

ISim HW Co-Simulation 13.4 Quick Reference

Powering Up a Board

The following procedure describes how to install and set up the board to run Hardware (HW) Co-Simulation:

1. Make sure the hardware board is powered off.
2. If you are using a Xilinx® Parallel Cable IV:
 - a) Connect the DB25 Plug Connector on the Xilinx Parallel Cable IV to the IEEE-1284 compliant PC Parallel (Printer) Port Connector.
 - b) Using the narrow (14 pin) 6" High Performance Ribbon cable, connect the pod end of the Xilinx Parallel Cable IV to the FPGA JTAG header on the board.
 - c) Connect the attached power jack cable to the keyboard/mouse connector.
 - d) If necessary, connect the male end of the keyboard/mouse cable to the associated female connector on the Xilinx power jack cable (splitter cable).
3. If you are using a Xilinx Platform Cable USB:
 - a) Connect the Xilinx Platform Cable USB to the computer USB port.
 - b) Using the narrow (14 pin) 6" High Performance Ribbon cable, connect the pod end of the Xilinx Platform Cable USB to the FPGA JTAG header on the board.
4. If your board supports P2P Ethernet co-simulation, and you want to use Ethernet P2P for (faster) co-simulation, then in addition to the above steps, use an ethernet cable to connect the ethernet port of your PC to the Ethernet port of your board.
If your computer has more than one ethernet card/port, note the MAC address of the port that you connected to the board. Specify that address to the ISim engine using the ISim Tcl command: `hwcosim set ethernetInterfaceID <MAC address>` before issuing any run command to the ISim engine. See the section "Using Tcl to Set and Get HW Co-Sim Properties" on page 5 for more details.
5. Power on the board and check to make sure the LED on the cable is green. Install the Xilinx cable drivers when prompted.

Running From Project Navigator

1. Ensure the ISim is the selected project simulator.
2. Switch to the Simulation view.
3. From the design hierarchy view, select the instance to co-simulate in hardware.
4. Right-click and select Source Properties from the popup menu to open the source properties dialog box.
5. In the **Source Properties** dialog box, select Hardware Co-simulation from the category list.
6. Set the following properties for hardware co-simulation:
 - a) Check the Enable Hardware Co-simulation check box.
 - b) In the Clock Port field, specify the name of the clock port on the instance. For an instance with multiple clocks, specify the name of the fastest clock port.
 - c) Select a board from the Target Board for Hardware Co-simulation drop-down list. The list shows only the boards with an FPGA that is in the same device family chosen for the project.
 - d) If a previous hardware co-simulation bitstream is available and the instance under co-simulation remains unchanged, check the Enable Incremental Implementation checkbox to skip the implementation flow for hardware co-simulation.
 - e) Click OK to close the source properties dialog box.
7. Select the test-bench module in the hierarchy view to start the simulation. ISim Hardware co-simulation must be started at a level above the instance that is selected for co-simulation.
8. Go to the Process view of the test-bench and double click on Simulate Behavior Model to start the compilation and simulation process.

ISim HW Co-Simulation 13.4 Quick Reference

Running on the Command Line

Run ISim compiler 'fuse' with following HW Co-Sim specific options added to the regular ISim compilation options. Then, run the generated simulation executable.

Usage: fuse [options] {[<work>.]<design top>}

Options:

```
....
--hwcosim_instance arg (Mandatory) Specify the hierarchical name of the instance that
will be run on FPGA (Example: /testbench/UUT)
--hwcosim_clock arg (Mandatory) Specify the clock port name on the instance
--hwcosim_board arg (Mandatory) Specify the name of the board
--hwcosim_incremental 0|1 (Optional) Skip the implementation phase and reuse
the previously created bit-stream file if value is 1 (Default: 0)
--hwcosim_constraints (Optional for non-hybrid mode) Specify the custom UCF
constraints file to provide additional constraints to control
implementation. The constraints specified are concatenated
to the constraints used by the HW Co-Sim compiler. This option
is Mandatory for the hybrid mode of HW Co-Simulation in order
to be able to map certain ports to external I/O pins of the FPGA
```

Examples:

```
Compilation: fuse -lib unisims_ver -lib unimacro_ver -lib xilinxcorelib_ver -lib secureip -prj
fp_fft_tb_beh.prj -hwcosim_instance /fp_fft_tb/dut -hwcosim_clock clk -hwcosim_board kc705-
jtag work.fp_fft_tb_work.glbl -o fp_fft_tb_isim_beh.exe
Simulation: fp_fft_tb_isim_beh.exe -tclbatch my.tcl
```

```
Compilation: fuse -prj mig_hw_tb.prj mig_hw_tb.glbl -L unisims_ver -L secureip -o
mig_hw_tb.exe
-hwcosim_instance /mig_hw_tb/mig_inst -hwcosim_clock c3_clk0 -hwcosim_board sp601-jtag
-hwcosim_constraints mig_dut_hwcosim.ucf -o mig_hw_tb.exe
Simulation: mig_hw_tb.exe -gui -tclbatch my.tcl -view my.wcfg
```

Supported Xilinx Boards

The following Xilinx boards are supported by default. Support for additional boards can be added as explained in the "Adding Support for a New Board" section.

- ml401-jtag: Xilinx ML401 Evaluation Platform
- ml402-jtag: Xilinx ML402 Evaluation Platform
- ml403-jtag: Xilinx ML403 Evaluation Platform
- ml405-jtag: Xilinx ML405 Evaluation Platform
- ml410-jtag: Xilinx ML410 Evaluation Platform
- ml501-jtag: Xilinx ML501 Evaluation Platform
- ml505-jtag: Xilinx ML505 Evaluation Platform
- ml506-jtag, ml506-ppethernet: Xilinx ML506 Evaluation Platform
- ml507-jtag: Xilinx ML507 Evaluation Platform
- xupv5-jtag: Xilinx University Program XUPV5-LX110T
- ml510-jtag: Xilinx ML510 Evaluation Platform
- ml605-jtag, ml605-ppethernet: Xilinx ML605 Evaluation Platform
- kc705-jtag: Xilinx KC705 Evaluation Platform
- vc707-jtag: Xilinx VC707 Evaluation Platform
- s3e-sk-jtag: Xilinx Spartan®-3E Starter Kit
- s3e-mb-jtag: Xilinx Spartan-3E MicroBlaze Development Kit
- s3a-sk-jtag: Xilinx Spartan-3A Starter Kit
- s3an-sk-jtag: Xilinx Spartan-3AN Starter Kit
- s3adsp1800a-jtag: Xilinx Spartan-3A DSP 1800A Starter Platform
- s3adsp3400a-jtag: Xilinx Spartan-3A DSP 3400A Development Platform
- sp601-jtag, sp601-ppethernet: Xilinx SP601 Evaluation Platform
- sp605-jtag, sp605-ppethernet: Xilinx SP605 Evaluation Platform

ISim HW Co-Simulation 13.4 Quick Reference

Adding Support For a New Board

It is easy to support a new FPGA board for hardware co-simulation in ISim, provided that the board has a JTAG header. You must create an additional entry in the board support file that records the following information of the board:

- FPGA part information
- Period and pin location of the system clock
- JTAG boundary scan chain information

A default board support file is installed under ISE installation as `$(XILINX)/sysgen/hwcosim/data/hwcosim.bsp`. This file lists entries for all the boards already supported. To support additional boards, you can either add entries for the new boards directly to the default board support file, or you can create a local board support file named as `hwcosim.bsp` in the directory where ISim compiler fuse is invoked with new entries added to the file.

The board support file defines a list of board specification in the following format:

```
kc705-jtag' =>
{
  'Description' => 'KC705 (JTAG)',
  'Vendor' => 'Xilinx',
  'Type' => 'jtag',
  'Part' => 'xc7k325t-2ffg900',
  'Clock' =>
  [
    {
      'Period' => 5,
      'VariablePeriods' => [ 10, 15, 20, 30 ],
      'Pin' => {'Positive' => 'AD12', 'Negative' => 'AD11'},
    },
    {
      'Period' => 6.405,
      'Pin' => {'Positive' => 'K28', 'Negative' => 'K29'},
    },
  ],
  'BoundaryScanPosition' => 1,
},
```

In the previous example format, `kc705-jtag` is the board identifier that is provided to the ISim compiler fuse to compile the design for the given board. Following the board identifier is a list of properties:

- **Description** provides the descriptive name of the board.
- **Vendor** specifies the board vendor.
- **Type** specifies the type of co-simulation interface to be used. Allowed values are: **jtag** or **ppethernet** (for point-to-point Ethernet based HW Co-Simulation).
- **Part** specifies the part name of the FPGA on the board.
- **Clock** provides the system clock information, where **Period** (and **VariablePeriods**) specifies the supported clock period(s) in nanoseconds, and **Pin** specifies the clock pin location. For differential clock sources, both **Positive** and **Negative** clock pin location should be provided.
- **BoundaryScanPosition** tells the position of the FPGA on the JTAG boundary scan chain, beginning with "1." This information can be determined by running the Xilinx iMPACT tool.

Note: For P2P Ethernet HWCoSim, additional fields must be specified. See setup for 'ml605-ppethernet' in the `$(XILINX)/sysgen/hwcosim/data/hwcosim.bsp` file as an example.

ISim HW Co-Simulation 13.4 Quick Reference

Using TCL to set and get HwCoSim Properties

ISim provides the Tcl command `-hwcosim` to get or set a given HW Co-Simulation property for the instance in the design hierarchy that is HW co-simulated:

```
hwcosim get|set <property>
```

Use this command to get or set the property of the hardware co-simulation instance in the current scope. A few important points about this command are:

- This command is scope-sensitive, which means you need to first use the “scope” Tcl command to select the instance that is being hardware co-simulated in the design hierarchy. For example: “scope /tb/DUT” should be used to change the scope to the /tb/DUT.
- You must run the `-hwcosim set <property>` command before issuing any simulation run Tcl command (such as `run 100 ns`).
- The `-restart` Tcl command does not preserve the properties set earlier. Therefore, after “restart” has been run, you will need to set the properties again.

You can set and read the following hardware co-simulation properties:

- skipConfig:** Default is 0. Set to 1 if the FPGA configuration should be skipped. To skip the configuration, the FPGA should have been configured with a valid hardware co-simulation bitstream. Otherwise, unexpected behavior can occur.
- cableParameters:** Default is an empty string. This property is used to specify a third-party JTAG cable supported by the iMPACT/ChipScope™ tools. Refer to the *iMPACT/ChipScope User Guide* for details on specifying cable plugin parameters.
- shareCable:** Default is 0. This property is only available for JTAG-based co-simulation interfaces. Set to 1 if the JTAG cable should be shared with EDK XMD or ChipScope Analyzer for concurrent access. Share the JTAG cable only when necessary as this may decrease the hardware co-simulation performance substantially.
- ethernetInterfaceID:** Default is an empty string. This property is only applicable to Ethernet-based co-simulation. If you have multiple Ethernet cards available on your host machine, you need to select the Ethernet card that is connected to your FPGA board. You can select an Ethernet card by setting the value of this property to the MAC address (in the format of `xx:xx:xx:xx:xx:xx`) of the Ethernet card.

Examples:

- The following Tcl commands get the `cableParameters` property of the hardware co-simulation instance `/mytestbench/top/hwcosim_inst`:

```
ISim> scope /mytestbench/top/hwcosim_inst
ISim> hwcosim get cableParameters
```
- Set `skipConfig` property of the hardware co-simulation instance under `/mytestbench/top/hwcosim_inst` in two simulation runs:

```
ISim > scope /mytestbench/top/hwcosim_inst
ISim > hwcosim set skipConfig 1
ISim > run 1000 ns
ISim > restart
ISim > scope /mytestbench/top/hwcosim_inst
ISim > hwcosim set skipConfig 1
ISim > run 1000 ns
```
- Select a particular ethernet port for P2P Ethernet HW Co-Simulation:

```
ISim > scope /tb/DUT
ISim > hwcosim set ethernetInterfaceID "00:1D:09:C9:2C:D6"
```

Note: To obtain the MAC address of the Ethernet port on your computer, you can run `ipconfig /all` on a command prompt on Windows and `/sbin/ifconfig -a` on Linux. Review the entries that say “Physical Address” or “HWaddr” to identify the MAC address of the port that you have connected to the board. You must use “:” instead of “-” to separate the numbers while specifying the MAC address using the `hwcosim` command.

ISim HW Co-Simulation 13.4 Quick Reference

Frequently Asked Questions

Q: Are there any limitations to the kinds of designs that can be HW Co-Simulated?

- A: Hardware co-simulation in ISim currently has the following limitations:
- Only one instance in a design can be selected for hardware co-simulation, and it cannot be the top-level test-bench itself.
 - The selected instance for hardware co-simulation must be synthesizable using XST and implementable on the target FPGA device of the selected board.

The non-hybrid mode of hardware co-simulation has additional restrictions on clocking and I/Os:

- The instance co-simulated in hardware is clocked with an emulated clock source that is controlled by ISim in synchronization with the software simulation. Thus, the co-simulation does not exactly model the real-life scenario of the design running in hardware or serve as a timing simulation.
- The instance under co-simulation cannot have access to external I/Os or MGTs, or instantiates primitives (such as DCMs/PLLs) that require a continuous clock or a clock at a specific frequency.
- All ports of the instance under co-simulation must be routable to a slice register or LUT. Certain resources on the FPGA require dedicated routes such as to an IOB or to certain port of a primitive, and cannot be wired to any port of the instance under co-simulation.

Q: What happens if the DUT already instantiates a BSCAN primitive (such as ChipScope ICON or EDK MDM)?

A: The hardware co-simulation interface uses a BSCAN primitive at location 1, which might result in an error in MAP if the DUT also instantiates another BSCAN primitive at location 1. Therefore, you might need to change the ChipScope ICON or EDK MDM to use a different location for the BSCAN primitive. To share the JTAG cable with ChipScope Analyzer or EDK XMD, you need to run this Tcl command: `hwcosim set shareCable 1` before issuing the “run” Tcl command.

Q: Which clock is supplied to the DUT? How is the clocking of the DUT being handled during co-simulation?

A: The clock pin of the chosen hardware board specified in the board support file is the master clock source but is not directly used to drive the DUT. This clock source is scaled to a particular clock frequency (typically around 25 to 100 MHz) through a DCM/PLL. The scaled clock further goes through a gated clock buffer (BUFGCTRL) before driving the clock port of the DUT. The gated clock buffer generates a clock pulse to the DUT per simulation cycle in order to synchronize the software simulation and the DUT execution.

Q: Can I allow the DUT clock to run free?

A: In hybrid co-simulation, you can let a portion of the DUT free-running by mapping one or multiple clock ports to external I/Os. However, at least one clock of the DUT needs to be clocked in a lockstep fashion in synchronization with the software simulation. That effectively partitions the DUT into two portions, one running in lockstep and the other free running. Also note that ISim hardware co-simulation does not insert any clock domain crossing logic between the two portions. Some asynchronous buffer or clock domain domain logic is expected in DUT between the two portions.

Q: Can I run the DUT at a particular clock frequency?

A: You can supply external clocks for the free-running portion of the DUT in hybrid co-simulation. However, the clock in the lockstep portion is controlled by ISim in synchronous to the software simulation and is not fixed to a particular clock frequency. The effective clock rate of the lockstep portion is very slow regardless of its input clock frequency. Driving the DUT with a faster clock does not improve the simulation performance because the major bottleneck is the communication between software and hardware. We constrain the DUT with a lower clock frequency in order to make the compilation process (such as map and place-and-route) faster.

Visit <http://www.xilinx.com/tools/isim.htm> for more details

ISim HW Co-Simulation 13.4 Quick Reference



Introduction

With ISim HW Co-Simulation, you can run the Design Under Test (DUT) on Xilinx FPGA while keeping HDL test bench and some of the other modules in the ISim HDL simulator.

ISim Hardware co-simulation supports two use models: one for pure logic-based designs and the other for hybrid designs consisting of hard IPs or external hardware components (such as DDR3 memory) requiring external I/O.

Pure Logic-based Designs

Pure logic-based designs are co-simulated in a lockstep fashion with the ISim HDL simulator. The modules under co-simulation typically have the following characteristics:

- Modules are composed of only LUTs, FFs, Block RAMs, and DSP primitives.
- All ports are controlled by ISim and accessible from the software test-bench. That is, no external I/Os.
- No need to run on a continuous clock or a clock at a specific frequency. The functionality of the module is independent of the clock frequency at which it operates.

The lockstep-based hardware co-simulation provides the following advantages:

- Simulation acceleration for compute-intensive designs
- In-hardware functional verification
- Bit-and-cycle accurate with respect to pure software simulation

Hybrid Designs

The pure logic-based design use model is simple to set up, but is not suitable for designs that require hard IPs, external I/Os, and specific clock frequencies. Hardware co-simulation in ISim offers a hybrid co-simulation flow that supports designs with the following characteristics:

- Composed of hard IP blocks, DCMs/PLLs, and MGTs
- Has some clocks that are in lockstep with the software simulation using emulated clock sources while some other clocks are free-running using external clock sources
- Might have some ports mapped to external I/Os, which are neither controlled by ISim nor accessible from software test-bench

The hybrid co-simulation flow offers the following advantages:

- Provides simulation acceleration
- Has in-hardware functional verification
- Allows non-trivial software and hardware interactions

