Clk\_Inst

Integration Guide

IP Rev. 0.8

Intel Restricted Secret

Copyright © 2014, Intel Corporation. All rights reserved.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

\* Other names and brands may be claimed as the property of others.

This document contains information on products in the design phase of development.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED OR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked “reserved” or “undefined.” Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your Intel account manager or distributor to obtain the latest specifications and before placing your product order.

Copies of documents that have an order number and are referenced in this document or in other Intel literature can be obtained from your Intel account manager or distributor.

Contents

[1 Introduction 7](#_Toc425848163)

[1.1 Audience 7](#_Toc425848164)

[1.2 Supported Projects 7](#_Toc425848165)

[1.3 Terminology 7](#_Toc425848166)

[1.4 Related Documents 7](#_Toc425848167)

[1.5 Opens, Risks, and Assumptions 8](#_Toc425848168)

[1.6 Contact Information 8](#_Toc425848169)

[1.7 Document Revision History 8](#_Toc425848170)

[2 Quick Start 10](#_Toc425848171)

[2.1 Downloading Sub IP 10](#_Toc425848172)

[2.2 Integrity Checks for Standalone IP 10](#_Toc425848173)

[3 Quick Start 11](#_Toc425848174)

[3.1 IP Block Diagram 11](#_Toc425848175)

[3.2 Functional Top-Level Signals 11](#_Toc425848176)

[3.3 Timing Diagram 13](#_Toc425848177)

[3.4 Frequency changes 14](#_Toc425848178)

[3.5 clkreq / clkack interface support 15](#_Toc425848179)

[3.6 Clock edge jitter on the gal edge 15](#_Toc425848180)

[3.7 Enabling xclk[n] and xclk\_usync[n] outputs 15](#_Toc425848181)

[4 Design Information for Integration 17](#_Toc425848182)

[4.1 RTL Directory Structure 17](#_Toc425848183)

[4.2 Clock, Power and Reset Domains 17](#_Toc425848184)

[4.2.1 Clock Domain Diagram 17](#_Toc425848185)

[4.3 Embedded Building Blocks/Custom Logic 17](#_Toc425848186)

[4.4 RTL Configuration Parameters 17](#_Toc425848187)

[4.4.1 Mandatory Parameters 17](#_Toc425848188)

[4.4.2 Boundary Scan Parameters 17](#_Toc425848189)

[4.4.3 Test Data Register Parameters 18](#_Toc425848190)

[4.5 Testbench Parameters 18](#_Toc425848191)

[4.6 IP Straps 18](#_Toc425848192)

[4.7 Fuses 18](#_Toc425848193)

[4.8 Power Information 18](#_Toc425848194)

[4.8.1 Power Supply 18](#_Toc425848195)

[4.8.2 Static Clock Gating 18](#_Toc425848196)

[4.8.3 Power Gating 18](#_Toc425848197)

[4.8.4 Bumps and Their Power Domains 19](#_Toc425848198)

[4.9 Power-up Requirements 19](#_Toc425848199)

[4.10 Macros used by IP 19](#_Toc425848200)

[4.11 Other Design Considerations 19](#_Toc425848201)

[4.12 DFx Considerations 19](#_Toc425848202)

[4.12.1 DFx Top-Level Signals 19](#_Toc425848203)

[4.12.2 DFx Clock Definition 19](#_Toc425848204)

[4.12.3 Clock Crossings 19](#_Toc425848205)

[4.12.4 N/ADebug Registers 19](#_Toc425848206)

[4.12.5 Scan – Clock Gating in RTL 19](#_Toc425848207)

[4.12.6 Scan – Reset Override 19](#_Toc425848208)

[4.12.7 TAP and Associated Registers 19](#_Toc425848209)

[4.13 System Startup 19](#_Toc425848210)

[4.13.1 Power-up Sequence 19](#_Toc425848211)

[4.13.2 Initialization Sequence 20](#_Toc425848212)

[4.13.3 Device Configuration 20](#_Toc425848213)

[4.13.4 Header for Windows Boot 20](#_Toc425848214)

[4.14 Security Considerations 20](#_Toc425848215)

[4.14.1 Security Threats 20](#_Toc425848216)

[4.14.2 Security Tests 20](#_Toc425848217)

[4.14.3 Interface Signals Implemented for Security 20](#_Toc425848218)

[4.15 RTL Design Libraries 20](#_Toc425848219)

[4.16 RTL Uniquification 20](#_Toc425848220)

[4.17 Emulation Support 20](#_Toc425848221)

[5 Verification Information for Integration 21](#_Toc425848222)

[5.1 IP Testbench Overview 21](#_Toc425848223)

[5.2 IP does not use any unit level test bench or test island.Reusable IP Testbench Components 21](#_Toc425848224)

[5.2.1 Test Island 21](#_Toc425848225)

[5.2.2 Collage or Sandbox Files 21](#_Toc425848226)

[5.2.3 IP Environment 23](#_Toc425848227)

[5.2.4 N/A. IP does not has any fuse requirements.Sequences 24](#_Toc425848228)

[5.2.5 Ip does not used any sequences or extended sequences.Miscellaneous 25](#_Toc425848229)

[5.3 Environment Settings and Files 25](#_Toc425848230)

[5.3.1 Base Test 25](#_Toc425848231)

[5.3.2 N/AConfiguration Object 25](#_Toc425848232)

[5.3.3 API 26](#_Toc425848233)

[5.4 N/ADescription of Reusable Tests 26](#_Toc425848234)

[5.5 Description of Reusable Automation Scripts 26](#_Toc425848235)

[5.6 N/ASupported Compiler Options for Simulation 26](#_Toc425848236)

[5.7 Reusable Simulation RUNMODEs 26](#_Toc425848237)

[5.8 RTL Verification Libraries 26](#_Toc425848238)

[6 Tools and Methodology for Integration 27](#_Toc425848239)

[6.1 Supported Tools 27](#_Toc425848240)

[6.2 Environment Variables 27](#_Toc425848241)

[6.3 HIP Libraries Included in Release 27](#_Toc425848242)

[6.3.1 Register Files or SRAM 27](#_Toc425848243)

[6.3.2 M-PHY and Related Libraries 27](#_Toc425848244)

[6.4 Directory Structure 28](#_Toc425848245)

[6.5 Ace 28](#_Toc425848246)

[6.6 Lintra 28](#_Toc425848247)

[6.7 Synthesis 28](#_Toc425848248)

[6.7.1 Clocks 28](#_Toc425848249)

[6.7.2 Clock Diagram 29](#_Toc425848250)

[6.7.3 Constraint Files 29](#_Toc425848251)

[6.7.4 Scan Insertion 29](#_Toc425848252)

[6.8 Formal Verification 29](#_Toc425848253)

[6.9 CDC 29](#_Toc425848254)

[7 Physical Integration 30](#_Toc425848255)

[8 Integration Test Plan 31](#_Toc425848256)

[9 Appendix 32](#_Toc425848257)

[9.1 Subsystem connectivity details 32](#_Toc425848258)

About This Template

How to Use This Template

Do not remove any headings from this document. If you do not need the headings to describe your IP, enter “Not applicable” under the heading. This lets the reader know that you did not overlook this topic.

In the main document that follows, add new headings that you need to fully describe the integration of this IP. Add them in the appropriate chapters.

Most red text in this document contains instructions for filling out the section where it appears. The tag for most of this red text is called “Gaps.” You should replace this text with the content appropriate for that section, ensuring that the text is tagged appropriately (for example, with the BodyText or List Bullet style). If a section is not relevant, do not remove it; instead just replace the “Gap” text with “Not applicable” and apply the BodyText style.

Goal of This Document

This document should contain all information an integration team would need to accomplish the task without needing to seek help from another source. Try not to refer to other documents for required information; do so only if you include specific instructions for obtaining those documents, and only if you are sure your audience has access to them. Verify all links. This should be a self-contained guide for integration.

# Introduction

## Audience

The information in this document is intended for:

* Subsystem verification teams currently using their own copy of clk\_inst.
* Subsystem verification teams desiring to bring clk\_inst into their test bench.
* Chassis verification test bench

## Supported Projects

This document supports the following projects at the listed RTL maturity level. Type “NA” if this IP is not included in a specific project, or remove those project names from the table.

|  |  |
| --- | --- |
| Project Name | IP Maturity Level |
| CNX |  |
| KNH |  |
| ICX |  |
| ERL |  |
|  |  |
|  |  |
|  |  |

## Terminology

The table below defines uncommon terms used in this document.

|  |  |
| --- | --- |
| Term | Definition |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## Related Documents

If you need more information on this IP, you may find these documents helpful.

|  |  |
| --- | --- |
| Document Title | Location |
| HAS or EAD |  |
| Product Brief |  |
| Release Notes |  |
| Signal List |  |
| other |  |

## Opens, Risks, and Assumptions

|  |  |  |  |
| --- | --- | --- | --- |
| Item # | Description | Comment | Status (Open or Closed)/Date |
| 1 | Addition of clock gating so that the ratio to a PLL can be changed mid test while the output is held low. | Determing if required | Closed 2/27/2015 |
| 2 | Support for RO mode | Determing if required | Open |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Contact Information

If you need additional help, use the contact information below.

|  |  |  |
| --- | --- | --- |
| Function | Name | Email |
| IP Architecture | Richard Gammack | richard.gammack@intel.com |
| IP Verification | N/A |  |
| IP Integration | Ken Correll | ken.correll@intel.com |
| Doc Template Owner | Susann Flowers | susann.flowers@intel.com |
|  |  |  |
|  |  |  |

## Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Description of Change | Date | Revised By |
| 0.1 | Initial version using this template. Identical in content to revision 2 of the previous document | 15ww8d | Ken Correll |
| 0.2 | Modified file and module names to avoid naming conflicts  Added ability to control the frequency of generated clks during a test | 15ww9c | Ken Correll  Andy Scougal |
| 0.3 | Added constraints to the clock changing function | 15ww9 | Ken Correll |
| 0.4 | Modified instructions on how to pull clk\_inst into your testbench | 15ww11d | Ken Correll |
| **0.5** | Added note about what timescale to use, added support for clkreq/clkack interface | 15ww15 | Ken Correll |
| 0.6 | Fixed typo in the block diagram | 15ww17 | Ken Correll |
| 0.7 | Attempted to fix one vcs clock cycle gitter on generated clocks  Added an optional wait\_for\_lock field to update\_clk\_ratio  Added functions get\_freq\_change\_status\_by\_<name/number>  Added function to set the xtal frequency output | 6/22/15 | Ken Correll |
| 0.8 | Added functions clock\_gen\_enable\_by\_<name/number> | 7/29 | Ken Correll |

# Quick Start

## Integrating clk\_inst

**Integrating 0p5 clk\_inst version in IP/Subsytem:**

1. ***$MODEL\_ROOT/cfg/IPToolData.pm***

* Include the globalclk repo and following scope :

$ToolConfig\_ips{clk\_inst\_vc} = {

      OTHER   => {

                IMPORT => ["cfg/clk\_inst\_vc\_IPToolData.pm"],

      },

};

$ToolConfig\_ips{clk\_inst\_vc}{VERSION} = "globalclk-srvr10nm-15ww50d";

$ToolConfig\_ips{clk\_inst\_vc}{PATH} = "$IP\_MODELS/globalclk/" . $ToolConfig\_ips{clk\_inst\_vc}{VERSION};

IPToolDataExtras::import\_files("clk\_inst\_vc",\%ToolConfig\_ips);

***Note:*** ***globalclk-srvr10nm-15ww50d is 0p5 released version. It is receomned to check with clk\_iinst IP owner to use any version later than 15ww50d.***

* add val scope “clk\_inst\_vc”

***2. $MODEL\_ROOT/Cfg/<IP>\_integ.udf***

* add  "clk\_inst\_vc:cfg/clk\_inst\_vc\_hdl.udf",

## Integrity Checks for Standalone IP

Following are steps for running standalone integrity checks of this IP. It is assumed that the environment variable IP\_ROOT is set to the path of the IP collateral.

1. Run the environment script:

Not applicable

1. Run Lintra:

Not applicable

1. Compile the model:  
     
   Not applicable

# Quick Start

## IP Block Diagram

Clk\_inst is a behavioral block that supplies clocks and usync signals to the DUT compliant with 10nm products.



## Functional Top-Level Signals

Clk\_inst generates the following clocks:

* 1. x1clk and x1clk\_usync

A programmable 100MHz clk is generated with a period of (107/ratio) x 100 fs or 104 ps, where ratio is internally set to 100. This should not be changed. This clock is also assigned to:

Bclk\_p = x1clk

Bclk\_n = ~x1clk

refclk = x1clk

ClkRefX1H = x1clk

* 1. x3clk and x3clk\_usync

This clock is not programmable and is fixed at a 7.5ns period or 133.3333MHz

* 1. xclk[X] and xclk\_usync[X]

A programmable number of clkgen blocks can be instantiated. Each instantiation will accept an input ratio and supply a clk and usync output. By default three clocks are generated:

parameter NUM\_CLKS = 3

parameter [7:0] RATIO[NUM\_CLKS-1:0] = ‘{8’d12,8’d8, 8’d4};

logic [NUM\_CLKS-1:0] enable = {NUM\_CLKS{1’b1}};

logic [NUM\_CLKS-1:0] xclk;

logic [NUM\_CLKS-1:0] xclk\_usync;

|  |  |  |
| --- | --- | --- |
| X | RATIO | CLK FREQ |
| 0 | 4 | 400 MHz |
| 1 | 8 | 800 MHz |
| 2 | 12 | 1200 MHz |

Due to the common usage of x4clk and x8clk, they have been assigned to the following top level signals:

ClkRefX4H = xclk[0]

ClkM1H = xclk[1]

* 1. xtalclk

Three crystal clock frequencies are generated – 19.2MHz, 24MHz, 25MHz.

Each can be accessed in xtalclkgen.xtal\_19p2M, xtalclkgen.xtal\_24M and xtalclkgen.xtal25M.

There is an output pin xtalclk, where the frequency selected defaults to 19.2MHz, but can be modified in the test program with the call xtal\_frequency\_select(sel). Where:

0 -> 19.2MHz

1 -> 24MHz

2 -> 25MHz

3 -> no output

1. clkreq[i]/clkack[i]

These signals implement a req/ack interface for each of the NUM\_CLKS clock generators.

refclkgen.sv I/O

|  |  |  |  |
| --- | --- | --- | --- |
| **Signal Name** | **I/O** | **Initial value** | **Description** |
| ref\_ratio | Input | 100 | results in a 100MHz refclk/bclk frequency |
| ref\_enable | Input | 1 | 1: enable refclk, 0: disable refclk |
| x1clk | output |  | Connected in clk\_inst to bclk\_p, bclk\_n, refclk |
| x1clk\_usync | output |  | x1clk usync, delayed 10ps relative to x1clk |
| x3clk | output |  | 133.333MHz clk |
| x3clk\_usync | output |  | Usync associated with 133.333 MHz clk, delayed 10ps relative to x3clk |

clkgen.sv I/O – instantiated NUM\_CLKS time with set inside clk\_inst.sv to 2 (3 instances)

|  |  |  |  |
| --- | --- | --- | --- |
| **Signal Name** | **I/O** | **Initial value** | **Description** |
| clkref | Input |  | Connect to the x1clk output of refclkgen.sv |
| clkref\_usync | Input |  | Connected to the x1clk\_usync output of refclkgen |
| [7:0] RATIO[NUM\_CLKS-1] | Input | [0]: 4  [1]: 8  [2]: 12 | 400MHz  800MHz  1200MHz |
| [NUM\_CLKS-1] enable | Input | { NUM\_CLKS{1’b1}} | x1clk usync, delayed 10ps relative to x1clk |
| [NUM\_CLKS-1]xclk | output |  | [0]: 400MHz output phase locked to clkref  [1]: 800MHz output phase locked to clkref  [2]: 1200MHz output phase locked to clkref |
| [NUM\_CLKS-1]xclk\_usync | output |  | Usync, delayed 10ps relative to its xclk |

Other clk\_inst.sv signals available:

|  |  |
| --- | --- |
| Signal Name | Description |
| Bclk\_p | Connect to x1clk output of refclkgen.sv |
| Bclk\_n | Connected to ~x1clk output of refclkgen.sv |
| refclk | Connect to x1clk output of refclkgen.sv |
| ClkRefX1H | Connected to x1clk output of refclkgen.sv |
| ClkRefX4H | Connected to xclk[0] which defaults to 400MHz.  If the parameters are changed this might be affected. |
| ClkM1H | Connected to xclk[1] which defaults to 800MHz.  If the parameters are changed this might be affected. |

Xtalclkgen signals

|  |  |  |
| --- | --- | --- |
| Signal Name | I/O | Description |
| xtal\_freq\_sel | internal | 0: Connects 19.2MHz clk to output  1: Connects 24MHz clk to output  2: Connects 25MHz clk to output |
| xtal\_25M | internal | Internal 25MHz clk |
| xtal\_24M | Internal | Internal 24MHz clk |
| xtal\_19p2M | Internal | Internal 19.2MHz clk |
| xtalclk | output | Output frequency dependent on xtal\_freq\_sel |

## Timing Diagram

The following figure shows the relationship of the usync signal to the usync (or gal) edge.



This relationship will hold true for any clock generated by modifying the parameters.

## Frequency changes

The capability has been added for the test to dynamically change the ratio presented to one of the instantiated ipclkgen\_inst modules on the fly with the wait\_for\_lock not included or set to 1. When this happens the logic:

* 1. Gates the output clock synchronized to the next usync edge
  2. Waits for the ipclkgen\_inst module targeted to indicate the gate has been asserted
  3. Changes the ratio
  4. Waits for the PLL model to indicate lock
  5. Ungates the output clock synchronized to the next usync edge
  6. Waits fo the ipclkgen\_inst module targeted to indicate the gate has been deasserted

Timing diagram:



If wait\_for\_lock is set to ‘0’, then the new ratio is written and the program returns. The steps above still occur, the PLI call doesn’t wait for them to complete.

In this case multiple new ratio’s can be written at the same time and all will immediately be acted on.

To know when the frequency switch is done and the clocks are stable again, a new function to check the lock status has been added.

Notes:

A ratio of 1 is invalid.

Requesting a ratio change where the ratio does not change will hang the test if wait\_for\_lock is ‘0’.

## clkreq / clkack interface support

Support for the standard clkreq/clkack interface has been added. Use of this interface is not required.

If desired connect the clkreq / clkack signals in the IP to the corresponding clkreq / clkack signals that exist in clk\_inst.sv;

assign <ip>.clkack = clk\_inst.clkack[i];

assign clk\_inst.clkreq[i] = <ip>.clkreq;

Where i corresponds to one of the instanced ipclkgen\_inst modules.

Timing is illustrated below. Note that there is no relationship in this implementation between clkreq and usync.



## Clock edge jitter on the gal edge

In some test benches it has been seen that there is a single vcs cycle of uncertainty between generated clock edges on the gal edge. An attempt has been made to fix this.

## Enabling xclk[n] and xclk\_usync[n] outputs

Two functions have been added with support logic in the RTL to enable the capability of having the outputs drive X until enabled.

Parameter CLK\_ENABLE = {NUM\_CLKS{1’b1}}

The parameter CLK\_ENABLE sets the initial state of the enable signal. If not included or if set to ‘1, then that set of outputs (xclk[n] & xclk\_usync[n]) will behave as before, driving the frequency set by the RATIO parameter until changed.

If a CLK\_ENABLE bit is ‘0, then that set of outputs will drive ‘X until set by the new function call clock\_gen\_enable\_ by\_name(clk\_name, enable) or clock\_gen\_enable\_by\_number(clk\_number, enable). Enable can be set to ‘0 or ‘1. Setting to ‘0 will cause the associated set of outputs to drive ‘X. Setting to ‘1 will cause the associated outputs to drive the clock and usync generated.

Note that changing the ratio of a clock generator while not enabled (driving ‘X) will still have affect. When the appropriate clk\_enable bit is set the new frequency (if the change has completed) will be driven.

# Design Information for Integration

This chapter is targeted to the IP verification team responsible for integrating this IP into a local test bench.

## RTL Directory Structure

All files are located in $IP\_MODELS/globalclk/globalclk-srvr10nm-latest/verif/testbench/

## Clock, Power and Reset Domains

Not applicable

### Clock Domain Diagram

Not applicable

## Embedded Building Blocks/Custom Logic

|  |  |  |
| --- | --- | --- |
| Name | Library | Synthesis exchange? |
|  |  |  |
|  |  |  |
|  |  |  |

## RTL Configuration Parameters

The following tables list all RTL configuration parameters for this IP. If the parameter is derived, it must not be changed by the user.

### Mandatory Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter Name | Derived? | Range | Default | Descriptions (including interdependencies) |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

### Boundary Scan Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter Name | Derived? | Range | Default | Descriptions (including interdependencies) |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

### Test Data Register Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter Name | Derived? | Range | Default | Descriptions (including interdependencies) |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Testbench Parameters

The following table lists all testbench configuration parameters for this IP.

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter Name | Range | Default | Descriptions (including interdependencies) |
| NUM\_CLKS | 1 : unlimited | 3 | Defines the number of times the clkgen module is instantiated inside clk\_inst |
| RATIO | Each term  1 to 48 | {12,8,4} | The ratio defines the multiplier times 100MHz to generate the clock desired in each instance of clkgen. |

## IP Straps

|  |  |
| --- | --- |
| Strap | Purpose |
|  |  |
|  |  |
|  |  |

## Fuses

Not applicable.

## Power Information

### Power Supply

Not applicable.

### Static Clock Gating

Not applicable.

### Power Gating

Not applicable.

### Bumps and Their Power Domains

Not applicable

## Power-up Requirements

Not applicable.

## Macros used by IP

Not applicable.

## Other Design Considerations

Not applicable

## DFx Considerations

Not applicable.

### DFx Top-Level Signals

Not applicable

### DFx Clock Definition

### Clock Crossings

Not applicable

### N/ADebug Registers

### Scan – Clock Gating in RTL

Document the override signal that makes clocks free-running in scan mode.

### Scan – Reset Override

Describe the mechanism to override reset during scan mode.

### TAP and Associated Registers

## System Startup

### Power-up Sequence

Not applicable

### Initialization Sequence

Not applicable

### Device Configuration

Not applicable

### Header for Windows Boot

Not applicable

## Security Considerations

Not applicable

### Security Threats

Not applicable.

### Security Tests

Not applicable.

### Interface Signals Implemented for Security

Not applicable.

## RTL Design Libraries

Not applicable.

## RTL Uniquification

Not applicable

## Emulation Support

Not applicable

# Verification Information for Integration

## IP Testbench Overview

Clk\_inst is typically instantiated in a test benches soc\_pre\_ti\_include.sv file:

clk\_inst clk\_inst();

NOTE: clk\_inst uses timeunit and timeprecision of 100fs. Problems have been seen in IP’s that try to compile their Verilog with a timescale of 1ns/1ns. It is therefor suggested that 1ps/1ps be used.

## IP does not use any unit level test bench or test island.Reusable IP Testbench Components

### Test Island

Clk\_inst is a parameterized module. The parameters NUM\_CLKS and RATIO define how many clocks will be generated (in addition to the 100MHz x1clk and the 133.333MHz x3clk) and what the frequencies will be.

The default assignment is:

parameter NUM\_CLKS = 3; // number of clk generators

parameter [7:0] RATIO [NUM\_CLKS-1:0] = '{8'd12, 8'd8, 8'd4};

// array of ratio's for the clkgen units instantiated

// this parameter list sets xclk[2] at 1200MHz, xclk[1] at 800MHz and xclk[0] at 400MHz

If the desired output is 400MHz, 800MHz, 1300Mhz and 2400MHz, then the instantiation of clk\_inst would be:

Clk\_inst #(NUM\_CLKS=4, RATIO = `{8`d24, 8`d13, 8`d8, 8`d4}) clk\_inst();

Following are details of interfaces that are to be connected at the SoC level.

|  |  |  |
| --- | --- | --- |
| Signal | Connect to | Description |
|  |  |  |
|  |  |  |

### Collage or Sandbox Files

**Integrating 0p5 clk\_inst version in IP/Subsytem:**

1.The file: ***$MODEL\_ROOT/cfg/IPToolData.pm***

must include the globalclk repo and following scope :

$ToolConfig\_ips{clk\_inst\_vc} = {

      OTHER   => {

                IMPORT => ["cfg/clk\_inst\_vc\_IPToolData.pm"],

      },

};

$ToolConfig\_ips{clk\_inst\_vc}{VERSION} = "globalclk-srvr10nm-15ww50d";

$ToolConfig\_ips{clk\_inst\_vc}{PATH} = "$IP\_MODELS/globalclk/" . $ToolConfig\_ips{clk\_inst\_vc}{VERSION};

IPToolDataExtras::import\_files("clk\_inst\_vc",\%ToolConfig\_ips);

2. ***Cfg/\*\_integ.udf***

* add val scope “clk\_inst\_vc”
* add  "clk\_inst\_vc:cfg/clk\_inst\_vc\_hdl.udf",

Where the version could be any of the released versions from 15ww08b forward. Specifying latest is preferred as any bug fixes or feature enhancements will automatically be picked up. Any feature enhancement will be backwards compatible.

**In all cases add:**

In the testbench .hdl file:

-vlog\_files => [

&ToolConfig::get\_tool\_path("ipconfig/globalclk") . "/verif/testbench/clk\_inst.sv",

],

-vlog\_lib\_dirs => [

&ToolConfig::get\_tool\_path("ipconfig/globalclk") . "/verif/testbench",

],

-vlog\_opts => [

"+libext+.sv",

"+libext+.vs",

],

Note that it is important to only point to the top level file. Doing this and added the vlog\_lib\_dirs and vlog\_opts entries will avoid any naming conflicts in sub modules.

Any reference of local copies of the old versions of these files in the testbench should be deleted.

**Adding optional control of output frequency from test sequences**

If it is desired to modify the output frequency of one of the ipclkgen\_inst modules during a test the following setup is required.

In the testbench \*\_env\_pkg.sv file, import the clk\_ctrl\_pkg.sv, e.g.

import clk\_ctrl\_pkg::\*;

In the testbench env class, add something like the following:

clk\_control m\_clk\_ctrl;

In build() method of the env class, add something like the following:

m\_clk\_ctrl = clk\_control::type\_id::create("clk\_ctrl", this);

In connect() method of the env class, add path to the clk\_inst, e.g.

m\_clk\_ctrl.set\_clk\_path("soc\_tb.clk\_inst");

Optionally - setup name to number map for easier reference, for example:

m\_clk\_ctrl.set\_clk\_map({"clk400":1, "clk800":2, "clk200":3, "clk100":4});

m\_clk\_ctrl.update\_clk\_map("newclk", 5);

**Using clk control in test sequences**

Usage - in test sequence, etc:

env.m\_clk\_ctrl.update\_ratio\_by\_name("clk400", 4, wait\_for\_lock); // Change to 400 Mhz

env.m\_clk\_ctrl.update\_ratio\_by\_number(5, 8, wait\_for\_lock); // Change to 800 Mhz

Note that wait\_for\_lock is a new parameter initialized to ‘1’. In order for multiple clocks to change frequency at the same time this must be included and set to ‘0’.

To determine if the frequency change requested above is complete when programmed with wait\_for\_lock = 0:

done = env.m\_clk\_ctrl.get\_freq\_change\_status\_by\_number(“clk400”, <new ratio>);

done = env.m\_clk\_ctrl.get\_freq\_change\_status\_by\_number(2, <new ratio>);

This function returns ‘1’ when the clock specified is locked to the new frequency

To change the xtal clock frequency, use:

env.m\_clk\_ctrl.xtal\_frequency\_select(select)

Note that there is not attempt to make a clean, glitch free change in xtal clock frequencies.

**Updates to testbench udf files to support clk\_ctrl**

1. In \*\_integ.udf ensure "Includes" section has line with globalclk:cfg/globalclk\_hdl.udf
2. In IPToolData.pm ensure SUB\_SCOPES has line with "globalclk"
3. In \*\_hdl.udf ensure “clk\_verif\_lib” is excluded from lintra and sglp checks: clk\_verif\_lib
4. In \*\_hdl.udf ensure “clk\_verif\_lib” is imported e.g.

HDLSpec => {

InterfaceVars => {

-vlog\_opts => ["-full64",],

},

$scope => {

import => {

globalclk => { libs => ["clk\_verif\_lib", ], },

1. In \*\_hdl.udf update dependent libs to include “clk\_verif\_lib”, e.g.

libs => {

sbbridge\_env\_lib => {

-hdl\_spec => ["verif/tb/bridge/ace/sbbridge\_env.hdl",],

-dependent\_libs => ["sip\_shared\_lib", ..., "clk\_verif\_lib"],

},

1. In \*\_hdl.udf, in model section add “clk\_verif\_lib” to list of libs, e.g.

models => {

sbbridge => { top => "sbbridge\_top\_lib.sbbridge\_cfg",

libs => [

"sip\_shared\_lib",

"clk\_verif\_lib",

### IP Environment

Not applicable

#### Configuring the IP Environment

Not applicable

#### Saola Environment Walkthrough

Following are the components of RAL:

|  |  |
| --- | --- |
| File | Description |
| N/A | N/A |
|  |  |
|  |  |

#### Saola/RAL Components

|  |  |  |  |
| --- | --- | --- | --- |
| SAOLA components | Description | SoC recommendations | Required? |
| N/A |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| RAL components | Description | SoC recommendations | Required? |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

#### System Manager

Not applicable

### N/A. IP does not has any fuse requirements.Sequences

Sequences are located here: N/A

#### Sequence for Bringing up the IP

#### N/ABFM Sequences

|  |  |  |  |
| --- | --- | --- | --- |
| Sequence Name | Description | Parameters | Saola Phase |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

#### IOSF Primary/Sideband BFM Sequences

|  |  |  |  |
| --- | --- | --- | --- |
| Sequence Name | Description | Parameters | Saola Phase |
| N/A |  |  |  |
|  |  |  |  |
|  |  |  |  |

#### Other Reusable Sequences

|  |  |  |  |
| --- | --- | --- | --- |
| Sequence Name | Description | Parameters | Saola Phase |
| N/A |  |  |  |
|  |  |  |  |
|  |  |  |  |

#### IP Test Sequences

|  |  |  |
| --- | --- | --- |
| Test Sequence Name | Parameters | Function |
| N/A |  |  |
|  |  |  |
|  |  |  |

#### SoC Requirements for Sequence Reuse

#### Sequence File Dependencies

### Ip does not used any sequences or extended sequences.Miscellaneous

#### Using the Runtime or Post-Processing Checkers

Not applicable

#### Environment Files

Not applicable

#### Coverage

Not applicable.

## Environment Settings and Files

### Base Test

### N/AConfiguration Object

Not applicable

### API

## N/ADescription of Reusable Tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Name | Runcmd | Range | Transaction | Source |
| N/A |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Description of Reusable Automation Scripts

## N/ASupported Compiler Options for Simulation

The table below summarizes the supported options.

|  |  |  |
| --- | --- | --- |
| Argument | Input | Example |
| N/A |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Reusable Simulation RUNMODEs

|  |  |
| --- | --- |
| RUNMODE | Description |
| N/A |  |
|  |  |
|  |  |

## RTL Verification Libraries

|  |  |  |
| --- | --- | --- |
| Library | Version | Special usage |
| N/A |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Tools and Methodology for Integration

## Supported Tools

The following tools are used in the integration of this IP. For versions supported by each release, see Release Notes in the "doc" directory of the release package.

* VCSMX
* OVM
* Ace
* SaolaLintra
* Design Compiler
* Conformal
* 0-In

## Environment Variables

Set the following environment variables as listed.

|  |  |
| --- | --- |
| Variable | Description |
| N/A |  |
|  |  |
|  |  |

## HIP Libraries Included in Release

|  |  |  |
| --- | --- | --- |
| Library | Version | Location |
| N/A |  |  |
|  |  |  |
|  |  |  |

### Register Files or SRAM

Not applicable

### M-PHY and Related Libraries

|  |  |  |
| --- | --- | --- |
| Library | Version | Date |
| N/A |  |  |
|  |  |  |
|  |  |  |

## Directory Structure

Doc

Cfg

Bin

Src

Tools

Target

## Ace

Paths to acerc: $MODEL\_ROOT/cfg/globalclk.acerc

Location of udf file:$MODEL\_ROOT/cfg/globalclk\_hdl.udf

Model to use when importing libraries:globalclk

Elaboration options not exported:N/A

Required content in sip\_shared\_libs:N/A

## Lintra

Lintra Version: 14.1p9\_shOpt64

Lintra location : tools/lintra/

Location of waiver files: tools/lintra/waivers

Location of Lintra patches & configuration: N/A

Location of Lintra report file for warnings and errors:

$MODEL\_ROOT/target/lint/<model\_name> /<model\_name>.log

## Synthesis

Not applicable

### Clocks

1. Primary Clocks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Clock name | Clock period | Clock waveform | Clock source |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1. Generated Clocks

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | Clock Name | Master Clock Name | Master Clock Source | Edges | Source |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

### Clock Diagram

### Constraint Files

Not applicable

### Scan Insertion

Not applicable

## Formal Verification

Not applicable

## CDC

Not applicable

# Physical Integration

This chapter is intended to capture the aspect ratio requirements and any fixed size impact, etc., of memories that will be used in the IP.  It is not intended to be “accurate” so much as an indication of what the impact and limitations might be.  As this information will be based on the current memories, it would be only as accurate as the current design.

|  |  |
| --- | --- |
| Array type and number of instances | N/A |
| Functional usage (how many bits are used) |  |
| Highest functional clock frequency |  |
| Floorplan details |  |
| Security requirements |  |
| IP power draw limitations for array testing |  |

# Integration Test Plan

Not applicable

# Appendix

## Subsystem connectivity details