2014 DESIGN AND VERIFICATION DVCON CONFERENCE AND EXHIBITION INDIA
Sept 25-26, 2014
Hotel Park Plaza
Bangalore, India

Simulation Based Pre-Silicon Characterization

Saurabh Pandey, Senior Design Engineer, Texas Instruments, Bangalore, India (s-pandey@ti.com)
Venkatseema Das, Senior Design Engineer, Texas Instruments, Bangalore, India (venkatseemadas@ti.com)
Arif Mohammed, Senior DV Lead Engineer, Texas Instruments, Bangalore, India (arifm@ti.com)
Nishant Gurunath, Summer Intern, Texas Instruments, Bangalore, India (nishantgurunath@gmail.com)

***Abstract*— Modern System-on-Chips are dynamic and complex. With technology variability and scaling faster than Moore's prediction, guaranteeing seamless integration of SoCs on board has become more challenging than before. For reliable communication between multiple devices on a system, it is crucial to understand their I/O performance characteristics. SoC manufacturers provide input switching characteristics such as setup/hold time requirement and output delay information in device data sheets usually obtained through post-silicon characterization. Pre-silicon verification to ensure that the design meet these requirements is really an arduous task for engineers, which is further exacerbated by shortening product design duration and increasing time-to-market pressure. Static Timing Analysis (STA) provides a solution by estimating the chip input timing requirements in pre-silicon phase. But what if there is a mistake in STA? Is there a convenient way to stress the timing limits in simulations to verify STA data? This paper introduces a method of simulation based characterization in pre-silicon phase to address this challenge. It describes how to stress a peripheral's timing limits during Gate Level Simulation by changing the chip IO delay and the way the implement the module in test bench.**

Keywords— STA - Static Timing Analysis, GLS – Gate Level Simulation, Characterization, Simulation

I. INTRODUCTION : WHAT IS INPUT CHARACTERIZATION?

Chip manufacturer has to specify the timing requirements of all the chip inputs for the chip to operate properly in a system. The input timing specification could be timing relation between two chip inputs or between a chip input and a chip output. This process of finding out actual timing specification is known as characterization. These timing numbers are put in the device datasheet as setup and hold time with respect to the reference signal. End user uses this information for designing his system.

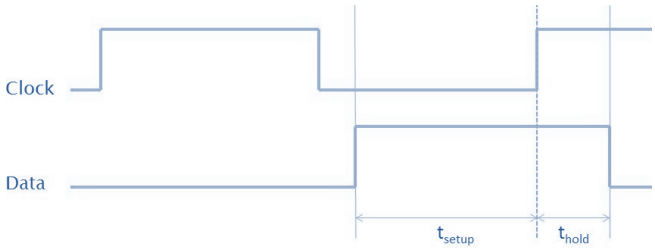

Figure 1: Setup and Hold time margin for Data w.r.t reference Clock

II. HOW TO DO CHARACTERIZATION?

The whole process of characterization can be divided into two phases (a) Pre-silicon phase and (b) Post-silicon phase. In pre-silicon phase design timing behaviour can be analysed by two methods

- Static Timing analysis
- Gate Level simulations

Static Timing Analysis (STA) is a method of validating the timing performance of a design by checking all possible paths for timing violations under worst case conditions. It considers the worst possible delay through

each logic element, but not the logical operation of the circuit. Please note that STA checks the design for proper timing, not for correct logical functionality.

Gate Level Simulation (GLS) is done on the timing annotated netlist where the actual design library components are instantiated with timing delays. Gate level simulations on timing annotated netlist are carried out to verify protocol and data integrity as they excite the actual logical paths in the design at the rated frequency.

All the steps discussed so far were based on some or other kind of modelling of the design. Any manufacturer cannot release the chip in the market based on just the pre-silicon data .The final step of characterization happens in the post-silicon phase. Post silicon characterization is done on high precision testers, which are capable of driving and sampling chip pins with accuracy and high time resolution. Functional patterns developed by design team during pre-silicon phase are used here. These patterns have the information of what data has to be driven to the chip to check certain functionality and also what to expect on output pins to decide pass/fail. While doing input characterization, tester reads the functional pattern, drives the input data and makes sure that the pattern passes. Then input data is swept close to the relevant clock edge in steps, till pattern fails. Pattern pass to fail transition helps in determining the actual setup/hold requirement of the design.

III. ISSUES WITH THE CURRENT APPROACH

We have seen few cases where errors in STA constraints result in timing issues on silicon. If any major STA miss is caught during the post-silicon characterization phase, design re-fabrication remains the only option to fix the issue which involves delay in time to market of the device and cost to company. This encouraged us to introduce an additional check based on simulation in pre-silicon phase to ensure reliability of STA data and avoid any surprises on silicon arrival.

To understand the issue further let's consider a case depicted in figure 2. If during STA due to specification of wrong case analysis, the path shown in red is taken instead of actual functional path shown in green, then design timing numbers will not match STA numbers.

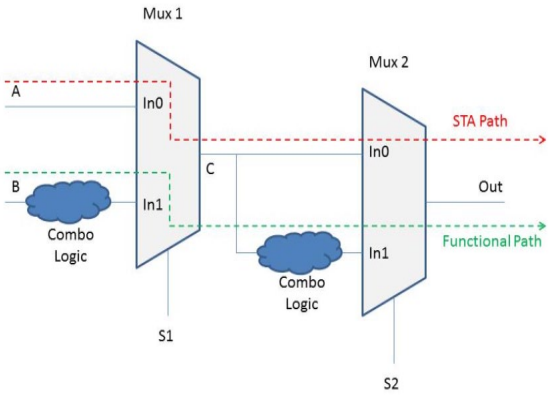

Figure 2: STA path vs Functional path

Now one obvious question arises that why it was not caught in GLS phase. GLS simulation needs testbench to drive the data on chip inputs. Usually the testbench components drive the data with zero delay or with a compile time fixed delay. One can argue that varying the compile time fixed delay, data can be swept across and such STA issues can be caught. But creating compile and then running the test each time is time consuming. Even if someone does this, we will discuss later that just delaying the input data is not enough to find out both setup and hold time requirements. Also, in many cases the third party TB components are encrypted and it is not possible to edit them and change delay. This paper introduces a convenient way to stress the timing limits during GLS by changing the delay at runtime in steps for the same compile. We propose the use of configurable delay module in test bench for simulation based pre-silicon characterization of the design.

IV. WHAT IS CONFIGURABLE DELAY TESTBENCH?

The configurable delay TB module consists of multiple instantiation of basic configurable delay element block, one for each chip IO. Key features of the configurable delay testbench module are

(a) It is a verilog module which can be plugged-in in any existing testbench as shown in figure 3. The configurable delay TB module sits between the conventional TB and the chip. Existing TB components are capable of driving data according to the protocol and the configurable delay TB just takes their outputs modifies them and drives them on chip IOs. As the configurable delay testbench interacts only with the ports of existing TB components, it can be used with any kind of BFMs i.e modules written in HDL like verilog, VHDL or HVL like 'e', system verilog and even encrypted third party modules.

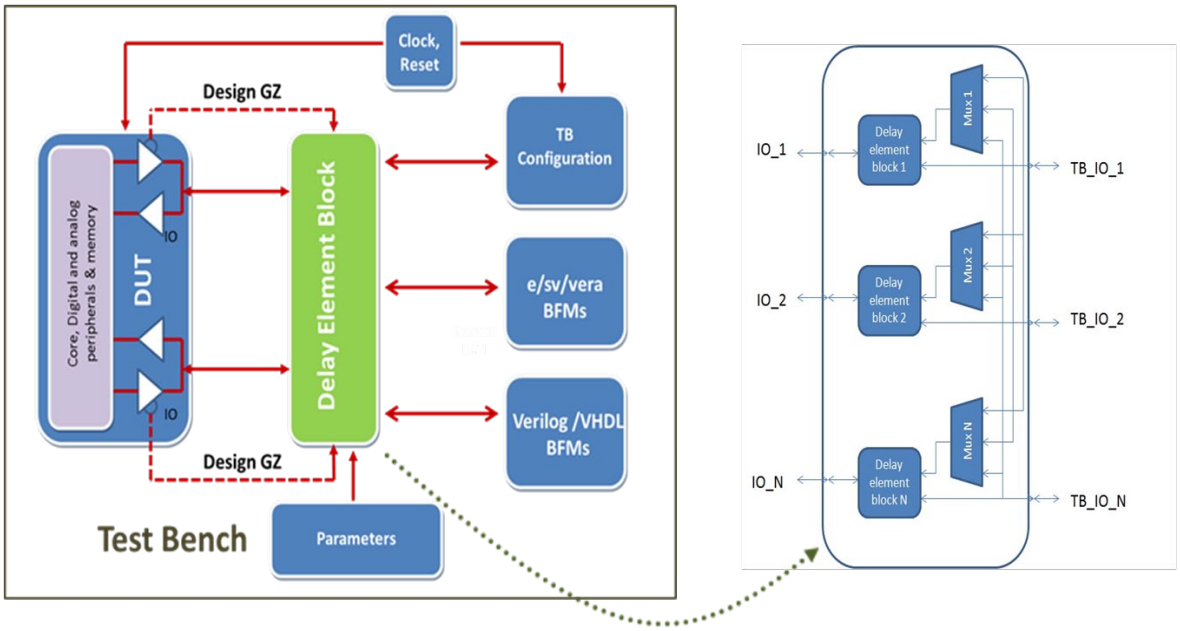

Figure 3: Block diagram of test-bench with configurable delay TB module

(b) The configurable TB module should be able to pass the chip outputs as it is to the other testbench components without any delay, but it should add desired delay in the path from testbench to chip inputs. Please note that chip IO direction is not static and can change dynamically during the testcase run. To know whether a specific design IO is in input or output mode, each delay element looks at design IO direction signal (GZ).

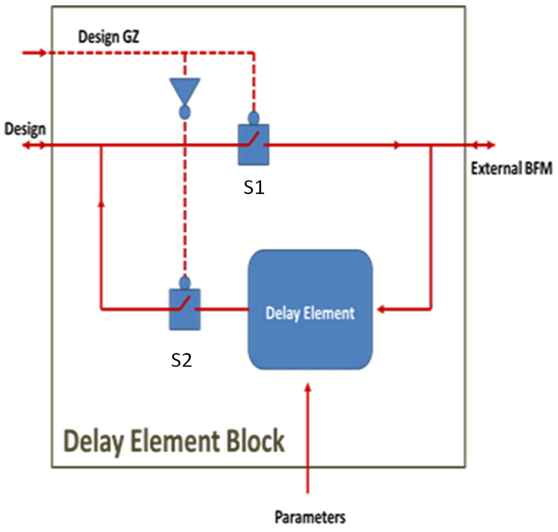

Figure 4: Block diagram of individual Delay Element

From figure 4, it is clear that when GZ is low i.e. chip IO is in output mode, switch S1 is on and S2 is off and thus design output goes to the existing TB components undisturbed. When GZ is high i.e. design IO is in input mode, S2 is on and S1 is off and thus output of TB components pass through the delay element before and then reach the design.

(c) Third and the most important feature of the configurable delay element is that it can be used to find both setup and hold requirement of a chip input. To find the setup time just delaying the testbench driven design inputs is sufficient but the challenge comes in finding out the hold time where data has to be moved backwards in time to bring it closer to the relevant clock edge. Let us take an example where data input to a specific peripheral in the design is driven from external world with reference to clock. *Clock* and *data* are design IOs and datasheet requirement is to document setup/hold time requirement of *data* with respect to the positive edge of *clock*. In testbench environment, *data* signal comes directly from drivers in the existing TB modules according to the required protocol. To find the setup time the data has to be simply delayed to move its positive edge closer to the positive edge of *clock* edge. But for finding out the hold time requirement the negative edge of data has to be moved back in time to bring it closer to the positive clock edge.

V. WORKING PRINCIPLE OF DELAY ELEMENT

In this section we discuss how the above mentioned features are achieved in simulation. Let's take an example where data is launched by the existing TB modules at the positive edge of clock and sampled by the design at the next positive edge. Please refer to the waveform shown in the figure 5. The idea is based on extracting the information of what *data* value is driven by the existing TB in each *clock* cycle and then using that value to create a *data* pattern which can be used to find out setup/hold time boundaries.

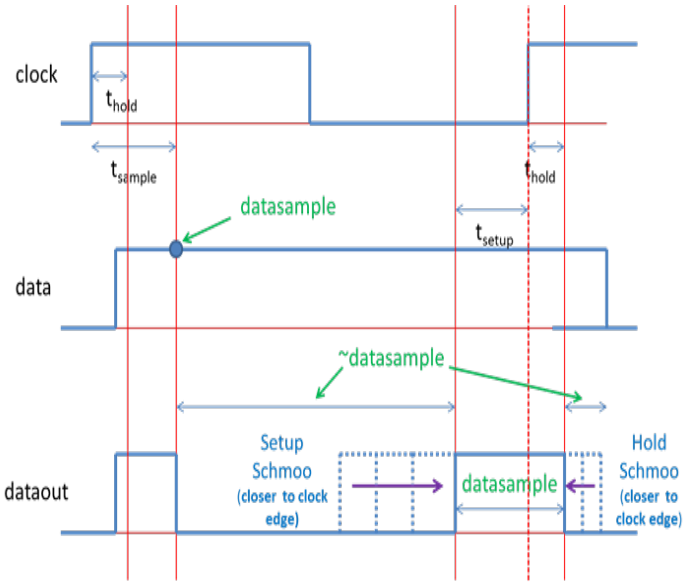

Figure 5: Waveform for core delay element implementation

The basic working principle of delay element module depends on below three equations.

$$\#(t_{sample})\ \text{dataout} = \sim\text{datasample};$$

$$\#(t_p - t_{sample} - t_{setup})\ \text{dataout} = \text{datasample};$$

$$\#(t_{setup} + t_{hold})\ \text{dataout} = \sim\text{datasample};$$

Note: These equations are written in VERILOG syntax. They are executed sequentially and triggered at every launch edge of clock (positive edge in this example)

Let's visualize the core working principle of delay element and its implementation below.

- Each delay element block has two inputs - *clock* and *data* and one output - *dataout.*
- As the name suggests *clock* input should be connected to the reference clock pin against which the timing specifications have to be found.
- *data* input is connected to the existing TB component's output which is supposed to drive the design data pin.
- *dataout* is the modified version of *data* which will be connected to the design's data pin which we want to characterize .
- The delay element samples the *data* at t_{sample} time units after positive edge of *clock*. This data is named as *datasample*.
- *dataout* waveform now completely depends upon three parameters –t_{sample}, t_{setup} ,t_{hold} and the *datasample* value.
- As shown in figure 5 the delay element block starts driving inverted value of *datasample* t_{sample} time units after positive edge of the *clock*.
- *dataout* is same as *datasample* at t_{setup} time units before the next positive edge,. This creates a positive edge on the actual data going to the chip and by playing around with t_{setup} value this edge can be moved closer or farther from the positive clock edge.
- t_{hold} time units after the positive edge, *dataout* is again made invert of *datasample* . This creates the negative edge of actual data going to the pin and by playing around with t_{hold} the negative edge can be moved closer or farther from the positive edge of the clock.
- The delay element block takes t_{setup} and t_{hold} as runtime parameters and by changing their values in steps for each run we will be able to sweep the data and find out setup/hold timing requirements.
- To find the setup time one can start with a larger value of t_{setup} and reduce it in steps till the value of data sampled is wrong which again should get indicated by test failure. Same scheme should be followed for finding out the hold time.
- To make sure that the data sampled by delay element block is always correct, the operating frequency is chosen to be very low which enables us to pick a large value for sampling time (t_{sample}). This ensures t_{sample} chosen works fine with any process, voltage and temperature variations. Running at very low frequency is based on the fact that setup/hold timing requirement of data coming from external world w.r.t clock pin will not get affected by change in the frequency.
- Changing the data in steps for every run has been automated using an automated flow based on Perl scripts and explained later in the paper.

VI. SUMMARY OF DELAY ELEMENT BLOCK PARAMETERS

- Run time parameters – t_{sample}, t_{setup}, t_{hold} and t_p
- t_{setup} and t_{hold} are used to move the relevant data edge
- t_{sample} is used to sample the correct value of *data* which has to be driven
- t_p is the time period of reference clock
- t_{sample} and t_p once decided do not change in every run
- The launch and capture edge of *clock* are also configurable. For example it will work for peripherals where data is driven at negative edge and gets captured at positive edge
- *clock* input of a delay element comes from the output of a N-to-1 multiplexer where N is the number of IOs in the design. N such multiplexers are present, one for each delay element block. The multiplexer

selection bits are also passed at runtime parameter .Thus any design pin can act as *clock* depending upon the requirement.

VII. TEST AUTOMATION

We wanted to have an automated way of correlating gate-level-simulation values with STA numbers on a specific design peripheral. We preferred to opt for perl based script-ware to achieve this.

- What is test automation?

Test automation script-ware is nothing but a set of perl scripts to derive and apply configuration parameters (t_{setup}, t_{hold} etc) to the delay element module, control the execution of tests and generate result reports. Then the timing numbers derived from these reports are compared with STA data manually.

- Why test automation?

The tasks for deriving configuration parameters (such as stepping through set-up or hold time values till violation) can be laborious and time consuming to do manually. In addition, a manual approach might not always be correct due to human errors. Perl script offers a mechanism to perform these tasks in an automated manner effectively. Once automated tests have been developed, they can be run quickly and then another script can decode the results to obtain final characterization numbers for comparison. This set of perl scripts is referred as test-automation script-ware.

- How to do test automation?

We can feed the script-ware all the required input through a spreadsheet in xls or csv format. It includes the testcase name, number of iterations, directive to stress setup or hold margin, desired step size and IO information etc. The script-ware should be able to generate a regression list of testcases along with all required parameters for configuring delay element block. This regression suite can be run on any vendor provided or in-house regression tool to simulate the testcases. Then we can use another script to decode the results and publish a final report for further analysis by user.

Figure 6 illustrates the flow diagram for test automation procedure.

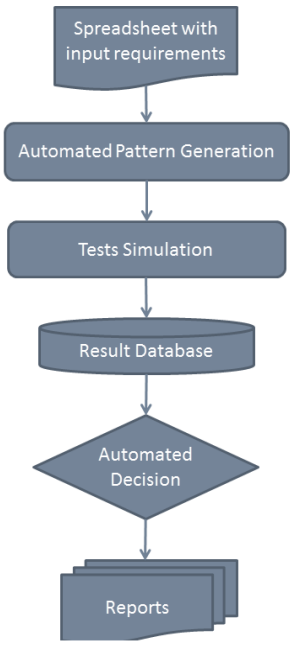

Figure 6: Flow diagram for characterization test automation

Input spreadsheet to the script-ware looks somewhat like an example given in figure 7.

	A	B
1	**Module Name**	SD-RAM
2	**Test Name**	data_check_sdram_32bit
3	**No of Iterations**	5
4	**Timescale**	ps
5	**Step size**	100
6	**Variable Parameter**	Setup
7	**IO-28**	reference
8	**IO-29**	tsample=20;tsetup=1000;thold=1000;tp=40000
9	**IO-30**	tsample=20;tsetup=1000;thold=1000;tp=40000
10	**IO-31**	tsample=20;tsetup=1000;thold=1000;tp=40000
11	**IO-32**	tsample=20;tsetup=1000;thold=1000;tp=40000

Figure 7: Input spreadsheet to the test automation script-ware

Note: IO-28 is the chip IO through which SD-RAM receives reference (clock) signal and IO-29, IO-30 etc. drives data inputs to the peripheral.

VIII. RESULTS

The reliability of this technique is checked by correlating the gate level simulation values with STA numbers on design peripherals like Serial Peripheral Interface (SPI) and SD-RAM etc. The results for 32bit SD-RAM's input data setup/hold characterization are presented in this section.

Let's start with setup margin analysis from SDRAM simulation results, and then proceed to hold margin calculation.

We needed to step through the setup time margin for data with respect to reference clock to eventually sweep the data transition edge close to clock edge. We started with an initial safe setup margin (t_{setup}) which is more than the actual setup requirement, so there was no timing violation and simulation passed. Then we kept on decrementing t_{setup} in steps and re-simulated the same testcase. The simulation result didn't fail in the first three steps till we reached the violation point in fourth step when data transition occurred close to clock edge and within the setup margin. Then our testcase failed because of timing violation by one or more design flops. Hence the setup margin (t_{setup}) provided in the last passed simulation (3rd run) can be treated as the approximate setup requirement for the peripheral under test. Figure 8 depicts the above mentioned simulation behaviour.

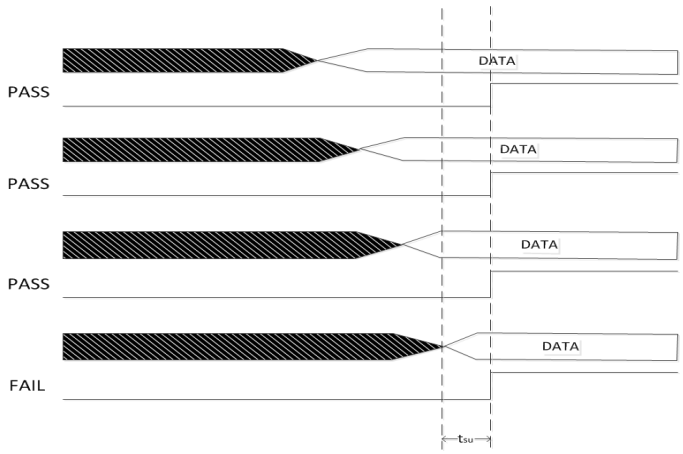

Figure 8: Set-up margin calculation by sweeping data transition edge close to clock edge

Using the similar approach, we started with higher value of hold time (t_{hold}) and reduced it in steps to bring data transition edge with in hold limit of clock edge. Also in this case the last passed simulation data gives us the approximate hold margin requirement of the peripheral. Figure 9 represents the actual simulation behaviour.

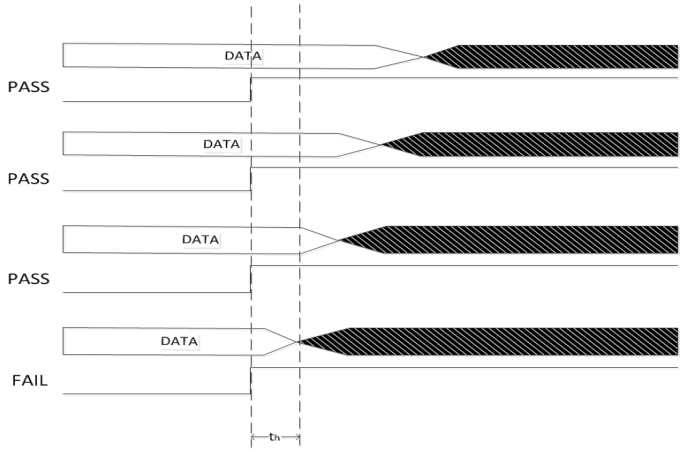

Figure 9: Hold margin calculation by sweeping data transition edge close to clock edge

Standard Delay Format (SDF) used for timing simulation doesn't capture derating information and hence we should take the non derated numbers from timing analysis for comparison. Setup and hold time margins (+ve/-ve timing slack) for SDRAM data input obtained from STA were 700ps and 0ps respectively. The numbers derived from simulation based characterization procedure matched STA numbers, which can be witnessed in table I and table II.

Table I. Simulation results for characterization to determine set-up margin

Stressing Input Data Setup Time			
Simulation	t_{setup}	t_{hold}	Status
Run0	**900**	1000	**Pass**
Run1	**800**	1000	**Pass**
Run2	**700**	1000	**Pass**
Run3	**600**	1000	**Fail**

Table III. Simulation results for characterization to determine hold margin

Stressing Input Data Hold Time			
Simulation	t_{setup}	t_{hold}	Status
Run0	7000	**100**	**Pass**
Run1	7000	**0**	**Pass**
Run2	7000	**-100**	**Pass**
Run3	7000	**-200**	**Fail**

The correctness of this method has also been checked on other peripherals in existing devices.

IX. CONCLUSION

The proposed method of simulation based characterization provides a vital way to double-check STA calculations and can be adopted for timing critical peripherals. This ensures verification of data sheet input switching characteristics in pre-silicon phase and helps in avoiding post-silicon surprises.