2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Automation of Delay Tuning in TSV aware Heterogeneous 3D Inter-Die memory controller

Rahul Laxkar, Ananya Sinha, Naveen Srivastava, Sekhar Dangudubiyyam
SAMSUNG SEMICONDUCTOR INDIA RESEARCH
Bangalore, 560048

***Abstract*- Heterogeneous 3D Integrated Circuits (3DIC) are a technology that allows multiple dies (chips) to be stacked vertically to create a single compact package. This technique provides several benefits, including a reduction in power consumption, increased performance, and a smaller form factor. But it also comes with some challenges when 2 heterogeneous blocks are communicating through TSVs (Through-Silicon Via) like, data synchronization between dies, timing meet requirement between both the blocks on silicon and stuck at fault errors at TSVs.**

This paper will share an approach on how to overcome these challenges and to make the communication efficient between two dies. Our experimental results demonstrate the effectiveness of our approach in the verification of 2 heterogeneous blocks presents in different die. These blocks are connected through silicon vias(TSVs) in which one of them is master(LOGIC) and the other block is slave(Memory). The approach will explain the purpose and importance of DLINE (Delay-LINE) configuration and how can we achieve the timing requirement or data synchronization between 2 blocks present in different dies using DLINE configuration. The approach will also share an automation of delay tuning which gives the exact DLINE configuration needed for communication between the blocks with any kind of tech node.

The approach can be integrated into existing design flows for 3DIC, enabling designer to ensure reliable inter-die communication with TSV-aware Logic and SRAM.

I. INTRODUCTION

In 3D IC, the TSVs are interconnects that act as bridge between the dies, allowing them to communicate with each other. The TSVs can be used for various types of communication such as power delivery, clock distribution and data transfer. However, they introduce unique challenges, particularly in terms of signal integrity and timing delays.

In 3D IC, the inter-die MEM_CONTROLLER (LOGIC) acts as master which manages data communication with MEMORY die (slave) through TSVs. MEM_CONTROLLER and MEMORY are heterogeneous designs with different tech nodes, making the simulation a tedious process due to clock frequency differences and the challenge of ensuring proper data synchronization between blocks during simulation. TSVs have their delay lines and its dedicated delay registers (Delay-LINE/DLINE) for configuring the delays in each delay line to address above issues. The delay (Delay-LINE) tuning process can be used for DLINE (Delay-LINE) configuration which is essential to ensure reliable and efficient data transfer across TSVs. Manual or conventional delay tuning method is time-consuming and prone to errors, hence the need for automation of delay tuning is required.

Automating the delay tuning process involves developing algorithms that can adjust the timing parameters dynamically to optimize performance and minimize errors. By leveraging this approach enhances the performance, reliability, and scalability of 3D ICs.

Fig.1 illustrates the block connectivity of MEM_CONTROLLER (LOGIC) and MEMORY through TSV for memory access communication.

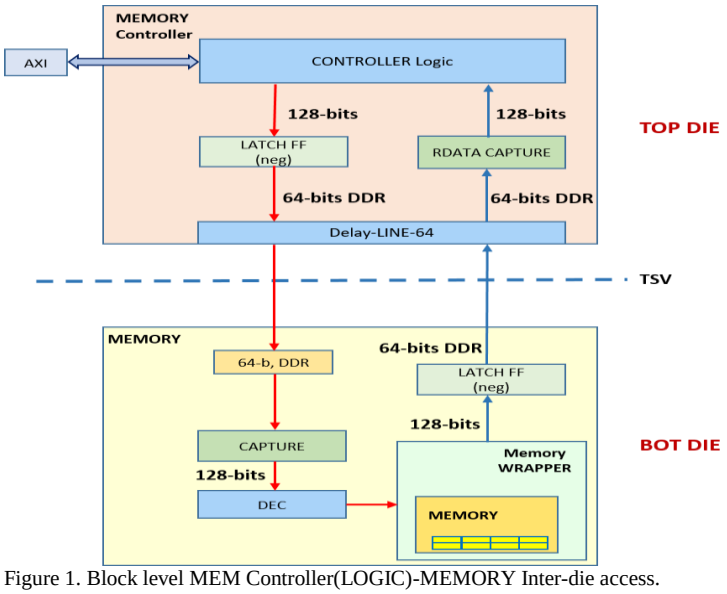

Figure 1. Block level MEM Controller(LOGIC)-MEMORY Inter-die access.

II. Conventional Delay-LINE Tuning Approach and Concept of Smart Tuning of Delay-LINE

A. Conventional Delay-LINE tuning approach and limitations

With conventional Delay-LINE tuning method, test is run manually by configuring required Delay-LINE registers with some specified delay values which is provided by designer based on STA expected timing delays on each Delay-LINE. The Delay-LINE associated register's configuration can be used to set required delays at the transmitting (LATCH) as well as receiving (CAPTURE) end for proper data sampling. The communication between the two dies is facilitated by Latch and Capture modules present on different blocks (inter-die) as shown in Fig.2. The LATCH module is responsible for transmitting 128-bit data to 64-bit DDR data, while the CAPTURE module converts 64-bit DDR data back into 128-bit data.

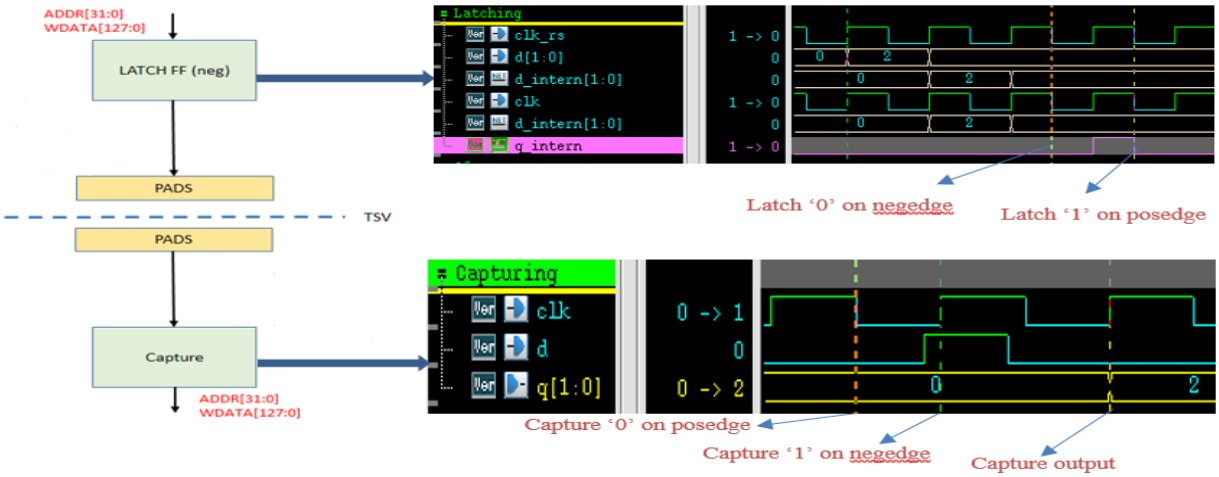

Figure 2. Latching and Capturing of data between the two dies (Inter-die)

The impact of configuring Delay-LINEs can be clearly seen on the sampling clock of the Capture module in Fig.3, which demonstrates the phase delay between the driving and sampling (LATCH and CAPTURE module respectively) of the data which was transmitted from MEM_CONTROLLER (Latch) die to MEMORY (Sampling) die. On the other hand, Fig.4 shows the communication between MEM_CONTROLLER and MEMORY without using the Delay-LINE configuration, resulting in no delay of capturing clock at MEMORY die in turn resulting in the conversion of wrong data at the output of Capture module end (MEMORY die).

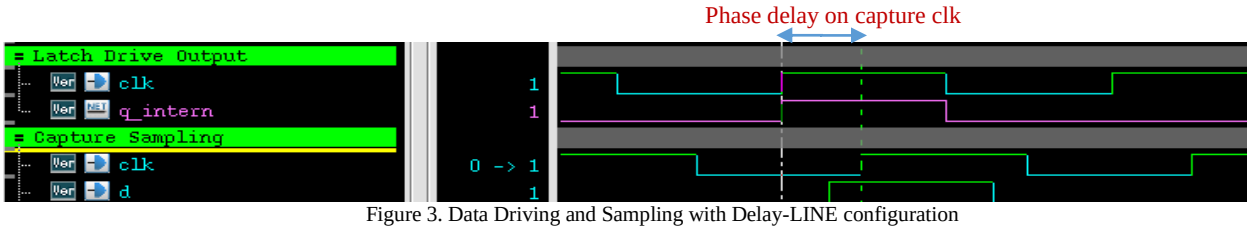

Figure 3. Data Driving and Sampling with Delay-LINE configuration

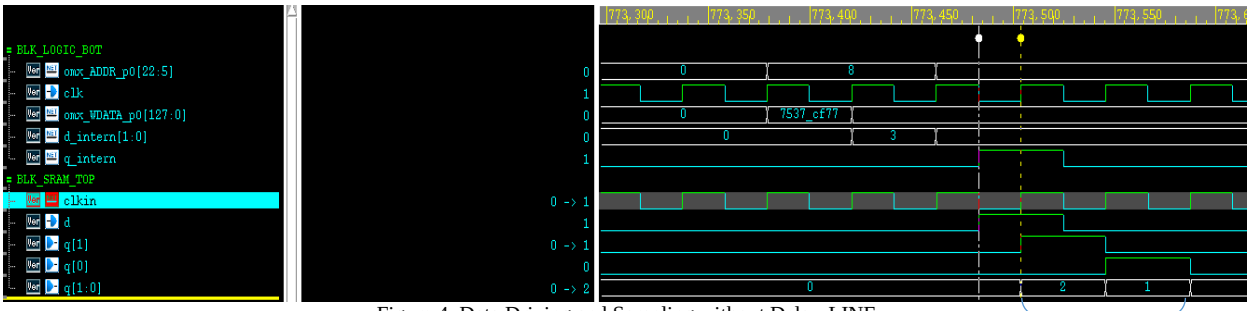

Figure 4. Data Driving and Sampling without Delay-LINE

Wrong data sampling at Capture

If data synchronization between the inter-die fails, then the Delay-LINE registers which includes phase (clock) delay, write data delay and read data delay registers are reconfigured by checking following issues:

1. Whether phase delay between the driving and sampling (LATCH and CAPTURE) of the data is sufficient enough to latch the correct data at controller die and sample correct data at memory die. Clock delay registers are reconfigured in case of insufficient phase delay.
2. During data write from controller die to memory die, whether there is sufficient capturing window in order to drive correct data to memory die. Write data delay registers are configured accordingly in case of insufficient capturing window.
3. During data read from memory to controller die, whether sufficient sampling window is there in order to sample correct data from memory. Read data delay registers are configured accordingly in case of insufficient sampling window.

Tuning Delay-LINE using conventional method is not full proof solution since due to various physical aspects of silicon, parameters on both the inter-die blocks can vary with different tech nodes. This process has several significant limitations:

1)High Complexity: It requires manual effort to check delay in each data line (phase, write data, read data and strobe lines) where data bits have changed either due to wrong sampling of data in controller block or wrong capturing of data in memory block.

2)Time-Consuming Process: Manual tuning requires extensive trial and error of delay setting, simulations, and iterations, leading to prolonged design cycles.

3)Inconsistent Results: Since there are many Delay-LINE registers to be configured which takes manual effort to change the Delay-LINE configurations. Configuring many registers manually can lead to errors which take multiple iterations to run the test for the data synchronization check with correct latency.
4)Difficulty in Managing Interdependencies: It can fail at validation time as the delay parameters can vary due to different tech nodes and temperature aspects. There are dependencies on RTL designer and STA reports to extract the exact configuration of DLINE registers.

B. Concept of Smart Tuning of Delay-LINE

To overcome above limitations, the conventional approach of Delay-LINE tuning can be replaced with the smart Delay-LINE tuning approach. With smart tuning approach, the Delay-LINE configurations are automated to get correct delay on each Delay-LINE of the data lines (phase, write data and read data) in order to ensure data synchronization between both the memory controller (LOGIC) and memory blocks which are residing on different dies.

For smart tuning approach, a single sequence is run in which each Delay-LINE register's (each Delay-LINE register is associated with its respective Delay-LINE) re-configuration process happens by randomizing and incrementally iterating all possible delay values till the time correct delay is achieved in each Delay-LINE which is sufficient enough for correct data capture at memory die and correct data sampling at controller die for accurate data transfer between both dies. Fig.5 represents the flow of Delay-LINE automation for one Delay-LINE register.

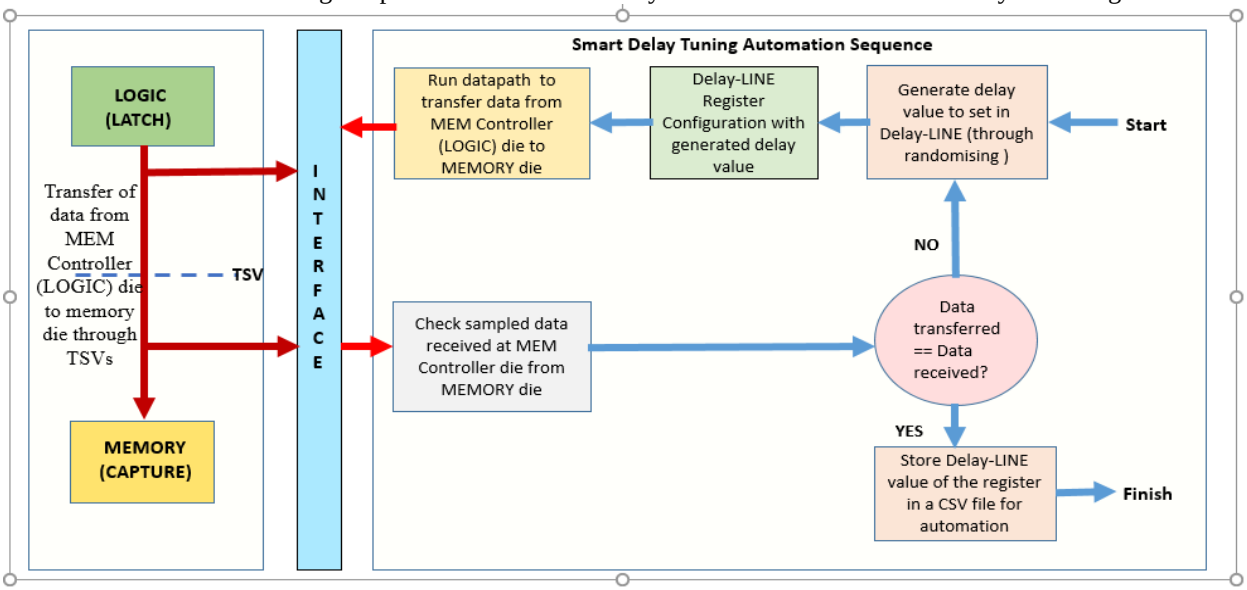

Figure 5. Delay Tuning Automation Flow in Sequence

There are multiple Delay-LINE registers (including phase delays, write data delays and read data delays) present in Controller die which are automated for data synchronization between controller and memory die. As one delay value is configured correctly, it gets stored in an input CSV report file which can used as a reference for configuring other delays values. Then, another delay line configuration iteration happens till correct delay is achieved. This process is followed till all correct delay values of all Delay-LINES are achieved and stored. The CSV report file generated can used as input to sequences for using stored delay values for generating new delays values when required.

The automation with smart tuning of Delay-LINE in 3D IC provides several advantages:
1. Improved Performance*:* With smart tuning of Delay-LINE, no manual effort required to check delay in each data lines as smart tuning will iterate till it gets optimized delay values for each delay line.

2. Enhanced Reliability: Smart tuning can ensure no human errors as no manual effort required to reconfigure delay values since smart tuning stops at the correct delay values during its automated iterations for delay check.
***3. Reduced design time*:** Smart tuning approach sets efficient Delay-LINE in one or two iterations, which in turn accelerates the design process as multiple iterations will not be required for delay check.
4. Scalability: Dependencies on designer and STA can be reduced by smart tuning. Automated verification can be easily adapting to different designs and technologies, making them versatile for various applications.

III. Implementation and Practical Consideration

The successful implementation of the smart tuning of Delay-LINE approach involves several key consideration and steps which are as follows:
1. ***Generate delay value to set in Delay-LINE:*** The initial step of smart tuning of Delay-LINE approach involves setting of Delay-LINE parameters based on simulation data. For getting simulation data, we run initial simulations in order to collect baseline performance data such as identifying critical paths and clock delays, write data path delays and read data path delays and other delay parameters. Delay values configured for initial simulations are usually received from designers. In cases, when we don't get optimized delay values from STA reports designers, then delay values are generated by randomizing each delay value with some constraints.
2. ***Delay-LINE configuration with new delay value:*** The delay values generated above, either from STA reports provided reports or through randomization, are then stored in the Delay-LINE configuration registers followed by the datapath run to check for proper inter-die data synchronization.
3. ***Transfer data from Controller die to memory die:*** After configuring delay registers, we initiate a set of data transfer check from memory controller (LOGIC) die to memory die. The data transferred from memory controller die are stored for check as write data at memory die.
4. ***Check sampled data received at controller die from memory die:*** The data then received back at controller die from memory die are stored as read data. Now, both the write and read data are compared for inter-die signal integrity check. Fig.6 shows write data and read data comparison to check for inter-die data integrity.

```
1055            xactor_write(addr1, wdata1);
1056            # 0.2us;
1057            xactor_read(addr1, rdata1);
1058
1059            if(rdata1 != wdata1)
1060              `uvm_error(get_type_name(), $psprintf("MEMORY ACCESS : Data Check Failed !! WDATA (%0h) != RDATA (%0h)", wdata1, rdata1))
1061            else
1062              `uvm_info(get_type_name(), $psprintf("MEMORY ACCESS : Data Check PASSED !!"), UVM_LOW)
```

Figure 6. Data integrity check for delay value configured in sequence

5. ***Feedback loop:*** If write data matches with read data, it indicates that delay values configured in each Delay-LINE are correct and those delay values are stored in a delay storing csv file which is used as input in the automated script that is used in the sequence for re-usage of generated delay values.
If write data doesn't match with the read data, it shows that delay values configured are wrong and needs to be re-configured. In such case, new delay values get automatically generated with the help of delay value iteration incrementing automation process. Below Fig.7 shows one of the write data Delay-LINE's delay value generation through iteration incremented. The new delay values now generated are again stored in delay registers and above steps are repeated for delay check.

```
1131            for(int ii = 1; ii < 16; ii++) begin
1132               `uvm_info(get_type_name(),$psprintf("NDLINE_SEL_WDATA configuration starts for value %0h......", ii),UVM_LOW)
1133               NDLINE_SEL_WDATA[6:0]   =  ii;
1134               xactor_write(36'h1_2303_0408, NDLINE_SEL_WDATA);
1135               xactor_read(36'h1_2303_0408, rdata);
```

Figure 7. Delay value generation through iteration incrementing in sequence

IV. Impact on the Verification Process

The automation with smart tuning of Delay-LINE plays a significant role in the verification process of Memory Controller and Memory for inter-die communication. Here are some of the key impacts:

1. ***Optimized timing:*** Smart tuning of Delay-LINE ensures that delay parameters are finely tuned to achieve optimal timing, leading to proper data synchronization between inter-die blocks. More efficient delay management allows for higher memory bandwidth, enhancing overall performance of inter-die data communication.

2. ***Reduced Errors:*** Automation with smart tuning of delays minimizes the human errors which might have come by manually tuning of multiple delay lines. It leads to more reliable and consistent delay tuning results.

3. ***Reduction in Turnaround Time****:* Unlike conventional approach of delay tuning that requires multiple simulation and then subsequent analysis, smart delay tuning of approach significantly achieves required delays in single simulation. This reduction in turnaround time accelerates the verification process which in turn shortens the design cycle by eliminating the time-consuming manual delay tuning process for data synchronization between inter-die. With quicker design iterations and validation, products can reach the market faster.

4. ***Real-Time Adjustments:*** Automated smart delay tuning can make real-time adjustments to various delay parameters of different blocks on different dies in response to changing conditions, such as varying timing parameters, different tech nodes for increased performance of 3D IC.

V. Case Studies and Results

A. Case Study

1. Implementation on SoC

The automation of Delay-LINE tuning has been tested on a set of heterogeneous blocks in SoC, shown in Fig8. These Delay-LINE registers are basically configured in the Memory Controller block. Each controller block in SoC has its own Delay-LINE register set. According to the data sheet of Delay-LINE, 25-30% clk phase delay between LATCH and CAPTURE block would be good enough to achieve higher performance.

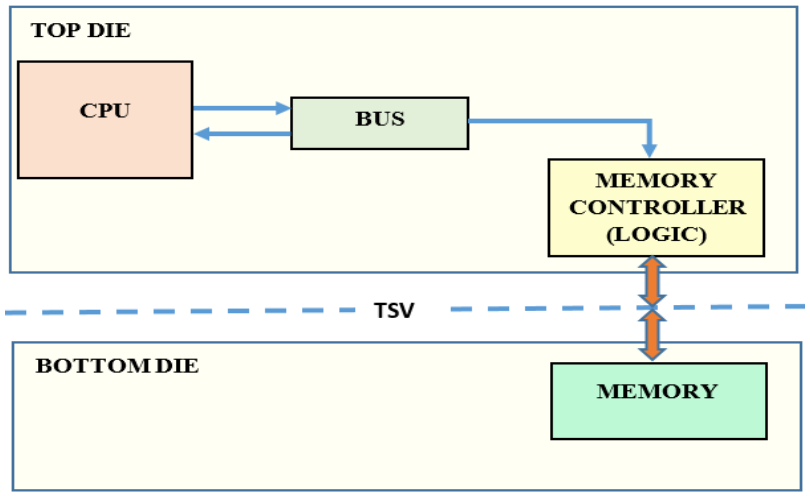

Figure 8. Block level representation of Memory Controller and Memory in 3D IC SOC with TSVs

The Automated Delay Tuning sequence runs the datapath till 25-30% phase delay and provide the Delay-LINE value which has lower latency and proper data synchronization. We can also set different delays for each MEMORY block according to the design requirement and performance. This provides flexibility in our testcases to verify different kind of performance from Memory Controller to MEMORY in 3D IC SoC design.

Below Fig.9 represents automation of one write data Delay-LINE register with different values which provides different write data value on different iterations of Delay-LINE value configuration.

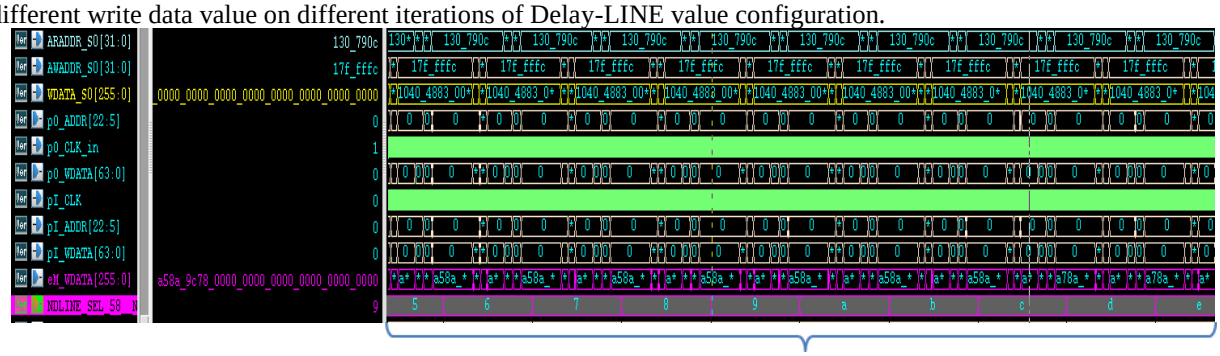

Figure 9. Different Delay-LINE setting with automation to check correct Delay value

B. Results

As a result of automating the delay tuning process in the TSV-aware heterogeneous 3D inter-die memory controller, we are able to significantly reduce the time and effort required to perform the simulation. The use of Delay-LINE allowed us to ensure proper data synchronization and required performance between the Controller and Memory blocks, resulting in accurate data transfer between the two dies. The automated delay tuning sequence determines the efficient delay values to achieve the lower latency between the dies and solves following write as well as read delay issues:

1. During data write from controller die to memory die, to set sufficient capturing window in order to drive correct data to memory die, write data delay registers are configured accordingly. In Fig.10(a) wrong write data capturing can be seen because of insufficient capturing window. On the other hand, Fig.10(b) clearly shows correct write data captured as write data delay registers configuration solves the issue of insufficient capturing window at memory die.

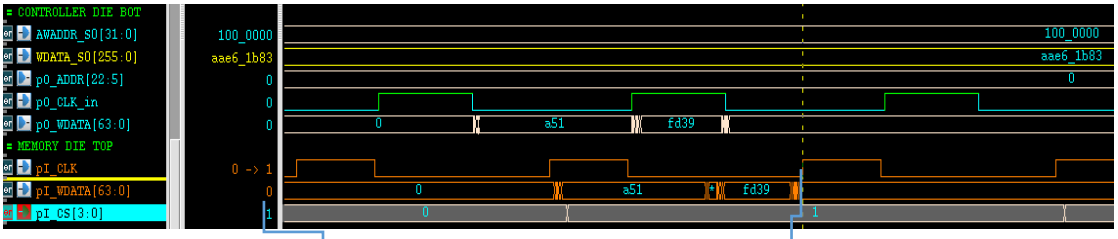

Figure 10(a). Wrong data captured with inaccurate write data delay in Delay-LINE

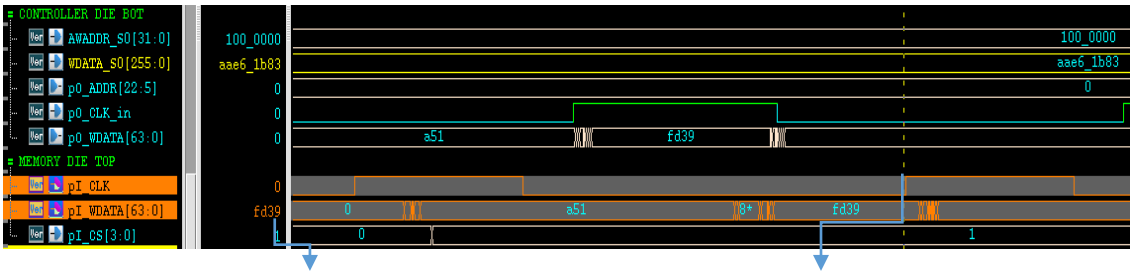

Figure 10(b). Correct data captured with accurate write data delay in Delay-LINE as provided by Designer

2. During data read back from memory to controller die, to set sufficient sampling window in order to sample correct data from memory die, read data delay registers are configured accordingly. Fig.11(a) shows wrong read data sampled at mem controller die due to insufficient sampling window. On the other hand, Fig.11(b) shows correct read data delay register tuning solves this issue of insufficient sampling window.

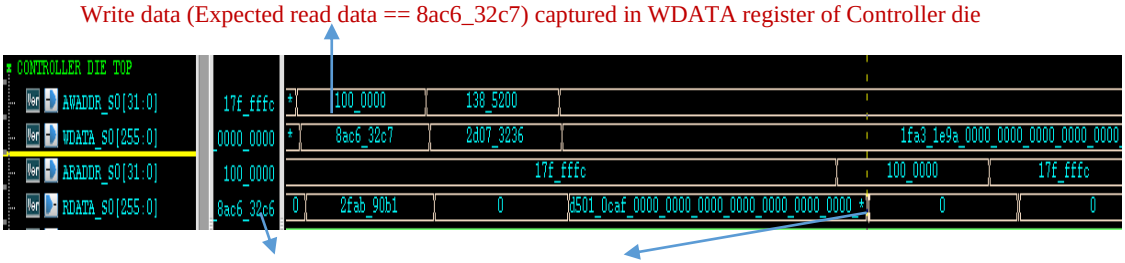

Figure 11(a). Wrong read data sampling with inaccurate read data delay in Delay-LINE

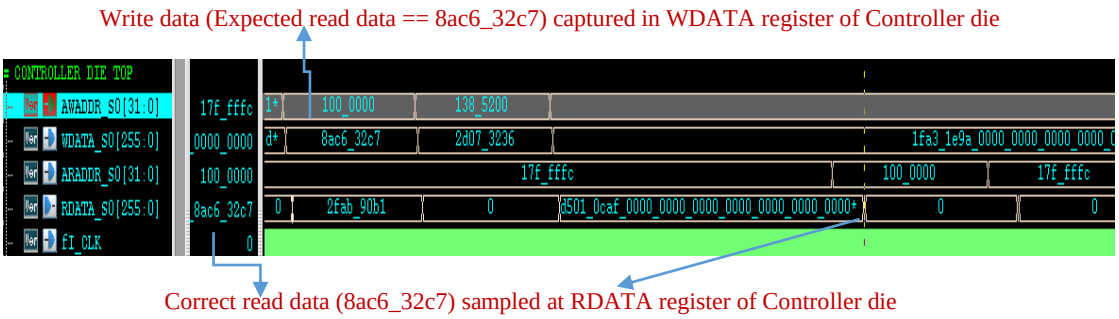

Figure 11(b). Correct read data sampling with accurate read data delay in Delay-LINE

Below Fig12(a). is snippet of log file from test run for gate level simulations showing how out of different iterations of delay value tuning, data access from memory controller die to memory die clearly fails when wrong delay value “a” is tuned in Delay-LINE registers. However, Fig12(b) shows how correct Delay-LINE configuration “c” during delay value tuning process leads to proper and accurate data driving from controller to memory die and correct data sampling back from memory die to controller die end. Fig.13 represents correct data capture due to correct delay value “c” tuned at Delay-LINE registers.

```
[3063355] UVM_INFO (sanity_ext_sram_access_seq_c) ************************************************************
[3063355] UVM_INFO (sanity_ext_sram_access_seq_c) NDLINE_SEL_WDATA configuration starts for value a......
[3063648] UVM_INFO (cdn_ps_ahb0_reg_adp) UVM_WRITE, addr = 'h0000000123030408(ndline_sel_58), data = 'h1a3468a
[3063968] UVM_INFO (cdn_ps_ahb0_reg_adp) UVM_READ, addr = 'h0000000123030408(ndline_sel_58), data = 'h1a3468a
[3063968] UVM_INFO (base_ahb_seq_c) ahb_master_burst.Data.size() =           4!! (bytes)
[3064101] UVM_INFO (cdn_ps_ahb0_reg_adp) UVM_WRITE, addr = 'h00000000b1000000, data = 'haae61b83
[3064235] UVM_INFO (cdn_ps_ahb0_reg_adp) UVM_WRITE, addr = 'h00000000b130790c, data = 'ha78a9c78
[3064728] UVM_INFO (cdn_ps_ahb0_reg_adp) UVM_READ, addr = 'h00000000b1000000, data = 'haae61b83
[3064728] UVM_INFO (base_ahb_seq_c) ahb_master_burst.Data.size() =           4!! (bytes)
[3064891] UVM_INFO (cdn_ps_ahb0_reg_adp) UVM_READ, addr = 'h00000000b17ffffc, data = 'h10404883
[3064891] UVM_INFO (base_ahb_seq_c) ahb_master_burst.Data.size() =           4!! (bytes)
[3065055] UVM_INFO (cdn_ps_ahb0_reg_adp) UVM_READ, addr = 'h00000000b130790c, data = 'ha58a9c78
[3065055] UVM_INFO (base_ahb_seq_c) ahb_master_burst.Data.size() =           4!! (bytes)
[3065055] UVM_INFO (sanity_ext_sram_access_seq_c) SRAM ACCESS : Data Check PASSED !!
[3065055] UVM_INFO (sanity_ext_sram_access_seq_c) SRAM ACCESS : Data Check PASSED !!
[3065055] UVM_ERROR (sanity_ext_sram_access_seq_c) SRAM ACCESS : Data Check Failed !! WDATA (a78a9c78) != RDATA (a58a9c78)
[3065055] UVM_INFO (sanity_ext_sram_access_seq_c) NDLINE_SEL_WDATA[6:0] value needs to be incremented as Data Check Failed for delay value = a
```

Figure 12(a). Write data Delay-LINE value “a” which failed for data access between controller and memory die

```
[3066751] UVM_INFO (sanity_ext_sram_access_seq_c) NDLINE_SEL_WDATA[6:0] value for which Data Check Passed = b
[3066751] UVM_INFO (sanity_ext_sram_access_seq_c) ************************************************************
[3066751] UVM_INFO (sanity_ext_sram_access_seq_c) NDLINE_SEL_WDATA configuration starts for value c......
[3067038] UVM_INFO (cdn_ps_ahb0_reg_adp) UVM_WRITE, addr = 'h0000000123030408(ndline_sel_58), data = 'h1a3468c
[3067351] UVM_INFO (cdn_ps_ahb0_reg_adp) UVM_READ, addr = 'h0000000123030408(ndline_sel_58), data = 'h1a3468c
[3067351] UVM_INFO (base_ahb_seq_c) ahb_master_burst.Data.size() =           4!! (bytes)
[3067481] UVM_INFO (cdn_ps_ahb0_reg_adp) UVM_WRITE, addr = 'h00000000b1000000, data = 'haae61b83
[3067611] UVM_INFO (cdn_ps_ahb0_reg_adp) UVM_WRITE, addr = 'h00000000b130790c, data = 'ha78a9c78
[3067745] UVM_INFO (cdn_ps_ahb0_reg_adp) UVM_WRITE, addr = 'h00000000b17ffffc, data = 'h10404883
[3068105] UVM_INFO (cdn_ps_ahb0_reg_adp) UVM_READ, addr = 'h00000000b1000000, data = 'haae61b83
[3068105] UVM_INFO (base_ahb_seq_c) ahb_master_burst.Data.size() =           4!! (bytes)
[3068268] UVM_INFO (cdn_ps_ahb0_reg_adp) UVM_READ, addr = 'h00000000b17ffffc, data = 'h10404883
[3068268] UVM_INFO (base_ahb_seq_c) ahb_master_burst.Data.size() =           4!! (bytes)
[3068431] UVM_INFO (cdn_ps_ahb0_reg_adp) UVM_READ, addr = 'h00000000b130790c, data = 'ha78a9c78
[3068431] UVM_INFO (base_ahb_seq_c) ahb_master_burst.Data.size() =           4!! (bytes)
[3068431] UVM_INFO (sanity_ext_sram_access_seq_c) SRAM ACCESS : Data Check PASSED !!
[3068431] UVM_INFO (sanity_ext_sram_access_seq_c) SRAM ACCESS : Data Check PASSED !!
[3068431] UVM_INFO (sanity_ext_sram_access_seq_c) SRAM ACCESS : Data Check PASSED !!
[3068431] UVM_INFO (sanity_ext_sram_access_seq_c) NDLINE_SEL_WDATA[6:0] value for which Data Check Passed = c
[3068431] UVM_INFO (sanity_ext_sram_access_seq_c) ************************************************************
[3068431] UVM_INFO (sanity_ext_sram_access_seq_c) NDLINE_SEL_WDATA configuration starts for value d......
```

Fig.12(b) Write data Delay-LINE value “c” which passed for data access between controller and memory die

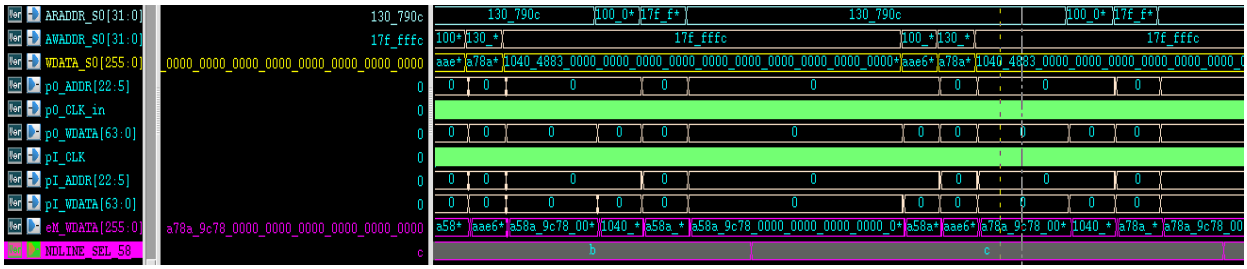

Figure 13. Correct data (a78a_9c78) capture with correct write data Delay-LINE value “c”.

VI. CONCLUSION

The smart tuning of Delay-LINE in TSV aware heterogeneous 3D inter-die controller provides efficient delay value to achieve lower latency between dies. This approach uses Delay-LINE configuration to ensure proper communication between blocks during simulation. Automating the delay-tuning process improves the accuracy and speed of the simulation while reducing the risk of errors. The results of implementing this automation technique shows that it significantly improves the performance and overall communication between Controller and Memory present in separate dies. Below table shows the advantages of the proposed approach over conventional approach.

TABLE I
ADVANTAGES OF DELAY-TUNING APPROACH OVER CONVENTIONAL APPROACH

Attributes	Conventional Approach	Smart Delay Tuning Approach
Dependency on Designer	Designer provide the Delay values from the STA report	No dependencies on designer
Accuracy	Delays extracted from STA report cannot be accurate in real time scenario due to effect of thermal or voltage fluctuation	Smart delay tuning can be used for real time scenarios as well to capture the precise delays
Iterations and Time consumption	The conventional approach requires multiple iteration to sync all the TSV signals with proper delays which is in cost lot of time during GLS simulations	It takes only one simulation to capture all the delay values for the TSV signals which result in the left shift of the GLS cycle

REFERENCES

[1] "IEEE Standard for SystemVerilog--Unified Hardware Design, Specification, and Verification Language," in IEEE Std 1800-2017 (Revision of IEEE Std 1800-2012) , vol., no., pp.1-1315, 22 Feb. 2018, doi: 10.1109/IEEESTD.2018.

[2] "IEEE Standard for Universal Verification Methodology Language Reference Manual," in IEEE Std 1800.2-2020 (Revision of IEEE Std 1800.2-2017) , vol., no., pp.1-458, 14 Sept. 2020, doi: 10.1109/IEEESTD.2020.9195920


2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
INDIA


2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
INDIA