DesignWare Cores LPDDR54 PHY On-Chip Clocking Implementation using LPDDR5/4/4X PHY with internal PLL Application Note

OCC using LPDDR5/4/4X PHY PLL

Version: 1.10a
March 2021

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

Contents

Revision History 5

Reference Documents 7

1 Introduction 9
1.1 Purpose 9
1.2 Overview 9
1.2.1 Design Structure with LPDDR5/4/4X 9
1.2.2 Clocks Usage 11
1.2.3 OCC Controller Implementation Flow 12

2 PLL Set-up and Initialization 15
2.1 Reset sequence 15
2.2 PLL configuration 16
2.2.1 Process/Frequency independent PLL inputs 17
2.2.2 Process/Frequency dependent PLL inputs 18
2.2.3 Dynamic PLL inputs 18
2.3 PLL power-up sequence 19

3 OCC Insertion using DFT Compiler 21
3.1 Design Preparation 22
3.1.1 Clock Dividing Logic 22
3.1.2 Ports for OCC Control Signals 22
3.1.3 Create Buffers as OCC Controller Insertion Point 23
3.2 General DFT Configuration 24
3.3 OCC Controller Configuration 24
3.3.1 Define OCC controller signals 25
3.3.2 OCC Controller definition 26
3.3.3 PLL Control Bus Constraints 26
3.4 OCC insertion and Scan Enable connection 26
3.4.1 OCC Insertion 26
3.4.2 OCC controller Scan Enable Connectivity fix 27
3.5 Output files 27
3.6 Warnings Analysis 28

4 ATPG with OCC using Tetramax 29
4.1 ATPG DRC 29
4.1.1 SPF file generated by DFT Compiler 29
4.1.2 Tetramax commands 29
4.2 Pattern Generation 31
4.3 Gate Level Simulation 31
4.3.1 Example Waveform 31

Revision History

Revision	Date	Description
1.10a	March 2021	Updating “Clocks Usage”
1.00a	July 2020	Initial release

Reference Documents

The following reference documents are used in this application note:

- *SolvNetPlus article 000024322- DFT-inserted OCC controller Data Sheet*
- *TetraMAX ATPG and TetraMAX II ADV ATPG User Guide*
- *DFTMAX Design-For-Test user guide*
- *Designware Cores LPDDR5/4/4X PHY Implementation Guide*
- *DesignWare Cores LPDDR5/4/4X PHY Databook*

Note	This application note focuses on OCC implementation and testing using Synopsys tools · DFT Compiler- 2018.06-SP2 · TestMAX- 2019.12 · VCS- 2019.06-SP2

1 Introduction

1.1 Purpose

The purpose of this application note is to provide guidance for implementing On-Chip Clocking (OCC) using the LPDDR5/4/4X PHY IP PLL. This application note applies to all the process nodes for which the LPDDR5/4/4X PHY IP is available.

In this document users will find:

- Guidance on setting up the PHY PLL
- Guidance on OCC controller insertion
- Guidance on ATPG pattern generation and simulation with an implemented OCC controller.

The structure and detailed functionality of OCC controller inserted by DFT Compiler (DFTC) are not covered in this document.

For information regarding OCC controller itself, refer to:

- SolvNetPlus article 24322, "DFT-inserted OCC Controller Data Sheet"

For details on OCC implementation with DFT Compiler, refer to:

- On-Chip Clocking Support in DFTMAX Design-For-Test User Guide.

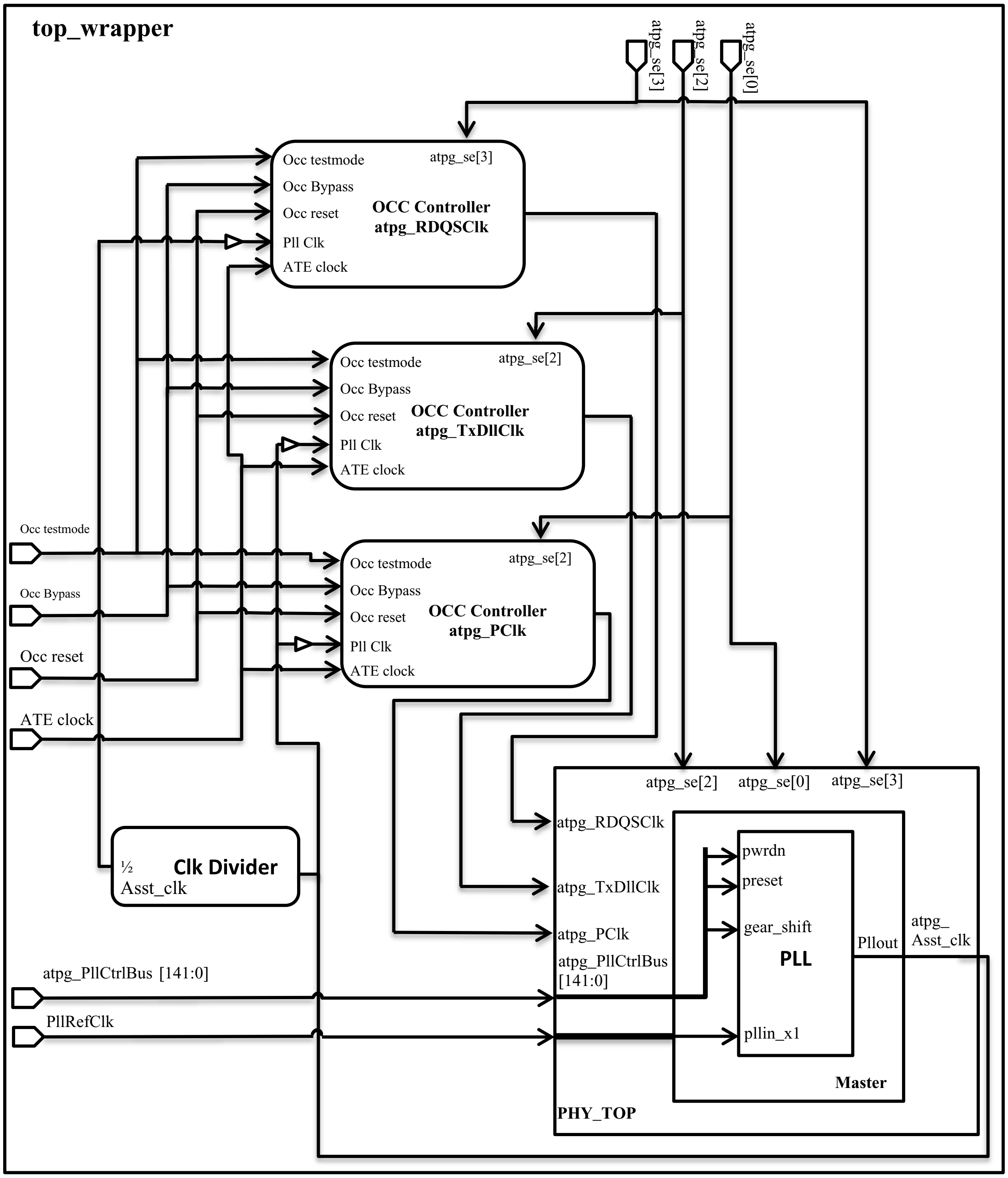
1.2 Overview

1.2.1 Design Structure with LPDDR5/4/4X

Figure 1.1 illustrates one of the possible ways of implementing OCC controller logic alongside LPDDR5/4/4X PHY_TOP design. Several key points are listed below:

- Unlike regular stuck-at ATPG, At Speed Scan Testing (ASST) requires higher clock rates to test transition faults. Chapter 2 describes how to connect and setup the PHY PLL to be the clock source for At-Speed Scan testing of the LPDDR5/4/4X PHY.
- The maximum achievable speed for atpg_Asst_Clk, atpg_PClk, atpg_TxDllClk and atpg_RDQSClk will depend directly on implementation. OCC controllers should be placed as close as possible to Hard-IP clock pins and customers should use a standard cell library which is optimized for speed.
- Clock tree synthesis (CTS) of high speed ATPG clocks will require special consideration on the placement, routing layer width, spacing and shielding.
- If minimum pulse width requirements for high frequency clocks cannot be met across PVT or can be achieved but at the cost of higher power and area, it is suggested to lower high speed ATPG clocks' frequency as the impact on At-Speed test coverage is minimal.
- LPDDR5/4/4X PHY includes scan enable signals (atpg_se[n:0]) dedicated to each clock domain described in Table 1.1. To support these independent clock domains, a single OCC controller per clock domain is required.

Figure 1.1: Example OCC Controller implementation



1.2.2 Clocks Usage

LPDDR5/4/4X PHY_TOP design provides multiple primary inputs for ATPG mode clocks which are listed in Table 1.1. This application note refers to OCC controller support for high speed clocks generated by the PLL located inside the PHY. The supported frequency ranges are included in LPDDR5/4/4X PHY databook.

Table 1.1: ATPG clocks in LPDDR5/4/4X

Name	Max Freq	Used by PUB	Used by Hard Macro	Used for ASST	Scan Enable
atpg_PClk	DATA_RATE	No	Yes	Yes	atpg_se[0]
DfiClk	DfiClk	Yes	Yes	No	atpg_se[1]
atpg_TxDllClk	DATA_RATE	No	Yes	Yes	atpg_se[2]
atpg_RDQSClk	DATA_RATE/2	No	Yes	Yes	atpg_se[3]
atpg_DlyTestClk	DfiClk	No	Yes	No	atpg_se[4]
APBCLK	APB	Yes	No	No	atpg_se[5]
atpg_UcCLK	DfiClk/2	Yes	No	No	atpg_se[6]
TDRCLK	TDR	Yes	No	No	atpg_se[7]
ATE clock (ATPG Shift clock)	100 MHz				

During At-Speed Scan Testing (ASST) the output of the PLL (which is located inside MASTER Hard-IP) is connected to atpg_Asst_Clk output port of LPDDR5/4/4X PHY TOP. The atpg_Asst_Clk or a divided down version of the clock is expected to be used as the clock source for each OCC controller.

The inserted OCC controllers select between ATE clock and PLL clock based on scan shift or scan capture operation; ATE clock is used during scan shift (100 MHz) and PLL clock is used during scan capture. The output of the OCC controllers is used to drive LPDDR5/4/4X PHY high speed clocks during ATPG.

The at-speed (scan capture) frequency of the clocks is determined based on customers' implementation and does not have to be the maximum supported frequency by the PHY.

Fast Atpg_* clock trees should be implemented with buffers/inverters from standard cell library which are optimized for slew. Shielding and non-default routing for clock nets should be used in order to prevent crosstalk. Net length should be balanced from Root to Sink. Special attention should be paid to the power grid used for buffers and inverters that are part of the clock tree to control Dynamic IR drop.

The following example implementation describes support for two different frequencies during ASST:

- MEMCLK: atpg_PClk, atpg_TxDllClk
- MEMCLK/2: atpg_RDQSClk

In this case, clock dividers should be inserted to generate half frequency clock from atpg_Asst_Clk.

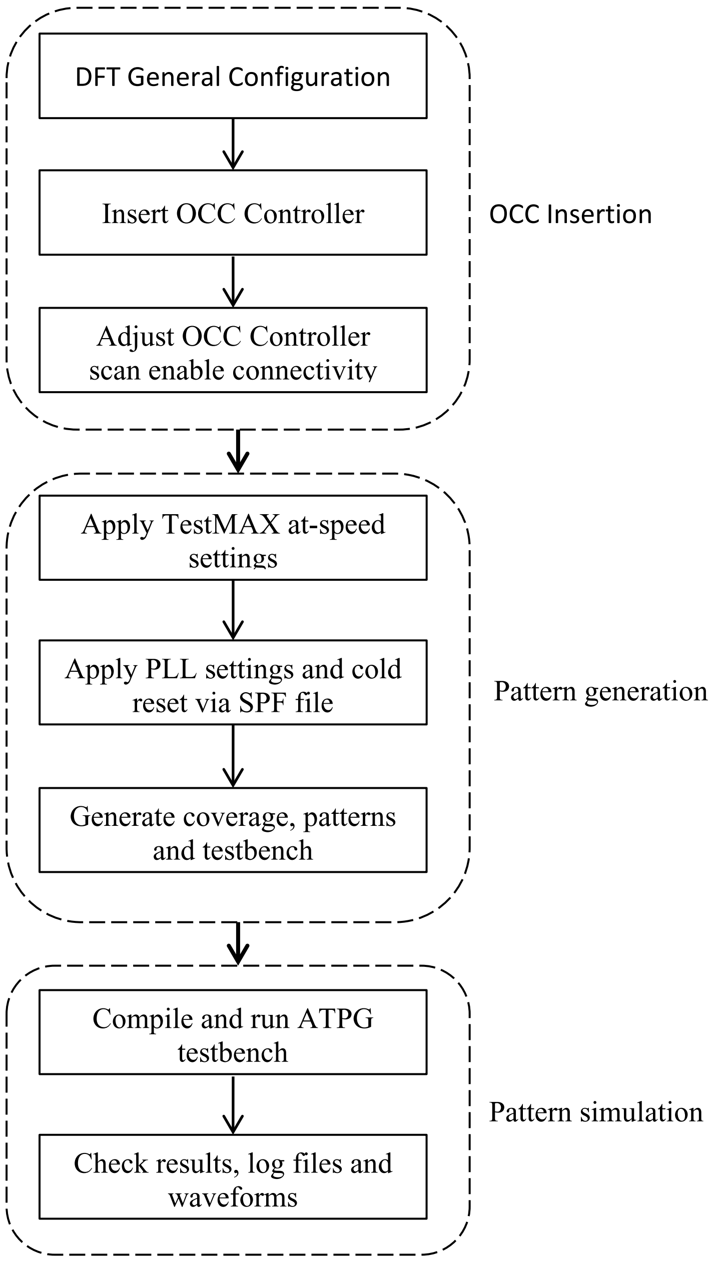
1.2.3 OCC Controller Implementation Flow

The steps which are part of the OCC implementation flow can be integrated into any regular DFT flow by including a few additional settings.

The complete process (OCC + DFT + ATPG) can be divided into 3 major sections:

1. OCC controller insertion using DFT compiler
2. Pattern generation flow using TestMAX
3. Pattern simulation flow using VCS

Figure 1.2: OCC insertion flow


DFT General Configuration
Insert OCC Controller
Adjust OCC Controller scan enable connectivity
OCC Insertion
Apply TestMAX at-speed settings
Apply PLL settings and cold reset via SPF file
Generate coverage, patterns and testbench
Pattern generation
Compile and run ATPG testbench
Check results, log files and waveforms
Pattern simulation

2 PLL Set-up and Initialization

During At-Speed Scan Testing, the PLL should be powered up and configured to the highest functional frequency at which the LPDDR5/4/4X PHY is implemented. To obtain control over the PLL using PHY_TOP atpg_PllCtrlBus[141:0] port, atpg_mode and atpg_Asst_Clken TOP level ports should be HIGH.

The PLL initialization sequence can be divided into following steps:

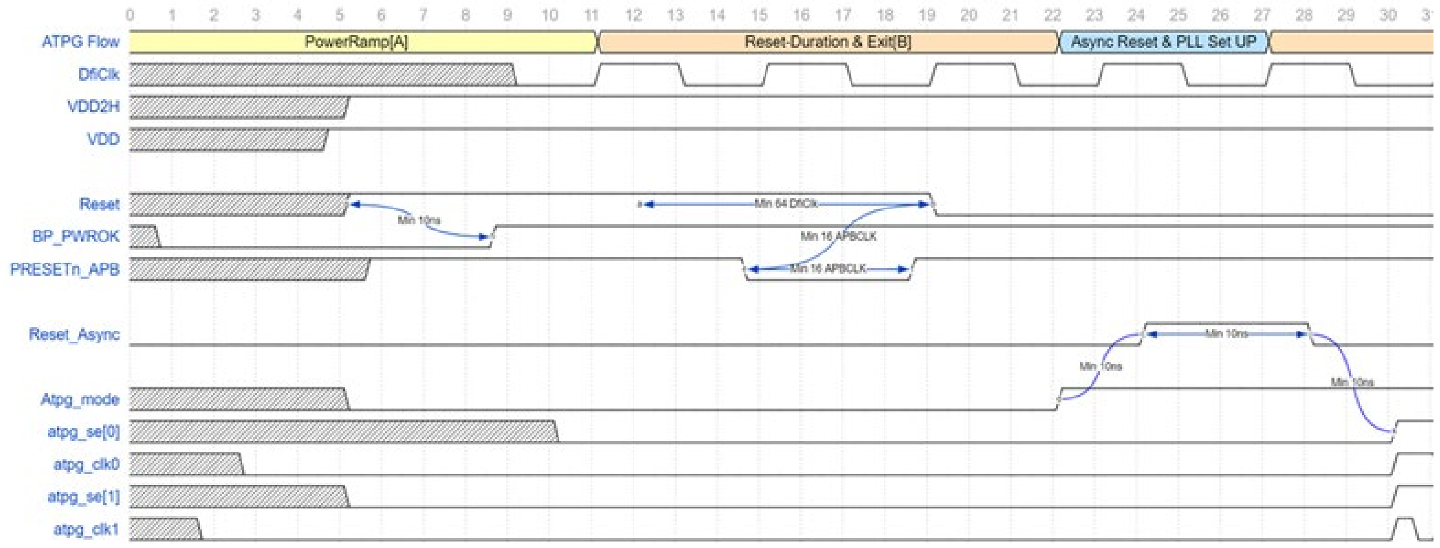
- Reset sequence- described in Section 2.1
- PLL configuration to generate mission mode frequency- described in Section 2.2
- PLL power-up sequence- described in Section 2.3

More information about PLL can also be obtained from DesignWare Cores LPDDR5/4/4X PHY Databook.

2.1 Reset sequence

- Generate DfiClk and PllRefClk (*pllin_x1* signal in Figure 2.2)
- Assert Reset
- Assert BP_PWROK 10ns after Reset has been asserted (BP_PWROK can also be held permanently high)
- De-assert PRESETn_APB at least 44 DfiClk cycles after BP_PWROK has been asserted
- Keep PRESETn_APB de-asserted at least 16 DfiClk cycles
- De-assert Reset at least 4 DfiClk cycles after PRESETn_APB has been asserted
- Set static PLL inputs- see Section 2.2
- Assert atpg_mode and atpg_Asst_Clken to enable PLL control via atpg_PllCtrlBus[]

Figure 2.1: Cold Reset sequence



2.2 PLL configuration

PLL inputs are grouped into 3 categories:

- Process/Frequency independent PLL inputs
- Process/Frequency dependent PLL inputs
- Dynamic PLL inputs

2.2.1 Process/Frequency independent PLL inputs

Inputs in Table 2.1 are static and should be set to a fixed value before atpg_mode is asserted HIGH.

Table 2.1: Process/Frequency independent PLL inputs

PHY_TOP atpg_PllCtrlBus[]	PLL Signal	Recommended setting
atpg_PllCtrlBus[1]	byp_mode	1’b0
atpg_PllCtrlBus[20]	x4_mode	1’b0
atpg_PllCtrlBus[21]	standby	1’b0
atpg_PllCtrlBus[40:36]	dacval_in[4:0]	5’b0
atpg_PllCtrlBus[41]	force_cal	1’b0
atpg_PllCtrlBus[42]	en_cal	1’b0
atpg_PllCtrlBus[43]	lock_count_sel	1’b0
atpg_PllCtrlBus[45:44]	lock_phase_sel[1:0]	2’b01
atpg_PllCtrlBus[64:60]	maxrange[4:0]	5’b1_1111
atpg_PllCtrlBus[65]	pll_ana_test_en	1’b0
atpg_PllCtrlBus[71:66]	pll_ana_test_sel[5:0]	6’b00_0000
atpg_PllCtrlBus[77:72]	pll_dig_test_sel[5:0]	6’b00_0000
atpg_PllCtrlBus[141:78]	upll_prog[63:0]	upll_prog<23:0>-2531’h upll_prog<24> 0'h. upll_prog<35:25>-18C’h upll_prog<58:36>-0’h upll_prog<63:59>-0’h

2.2.2 Process/Frequency dependent PLL inputs

PLL inputs in Table 2.2 are static and should be set to a fixed value before atpg_mode is asserted HIGH.

Refer to PHY Databook section 2.1.12.1 (Optimal PLL Settings) for required value.

Table 2.2: Process/Frequency dependent PLL inputs

PHY_TOP atpg_PllCtrlBus[]	PLL Signal	Recommended setting
atpg_PllCtrlBus[13:4]	div_sel[9:0]	Frequency dependent
atpg_PllCtrlBus[16:14]	v2i_mode[2:0]	Frequency dependent
atpg_PllCtrlBus[19:17]	vco_low_freq[2:0]	Frequency dependent
atpg_PllCtrlBus[28:22]	cp_int_ctrl[6:0]	Process dependent
atpg_PllCtrlBus[35:29]	cp_prop_ctrl[6:0]	Process dependent
atpg_PllCtrlBus[52:46]	cp_int_gs_cntrl[6:0]	Process dependent
atpg_PllCtrlBus[59:53]	cp_prop_gs_cntrl[6:0]	Process dependent

2.2.3 Dynamic PLL inputs

PLL inputs in Table 2.3 are needed for PLL initialization, so they should be dynamically changed during PLL power-up sequence. After PLL is up, these inputs should be set to a fixed value of 0.

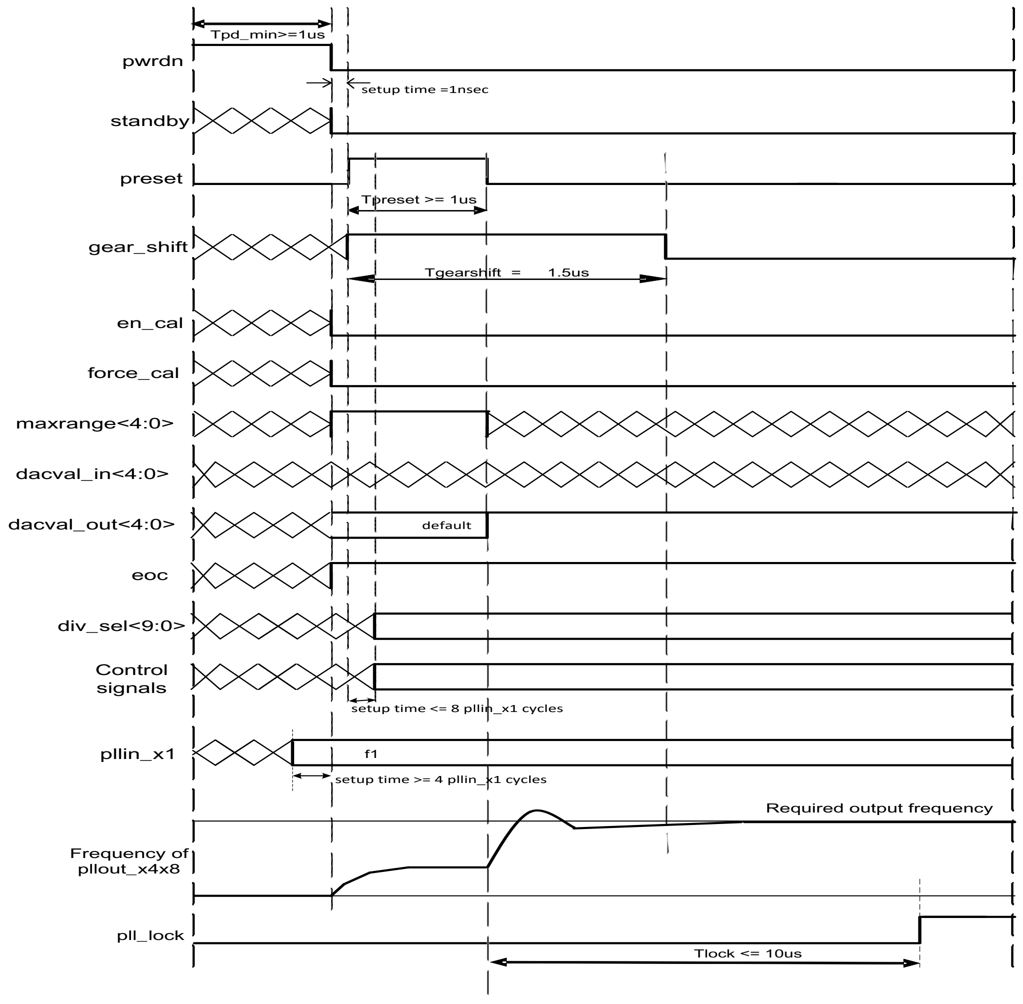
Table 2.3: Dynamic PLL inputs

PHY_TOP atpg_PllCtrlBus[]	PLL Signal	Recommended setting
atpg_PllCtrlBus[0]	preset	Controls PLL power-up sequence
atpg_PllCtrlBus[2]	pwrdn	Controls PLL power-up sequence
atpg_PllCtrlBus[3]	gear_shift	Controls PLL power-up sequence

2.3 PLL power-up sequence

- Assert *pwrdn* (atpg_PllCtrlBus[2]) for minimum time 1us.
- De-assert *pwrdn*, wait 1ns and assert *preset* (atpg_PllCtrlBus[3]) and *gear_shift* (atpg_PllCtrlBus[0]).
- *preset* should be kept asserted for 1.0us
- De-assert *preset*, while keeping *gear_shift* HIGH for another 0.5us
- By de-asserting *gear_shift* the PLL is switched to final bandwidth
- Maximum time for the PLL to lock is 10us
- After the PLL is locked, Reset_async should be asserted for 10ns

Figure 2.2: Initial PLL power-up sequence



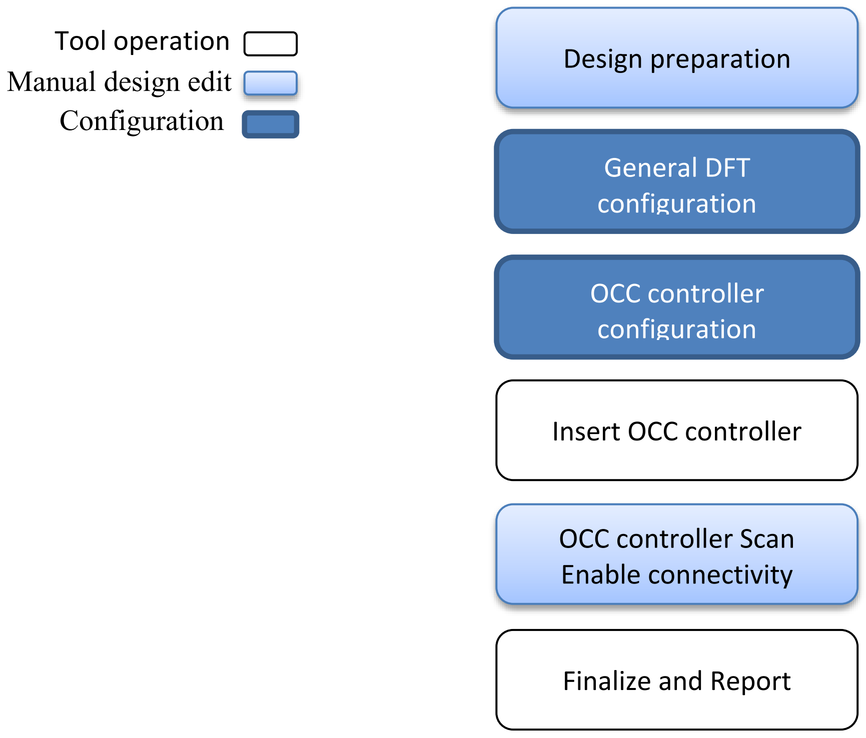
3 OCC Insertion using DFT Compiler

OCC controller insertion requires a netlist which is ready for scan chain insertion. The OCC controller insertion can be either integrated into the main DFT step or can be done separately on its own.

For general LPDDR5/4/4X DFT guidelines, refer to Chapter 7 of DesignWare Cores LPDDR5/4/4X PHY Implementation Guide.

Figure 3.1 provides general overview of OCC controller insertion flow.

Figure 3.1: OCC Controller Insertion Flow



3.1 Design Preparation

There are 3 main steps which need to be completed before OCC controller insertion:

- add clock dividing logic to generate atpg_RDQSClk from atpg_Asst_Clk
- add ports for OCC control signals
- add clock buffers used as OCC controller insert points

Note	All design modifications introduced in this section should be done before `creat_test_protocol` or `read_test_protocol` commands to avoid the creation of invalid test protocol. It is suggested to edit the design before any DFT operation.

3.1.1 Clock Dividing Logic

Clock dividing logic must be instantiated in the design to generate half speed clock out of atpg_Asst_Clk. To avoid malfunction of clock dividing logic, note:

- All flip-flops in clock dividing logic must remain as non-scan cells
- The reset pin of clock dividing logic should be connected to the signal used to drive Reset_async port of LPDDR5/4/4X PHY

3.1.2 Ports for OCC Control Signals

Table 3.1 lists all I/O ports that are necessary for OCC control.

Table 3.1: OCC control signals

Port name	I/O	Description
ate_clk	I	Slow test clock from ATE (100 MHz; used for scan shift)
occ_testmode	I	Test mode control signal for OCC controller
occ_bypass	I	Bypass control signal for OCC controller
occ_reset	I	Reset signal for OCC controller

To create ports listed in Table 3.1 (if they do not already exist) customers can use

```
create_port -direction “in” <port_name>
```

Note	A dedicated TestMode port must be created for OCC controller. It must be active in all test modes and inactive in mission mode. It cannot be shared with TestMode signals used for other purposes, such as AutoFix or multiple test-mode selection.

3.1.3 Create Buffers as OCC Controller Insertion Point

OCC controller is inserted at the source of the PLL clock. If several clock domains share the same PLL clock source and separated OCC controllers are required, buffers must be inserted to separate the insertion points of OCC controller for each clock domain.

For example, atpg_Asst_Clk is used as the PLL clock source for both atpg_PClk and atpg_TxDllClk, making it impossible to create sperate OCC controllers for these two clock domains. In this case, to enable the insertion of different OCC controllers which utilize the same PLL output, customers can:

1. insert 2 additional buffers, *OCC_BUF_atpg_PClk* and *OCC_BUF_atpg_TxDllClk*, which are to be connected directly to PHY_TOP output atpg_Asst_Clk
2. define the outputs of *OCC_BUF_atpg_PClk* and *OCC_BUF_atpg_TxDllClk* as the PLL clock source instead of atpg_Asst_Clk.

By doing so, DFT compiler will insert separate OCC controllers for atpg_PClk and atpg_TxDllClk during 'insert_dft'.

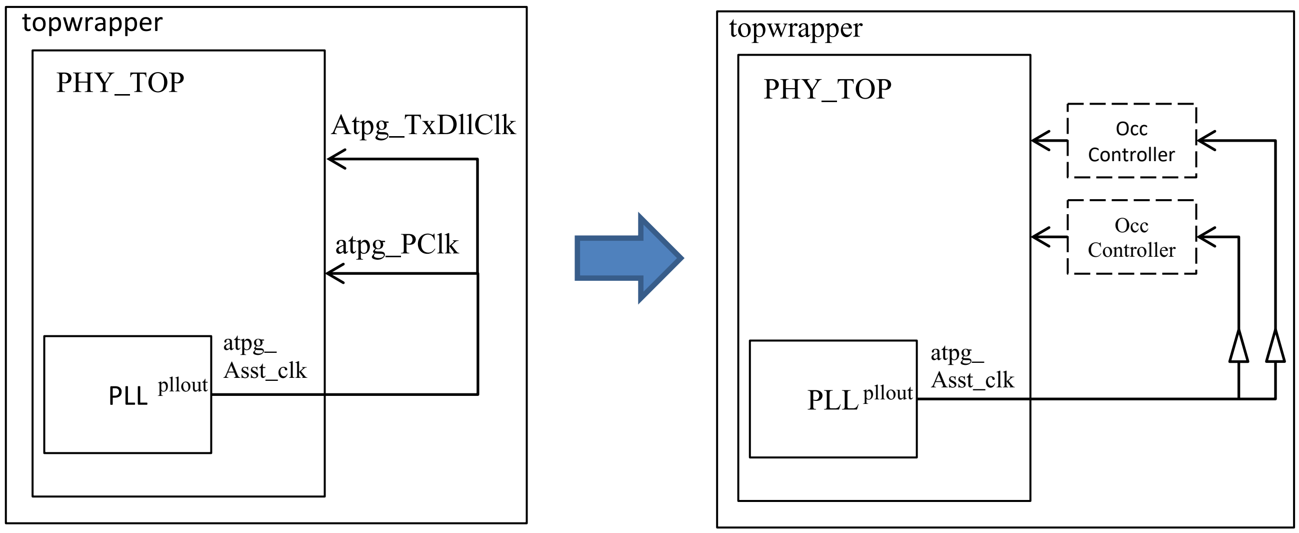


Table 3.2 shows all necessary buffers for occ controller insertion. All names in Table 3.1 and 3.2 are provided as examples. Buffer and port names can be tailored to fit customers' design naming convention.

Table 3.2: Buffer for OCC Controller Insertion

Instance name	Type	Description
OCC_BUF_atpg_PClk	Buffer	A buffer used as OCC controller insert point for atpg_Pclk
OCC_BUF_atpg_TxDllClk	Buffer	A buffer used as OCC controller insert point for atpg_TxDllClk
OCC_BUF_atpg_RDQSClk	Buffer	A buffer used as OCC controller insert point for atpg_RDQSClk

`create_cell` and `connect_pin` can be used to create buffers and their connection, e.g.

```
create_cell OCC_BUF_atpg_PClk $tech_lib/$clock_buf
connect_pin -from [get_pins dwc_ddrphy_top_inst_0/atpg_Asst_Clk] -to
[get_pins OCC_BUF_atpg_Pclk/A]
connect_pin -from [get_pins OCC_BUF_atpg_PClk/X] -to
[get_pins dwc_ddrphy_top_inst_0/atpg_PClk] -port_name atpg_PClk
```

3.2 General DFT Configuration

To perform On-Chip Clocking controller insertion, the `-clock_controller` option of `set_dft_configuration` should be enabled when configuring DFT and scan settings:

```
set_dft_configuration      -clock_controller enable
set_scan_configuration     -style multiplexed_flip_flop -clock_mixing no_mix
```

Note	If the current design is to be used as a test model later in a hierarchical flow, it is important to avoid clock mixing. Clock mixing can cause the clock chain part of the On-Chip Clock controller to mix with flip-flops of opposite polarity on a single chain. As a result, the mixed scan chain cannot be combined with scan chains from other test models and the maximum scan chain count at the top level would be increased.

Scan chains and DFT signals including clocks, resets and scan constants can also be defined as part of the General DFT Configuration step.

3.3 OCC Controller Configuration

Configuring OCC controller can be separated into three main parts:

- Configure OCC controller signals including clocks and control ports
- Define OCC controllers
- Constraint PLL control bus to its required value

3.3.1 Define OCC controller signals

Three types of clocks should be defined during On-Chip Clocking implementation via `set_dft_signal`:

- ATE clock

  ATE clock should be defined as both `-type Oscillator` and `-type ScanClock` on ATE clock input port
- Reference clock

  LPDDR5/4/4X PHY uses PllRefClk as PLL reference clock. It should be defined as `-type PlRefClk` and its frequency should be based on customer's requirements
- PLL clock

  PLL clock sources are used as OCC insertion points. They should be defined at the output of inserted OCC buffers described in section 3.1.3 with `-type Oscillator` and `-hookup_pin`.

Note	The reference clock can only be defined as `-type refclock` when its period is different from default.

Example command usage

```
# ATE clock define
set_dft_signal -view exist -type Oscillator -port ate_clk -test_mode all'
set_dft_signal -view exist -type ScanClock -port ate_clk -timing [list [expr double
(2.8)] [expr double (7.8)]]
# PLL reference define
set_dft_signal -view exist -type refclock -port PllRefClk -period $customer_period
# PLL clock define
set_dft_signal   -view   exist   -type   Oscillator   -hookup_pin   [get_pins
OCC_BUF_atpg_PClk/X]
```

OCC controller insertion requires the definition of the following three OCC control signals.

- occ reset
- occ bypass enable
- occ test mode enable.

Following TCL command give an example of defining these control signals:

```
set_dft_signal -view spec -type TestMode -port occ_testmode
set_dft_signal -view spec -type pll_bypass -port occ_bypass
set_dft_signal -view spec -type pll_reset -port occ_reset
```

3.3.2 OCC Controller definition

The OCC controller can be defined with `set_dft_clock_controller` and should follow the definition of OCC clocks and control signals.

PHY_TOP ATPG test coverage experiments show that a 2-clock cycle capture can be used for all OCC to obtain maximum coverage. The number of clock chain can be a customized number. Configuration example:

```
set_dft_clock_controller -cell_name snps_pll_controller \
           -design snps_clk_mux -cycles_per_clock 2 -test_mode occ_testmode \
           -pllclocks $inter_clk -ateclocks ate_clk -chain_count 1
```

3.3.3 PLL Control Bus Constraints

During ASST ATPG, the PLL inside PHY_TOP MASTER Hard-IP can be brought up to generate functional clock frequencies by TOP level atpg_PllCtrlBus[*] ports (atpg_mode and atpg_Asst_Clken are required to be set to 1).

During OCC controller insertion in DFT Compiler, atpg_PllCtrlBus[*] should be constrained to values, specified in Section 2 to ensure stable at-speed clock can be output from the PLL.

Constraints on ports can be implemented with
`set_dft_signal -type Constant`

3.4 OCC insertion and Scan Enable connection

3.4.1 OCC Insertion

After configuring DFT and OCC controller settings, the OCC controller insertion process follows regular test insertion methodology:

create_test_protocol → preview_dft → dft_drc → insert_dft → dft_drc

After post-dft drc, PLL bypassed DRC can be run by enabling `-pll_bypass` option of
`set_dft_drc_configuration:`

```
set_dft_drc_configuration -pll_bypass enable
dft_drc -verbose -coverage_estimate
```

After running DRC with bypassed PLL, the SPF file written in next step can be utilized in both PLL enable mode and PLL bypass mode.

3.4.2 OCC controller Scan Enable Connectivity fix

All scan chains in LPDDR5/4/4X PHY are clock-domain-based and enabled by associated scan enable signals (see Table 1.1).

However, there is no option or setting through which DFT Compiler can be guided to use different scan enable signals for the different OCC controllers. When defining multiple scan enable signals, the last defined scan enable will be connected to all OCC controller in the design.

Scan enable connection to OCC controllers can be adjusted once final `insert_dft` has completed. This can be achieved by using `disconnect_net` and `connect_pin` commands.

It should be noted that changing controller SE connectivity using TCL commands does not affect the test protocol stored inside DFT Compiler internal memory. Incorrect scan enable connections would be output even if test protocol is written out after signal reconnection post `insert_dft`. This issue can only be fixed by editing `ScanEnable` specification for each clock chain block in the test protocol file manually.

Note	To enable easier pattern generation in TMAX, customers can use the modified SPF generated by DFT compiler. The SPF file can be used in ATPG DRC, thus allowing customers to avoid discrepancies between signal definition used in DFT Compiler and TMAX. Example command: `write_test_protocol -output <dir>/${design}_scan.spf`

3.5 Output files

After reconnecting scan enable, to facilitate debug and further analysis, the following files should be generated:

- scan inserted netlist
- DDC file
- CTL file
- scan reports

3.6 Warnings Analysis

All clock dividing logic and some of the OCC controller inserted logic are clock related cells which cannot be stitched into a scan chain. These cells will cause some DRC violations in the post DFT DRC step.

Table 3.3 lists all expected violations during DFT DRC.

Table 3.3: Expected violations

Violation Type	DRC Stage	Reported Cell	Reason to Waive
C16	Post DFT DRC	FFs in OCC controller	All clock dividing cells should remain as non-scan. It is excepted that there are violations identifying that these cells cannot capture data.
C17	Post DFT DRC	Clock connected to primary output	A clock should not have a path to a primary output consisting only of combinational gates. A violation of this rule introduces no danger of generating bad patterns.
S19	Post DFT DRC	FFs in OCC controller	These nonscan cells are not designed to be stable during whole load_unload. These violations should be expected.
S29	Post DFT DRC	Dependent slave	These violations are expected because in each OCC controller contains fast and slow clock

4 ATPG with OCC using Tetramax

Pattern generation and simulation for a design with inserted On-Chip Clocking (OCC) is similar to regular ATPG flow:

ATPG Build → ATPG DRC → Pattern Generation → Pattern Simulation

4.1 ATPG DRC

There are two approaches which can be adopted to run ATPG Design Rule Check (DRC) in Tetramax (TMAX):

- Using SPF from DFT Compiler (DFTC)
- Using TMAX commands

4.1.1 SPF file generated by DFT Compiler

PLL initialization sequence, introduced in Chapter 2, should be inserted in the SPF generated from DFTC.

SPF modification requires insertion of PLL initialization sequence described in Section 3.1- once SPF has been modified, it should be run through Tetramax and all Warnings should be carefully analyzed.

Note	PLL initialization is inserted as part of the `test_setup` macro. The final vector in `test_setup` macro should always assert ports to their constrained value. Therefore, the PLL initialization sequence should be inserted in the middle of `test_setup`, after the first condition vector and before final assertion vector.

4.1.2 Tetramax commands

All DFT signals and OCC controller related information can be defined using TCL commands.

In addition, a pre-shift vector must be inserted in the Tetramax SPF `load_unload` macro before `shift` to avoid S1 (scan chain blocking) issues in ATPG DRC. This is due to the synchronization mechanism of OCC controller.

The whole DRC process can be divided into following steps:

1. Define general DFT signals including clocks, resets and scan enables; note that ATE clock and PLL reference clock should be defined with `add_clocks -refclock`
2. Define scan chains.
3. Define general DRC settings and set maximum number of capture clocks during shift with

4. set_drc -num_pll_cycle <n>
5. Define OCC controller with `add_clocks`, for example:

```
add_clocks 0 snps_pll_controller_atpg_Pclk/U2/U2/X -intclock \
   a. -pll_source OCC_INBUF_DfiCtlClk/X -cycle { \
   b. 0 snps_clk_chain_0/clk_ctrl_data[0] 1 \
   c. 1 snps_clk_chain_0/clk_ctrl_data[1] 1 }
```

6. Write out SPF file of current settings.
7. Insert PLL initialization sequence described in Section 3.1 into the SPF from previous step.
8. Insert "pre-shift" vector in `load_unload` macro before shift.
9. Run DRC with the edited SPF file.

For more operation details on defining OCC controller in Tetramax, refer to chapter 15 in
TetraMAX ATPG and TetraMAX II ADV ATPG User Guide.
The pre-shift vector in step 8 asserts all scan enable signals in design and toggles ATE clocks for one cycle.
The procedure allows correct slow clock functionality.
An example pre-shift vector:

```
"Internal_scan_pre_shift" : V {
        "_clk" = 00P0;
       "atpg_se[0]" = 1;
        "atpg_se[1]" = 1;
        "atpg_se[2]" = 1;
        "atpg_se[3]" = 1;
        "atpg_se[4]" = 1;
        "atpg_se[5]" = 1;
}
```

4.2 Pattern Generation

During LPDDR5/4/4X PHY ATPG, at-speed test patterns should be generated per clock domain. This methodology avoids timing issues caused by cross clock domain paths.

Example pattern generation script for atpg_PClk clock domain:

```
remove_faults -all
add_faults -launch $clock -capture $clock
run_atpg  -auto_compression
write_patterns ${PATTERNS_DIR}/${TOP}_atspeed_${clock}.stil.gz -internal \
       -format stil -compress gzip -replace \
       -nopatinfo -nocore -vcs -parallel
write_testbench -input ${PATTERNS_DIR}/${TOP}_atspeed_${clock}.stil.gz \
       -output ${PATTERNS_DIR}/${TOP}_atspeed_${clock}_tb -replace
write_faults ${FAULTS_DIR}/${TOP}_atspeed_${clock}_faults.rpt -all -replace
```

4.3 Gate Level Simulation

The pattern simulation steps for ATPG patterns with OCC are identical with regular ATPG pattern simulation steps.

4.3.1 Example Waveform

OCC controller signals during an ATPG pattern are shown in Figure 4-1 and Figure 4-2.

During scan shifting, the design uses slow clock to operate; during scan capture the clock output of the OCC controller is the fast clock which allows the design to operate at functional frequency.

Figure 4.1: Slow clock operation

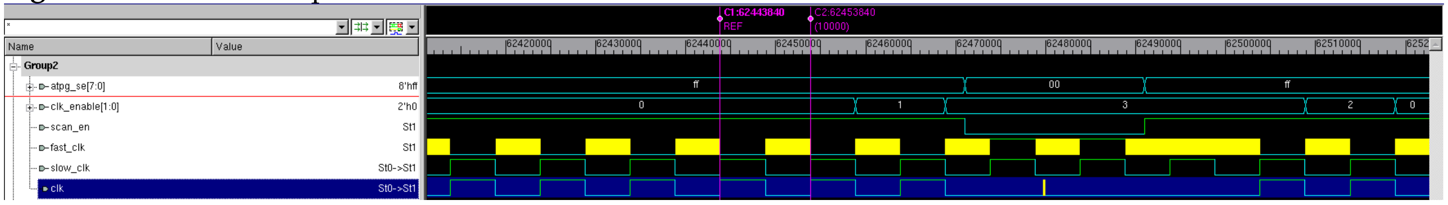
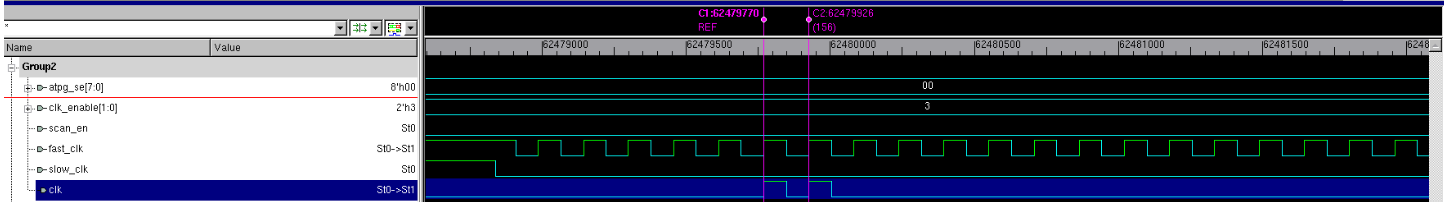


Figure 4.2: Fast clock operation


C1:62479770
REF
C2:62479926
(156)
Name
Value
Group2
atpg_se[7:0] 8'h00
clk_enable[1:0] 2'h3
scan_en St0
fast_clk St0->St1
slow_clk St0
clk St0->St1
00
3