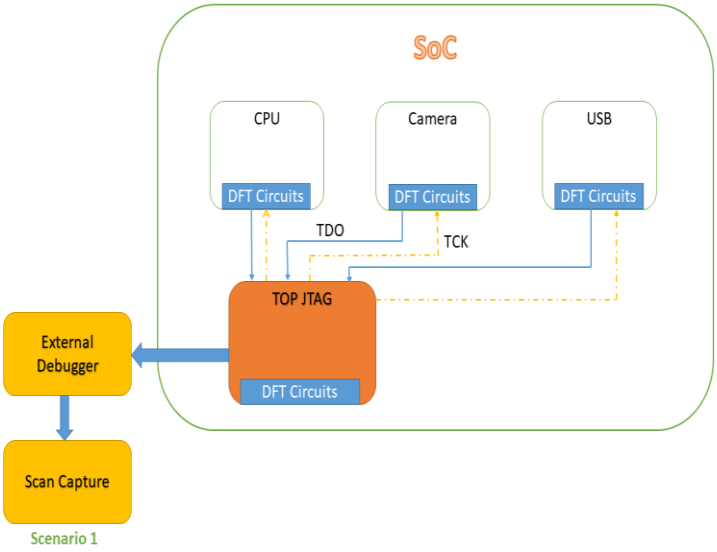

Figure 8: SoC SCANDUMP Capture Mechanism

VII. RESULTS

Observations with regard to Exynos Flagship SoC is as follows:

i. SoC SCANDUMP capture in 54mins.
ii. Accurate switching activity capture in 42mins for a Multimedia block of SoC
iii. Basic Processor boot up sequence is completed in 12mins.

TABLE II. GLS VS GLE TIME COMPARISON

Application Scenarios	GLS (Hrs)	GLE (Hrs)
Scenario 1	UNREALISTIC	0.9
Scenario 2	UNREALISTIC	2.2
Scenario 3	12	0.25

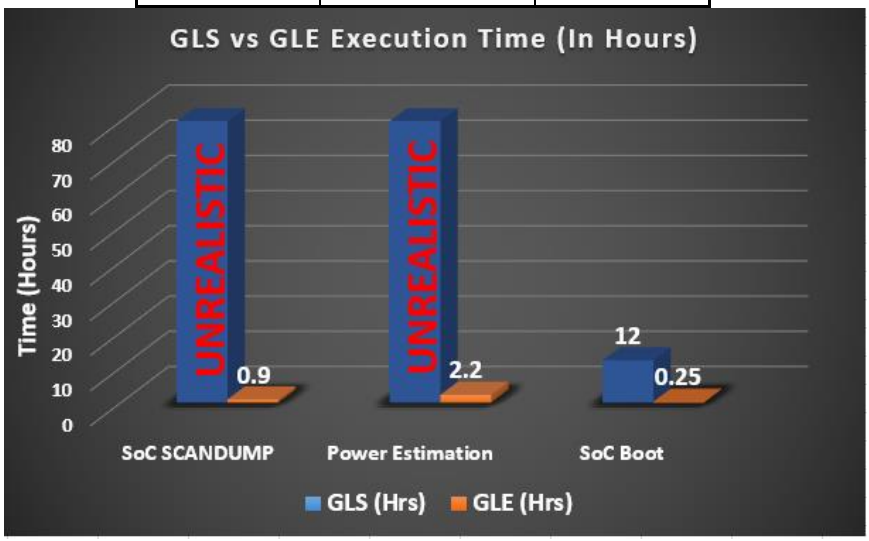

Figure 9. GLS vs GLE Execution Time Comparison

VIII. CONCLUSION AND FUTURE WORK

A Methodology flow is proposed for integrating full SoC netlist files to create a compiled database with the aid of an emulator compiler and map these files on the actual hardware for complex SoCs consisting of billions of gates.

Additionally, real use case scenarios like capturing the SCANDUMP of a SoC which is vital for debugging in case of a SoC hung, precise switching activity capture for accurate Power Estimation, CPU Boot up are validated efficiently. As compared to simulation that takes days to complete the verification, execution time for emulation takes only in order of minutes. As a result, we are able to effectively validate the long-running scenarios with Gate level netlist enabled on emulation environment and detect the SoC issues early during the pre-silicon stage only.

There's no support of full timing Standard Delay Format (SDF) currently in Emulation environment and all the delays will be ignored by the compiler. Hence, the zero delay or unit delay netlist files are used to validate all of the scenarios.

There's a mechanism to accurately capture the prime power using a glitch generated on the ZeBu empower emulator test bench. This will be examined as a future enhancement to the emulation environment which assists in the Power Analysis.

In advancements to these, we will be working on Unified Power Format (UPF) enablement integrated with GLE. This helps us to validate all the power scenarios.

REFERENCES

[1] K. S. Das, P. Prakash and A. Zala, "Accelerating GLS Simulation closure in DFT with Emulator," 2021 IEEE International Test Conference India (ITC India), Bangalore, India, 2021, pp. 1-6, doi: 10.1109/ITCIndia52672.2021.9532898.

[2] J. Aggarwal, T. Nguyen and P. K. Gupta, "DFT+Emulation – A Faster Way to Close Verification," 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Lisbon, Portugal, 2018, pp. 11-12, doi: 10.1109/QRS-C.2018.00015.

[3] Mohammad Ali Ghodrat, Kanishka Lahiri, Anand Raghunathan, "Accelerating system-on-chip power analysis using hybrid power estimation," DAC'07:Proceedings of the 44th annual Design Automation Conference, June 2007, pp. 883-886.

[4] B. J. Tomas, Y. Jiang and M. Yang, "SoC Scan-Chain verification utilizing FPGA-based emulation platform and SCE-MI interface," 2014 27th IEEE International System-on-Chip Conference (SOCC), Las Vegas, NV, USA, 2014, pp. 398-403, doi: 10.1109/SOCC.2014.6948962.

[5] Synopsys ZeBu5 User Guide.