Questa® SIM Multi-core Simulation User's Guide

Software Version 10.1b

Contractor/manufacturer is:
Mentor Graphics Corporation
8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777.
Telephone: 503.685.7000
Toll-Free Telephone: 800.592.2210
Website: www.mentor.com
SupportNet: supportnet.mentor.com/
Send Feedback on Documentation: supportnet.mentor.com/doc_feedback_form

Table of Contents

Chapter 1
Using Questa SIM With Multiple Cores

Introduction

This document provides information on how to use Questa SIM to compile and simulate a design on multiple processors. You can do this on a computer that uses multiple CPUs or in a ring of multiple single-CPU computers.

Configuring Your Operating Environment

Prerequisites

- You must have python v2.3 (or later) installed on all machines.
- You must have the “modeltech” tree in your search path.

  This contains all the Questa SIM simulation and multi-core utilities, as well as the MPICH package which are required for multi-core simulation. The full path to the modeltech tree includes the platform name, which is either linux or linux_x86_64.

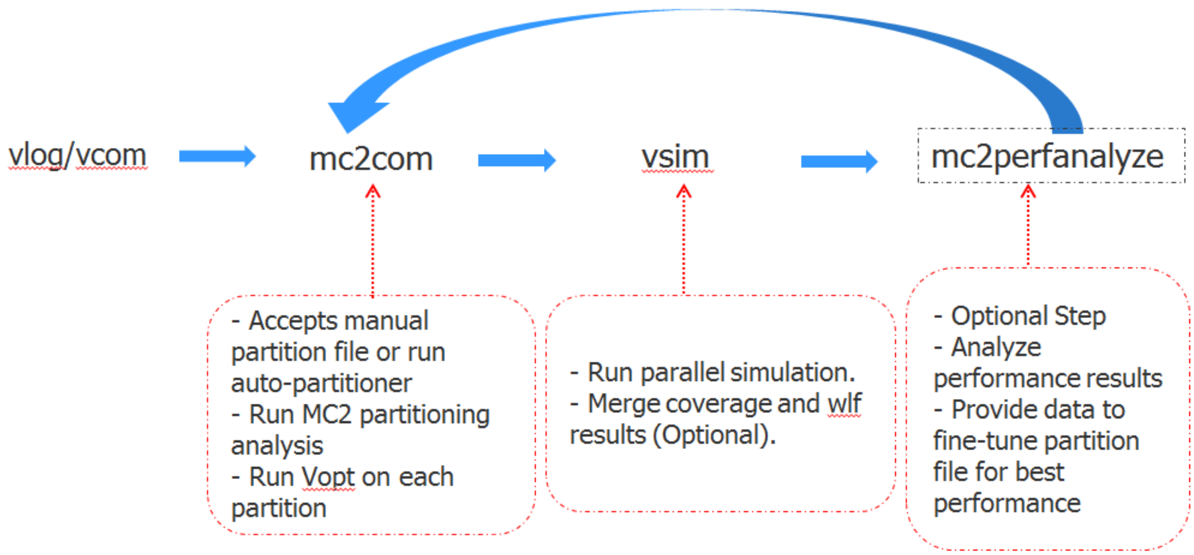
  Example — setup 32bit Questa SIM multi-core to run on Linux:

  ```
  Set path = (<path_to_your_mpich_package>/linux $path)
  ```

You are ready to run Questa SIM multi-core simulation as described in "Multi-core Simulation Flow" on page 6.

Multi-core Simulation Flow

Figure 1-1. Three Step Flow



Running multi-core simulation consists of three basic steps:

- Compile your design.

  Use similar vlog/vcom command options to compile your design for multi-core simulation, as you would do for Unisim.

- Compile for multi-core simulation.

  Multi-core simulation compilation consists of the following phases:

  - Auto/Manual partitioning ("Create a Partition File" on page 7)
  - Multi-core Partition Analysis ("Standalone Partition Analysis" on page 12)
  - Partition's vopt Optimization ("Standalone Partition vopt" on page 13)

  Use the mc2com command to run any of the three phases, together or individually, as suited in your flow.

- Run multi-core simulation.

  This step will run the simulation in multi-core simulation mode.

  For detailed explanation of multi-core simulation vsim options, see "Running Multi-core Simulation" on page 13.

For examples of using these steps to run Questa SIM multi-core simulation, refer to "Examples of Multi-core Simulation" on page 16.

Compiling for Multi-core Simulation

Create a Partition File

The partition file is a text file that defines how the design is to be partitioned for the Questa SIM multi-core simulation run.

You can create a partition file in any of the following ways:

- Running a design in multi-core simulation flow without a partition file will cause mc2com to perform auto-partitioning on your design. ("Partitioning With Auto-Partitioner" on page 8)
- Running a design in multi-core simulation flow with a modified auto-partition file or a manually created partition file. ("Partitioning Without Auto-Partitioner" on page 8)
- Run the auto-partitioner without multi-core compilation or vopt phases. Use this flow when you want to experiment with different partitions. ("Running Auto-Partitioner Phase Only" on page 9)
- Generate a partition file automatically using the mc2com command with profiler information. ("Using Auto-Partitioner With Profile Data" on page 10)

About Automatic Partitioning

The auto-partitioner partitions the design at an instance boundary based on the cost/weights associated with the instances. In its default mode, the auto-partitioner provides a static estimation of the cost of individual instances. While the partitioning resulting from this estimation approach is satisfactory in designs where the static performance estimation closely corresponds to the dynamic behavior of the design, it might not be effective in generating good partitioning for other types of designs.

The Questa SIM profiler can collect information on both performance and memory utilization. The auto-partitioner can use this information from the profiler to make partitioning decisions based on either performance alone, memory alone, or both.

If there are not enough instances with significant weight in the design to create load-balanced requested number of partitions, or if there are not enough instances that meets the criteria to put them on partition boundary, auto-partitioner won't generate requested number of partitions. In such cases, auto-partitioner will automatically decide and generate optimal number of load-balanced partitions.

Partitioning With Auto-Partitioner

If you do not have a partition file you can use auto-partitioning to create the partition file. The mc2com command will first run auto-partitioner, and then compile based on generated partition file, and run vopt on all partitions. Your design is ready for multi-core simulation after these steps complete.

Required Syntax:

mc2com <design_top> -o <optimized_design> -mc2numpart <num>

Example:

```
mc2com testbench -o top_opt -mc2numpart 3

Model Technology ModelSim SE-64 mc2com DEV Compiler 2003.05 Jul  7 2011

Top level modules:
        testbench

Analyzing design...
-- Loading module testbench
-- Loading module mod1
-- Loading module mod2
-- Loading module bottom
-- Loading module mod3

Running Auto-partitioner...
-- Auto-partitioner finished creating file 'top_opt.part' with 3
partitions

Running MC2 Partitioning Analysis...
-- Cleaning up interface directory
-- Running analysis
-- Compiling interface files

Running MC2 Vopt Optimization...
-- Running Vopt for partition 'master'
-- Running Vopt for partition 'p1'
-- Running Vopt for partition 'p2'
-- Vopt for all partitions completed successfully
```

Partitioning Without Auto-Partitioner

If you have a manually created partition file, or a modified auto-partition file, then you can skip the auto-partitioning step and directly start with multi-core compilation. For information on partition file see "Partition File Structure and Syntax" on page 49.

Required Syntax:

mc2com <design_top> -o <optimized_design> -mc2partfile <filename>

Example:

```
mc2com testbench -o run_mp -mc2partfile manual.part
```

```
Model Technology ModelSim SE-64 mc2com DEV Compiler 2003.05 Jul  7 2011

Top level modules:
        testbench

Analyzing design...
-- Loading module testbench
-- Loading module mod1
-- Loading module mod2
-- Loading module bottom
-- Loading module mod3

Running MC2 Partitioning Analysis...
-- Cleaning up interface directory
-- Running analysis
-- Compiling interface files

Running MC2 Vopt Optimization...
-- Running Vopt for partition 'master'
-- Running Vopt for partition 'p1'
-- Running Vopt for partition 'p2'
-- Vopt for all partitions completed successfully
```

Running Auto-Partitioner Phase Only

Optionally, you can run just auto-partitioner phase within mc2com, without performing the multi-core simulation compilation or vopt phases. Use this flow when you want to experiment with different partitions.

Required Syntax:

mc2com <design_top> -o <optimized_design> -mc2numpart <num> -mc2noanalyze \
    -mc2novopt

Example:

```
mc2com testbench -o top_opt -mc2numpart 3 -mc2noanalyze -mc2novopt

Model Technology ModelSim SE-64 mc2com DEV Compiler 2003.05 Jul  7 2011

Top level modules:
        testbench

Analyzing design...
-- Loading module testbench
-- Loading module mod1mc2inst.wlfmc2inst.wlf
-- Loading module mod2
-- Loading module bottom
-- Loading module mod3

Running Auto-partitioner...
-- Auto-partitioner finished creating file 'top_opt.part' with 3
partitions
```

Using Auto-Partitioner With Profile Data

To automatically create a partition file using profiler information, you must first simulate the design in unisim with the Questa SIM profiler enabled, using either the command-line interface (CLI) or the graphical user interface (GUI). Once the profiler database file has been created, you then auto-partition using the mc2com -mc2useprofile <file> option.

Required Syntax:

mc2com <design_top> -o <optimized_design> -mc2numpart <num> -mc2useprofile <file>.pdb

Example:

```
mc2com -o tb_3part -mc2numpart 3 tb -mc2useprofile tb3_profile.pdb
```

Run Profiler From The Command Line

From the command line, you can use the vsim -memprof command to enable memory profiler and the profile on -p command to enable performance profiler.

Example 1 — enables and saves only performance profiler data

```
vsim -c top -do "profile on -p; onfinish stop; run -a; \
profile save profile_perf.pdb;quit"
```

Example 2 — enables and saves only memory profiler data

```
vsim -memprof -c top -do "onfinish stop; run -a; \
profile save profile_mem.pdb;quit"
```

Note

The command “onfinish stop” is necessary to ensure that simulation does not quit before the profile save command.

Example 3 — enables and saves both performance and memory profiler data

```
vsim -memprof -c top -do "profile on -p; onfinish stop; run -a; \
profile save profile_both.pdb;quit"
```

Example 4 — enables and saves both performance and memory profiler data while disabling memory profiler after initial time zero initialization

```
vsim -memprof -c top -do "profile on -p; onfinish stop;run 1; \
profile off -m; run -a; profile save profile_both.pdb; quit"
```

Run Profiler From The GUI

1. Start vsim.
2. From the main menu, choose **Simulate** > **Start Simulation**.
3. From the Start Simulation dialog box, click the **Others** tab.
4. Under the Profiler region, select **Enable memory profiling** (if you want memory profiling). This enables memory profiling during elaboration.
5. Click the **Design** tab, select your compiled design, and click **OK**.
6. After the design loading is complete, choose either or both of the following from the main menu:
    - **Tools** > **Profile** > **Performance**
    - **Tools** > **Profile** > **Memory** (this will already be checked if you selected Enable memory profiling in Step 4, above)
7. Choose **Simulate** > **Run** > **Run 100** (or any other desired Run option).
8. When prompted with * "Are you sure you want to finish" * click **No**.
9. Save the profile data by entering the command **profile save** <filename> in the command window.
10. Choose **File** > **Quit**.

Run Auto-partitioning Using Performance Profile Data

Once you have obtained performance profile data, you can use it to perform automatic partitioning of your design. This mode of auto-partitioning uses the mc2com command with the profile argument (-mc2useprofile) and the partitioning argument (-mc2numpart).

Required Syntax

mc2com <design_top> -o <optimized_design> -mc2useprofile=perf <filename>.pdb \
-mc2numpart <num>

Example:

Create eight partitions using only performance profiler data.

```
mc2com -mc2useprofile=perf profile_perf.pdb -mc2numpart 8 \
         top -o part8
```

Run Auto-partitioning Using Memory Profile Data

Once you have obtained memory profile data, you can use it to perform automatic partitioning of your design. This mode of auto-partitioning uses the mc2com command with the profile argument (-mc2useprofile) and the partitioning argument (-mc2numpart).

Required Syntax

mc2com <design_top> -o <optimized_design> -mc2useprofile=mem <filename>.pdb \
-mc2numpart <num>

Example:

Create eight partitions using only memory profiler data.

```
mc2com -mc2useprofile=mem profile_mem.pdb -mc2numpart 8 top -o part8
```

Standalone Partition Analysis

You can just run the multi-core simulation partitioning analysis, without auto-partitioning and optimizing partitions, by specifying an existing partition file to mc2com and using the -mc2novopt option.

Required Syntax

mc2com <design_top> -o <optimized_design> -mc2numpart <num> -mc2novopt

Example:

```
mc2com testbench -o top_opt -mc2partfile XXX -mc2novopt

Model Technology ModelSim SE mc2com DEV Compiler 2003.05 Jul 12 2011

Top level modules:
        testbench

Analyzing design...
-- Loading module testbench
-- Loading module mod1
-- Loading module mod2
-- Loading module bottom
-- Loading module mod3

Running MC2 Partitioning Analysis...
-- Cleaning up interface directory
-- Running analysis
-- Compiling interface files

Skipping MC2 Vopt Optimization...
```

Standalone Partition vopt

You can just run vopt for all partitions, without running the auto-partitioning or partition analysis phases of mc2com using -mc2noanalyze option.

Required Syntax

mc2com <design_top> -o <optimized_design> -mc2numpart <num> -mc2noanalyze

Example:

```
mc2com testbench -o top_opt -mc2partfile XXX -mc2noanalyze

Model Technology ModelSim SE mc2com DEV Compiler 2003.05 Jul 12 2011

Top level modules:
        testbench

Skipping MC2 Partitioning Analysis...

Running MC2 Vopt Optimization...
-- Running Vopt for partition 'master'
-- Running Vopt for partition 'p1'
-- Running Vopt for partition 'p2'
-- Running Vopt for partition 'p3'
-- Vopt for all partitions completed successfully
```

Running Multi-core Simulation

Once the design has fully completed multi-core compilation, it is ready for multi-core simulation. You run the simulation by specifying the -mc2 option to vsim, along with any other necessary arguments described in "Multi-core Simulation Arguments for the vsim Command" on page 15.

By default, '-l <partition_name.log>' option is specified to each partition to create its own transcript file. you can override default transcript file name using -mc2vsimargs option for each partition. In addition, a transcript file containing output from all partitions combined is generated, following the normal conventions for Vsim transcript files. The default name for Vsim transcript file is "transcript". You can override default transcript filename using -l option at vsim command line.

If you do not specify -do option, then by default, "-do 'run -a; quit -f'" is specified to all the partitions. You can override it by specifying -do option at the command line. You can also override it for a specific partition by using -mc2vsimargs option. If you override -do option for individual partitions, make sure that all the partitions are executing same 'run' commands, otherwise simulation may hang.

When you partition a design, sometimes partitions end up with different minimum time resolutions. If this happens, you may see time resolution mismatch error after elaboration. Time resolution for all the partitions need to match in order to run simulation correctly. You can overcome this situation by specifying -t option with overall design's time resolution at vsim command line.

Example:

```
vsim -mc2 top_opt -do run.do

Reading /u/dvtbata/rkjain/mainline/modeltech/tcl/vsim/pref.tcl

# DEV

# Python 2.3.4
# Reading /u/dvtbata/rkjain/mainline/modeltech/tcl/vsim/pref.tcl
#
# //
# [MC2-STAT] Partition "p2", Proc Id = 1064
# vsim +nowarnTFMPC +nowarnTSCALE -do run.do -l p2.log -lib
/u/dvtbata/rkjain/mainline-ws/tests_mp/base/param/param19/work/_mc2/p2_work -
mc2partfile top_opt.part -c -wlf p2.wlf -wlfnoopt p2
# [MC2-STAT] Partition "p1", Proc Id = 1065
# vsim +nowarnTFMPC +nowarnTSCALE -do run.do -l p1.log -lib
/u/dvtbata/rkjain/mainline-ws/tests_mp/base/param/param19/work/_mc2/p1_work -
mc2partfile top_opt.part -c -wlf p1.wlf -wlfnoopt p1
# [MC2-STAT] Partition "master", Proc Id = 1063
# vsim +nowarnTFMPC +nowarnTSCALE -do run.do -l master.log -lib
/u/dvtbata/rkjain/mainline-ws/tests_mp/base/param/param19/work/_mc2/master_work -
mc2partfile top_opt.part -c -wlf master.wlf -wlfnoopt master
# Loading ./work.testbench(fast)
# Loading ./work.mod1(fast)
# Loading ./work.mod2(fast)
# Loading sv_std.std
# Loading work.p1_master__intf_1(fast)
# Loading work.p1__intf_3(fast)
# Loading work.p1__intf_8(fast)
# do run.do
# Loading ./work.testbench(fast)
# Loading ./work.mod2(fast)
# Loading ./work.mod3(fast)
# Loading sv_std.std
# Loading work.master_p1__intf_0(fast)
# Loading work.master__intf_2(fast)
# Loading work.master_p2__intf_4(fast)
# Loading work.master__intf_7(fast)
# do run.do
# Loading ./work.testbench(fast)
# Loading ./work.mod1(fast)
# Loading ./work.mod3(fast)
# Loading sv_std.std
# Loading work.p2_master__intf_5(fast)
# Loading work.p2__intf_6(fast)
# Loading work.p2__intf_9(fast)
# do run.do
#          0 testbench.mod1_inst p2=          3
#          5 testbench p1=          2
#          5 testbench.mod1_inst p2=          3
#         10 testbench p1=          2
#         10 testbench.mod1_inst p2=          3
#          0 testbench.mod1_inst.mod3_inst.inst3 p4=          7
#          0 testbench.mod1_inst.mod3_inst.inst2 p4=          6
#          0 testbench.mod1_inst.mod3_inst p3=          5
#          5 testbench.mod1_inst.mod3_inst p3=          5
#          6 testbench.mod1_inst.mod3_inst.inst2 p4=          6
#          7 testbench.mod1_inst.mod3_inst.inst3 p4=          7
#         10 testbench.mod1_inst.mod3_inst p3=          5
#          0 testbench.mod1_inst.mod2_inst.inst1 p4=        100
#          0 testbench.mod1_inst.mod2_inst p3=         20
```

```
#          5 testbench.mod1_inst.mod2_inst.inst1 p4=         100
#          5 testbench.mod1_inst.mod2_inst p3=         20
#         10 testbench.mod1_inst.mod2_inst.inst1 p4=         100
#         10 testbench.mod1_inst.mod2_inst p3=         20
#
# [MC2-STAT] Partition     p2 (2)=> Value Triggers: 0, User Sync Events: 0, Total
Sync Events: 15, Inout Sync Events: 0, Dataless Sync Events: 6, Idle Sync Events:
0, Max Sync Reloop: 2, Max Data Reloop: 1, Events: 13, Processes: 33, Suspend Opt
: 0
#
# [MC2-STAT] Partition     p1 (1)=> Value Triggers: 96, User Sync Events: 0, Total
Sync Events: 15, Inout Sync Events: 0, Dataless Sync Events: 6, Idle Sync Events:
0, Max Sync Reloop: 2, Max Data Reloop: 1, Events: 13, Processes: 23, Suspend Opt
: 0
#
# [MC2-STAT] Partition master (0)=> Value Triggers: 576, User Sync Events: 0,
Total Sync Events: 15, Inout Sync Events: 0, DatalessSync Events: 6, Idle Sync
Events: 0, Max Sync Reloop: 2, Max Data Reloop: 1, Events: 18, Processes: 43,
Suspend Opt : 0
#
# real   0m2.616s
# user   0m0.056s
# sys    0m0.014s
```

Multi-core Simulation Arguments for the vsim Command

The arguments specific to multi-core simulation operation with the vsim command are detailed in the vsim command reference section in this document. Refer to the *Questa SIM Reference Manual* for complete documentation of the vsim command.

Running Multi-computer Simulation

By default, multi-core simulation is run on multiple cores of single machine. However, simulation can also be run on multiple machines connected via network using vsim option -mc2network.

- vsim -mc2 <optimized_design> -mc2network <hostfile>

  The host file contains the list of hosts you want to run the executable on, and optionally specifies how many processes can run on each host. The maximum number of processes should not exceed the number of available cores on that machine, and you may want to specify fewer processes than the available cores to achieve a particular load balancing between the different hosts. Also, you should consider the memory capacity of each host, and the expected memory requirements of each multi-core simulation partition. If you do not specify a number of processes for each host, multi-core simulation will determine a maximum according to how many cores are present on each host.

  Example host file:

  ```
  host1:1 # Run 1 process on host1
  host2:4 # Run 4 processes on host2
  host3:2 # Run 2 processes on host3
  host4:1 # Run 1 process on host4
  ```

Examples of Multi-core Simulation

This section contains two examples of multi-core simulation:

- Normally Synchronized Example
- User-synchronized Example

Each example consists of the following:

- Design files
- Partition file
- Compiling and Simulating the design in multi-core simulation

Normally Synchronized Example

Design Files

top.v

```
`define     LOOP_NUM  123456

module top ;
reg[0:31]   val ;
reg         clk ;
integer     i ;

wire[31:0]  res1, res2, res3, res4 ;

initial begin
   #20  clk = 0 ;
   forever
       #50 clk = ~clk ;
end

initial begin
   #10  val = 32'hdeadbeef ;
   for( i = 0 ; i < `LOOP_NUM ; i = i + 1 ) begin
      #200 val = $random() ;
   end
   #400 $finish ;
end

child   c1( clk, val, res1 ) ;
child   c2( clk, res1, res2 ) ;
child   c3( clk, res2, res3 ) ;
child   c4( clk, res3, res4 ) ;

always @( val or res1 or res2 or res3 or res4 )
    $display( $stime,,,"%x--%x %x %x %x", val,res1,res2,res3,res4 ) ;
```

```
endmodule
```

child.v

```
module child( input wire clk, input[0:31]  in, output reg [0:31] out ) ;
always @(posedge clk)
    #5 out <= { in[31], in[30],in[29],
                  in[28], in[27],in[26],in[25],
               in[24], in[23],in[22],in[21],
               in[20], in[19],in[18],in[17],
               in[16], in[15],in[14],in[13],
               in[12], in[11],in[10],in[ 9],
               in[ 8], in[ 7],in[ 6],in[ 5],
               in[ 4], in[ 3],in[ 2],in[ 1], in[ 0] };


endmodule
```

Partition File

```
XXX :
----------
sync_control {
   Verilog = 1 ;
}

partition p1 {
   mod_inst = top.c1;
}

partition p2 {
   mod_inst = top.c2 ;
}

partition p3 {
   mod_inst = top.c3 ;
}

partition p4 {
   mod_inst = top.c4 ;
}

partition master {
   mod_inst = top ;
}
```

Compiling and Simulating the Design in Multi-core Simulation

```
vlib work
vlog -mfcu top.v child.v
mc2com -mc2partfile XXX top -o fast_top
```

```
vsim -mc2 fast_top
```

User-synchronized Example

Design Files

top.v

```
`define LOOP_NUM        123456
module top ;
reg[0:31]       val ;
reg             clk ;
integer         i   ;

wire[31:0]      res1, res2, res3, res4 ;

initial begin
   #20  clk = 0 ;
      forever
         #50 clk = ~clk ;
end

// MC2 EVENT SYNC CONTROL
event   sync_cntl ;

always @( posedge clk )
   #5 ->sync_cntl ;

initial begin
   #10  ->sync_cntl ;
   forever #200 ->sync_cntl ;
end

// END EVENT MC2 SYNC CONTROL

initial begin
   #10  val = 32'hdeadbeef ;
   for( i = 0 ; i < `LOOP_NUM ; i = i + 1 ) begin
      #200 val = $random() ;
   end
   #400 $finish ;
end


child   c1( val , res1 ) ;
child   c2( res1, res2 ) ;
child   c3( res2, res3 ) ;
child   c4( res3, res4 ) ;

always @( val or res1 or res2 or res3 or res4 )
   $display( $stime,,,"%x--%x %x %x %x", val,res1,res2,res3,res4 ) ;
```

```
endmodule
```

child.v

```
module child(
      input[0:31]  in, output reg [0:31] out ) ;

//  MC2 : clk replicated
reg            clk ;
initial begin
   #20  clk = 0 ;
   forever
      #50 clk = ~clk ;
   end

always @(posedge clk)
   out <= #5 { in[31], in[30],in[29],
               in[28], in[27],in[26],in[25],
               in[24], in[23],in[22],in[21],
               in[20], in[19],in[18],in[17],
               in[16], in[15],in[14],in[13],
               in[12], in[11],in[10],in[ 9],
               in[ 8], in[ 7],in[ 6],in[ 5],
               in[ 4], in[ 3],in[ 2],in[ 1], in[ 0] };

// MC2 SYNC EVENT
event   sync_cntl ;
always @( posedge clk )
   #6 ->sync_cntl ;

initial begin
   #10 ->sync_cntl ;
   forever
      #200 ->sync_cntl ;
end

endmodule
```

Partition File

```
XXX:
----
sync_control {
    Verilog = 1 ;
}

partition p1 {
    mod_inst = top.c1 ;
    sync_event top.c1.sync_cntl ;
}

partition p2 {
    mod_inst = top.c2 ;
    sync_event top.c2.sync_cntl ;
}
```

```
partition p3 {
    mod_inst = top.c3 ;
    sync_event top.c3.sync_cntl ;
}

partition p4 {
    mod_inst = top.c4 ;
    sync_event top.c4.sync_cntl ;
}

partition master {
    mod_inst = top ;
    sync_event top.sync_cntl ;
}
```

Compiling and Simulating the Design in Multi-core Simulation

```
vlib work
vlog -mfcu top.v child.v
mc2com -mc2partfile XXX top -o fast_top

vsim -mc2 fast_top
```

Globally Shared Memory Objects

Globally shared objects are supported by multi-core simulation. In general, a globally shared object is a multi-dimensional HDL object of the form:

```
reg[0:31] MEM[0:32767] ;
```

that you must access directly by using Hierarchical references, such as:

```
top.MEM
```

where the references (read or write) occur in multiple partitions.

Defining A Globally Shared Memory Object

By default, multi-core simulation allows implementation of memory as globally shared objects, unless you explicitly instruct the mc2com partitioner to disable this capability. You can disable global memory using the -mc2noglobalmem argument to the mc2com command:

```
mc2com top -mc2numpart 3 -mc2noglobalmem -o top_opt
```

When -mc2noglobalmem is specified, the auto-partitioner will create partitions which avoid cross-partition references to memory arrays. If the switch is specified together with a partition file (no auto-partitioning), then mc2com will report an error if there are any cross-partition memory references.

Usage Notes

To use globally shared objects, you must be sure that they meet the following criteria:

- Contention-free — accessing the shared object is done in a contention-free manner in the same time step.
- Sequentially neutral — accessing the shared object does not depend on the access sequences from other partitions in the same time step.
- Exclusive access — accessing the "word address" of the form:

  ```
  MEM[ A ][ 21:40 ] = expr1 ; // partition #1
  expr2 = MEM[ A ][ 41:60 ] ; // partition #2
  ```

  at the same time step can produce unexpected results. Even though the accesses look mutually exclusive, they are not word exclusive. This restriction applies only when one or more accesses is a memory write.
- Event triggering — the object cannot be used as trigger if the writer of the shared memory object resides in a different partition, an event trigger of the following form does not work:

  ```
  @top.MEM
  ```

Merging Partition's UCDB and WLF Files

By default, each partition produces individual files for WLF (<partition_name>.wlf) and UCDB (<partition_name>.ucdb). Each file has data specific to its partition's hierarchy. Vsim provides options to automatically merge individual partition files to generate design-wide data files.

The **-mc2mergewlf <filename>** option automatically merges WLF files from all the partitions and creates a single design-wide WLF file. Optionally, you can also merge all the partitions' WLF files manually, as follows:

```
wlfman merge -mark -opt -o <filename> <partition_name1>.wlf \
      <partition_name2>.wlf <optional_args>'
```

Use the **-mc2mergeucdb <filename>** option to save a UCDB file for each partition, merging the UCDB files from all the partitions, and creating a single design-wide UCDB data file. When you specify this option, you don't need to explicitly save UCDB files, as the simulator will automatically save the UCDB files on exit.

Optionally, you can also merge all the partitions' UCDB files manually, as follows:

1. add vsim command **coverage save -onexit <partition_name>.ucdb** for each partition at the end of simulation
2. after simulation completes, use the **vcover** utility to merge the files:

```
vcover merge -combine <filename> <partition_name1>.ucdb \
      <partition_name2>.ucdb <optional_args>
```

Debugging Standalone Partitions with VCD Flow

It is possible to debug and run a standalone partition without communicating with (or being dependent on) other partitions. You can do this by recording Value Change Dump (VCD) values for a given partition.

To be able to run standalone partition, the simulator needs to know what values are arriving at boundary from other partitions and at what time. Once it has this information, simulation can be run on a standalone partition without needing to run other partitions at the same time.

Usage Notes

VCD debugging consists of a fully automated two step flow:

1. Run full multi-core simulation and record VCD values. (VCD Recording)
2. Select the partition you want to run and re-simulate it using earlier VCD recorded values. (VCD Resimulate)

Step 1—VCD recording

While running full multi-core simulation, you need to specify the vsim -mc2vcddump option (refer to "Multi-core Simulation Arguments for the vsim Command" on page 15). This option will automatically append required VCD commands in each partition to dump correct VCD values.

After the simulation is run, you will see various VCD files generated for each boundary instance.

Step 2—VCD Resimulate

After recording VCD values from multi-core simulation, you are ready to run single partition in standalone mode. You need to pick a partition and run vsim using the -mc2sal option. When you specify this option to vsim, it automatically turns off compilation and full flow multi-core simulation, and runs the simulation only for the given partition. It also automatically appends required VCD commands to the vsim command. It also changes the default values for -l and -wlf options (notice the suffix _sal).

Examples

The following examples show how to use vsim command options for VCD debugging.

- Run multi-core simulation in standalone mode for partition p1:

```
vsim -mc2 <vsim args> -mc2vcddump
vsim -mc2 <vsim args> -mc2sal p1
```

- Checkpoint-Restore with VCD

  It is also possible to use checkpoint-restore with VCD standalone flow:

```
vsim -mc2 <vsim args> -mc2sal p2 -mc2vsimargs=p2 "-do 'run 100; \
      checkpoint p2.chkpt; run -a; quit -f '"
vsim -mc2 <vsim args> -mc2sal p2 -mc2vsimargs=p2 "-l results/p2_sal1.log \
      -restore p2.chkpt"
```

Limitations

The current VCD debugging system has the following limitations:

- Reg type ports at boundary are not supported.
- Bit-select, part-select ports at boundary are not supported.
- Complex SystemVerilog (SV) and VHDL types (except std_logic) are not supported.
- Objects (hrefs) crossing a partition boundary cannot use this flow.
- Designs with user-defined events are not supported.

Debugging Simulation Mismatches With Unicore Flow

You can use this flow for debugging purposes to do a self-test between multi-core simulation and Unisim runs, in order to find out if multi-core simulation and Unisim simulations produce different results.

This is an optional mode of multi-core simulation operation where you can run the entire design as a partition in addition to the other multi-core simulation partitions of the design (refer to the -mc2unicore argument described in "Multi-core Simulation Arguments for the vsim Command" on page 15).

For example, if you have partitioned the design into three partitions, then specifying a unicore mode would actually run four partitions, where the fourth partition is the full design itself. The multi-core simulation operation would run as it is (with three partitions), but the entire design would also be loaded as another partition (that is, a fourth partition). This fourth partition is referred to as the unicore partition. The conventional multi-core simulation partitions communicate normally and move forward with simulation. The unicore partition is independently running the full design (as the last partition), but it is run in time lockstep with other multi-core simulation partitions. Unicore partition communicates with all multi-core simulation partitions to check on time lockstep and multi-core simulation partition boundary values.

Note
Note that "unicore" is a reserved word—you cannot use it for a partition name.

Usage Notes

- Unicore debugging is an optional mode of multi-core simulation operation that you invoke with the -mc2unicore argument to the mc2com command:

  ```
  mc2com top -o run_mp -mc2partfile part01.part -mc2unicore all
  ```

  For more information on this argument, refer to the -mc2unicore {all | out | in} argument described in "Multi-core Simulation Arguments for the vsim Command" on page 15.

- General options for mc2com and vsim which are passed to all partitions will also be passed to the unicore partition. If necessary, you can specify arguments specific to the unicore partition using the -mc2voptargs=unicore option for mc2com, or -mc2vsimargs=unicore option for vsim.

- You can specify runtime option -mc2nozerochk to avoid getting errors for value mismatches at time 0.

Unicore Error Checking

Using this flow, you can perform error checking between multi-core simulation and Unicore modes along with the simulation, which immediately points out wherever different behavior occurs between modes. The Unicore flow performs two kinds of error checking to ensure that Unicore and multi-core simulation are in same state at the end of each time unit:

- After resolving the next time delta for multi-core simulation system, the simulation checks that this delta matches with next time delta for Unicore. Any difference found indicates that multi-core simulation and Unicore simulation are not progressing in same state. This difference may or may not be significant, based on how and when it occurs.

- At the end of each time unit, the simulation verifies that current values of all partition boundary ports are matching with their counterpart Unicore values. Any difference found indicates that something went wrong in multi-core simulation, and needs attention. Value differences at Time 0 may be exception.

If multi-core simulation detects a discrepancy with a Unicore run, it reports one of the following kinds of message:

- Delta time mismatch
  - o If next time value is different, it displays a message as such.

- If next time in multi-core simulation is less than in Unicore, a warning message is displayed. It is okay to have extra events in multi-core simulation flow than in the Unicore run. Such messages are reported as warning, so user can still investigate them if needed. For example:

```
** Warning: (vsim-8596) time 0, next time delta mismatch between MC2
core (dt = 1) and Uni core(dt = 10)
```

- If next time in multi-core simulation is higher than in Unicore, an error message is displayed. This is always an error that occurs because of a problem that you need to examine and correct. For example:

```
** Error: (vsim-8596) time 10, next time delta mismatch between MC2
core (dt = 105) and Uni core(dt = 7)
```

- Boundary port mismatch

  - If at any time, there is a value mismatch for partition boundary port in Unicore and multi-core simulation, an error message is displayed. This error will also occur for any hierarchical ref usage across partition. For example:

```
** Error: (vsim-8647) Value mismatch between unicore (val = 0) and
partition ’p1’ (val = x) for port ’tb.U1.a_n1.z’ (id = 0).
Time: 101 ns Iteration: 0 Region: /tb
```

  - Sometimes the multi-core simulation value has a mismatch at time 0. This can happen because of different events ordering resulting from partitioning. It may or may not be an issue. You can use the vsim -mc2nozerochk command to convert these errors to warnings at time 0. For example:

```
** Warning: (vsim-8647) Value mismatch between unicore (val
=xxxxxxxxxxxxxxxx) and partition ’p1’ (val = 0000000000000001) for
port ’top.sub1.so’ (id = 0).
Time: 0 ns Iteration: 3 Region: /top File: test.v
```

  - Sometimes when an inout port is at partition boundaries and some of the elements of the port are undriven. By LRM rules it should be driven by its initial value of the port. With synthesis tools we don't create an extra driver. Also unisim under some optimizations may stop creating the extra driver. To remove the simulation vs. synthesis mismatch under remaining cases, use the unisim -defaultstdlogicinittoz option. Because of partitioning these optimizations might not kick in with multi-core simulation run, and there might a mismatch between unisim and multi-core simulation. Use the -defaultstdlogicinittoz option with both unisim and multi-core simulation to verify if the simulation mismatches are removed.

Collecting Multi-core Simulation Related Statistics from Unisim

This section describes the flow to collect data from unisim (regular vsim simulation) runs that can be used for early qualification of a design for multi-core simulation. This flow is intended to collect the performance profiler database and the WLF log of the ports of the top-level and computationally significant instances.

Information Useful for Multi-core Simulation Evaluation

- Performance profile database obtained from vsim unisim run
- WLF logs

Prerequisites

- You already have a good flow running through vopt and unisim simulation.

Information Collection Flow for Multi-core Simulation

1. Collect the performance profiler data from a regular vsim run.

```
vsim -c top_opt -do "profile option keep_unknown on; profile on -p; \
      onfinish stop; run -a; profile save profile_perf.pdb;quit"
```

2. To view the profiler structural report you may run the following command. For details on the options supported by 'profiler report' command refer to the Questa SIM user documentation. Note that generating the structural report is optional and is not a necessary step of the mc2collect flow.

```
vsim -c -do "profile open profile_perf.pdb; profile report -structural"
```

3. Run the vopt on the design with the -mc2collectinfo option (see vopt Option -mc2collectinfo Details).

   This option first shortlists instances, then applies +acc=p to preserve the ports of these instances, and produces a vsim do file (mc2collect.do) with 'add wave' commands to log these ports in the WLF. If the profile database (obtained in step1) is provided to the vopt, then the short list of instances is based on the profile data. In the absence of the performance profiler database, the short list is generated by walking the Vtree (Breadth First walk) and selecting the top level instances. Currently the shortlist is limited to a maximum of 30 most intensive instances and can be overridden using the -mc2instlimit option. The mc2collect.do file and the instance list will be located in the _mc2collect/ directory. Following are some examples of using this option.

```
vopt top -o top_portlog -mc2collectinfo=profile_perf.pdb [-mc2instlimit N]
```

```
vopt top -o top_portlog -mc2collectinfo  [-mc2instlimit N]
```

4. Run vsim again with optimized design from step 2 using the *mc2collect.do* file

   The newly created mc2collect.do file contains add wave command(s) generated by the -mc2collectinfo option. Note that mc2collect.do should be included in front of any user commands that are needed for the normal simulation of the design. The simulation will produce the .wlf file.

```
vsim -c top_portlog -do _mc2collect/mc2collect.do -wlf mc2inst.wlf
```

5. Send following files to the factory: profile_perf.pdb, mc2inst.wlf files, and _mc2collect dir.

Note

This flow is not applicable to the Pre-optimized Design Unit (PDU) flow. Some modifications are needed to make this work if PDUs are present in the design.

vopt Option -mc2collectinfo Details

The -mc2collectinfo option prepares a short list of instances and internally applies +acc on the ports of these instances to preserve them from optimization. It produces a do file (_mc2collect/mc2collect.do) containing the add wave commands on the ports of these instances that can be provided to the vsim for WLF logging. The short list of instances is determined based on the input performance profile database provided as -mc2collectinfo=. If the profile database is not provided (plain -mc2collectinfo) then top-level instances are short listed.

The number of instances short listed can be controlled by the -mc2instlimit option. By default, ports of 30 instances are chosen for logging. If the profile data base is provided, the default number 30 is usually good enough in capturing the port activity of the computationally significant instances (as the instances with most number of hits are chosen). However, if the profile database is not specified then based on design hierarchy, 30 might be insufficient and you may have to specify higher number through the -mc2instlimit option.

Also, note that since we are applying +acc=p on the ports of the chosen set of instances, some of the vopt optimizations will be different from the regular vopt run on these instances.

Analyze Collected Information

Refer to sections "Analyzing Unisim Data", and "Unisim Data Analysis Reports", for details.

Simulation Performance Data

By default, Questa SIM always displays basic information at the end of an multi-core simulation. For example:

```
# [MC2-STAT] Partition master (0)=> Value Triggers: 90, User Sync Events:
0, Total Sync Events: 389085, Inout Sync Events: 0, Dataless Sync Events:
385122, Idle Sync Events: 76, Max Sync Reloop: 1, Max Data Reloop: 0,
Events: 770507, Processes: 389263, No More Event Opt : 0
```

where

- Value Triggers — Total number of value triggers to be sent to other partition.

- <Various> Sync Events — Number of various types of sync events that were executed during simulation.
- Max Sync Reloop — Maximum number of times the Sync FSM relooped in a single time step. Usually, a larger number means changes are being “dripped” across a partition, which indicates a potential performance issue.
- Max Data Loop — The largest sync FSM reloop count where data exchange really took place. If this number is large, it is also a sign of potential performance problem.
- Events — Number of events executed in the partition.
- Processes — Number of process executed in the partition.

Obtaining a Detailed Performance Report

You can obtain a more elaborate report on multi-core simulation performance in the following way:

- Pass option -mc2commstat directly to vsim to produce communication statistics report at the end of the multi-core simulation run.

```
vsim -mc2 top_opt -mc2commstat
```

This method reports performance information at the end of an multi-core simulation run. The report contains data on command statistics, execution time, and communication sizes, as described below.

Command Statistics

The first part of the report displays a count of the various types of communications that occurred in the multi-core simulation run, as in the following example:

```
# (1) Command Statistics
#              Command            Send            Recv
# --------------------------------------------------
#                   Go               3               0
#                 Sync               0               3
#           Inter-Proc               0               0
#                  End               3               0
#                 Data         1167027         1167030
#            No-Events               0               0
#           Next-Delta         1155243         1155366
#               Active               0               0
#     Next-Time-Iter-Q             123               0
#                 Idle               0               0
#              Cfg-Msg               0               3
#             Cfg-Sync               3               3
# --------------------------------------------------
```

The report from the master partition provides the most important data, since it is the most communication-loaded partition.

Execution Time Distribution

The second part of the report shows the respective times taken for simulation and communication in a given multi-core simulation run, as in the following example:

```
#   (2) Execution Time Distribution
#
#       Total Time: 8.81s
#          Sim Time:  6.68s (75.86%)
#          MC2 Overheads:  2.13s (24.14%)
#              Idle Time: 0.01s (0.12%)
#              Dataless Comm: 0.65s (7.35%)
#              Data Comm:   1.06s (11.98%)
#                  Send Prep:       0.44s (5.01%)
#                  Send Data:       0.09s (0.98%)
#                  Recv Data:       0.38s (4.36%)
#                  Recv Apply:       0.14s (1.63%)
#              Time Sync:  0.41s (4.69%)
#                  Recv Time:       0.24s (2.78%)
#                  Misc Time:         0.17s (1.91%)
```

The simulation run time distribution is categorized into the following:

- Time actually running simulation
- Time actually doing multi-core simulation sync, which consists of:
    - Time to prepare the data to be sent
    - Actual send time
    - Actual receive time
    - Time to process the received data from other partitions

Inter-Partition Data Communication

The third part of the report displays a column listing of the size of communications that occurred in the multi-core simulation run, as in the following example:

```
# Partition master (Id = 0): Inter-partition Data Exchange Summary:
#                          Sent To                                   Received From
# Part        Words      Count   %Ave      Total  %Rate        Ports      Count   %Ave      Total  %Rate
# -----------------------------------------------------------------------------------------------------
#    1      1042779    347593    3.00    2085560   16.7 |       347593     347593   1.00    2085560   16.7
#    2      1042782    347594    3.00    2085560   16.7 |       347593     347593   1.00    2085560   16.7
#    3      1042782    347594    3.00    2085560   16.7 |       347593     347593   1.00    2085560   16.7
#    4      1042782    347594    3.00    2085560   16.7 |       347593     347593   1.00    2085560   16.7
# -----------------------------------------------------------------------------------------------------
```

Where the “Part” column shows the partition ID number.

Also, note that there are two types of data in this part of the report:

- Sent Data (“Sent To”), containing the following columns:
    - Words: Total number of words sent to another partition.
    - Count: Total number of times communication has been sent with word packets (non-empty packets). All sending words from a partition are stuffed into a single packet, and then sent across to another partition.
    - %Ave: Average number of words sent per packet.
    - Total: Total number of times communication has been sent, with and without words, empty packet. If there is no data to send, partition still needs to communicate that. This is called an empty packet.
    - %Rate: Percentage of good and meaningful data communication sent. The higher this number, the better the communication.

- Received Data (“Received From”), containing the following columns:
    - Ports: Total number of port’s data received by this partition. Each boundary communication port may contain various number of words based on their data size.
    - Count: Total number of times communication has been received with ports (non-empty packets).
    - %Ave: Average number of ports received per packet.
    - Total: Total number of times communication has been received, with and without ports, empty packet.
    - %Rate: Percentage of good and meaningful data communication received. The higher this number, the better the communication.

In this example, the Sent To side shows the master partition sent 1042779 words of data to partition #1. However, only 16.7% of the data exchange contained good data (83.3% are empty exchange). On average, the exchanged data size is 3 words.

The Received from side shows that master received data for 347593 boundary ports from partition #1. Only 16.7% of the sync contain good data (83.3% empty), and the average size of the exchange is 1 port. The size of the actual exchange is not reported here; however, it can be found in similar report for partition #1.

From this report, you can evaluate the quality of the partitioning and simulation. In the example report shown above, the %Rate values are very low, which suggest that multi-core simulation may be over-syncing, or was ask to do too many types of sync. The aggregated low %Rate also suggested that the test+DUT may have serialized execution behavior, that behavior is exposed by the partitioning. Either the design is not a good fit for multi-core simulation, or the partitioning can use some improvements.

Saving Detailed Performance Data

The -mc2savestat option (given to the vsim command) captures the run-time performance data of all the partitions into a single SQL database (mc2data.mdb). Use the command mc2perfanalyze to analyze the database. Database analysis is discussed in "Multi-core Simulation Performance Data Analysis" on page 31. The -mc2savestat switch collects more comprehensive information than -mc2commstat and the results presented through the stand-alone mc2perfanalyze tool is better readable than the reports generated by -mc2commstat. The type of information collected by -mc2savestat is documented in "Type of Data Captured" on page 31.

This functionality can be invoked by passing option -mc2savestat[=filename] to vsim. For example:

```
vsim -mc2 -mc2savestat <optimized_design>
```

The type of reports that can be produced from the multi-core simulation SQL database using the mc2perfanalyze tool are documented in the Section Multi-core Simulation Performance Data Analysis.

Multi-core Simulation Performance Data Analysis

Multi-core simulation collects run time data during the course of the simulation and dumps the data into a SQL database. This page gives information about the type of data collected, enabling this functionality and utility to produce reports based on the contents of the database.

Enabling Data Capture Functionality

Currently, this functionality can be enabled by the option -mc2savestat[=filename] to the vsim command. Note that this option has to be passed to all the partitions. Each partition collects the data during the simulation and at the end synchronizes to write the collected data into a SQL database.

```
vsim -mc2 top_op -mc2savestat
```

Note that this option captures more data than -mc2commstat. Turning on this option will also internally enable -mc2commstat functionality.

Type of Data Captured

For each partition the -mc2savestat option captures and saves the following data into a single SQL database. In the absence of the user specified filename the data is saved to the SQL file named **mc2data.mdb**.

- Time distribution (sim.time, comm.time...)

- Data traffic information
- Commands exchange summary
- Event statistics (such as, number of value change triggers, number of process triggered, number of idle sync events, and so on)
- Port trigger information
- Instance trigger information
- Instance trigger pattern

Report Generation Tool

The standalone utility, *mc2perfanalyze*, produces formatted text reports from the raw data collected during the multi-core simulation (using -mc2savestat) and unisim run (using -mc2collectinfo flow). The utility can be used to produce several different types of reports. The complete list of supported options is listed in "mc2perfanalyze Options" on page 32.

mc2perfanalyze Options

Detailed description of the mc2perfanalyze command and its arguments, see the mc2perfanalyze command reference page.

Analyzing Multi-core Simulation Data

The utility mc2perfanalyze can produce several types of reports from the raw data in the multi-core simulation SQL database. For new users of the tool this can be overwhelming. Following the step-by-step approach below can help users understand these reports and identify the performance issues.

- Three main factors affect multi-core simulation performance: Load balancing, concurrency, and communication between partitions. Start with the command:

  ```
  mc2perfanalyze mc2data.mdb -analyze
  ```

  The mc2perfanalyze command produces a summary report that shows any anomalies in the above 3 areas. Another useful approach is getting partition connectivity information using:

  ```
  mc2perfanalyze mc2data.mdb -info
  ```

  Where the resulting output is formatted in a table.

- To take deeper look at load balancing try:

  ```
  mc2perfanalyze mc2data.mdb -time
  ```

  This reports a comparison table showing how much time each partition spent executing different tasks, such as active simulation, communication and so on.

To produce a similar table that compares the partitions against different counters, including the Processes count (which is the number of processes executed by each partition) use:

```
mc2perfanalyze mc2data.mdb -event
```

Comparing Processes counters of partitions can also reveal load balancing, but it may not be as precise as comparing simulations times.

- To understand concurrency issues try:

  ```
  mc2perfanalyze mc2data.mdb -time
  ```

  Compare the Idle Time of each of the partitions; if the idle times are significant then it indicates a lack of concurrency between the partitions. Partitions with high No Comm. Sync Events/Idle Sync. Events in the table generated by:

  ```
  mc2perfanalyze mc2data.mdb -event
  ```

  are the individually active partitions. Using -pattern switch produces additional information about the concurrent triggering pattern of the instances at the partition boundary, which is useful for fine-tuning partitioning for higher concurrency.

- The communication between partitions involves data and data-less communication. To fetch more details on data-communication try:

  ```
  mc2perfanalyze mc2data.mdb -info
  ```

  This reveals the number of ports between the partitions. Number of ports between partitions is a very rough indication of the data communication between partitions. mc2perfanalyze mc2data.mdb -data reports the amount of data that is exchanged between different pairs of partitions. One of the important fields in the data report is the 'Count' column, which shows the number of synchronizations between partitions that involved actual transfer of data. A high number of 'Count' indicates heavy communication between partitions. Data-less communication occurs at delta and time advances.

In addition, when we have couple of partitions communicating data with each other, the other partitions engage in data-less communication. A varying percentage of data-less syncs is a cause of concern, as it indicates an uneven distribution of load. The data-less sync information can be viewed by mc2perfanalyze mc2data.mdb -event.

Multi-core Simulation Data Analysis Reports

Use mc2perfanalyze can be used to analyze the runtime information previously captured by the -mc2savestat option in mc2data.mdb. Far more options are supported to analyze multi-core simulation data than unisim data. Below are some example invocations of the commands. Note that most switches are optional, in the absence of these switches the tool produces are applicable reports. The switches -sortorder, -sortby, and -limit can be used to reformat and limit the size of the tables.

```
mc2perfanalyze mc2data_m256.db -analyze
##########################################################################
                              MC2 ANALYSIS REPORT
##########################################################################
# LOAD BALANCING ANALYSIS
Partitions NOT balanced.
        Based on active simulation times, following partitions are overloaded :
                'master' (sim time (368.59s) >> min sim time (138.67s))
                'p2' (sim time (326.18s) >> min sim time (138.67s))
        Based on processes count in each partition, following partitions are
overloaded :
                'p2' (Processes (1.27B) >> min Processes (453.29M))

# CONCURRENCY ANALYSIS
Possible lack of concurrency between partitions.
        The following partition(s) were found to running alone during simulation,
while other partitions were idling : 'master'

# COMMUNICATION ANALYSIS
The following partition pair(s) communicated data a high number of times :
        'master' and 'p1' (7.78M)
The least communicated pair was :
        'master' and 'p2' (3.85M)

Significant difference in cross-partition data communication activity in
partitions. Number of sync events with data communication are:
        master (max): 10.81M, p2 (min): 7.58M.

# PARTITION ORDERING
For optimal performance, consider reordering partitions in partition file as
follows : master, p1, p2 (Suggestion is based on runtime load).
```

The -info switch shows the connectivity information between partitions:

```
mc2perfanalyze mc2data.mdb -info
#########################################################
             Partition Connectivity Info
#########################################################
        Partitions|            master|                p1|                p2|
------------------|------------------|------------------|------------------|
            master|                --|      15 sendports|     263 sendports|
                p1|       3 sendports|                --|                --|
                p2|     166 sendports|                --|                --|
```

Multiple options can be specified simultaneously:

```
mc2perfanalyze mc2data.mdb -time -event
################################################################################
                       Execution Time comparison (in seconds)
################################################################################
Quantity                  |master                    |p1                        |p2                        |
--------------------------|--------------------------|--------------------------|--------------------------|
Total Time                |939.24s (100.00%)         |939.24s (100.00%)         |939.24s (100.00%)         |
--------------------------|--------------------------|--------------------------|--------------------------|
    Sim Time              |    368.59s (39.24%)      |    138.67s (14.76%)      |    326.18s (34.73%)      |
--------------------------|--------------------------|--------------------------|--------------------------|
    MC2 Overheads         |    570.65s (60.76%)      |    800.57s (85.24%)      |    613.06s (65.27%)      |
--------------------------|--------------------------|--------------------------|--------------------------|
        Idle Time         |        0.00s (0.00%)     |        477.79s (50.87%)  |        344.26s (36.65%)  |
--------------------------|--------------------------|--------------------------|--------------------------|
        Dataless Comm     |        381.62s (40.63%)  |        195.90s (20.86%)  |        168.91s (17.98%)  |
--------------------------|--------------------------|--------------------------|--------------------------|
        Data Comm         |        94.00s (10.01%)   |        109.57s (11.67%)  |        74.81s (7.96%)    |
--------------------------|--------------------------|--------------------------|--------------------------|
            Send Time     |            6.94s (0.74%) |            4.45s (0.47%) |            3.18s (0.34%) |
--------------------------|--------------------------|--------------------------|--------------------------|
            Recv Time     |            64.10s (6.82%)|            63.10s (6.72%)|            60.76s (6.47%)|
```

PrepSend Time	12.36s (1.32%)	3.30s (0.35%)	2.32s (0.25%)
ApplyRecv Time	10.59s (1.13%)	38.72s (4.12%)	8.56s (0.91%)
Time Sync	95.03s (10.12%)	17.31s (1.84%)	25.07s (2.67%)
Recv Time	20.83s (2.22%)	0.00s (0.00%)	2.67s (0.28%)
Misc Time	74.20s (7.90%)	17.31s (1.84%)	22.39s (2.38%)
Active Time	549.18s (58.47%)	229.01s (24.38%)	401.25s (42.72%)
Waiting Time	390.05s (41.53%)	710.22s (75.62%)	537.99s (57.28%)

```
################################################################################
                          Partition stats comparison
################################################################################
```

Stats	master	p1	p2
Value Change Triggers	46.62M (78.56%)	7.82M (13.17%)	757.62K (1.28%)
Events	213.67M (58.73%)	54.61M (15.01%)	24.63M (6.77%)
Processes	764.30M (25.92%)	459.90M (15.60%)	1.27B (43.11%)
User Sync Events	0	0	0
Total Sync Events	62.38M (100.00%)	43.64M (100.00%)	48.06M (100.00%)
Inout Sync Events	0 (0.00%)	0 (0.00%)	0 (0.00%)
Dataless Sync Events	51.57M (82.67%)	32.84M (75.26%)	40.48M (84.22%)
Data Sync Events	10.81M (17.33%)	10.79M (24.74%)	7.58M (15.78%)
No-Comm Sync Events	59.78M (100.00%)	0 (0.00%)	0 (0.00%)
Max Sync Reloop	250	228	228
Max Data Reloop	34	34	34
Suspend Opt	46.34M	0	0

Option **-inst** will produce the trigger information of the instances in the boundary between each partition pair. Use the options **-part** and **-topart** to specify a partition pair. In the table below the columns mean the following:

- **instance_name** — Boundary instance name between the partition pair (master and p1).
- **nSendPorts** — Number of send ports in the specific instance between the partition pair.
- **ports_trig_count** — Total number of times the ports of the instances triggered (changed value) due to the activity in the instance.
- **instance_trig_count —** Number of times the instance triggered, this number will be less than or equal to the ports_trig_count.
- **%individual triggers —** Is indicative of ports in an instance triggering independently of each other. Higher percentage indicates that ports of the instances are triggering independently of each other. This number can range from 100% to 100/nSendports.

```
mc2perfanalyze mc2data.mdb -inst -part master -topart p1
```

```
##############################################################################
Instance trigger table from 'master' to 'p1' (sorted by: 'instance_trig_count')
      (Note: Higher individual triggers implies that ports of the instance are
             triggering independently)
##############################################################################
instance_name |nSendPorts |ports_trig_count |instance_trig_count |%individual |
              |           |                 |                    |   triggers |
--------------|-----------|-----------------|--------------------|------------|
tbleon.ram0   |         5 |            7324 |               6743 |     92.07% |
tbleon.ram1   |         5 |            7327 |               6743 |     92.03% |
tbleon.ram2   |         5 |            7322 |               6743 |     92.09% |
```

Option **-port** will produce the trigger information of the boundary ports in each partition pair. The partition pair can be specified by the options **-part** and **-topart**.

```
mc2perfanalyze mc2data.mdb -port -part master -topart p1

###################################################################
Port trigger table from 'master' to 'p1'  (sorted by: 'trig_count')
###################################################################
 port_num |    instance_name  |          port_name  |   trig_count |
----------|-------------------|---------------------|--------------|
        1 |     tbleon.ram0   |                D    |         3789 |
        6 |     tbleon.ram1   |                D    |         3789 |
       11 |     tbleon.ram2   |                D    |         3789 |
        0 |     tbleon.ram3   |                A    |         2798 |
        5 |     tbleon.ram4   |                A    |         2798 |
       10 |     tbleon.ram5   |                A    |         2798 |
        2 |     tbleon.ram6   |                CE1  |          320 |
        7 |     tbleon.ram7   |                CE1  |          320 |
       12 |     tbleon.ram8   |                CE1  |          320 |
       13 |     tbleon.ram9   |                WE   |          300 |
        8 |     tbleon.ram10  |                WE   |          297 |
        3 |     tbleon.ram11  |                WE   |          295 |
        4 |     tbleon.ram12  |                OE   |          120 |
        9 |     tbleon.ram13  |                OE   |          120 |
       14 |     tbleon.ram14  |                OE   |          120 |
```

The **-pattern** option provides information about the concurrent activity between the instances at the boundary. The fields mean the following:

- **Instance trig. count** — Total number of time the instance triggered resulting in the activity on its ports
- **Num. of Patterns** — A pattern is a set of active instances. This field indicates the number of different patterns this instance is part of
- **Concurrent triggers** — Number of times the given pair of instance triggered concurrently.

```
mc2perfanalyze mc2data.mdb -pattern -part master -topart p2 -limit 3
####################################################################################
Instance trigger pattern between master and p2  (sorted by: 'instance_trig_count')
####################################################################################
------------------------------------------------------------------------------------
+tbleon.tb.p0.leon0.mcore0.clkgen0     [Instance trig. count: 38624] [Num. of Patterns: 9]
------------------------------------------------------------------------------------
    =====> tbleon.tb.p0.leon0.iopadb5[10].iopadb51       [concurrent triggers: 24]
    =====> tbleon.tb.p0.leon0.iopadb5[12].iopadb51       [concurrent triggers: 12]
    =====> tbleon.tb.p0.leon0.iopadb5[14].iopadb51       [concurrent triggers: 12]

------------------------------------------------------------------------------------
```

```
+tbleon.tb.p0.leon0.mcore0.reset0     [Instance trig. count: 33485] [Num. of Patterns: 6]
-------------------------------------------------------------------------------------------
    =====> tbleon.tb.p0.leon0.iopadb8                    [concurrent triggers: 5]
    =====> tbleon.tb.p0.leon0.iopadb5[0].iopadb51        [concurrent triggers: 5]
    =====> tbleon.tb.p0.leon0.iopadb5[1].iopadb51        [concurrent triggers: 5]

-------------------------------------------------------------------------------------------
+tbleon.tb.testmod0     [Instance trig. count: 7356] [Num. of Patterns: 29]
-------------------------------------------------------------------------------------------
    =====> tbleon.tb.ram32d.rambnk[0].ramarr[1].ram0     [concurrent triggers: 7285]
    =====> tbleon.tb.ram32d.rambnk[0].ramarr[3].ram0     [concurrent triggers: 7285]
    =====> tbleon.tb.ram32d.rambnk[0].ramarr[0].ram0     [concurrent triggers: 7285]
```

Analyzing Unisim Data

mc2perfanalyze can render useful analysis based on the WLF file generated from -mc2collectinfo unisim flow. mc2perfanalyze analyzes the given WLF file and extracts important information into an SQL database (named mc2collect.mdb), which can be input back to mc2perfanalyze with different options to get port and instance activity information.

The command **mc2perfanalyze mc2inst.wlf -wlf2mdb[=filename]** produces the SQL database with the specified name. In the absence of the optional filename, the default database name is mc2collect.mdb. If WLF file is very large this command can take some time to finish, but this is a one-time analysis on WLF and for all subsequent unisim information the SQL database will be used. "mc2collect.mdb" contains information on port activity and instance triggering which can be obtained using -inst and -port information.

The commands **mc2perfanalyze mc2collect.mdb -port** and **mc2perfanalyze mc2collect.mdb -inst** produce reports that show the ports that are highly active and the instances that trigger very high number of times. Future enhancement will include options to report concurrency information between instances.

More details on analyzing unisim data can be found in Unisim Data Analysis Reports.

Unisim Data Analysis Reports

mc2perfanalyze can analyze the WLF files generated from unisim runs (using -mc2collectinfo flow) and provide information about the port activity and the instance trigger activity that can help in partitioning of the design. This analysis can be invoked using -wlf2mdb option as below.

```
mc2perfanalyze mc2inst.wlf -wlf2mdb[=filename]
```

-wlf2mdb produces 2 files *mc2collect.mdb* and *mc2_vsig.do*. mc2_vsig.do is a vsim dofile that the factory can use to further analyze the mc2inst.wlf file for concurrency. *mc2collect.mdb* is a SQL database containing port and instance trigger information and can be analyzed further by mc2perfanalyze and generate reports. Sample invocations of mc2perfanalyze on mc2collect.mdb are below:

The general info on the captured data can be obtained using -info option.

```
mc2perfanalyze mc2collect.mdb -info
########################################################################
                             Unisim Details
########################################################################
                    Quantity|         Data|
----------------------------|-------------|
           Num. of Instances|           56|
----------------------------|-------------|
               Num. of Ports|          233|
----------------------------|-------------|
        Num. of Sigs or Vars|           72|
----------------------------|-------------|
                  Start Time|            0|
----------------------------|-------------|
                 Start Delta|            0|
----------------------------|-------------|
                    End Time|       289860|
----------------------------|-------------|
                   End Delta|            0|
----------------------------|-------------|
               Time Advances|        29825|
----------------------------|-------------|
              Delta Advances|       203202|
----------------------------|-------------|
```

The instance trigger report can be obtained using -inst option. -limit option can used to limit the report size and by default the table is sorted in descending order of instance trigger count.

```
mc2perfanalyze mc2collect.mdb -inst -limit 5
################################################################################
Instance trigger table for Unisim data (sorted by: 'instance_trig_count')
     (Note: Higher individual triggers implies that ports of the instance are
            triggering independently)
################################################################################
instance_name      |nPorts |ports_trig_count |instance_trig_count |%individual |
                   |       |                 |                    |   triggers |
------------------ |------ |---------------- |------------------- |----------- |
/tbleon/proc0      |    27 |          235696 |             139606 |     59.23% |
/tbleon/m0         |    32 |          172630 |             125014 |     72.42% |
/tbleon/iu0        |    19 |          194870 |             103860 |     53.30% |
/tbleon/a0         |    12 |          118214 |              91895 |     77.74% |
/tbleon/c0         |    15 |          169688 |              81111 |     47.80% |
```

The port trigger report can be obtained using -port option. -limit option can used to limit the report size and by default the table is sorted in descending order of port trigger count.

```
mc2perfanalyze mc2collect.mdb -port -limit 5

############################################################
Port trigger table for Unisim data (sorted by: 'trig_count')
############################################################
port_num |       instance_name |port_name |trig_count |
-------- |-------------------- |--------- |---------- |
      77 |/tbleon/iu0          |iuo       |     40343 |
      94 |/tbleon/p0           |iuol      |     40343 |
     207 |/tbleon/a0           |iuo       |     40343 |
     244 |/tbleon/p0           |iuo       |     40343 |
```

```
        266  |/tbleon/m0           |iuo        |      40343  |
```

Viewing virtual signals file

The virtual signal file produced by the -wlf2mdb option of mc2perfanalyze can be viewed using vsim. There is one virtual signal for each instance and an event on the virtual signal indicates an activity on the ports of that instance. It is possible to infer concurrent activity of instances by viewing these virtual signals. Follow the steps below to view the virtual signals using vsim:

1. Open the WLF file using the vsim command in interactive mode.

   ```
   vsim -view mc2inst.wlf
   ```

2. In the vsim command window, run the do command on the file mc2_vsig.do, this will open the wave window and shows all the virtual signals.

   ```
   do _mc2_collect/mc2_vsig.do
   ```

3. Select all the virtual signals, right-click on the selected signals, and in the "Format" menu select "Event".

   This will change the format of the virtual signals from 'logic' to 'event'.

CPU or Core Binding (Affinity)

Affinity is the process of binding a process or memory to a CPU or core on a multi-processor system. Binding a process to a CPU eliminates the cost of process relocation between CPUs. The process is given the exclusive use of a CPU, which improves its performance with this type of affinity. With memory affinity, the physical memory that a process uses is assigned from the CPU/core that the process is run on. This gives the process better memory access performance, which can improve process performance.

Using affinity can improve multi-core simulation performance. The degree of performance improvement and type of settings required can vary according to the host hardware, including CPU type and memory architecture.

CPU/Core Binding Methods

You can control the CPU/core binding for multi-core simulation using vsim option -mc2binding. Following binding method are available for multi-core simulation:

- If you don't specify -mc2binding option — by default, each partition is bound to a specific core using taskset or numactl, based on the type of machine. Before starting simulation, vsim generates a file named .mc2.affine in the current working directory. This file needs to be generated specific to the machine the simulation is being run on. The .mc2.affine file contains the setting to bind cores to individual partition.

- If you specify the '-mc2binding affinity:user' option — then you are required to provide the .mc2.affine file in the current working directory. The .mc2.affine file can be manually created, modified from a previous run on the same machine, or it can be generated using the utility 'mc2_util affine'. With this option, vsim does not generate the .mc2.affine file, the existing .mc2.affine file is read and core binding is done as specified by you.
- If you specify the '-mc2binding cpu:sockets' option — partitions are bound to CPU/cores using MPICH2's built-in binding capability. With this binding method, binding is assigned to pack processes as closely to each other as possible without sharing a socket, unless the number of processes exceeds the number of sockets. This binding method results in same performance as the default numactl/taskset binding, but it sometimes can produce either faster or slower results based on the type of host used or design partitioning. You can use this option if you are fine-tuning performance to see if it improves your results.
- If you specify '-mc2binding none' option — CPU/core binding is disabled, and multi-core simulation will be run with no core binding. Using this option may affect performance. You might want to use this option when more than one multi-core simulations are run at the same time on the same machine, and the number of cores on the machine aren't equal to sum of all partitions running for all multi-simulations on a machine.

Auto-generated Binding (Default Method)

Questa SIM can automatically generate binding information for a machine and use it during multi-core simulation. Binding information is generated in the .mc2.affine file. This section describes the contents of this file and how it is used.

Based on Intel or AMD CPU, taskset or numactl commands are used to bind a CPU/core to a process during multi-core simulation. AMD CPU are NUMA (Non-Uniform Memory Access) systems. Newer Intel CPUs are also NUMA systems, some earlier Intel architectures are UMA based.

On AMD machine, the control command is as follows:

```
numactl -c C[,C] -m M[,M] ...
```

Where -c specifies the CPU(s)/core(s) the process is to be run on, and -m specifies the CPU/cores’ memory will be allocated from.

> **Tip**: On some NUMA systems, numactl may not be functional. Use taskset in that situation.

On Intel machine, the control command is as follows:

```
taskset -c C[,C]
```

Where -c specifies the CPU(s)/core(s) the process runs on.

Refer to man pages of numactl and taskset for more information.

Following are example .mc2.affine files for an AMD-based host and Intel-based host:

AMD Host:

```
MC2_AUTO_AFFINE_0="numactl --physcpubind 0 --membind 0"
export MC2_AUTO_AFFINE_0

MC2_AUTO_AFFINE_1="numactl --physcpubind 1 --membind 1"
export MC2_AUTO_AFFINE_1

MC2_AUTO_AFFINE_2="numactl --physcpubind 2 --membind 2"
export MC2_AUTO_AFFINE_2

MC2_AUTO_AFFINE_3="numactl --physcpubind 3 --membind 3"
export MC2_AUTO_AFFINE_3
```

Intel Host:

```
MC2_AUTO_AFFINE_0="taskset -c 7"
export MC2_AUTO_AFFINE_0

MC2_AUTO_AFFINE_1="taskset -c 6"
export MC2_AUTO_AFFINE_1

MC2_AUTO_AFFINE_2="taskset -c 4"
export MC2_AUTO_AFFINE_2

MC2_AUTO_AFFINE_3="taskset -c 5"
export MC2_AUTO_AFFINE_3
```

Binding information is generated in the .mc2.affine file based on available cores on the machine with sequential number in the format MC2_AUTO_AFFINE<num>. Binding with number 0 is assigned to master partition. Binding with subsequent numbers (1, 2, 3…) are assigned to slave partitions in the order partitions are defined in the partition file.

User-defined Binding

Vsim generated core binding file (.mc2.affine) or MPICH2's built-in binding assign binding based on processor's architecture. It does not take into account dynamic loading among the cores/CPUs on the system. You may be able to improve performance by applying binding based on the current load on the machine. You may also want to apply user-defined binding, if you think it will perform better than automatically generated binding. To specify user-defined binding, you can generate a new .mc2.affine file using 'mc2_util affine', and then modify the existing .mc2.affine file.

Before you modify this file, you should know the configuration of the system that you are running multi-core simulation on. The most important information is the number of cores and CPUs available on the system. You should also know how the cores are numbered and the core/CPU relation.

> **Tip**: On most Linux installations, you can examine the file /proc/cpuinfo to determine CPUs/cores configuration.

For platforms containing multi-core CPUs, you should assign master partition to a CPU where the least number of cores are being used to run the simulation. For example, assume you are running on a computer with dual Intel Quad-core CPUs, where one CPU administers all the cores with odd numbers and the other CPU administers the cores with even numbers. If your multi-core simulation runs in five partitions, this rule suggests you should assign the master partition to 0, and one of the remaining partitions to core 2 (same CPU as master), the remaining partitions should be assigned to cores 1, 3, and 5 (different CPU from master).

```
MC2_AUTO_AFFINE_0="taskset -c 0"
MC2_AUTO_AFFINE_1="taskset -c 2"
MC2_AUTO_AFFINE_2="taskset -c 1"
MC2_AUTO_AFFINE_3="taskset -c 3"
MC2_AUTO_AFFINE_4="taskset -c 5"
```

By default, Questa SIM uses multi-threading for WLF logging. Therefore, for each of the partitions where you are logging signals, results might be better if you assign those partitions to two cores from the same CPU. For example, if cores are assigned to CPUs using the even/odd method described above, and if master partition is doing the logging, you could assign 0 & 2 to master partition:

```
MC2_AUTO_AFFINE_0="taskset -c 0,2"
```

Binding for Multi-core Simulation on Multiple Machines

You can use -mc2network <hostfile> option to run multi-core simulation on multiple machines. The hostfile contains the information about which machines to run simulations on, and how many cores to use on each machine. In addition, you can also specify different bindings for each of the machines in the same hostfile. You can specify that processes are bound close to each other without sharing a socket (cpu:sockets), or you can specify your own ("user") core binding.

Syntax:

<hostname>[:<num_procs>] [binding=cpu:sockets|user:<c, c>]

Example:

```
host1:2                      # Run 2 process on host1 without any binding
host2:4 binding=cpu:sockets # Run 4 processes on host2 with cpu:sockets
```

```
                                #       type binding
host3:2 binding=user:0,3        # Run 2 processes on host3 with binding to
                                #       cores 0 and 3
```

Consider the load of each CPU when assigning binding. It is generally preferable to have the CPU with the master partition slightly less loaded than others. Each CPU should have a comparable total load.

Recommendations and Limitations

Recommendations for Running Multi-core Simulation

You can optimize multi-core simulation operation by observing the following recommendations for system preferences:

- Multi-core simulation runs better on a single multi-processor system (SMP) than on multiple multi-processor systems. Running multi-core simulation over a multi-system can have a significant negative effect on performance. If you absolutely must run multi-core simulation over multiple systems, you should make sure that at least one of the followings is true:
    - Simulation runtime footprint is too big to fit into the physical memory of a single system
    - The sync between the partitions can be reduced to offset the cost of communication between systems. (An example is to use user sync event to control sync.)
    - The systems are connected to the same network switch.
- Run multi-core simulation on its own dedicated system so that it can deliver the best performance possible.
- By default, multi-core simulation applies process and/or memory affinity to each partition. Running more than one multi-core simulation on the same system at the same can cause extreme CPU and memory contention. Therefore, you should turn off affinity for the second or later multi-core simulation run by specifying the '-mc2binding none' option to the vsim command:

    ```
    vsim -mc2binding none
    ```

Flexible Limitations

You should observe the following limitations as much as possible to reduce or avoid multi-core simulation failure and error conditions.

- The DUT contains few levels of hierarchy (partition designs to the higher levels of (functional) hierarchy).
    - If a design can be partitioned at higher level of hierarchy that represents well defined functional block of the design, it will better for multi-core simulation partitioning and will have better chances to deliver performance. It will also help during multi-core simulation debugging process, if one is needed.
    - You can also create a partition's hierarchy by partitioning the design in the same hierarchical path on top/bottom of another partition. However, if there is communication going across all levels of these partition hierarchies, it can slow down the simulation.
- The design (DUT and test bench) has no race conditions or is not sensitive to them.
    - A race-free design always produces the “correct” results that are easier for multi-core simulation to match.
    - If the design can handle a race condition (race-insensitive), it will be easier for multi-core simulation to run the design.
- Conditional force and change commands are not supported, if objects in condition and in force command belong to different partitions.

Rigid Limitations

You should always observe the following limitations to avoid multi-core simulation failure and error conditions.

- The $dumpvar() Command
    - Can only dump objects in the calling partition
- File I/O involving multiple partition
    - Not supported. You need to re-code their algorithm to avoid race conditions.
- PLI Traversal
    - Only access objects in the partition where the calls are made.
    - Static reference to objects not in current partition may fail.
    - File I/O may need to be redesigned to avoid collision.
- No bidirectional transistor on partition boundary
    - In general, multi-core simulation cannot handle the resolution requirement for a bidirectional transistor on a boundary. However, some designs may not require the fully bidirectional resolution to function. As a default, bidirectional transistor is allowed to sit on partition boundary. You can turn on the check against this situation by inserting the following option in the run script:

```
vsim -mc2 -mc2sanitycheck ...
```

Alternatively, you can instruct multi-core simulation to warn about their existence by inserting the following option in the run script:

```
vsim -mc2 -mc2sanitycheck=warn ...
```

- Strength not propagated across partition
    - You can use the -mc2sanitycheck option to detect strength at boundary. This check is not enabled by default as some designs may work correctly with presence of strength at partition boundary.
- Cross-partition task/function calls
    - Pure functions and tasks are supported provided that the function does not contain or reference:
        - Events, strings, interfaces, classes, covergroups, hierarchical references, nested functions or tasks, or struct or union types or variables.
        - You can turn off pure functions and tasks by using the mc2com -mc2nopurefunc argument.
    - Impure functions are supported provided that:
        - The function *reads* only variables outside its own scope. Those variables may not be in a package.
        - There are no hierarchical references elsewhere to variables within its own scope.
        - There are no local variables that are read before being written and the function is called from multiple partitions.
        - You can turn off impure functions and tasks by using the mc2com -mc2noimpurefunc argument.
- Globally shared memory using hierarchical references
    - Supported only on single multiprocessor (SMP) system.
    - Write access needs to be 32-bit “bounded.” For example, if the memory is declared as reg[0:63] Mem[0:10], and partition 1 is writing to Mem[5][17] at time 10ns, then no other partition can write to any bit/part of [0:31] of Mem[5] at the same time.
- SystemVerilog complex object types on partition boundary instances' ports, or in hierarchical references which cross partition boundaries. Complex object types include classes, structures, interfaces, and strings.
- Static object sharing

    - Static objects cannot be shared across partitions. If they are shared, each partition will have its individual copy. This may still work for some test bench situations when the information can be post-processed.

- No SystemC on partition boundary
    - Not currently supported.
- SDF
    - Observe caution when specifying uncompiled SDF files to vsim, as output file collisions may occur as a result of multiple partitions compiling the same SDF file to the same output location. It is recommended to compile SDF files explicitly using the sdfcom command, prior to running the mc2com or vsim commands. Uncompiled SDF files may also be specified to mc2com, but they will be compiled once for each partition (to separate locations), incurring extra overhead.
    - Module path delays, which are applied on a partition boundary net or its collapsed net, are not supported.
    - Inertial (non-transport) interconnect delays may not work when the destination port is on a partition boundary. Transport interconnect delays may not work when the source or destination port is on a partition boundary. An error will be reported by vsim for cases that are not supported.
    - Cross-partition multi-source interconnect delays are not supported.
- VCD Debug Flow
    - Reg type ports at partition boundary are not supported.
    - Bit-select, part-select ports at partition boundary are not supported.
    - Complex SV and VHDL types (except std_logic) at partition boundary are not supported.
    - Cross-partition hrefs cannot be driven with -vcdstim.
    - Partition file with user defined events is not supported.
- Multi-core Simulation VHDL Flow
    - Cross-partition impure function/procedure calls access, declared in packages.
    - Any limitations in current vopt while propagating generics, port constraints, complex configurations, generate unrolling.
    - FLIs/PLIs
    - Cross-partition hier-ref/signal spy support is limited to signal type only.
    - Cross-partition call of hier-ref of constant and variable type.
    - Cross partition access of package signals through hier-ref.

  - If hierarchical reference or signal spy call have unresolved paths (unrolled for-gen loop or complex expression).
  - The Following complex types at partition boundaries are not supported:
    - File type, access type, error type, incomplete type, or an array/record of these types.
    - Resolved record type at partition boundary is not supported.

- Signal spy support limitations

  Cross-partition signal spy calls have limited support in multi-core simulation flow. The limitations are detailed in the following sections.

  VHDL to VHDL limitations

  - signal_release, enable_signal_spy, disable_signal_spy and init_signal_spy with control state ON.
  - If signal spy calls have unresolved paths (unrolled for-gen loop or complex expression).

  Verilog to Verilog limitations

  - $enable_signal_spy, $disable_signal_spy, $init_signal_driver and $init_signal_spy with control state ON

  Mixed language limitations

  - Cross-partition usage of mixed signal spy calls has same limitations as above.
  - Port types, that are not supported cross-partition boundary, are also not supported in signal spy calls.

Feature Limitations

The following features are not yet supported in the multi-core simulation flow:

- GUI mode
- Limited CLI support — CLI that access cross-partition objects may not work correctly
- Power Aware Simulation
- Post-sim debug dataflow analysis
- Preoptimized Design Unit (PDU) flow

Chapter 2
Command and File Syntax Reference

Partition File Structure and Syntax

A partition file is segmented into two major sections:

- Synchronization Control
- Partition Definitions

Synchronization Control

You must specify synchronization (sync) points for determining when multi-core simulation partitions exchange data from any new events on cross-partition objects (using the sync_control command). The list of possible synchronization points is described below.

Syntax

```
sync_control {
    <list_of_values_spec> ;
}
```

Where list_of_value_spec is a semicolon-separated list of value assignments of the following form:

name = value

Where value is any non-negative integer and name is any of the following:

Active	If value = 1, sync will be performed at the end of active queue. This option is very costly and should be used only for a design that is highly sensitive to the number of propagations within a time step. Most designs do not require this control to be enabled in order to simulate correctly.
Events	If value = 1, perform communication of VHDL signal values when events occur on signals. Communication of VHDL signal values will usually occur when signals at port boundaries have activity. A common occurrence of activity that occurs without a value change is when a resolved signal has multiple drivers, a value of one of the drivers changes, but the resolved value remains unchanged. Use of the Events sync control will cause communication of signal values only when events occur on signals, rather than just activity. Unless a port on a partition boundary is connected to a signal that has the 'quiet, 'transaction, or 'active attributes, using the Events

sync control may reduce communication without affecting simulation accuracy.

HiPri	If value = 1, sync at the end of the High-priority queue. This is first the possible synchronization point for VHDL events.
Immed_CA	If value = 1, sync after all continuous assignments have been executed, but before their fanouts have taken effect.
Inactive	If value = 1, sync at the end of inactive queue.
LoopLimit	If value = n, the synchronization will be relooped a maximum of *n* times. By default, the reloop limit is set to match the simulator's delta iteration limit, which prevents infinite looping during simulation.
NBA	If value = 1, sync at the end of the Verilog non-blocking assignment event queue.
Observed	If value = 1, sync at the end of Observed queue.
Preponed	If value = 1, sync at the end of Preponed queue.
Postponed	If value = 1, sync at the end of Postponed queue.
ReActive	If value = 1, sync at the end of Re-active queue.
Reloop	If value = 1, reloop the synchronization when needed. If value = 0, disable relooping. When enabled, if data is exchanged between any partitions at any of the above sync points and new events are generated in a higher-priority queue, synchronization will "reloop" to the first synchronization point. By default, Reloop=1 (enabled).
ReNBA	If value = 1, sync at the end of Re-NBA queue.
Synch	If value = 1, sync at the point after all PLI read/write events are completed.

Sync Control Aliases

Sync Control Aliases specify several Sync Controls and are available for your convenience.

Verilog	If value = 1, this option enables the Synch and NBA sync controls. This alias should be used with Verilog partition boundary.
Verilog_conservative	If value = 1, this option enables the Active, Synch, and NBA sync controls. This alias should be used with Verilog partition boundary and when design partitions need tighter synchronization, for example, situations like 0-delay loop across partitions. This alias is *more expensive* than the

	Verilog sync control as it does more fine-grained synchronizations.
Verilog_aggressive	If value = 1, this option enables only the NBA sync control. This alias should be used with Verilog partition boundary and when the design is highly synchronous and does not need fine-grained synchronizations. This alias will produce faster results than other Verilog controls.
VHDL	If value = 1, this option enables the Active and HiPri sync controls. This option performs synchronization for VHDL design units at partition boundaries.
VHDL_aggressive	If value = 1, this option enables Active, HiPri, and Events sync controls. This option performs synchronization of VHDL design units at partition boundaries and only when the design does *not* have activity sensitive attribute usage (for example, 'quiet, 'transaction, or 'active). This option will set the individual sync points and reloop parameters above that are typically required for VHDL designs.
SV_conservative	If value = 1, perform synchronization for SystemVerilog constructs connected to boundary ports. This option enables all the SystemVerilog-specific sync points.

You can experiment with the combination of the name/value pairs above to improve performance. After modifying the sync controls in the partition file, re-run the multi-core simulation to test the changes. You do not need to re-run mc2com when changing the sync control portion of the partition file.

Generally, you should enable only the minimum set of sync points required for correct simulation, in order to maximize simulation performance.

For Verilog designs you should typically use the Verilog sync control alias. For VHDL designs should typically use the VHDL sync control alias.

Reloop should be enabled for either language, unless you can verify that data sent across partitions cannot trigger 0-delay events that propagate across partitions, or the synchronization points selected are sufficient to handle such propagation.

Mixed-HDL designs may require both Verilog and VHDL sync controls, even if partitions are at only one language boundary. That is, sync control does not depend on at what language in which partitions are made, but depends instead on what language construct is connected to ports on the partition boundary.

Mixed-HDL designs may require both Verilog and VHDL sync controls. Most of the time using just the Verilog sync control does work fine, unless you are partitioning at VHDL boundaries.

The VHDL sync control is required for mixed-HDL designs when delta-accurate results are required (this translates to more fine-grained synchronization and hence the use of VHDL).

There may be situations where you may need to use finer sync controls, such as race conditions or zero-delay feedback loops that exist between partitions. If these situations cannot be avoided by alternative partitioning, then you must use additional sync controls to ensure correct simulation results.

Once you get desired simulation behavior from a set of sync controls, you can try to reduce fine-grained sync controls (or use aggressive controls), if you are interested in further speeding up simulation. Take care when using aggressive controls. For example, you get correct results with aggressive controls, and then you make substantial changes to the design, that changes event scheduling, or communications patterns, or introduces 0-delay loops. Such design changes may now require you to do finer synchronization, and you may not get correct results using the same aggressive controls.

You can change sync controls in a partition file anytime during the flow. Changing sync controls does not require recompiling your design. You can change it as many times and proceed directly to simulation. However, sync controls cannot be changed in middle of a simulation.

Partition Definitions

You can define multiple partitions according to design hierarchy and synchronization events.

Syntax

```
partition <partition_name> {
    <list_of_partition_members> ;
}
```

Where <list_of_partition_members> is a semicolon-separated list of mod_inst definition and/or sync_event definitions. You must name one partition “master”. This master partition oversees communication and control during the simulation.

Module instance

Each partition requires at least one definition of a module instance, which represents the top-level module of that partition. When you specify a module instance for a partition, then this instance and all its children instances become part of the partition. If any of its children instance is specified as module instance for another partition, then this children instance and their children instances are not considered as part of partition that contains its parent instance as module instance, instead they become part of the partition that specify children instance as module instance.

Each partition can contain multiple module instance definitions.

Syntax

mod_inst = (*module_name*, *instantiation_name*) ;

or

mod_inst = *instatiation_name* ;

Where *instantiation_name* is the full hierarchical pathname of the instance.

Sync event control

You use this construct to identify a user-defined sync event you introduce into the partition. The event is then monitored by Questa SIM multi-core simulation. Data synchronization is performed when this event is detected.

In addition to this specification in the partition file, you also need to modify the netlist in order for event synchronization to take effect—see Section “Adding a User-defined Synchronization Event” on page 54.

Syntax

sync_event = *event*

Where *event* is full path name of the user sync event you added.

Common Module Definition

If your design has many hierarchical references to global objects such as Vcc or Ground signals, which are defined and driven inside a single module, you can improve performance by defining this as a "common module" in the partition file. This will cause the mc2com command to replicate the common module’s logic in all the partitions and avoid large numbers of cross-partition hierarchical references to objects contained within that module, which will improve simulation performance.

If your design has many such hierarchical references, but objects are not defined in a single module, you can modify the design to create a common module, and define and drive all such global objects from this common module.

Common modules must not be instantiated anywhere within the design, but instead must be specified along with other design top-level modules when compiling the design for multi-core simulation. If the module is instantiated within the original design (prior to using multi-core simulation), you must remove this instantiation before defining it as a common module.

The following syntax shows how to specify common modules in partition file:

Syntax

common_module {
list_of_partition_members ;
}

Where *list_of_partition_members* is defined in "Partition Definitions" on page 52.

Example Verilog Source for a Common Module:

```
module dsgn_globals;
    supply0 gnd;
   supply1 vcc;
endmodule
```

Example Partition File:

```
sync_control {
  Verilog = 1;
}

partition master {
  mod_inst = top;
}

partition p1 {
  mod_inst = top.sub1;
}

common_module {
  mod_inst = dsgn_globals;
}
```

Example mc2com Command:

```
mc2com top dsgn_globals -o run_mp -mc2partfile top.part
```

Adding a User-defined Synchronization Event

For each user sync event you define for a partition, you normally add the following type of construct to your Verilog netlist. In the following example, the event is named my_sync_event, and the top of the partition is named top_of_partition:

```
module top_of_partition ;
  event my_sync_event
  always @some_trigger
    -> my_sync_event
endmodule
```

You can also use an existing event from your design.

You should note following points when using user-defined synchronization:

1. Number of sync events used should be same for all partitions. It will result in error otherwise.
2. You can either create common events, using common module, and use it for all partitions, or create equivalent event logic in each partition's hierarchy.
3. If user sync event is triggered by clock, then you will also need to make sure that each partition is running its clock without needing to synchronize. Thus can be done by duplicating clock logic in each partition or defining clock logic in common module and include it in all partition. If you use common module, it is also easier to share same events by all partitions.
4. You should make sure that events trigger in all partitions around same time. Synchronization is controlled by this user-defined event. If event triggers in only few partitions and not in other partition, simulation may hang or produce incorrect results.

You should still provide regular sync controls, like Verilog etc. User-defined sync events only direct simulator when to start synchronizations. How fine grained synchronization should be in each time cycle is still determined by regular sync controls.

mc2com

Syntax for using the mc2com command and its arguments are described below.

Syntax

mc2com [options] <top-level_design> -o designName [-mc2numpart <num>]
[-mc2nopurefunc] [-mc2noimpurefunc] [-mc2vlogpart] [-mc2vhdlpart]
[-mc2ignoreexterns] [-mc2useprofile[=<mem | perf>] <file>]
[-mc2autopartreport[=verbose] <file>] [-mc2partfile <filename>]
[-mc2noglobalmem[=<mem_pathname>]] [-mc2unicore {all | out | in}]
[-mc2analysisreport <file>] [-mc2voptargs[="<prttn_name>"] "<vopt_args>"]
[-mc2allowgenhref] [-mc2makeinoutnets] [-mc2novopt] [-mc2noanalyze] [-vv]

Note

Currently, SDF compiled with mc2com does not work, unless it is limited to the netlist within a single partition. The incr SDF flow works even if it crosses partitions.

Arguments

- top-level_design

  Name of the top-level design that you want to perform multi-core simulation on.

- -mc2numpart <num>

  Specify number of partitions needed from auto-partitioner.

- -mc2nopurefunc

  Disallow cross-partition pure function usage in auto-partitioner.

- -mc2noimpurefunc

  Disallow cross-partition impure function usage in auto-partitioner.

- -mc2vlogpart

  Allow only Verilog/SV instances at partition boundary in auto-partitioner.

- -mc2vhdlpart

  Allow only VHDL instances at partition boundary in auto-partitioner.

- -mc2ignoreexterns

  Ignore failed externs during auto-partition analysis.

- -mc2useprofile[=<mem|perf>] <file>

  Specify profile database file to feed back to auto-partitioner.

  =mem — specifies memory database, =perf — specifies performance database.

  The default is performance profile database.

- -mc2autopartreport[=verbose] <file>

  Generates auto-partitioner report on the design nodes, algorithm parameters, and run-times. It reports the instances that could not be mapped to partitions and the reasons for not mapping them. The verbose mode (=verbose) reports more information about unsupported hierarchical references, signal-spys, and global variables that rendered the instances unmappable. On large designs, verbose report could be overwhelming. If you are using verbose report mode, it is recommended that you look at the "Unmappable nodes summary" first, and if you need more information about particular unmappable node, lookup the verbose report to get more details on unsupported hierarchical references, signal-spys or global variables.

- -mc2partfile <filename>

  Specifies a partition file to be used for multi-core simulation.

- -mc2noglobalmem[=<mem_pathname>]

  Disables shared memory usage to implement all cross-partition RTL memories. If full pathname of <mem_pathname> is specified, then only this memory will be disabled as shared memory. This option can be specified multiple times.

- -mc2unicore {all | out | in}

  Run unicore flow (See “Debugging Simulation Mismatches With Unicore Flow” on page 23), where:

  all — generates unicore ports for all boundary ports of current partition.

  out — generates unicore ports for only those boundary ports that are going out of current partition (that is, 'output/inout' module ports in top/higher boundary and 'input' module ports in bottom/lower boundary of current partition).

  in — generates unicore ports for only those boundary ports that are coming in the current partition. (that is, 'input' module ports in top/higher boundary and 'output/inout' module ports in bottom/lower boundary of current partition).

  For example, the following command generates all required information to run the entire design as separate partition:

  ```
  mc2com top -o run_mp -mc2partfile part01.part -mc2unicore all
  ```

- -mc2analysisreport <file>

  Generates a partition analysis report and save to the file specified.

- -mc2voptargs[="<prttn_name>"] "<vopt_args>"

  Specify vopt arguments for all or individual partitions.

  =<prttn_name> — specifies the partition name, optional. If you don't specify partition name, vopt arguments are passed to all partitions.

  <vopt_args> — vopt arguments specification

- -mc2allowgenhref

  Allows hierarchical references to bit- and part-selects with index expressions composed of genvars.

- -mc2makeinoutnets

  Makes port direction for all Verilog net connections at multi-core simulation partition boundary 'inout'. This option may be necessary for correct results, when a Verilog net connected to an "output" port on a partition boundary is forced within the driving partition, and also forced or released by another partition. This option can also be used to ensure nets connected to partition output ports, which have contributions (drivers) in other partitions, will always show the correct value when logged in a WLF file.

- -mc2novopt

  Disables partition's vopt optimization.

- -mc2noanalyze

  Disables multi-core simulation partition analysis.

- -vv

  Echo subprocess invocations to stdout.

vsim

Multi-core simulation specific arguments for the vsim command.

Syntax

vsim -mc2 [-mc2network <hostfile>] [-mc2binding <type>] [-mc2sal <partition_name>] [-mc2mergeucdb <filename>] [-mc2mergewlf <filename>] [-mc2vcddump] [-mc2commstat] [-mc2savestat[=filename]] [-mc2sanitycheck[=warn]] [-mc2nozerochk] [-mc2noidlesyncopt] [-mc2nosuspendopt] [-mc2noactivequeueopt] [-mc2activequeueopt] [-mc2vsimargs="<partition>" "<args>"] [-vv]

Arguments

- -mc2

  (required) Run simulation in multi-core simulation mode. This option also implies the vsim -c option.

- -mc2network <hostfile>

  Run simulation in multi-computer mode. See “Running Multi-computer Simulation” on page 15 for details on <hostfile> syntax.

- -mc2binding <type>

  Run simulation with specified core binding: none, cpu:sockets.

- -mc2sal <partition_name>

  Runs simulation on one standalone partition, <partition_name>, in VCD mode.

- -mc2mergeucdb <filename>

  Merge UCDB files from all partitions to the specified filename.

- -mc2mergewlf <filename>

  Merge WLF files from all partitions to the specified filename.

- -mc2vcddump

  Generate VCD info and enable later standalone mode simulation run on this partition.

- -mc2commstat

  Generate multi-core simulation statistics

- -mc2savestat[=filename]

  Save multi-core simulation statistics to the specified filename.

- -mc2sanitycheck[=warn]

  Perform sanity design checks for allowable boundary conditions. Optional '=warn' argument converts severity of error message to warning.

- -mc2nozerochk

  Converts severity of time 0 unicore value mismatch errors to warnings.

- -mc2noidlesyncopt

  Disables optimization, which suppresses synchronizations when all partitions but one are idle.

- -mc2nosuspendopt

  Use this option to disable an optimization on partitions without any activity. If time synchronization between any partitions is incorrect or we are getting incorrect output, then this option can be used. This option used together with -mc2noidlesyncopt option will force the partitions to synchronize at each simulation time step.

- -mc2noactivequeueopt

  Disables optimization on active queue communications. When only VHDL sync control is specified and design either shows incorrect results or design is a mixed HDL design you can use this option to see if results improve.

- -mc2activequeueopt

  Enable optimization for active queue communication.

- -mc2vsimargs="<partition>" "<args>"

  Specify additional vsim options, <args>, for a particular partition. By default, vsim command line options are applied to all partitions. This option can be used to apply specific vsim options to a particular partition. You can group multiple space-separated vsim arguments within the quotation marks.:

  ```
  -mc2vsimargs=p1 "-l users.log -wlf users.wlf"
  ```

- -vv

  Print verbose multi-core simulation command line information.

mc2perfanalyze

Syntax specification for the mc2perfanalyze command.

Syntax

mc2perfanalyze <filename> [-help | --help | -h] [-info] [-time] [-cmd] [-data] [-event] [-port] [-inst] [-pattern] [-analyze] [-list] [-part <from_partname>] [-topart <to_partname>] [-sortby <column_name>] [-sortorder <asc|desc>] [-limit <n>] [-wlf2mdb]

Arguments

- -help | --help | -h

  Prints this usage information
- -info

  Prints the connectivity info between partitions
- -time

  Prints the Exec. Time (comparison table) for all partitions
- -cmd

  Prints the Command exchange summary (comparison table) for all partitions
- -data

  Prints the Data traffic summary (comparison table) for all partitions
- -event

  Prints the event (comparison table) for all partitions
- -port

  Prints the port trigger information for every pair of communicating partitions
- -inst

  Prints the boundary instance trigger information for every pair of communicating partitions
- -pattern

  Prints the boundary instance concurrency information for every pair of communicating partitions
- -analyze

  Prints a summary analysis report for the collected data
- -list

  Prints the output in list format instead of comparison tables for options -time, -cmd, -event
- -part <from_partname>

  Prints information specific to this partition

- -topart <to_partname>

  Prints information specific to the partition pair specified by -topart and -part

- -sortby <column_name>

  Sorts the tables as per the specified column name, default is table specific

- -sortorder <asc|desc>

  Specifies the sort order for the sort by column, default is descending order

- -limit <n>

  Limit on the number of lines to be printed, applicable to -port, -inst, -pattern

- -wlf2mdb

  Produces a unisimdata.db file containing the port and instance trigger data and vsig.do file containing virtual signal expressions useful for viewing the WLF in the waveform window.

End-User License Agreement

The latest version of the End-User License Agreement is available on-line at:
www.mentor.com/eula

IMPORTANT INFORMATION

USE OF ALL SOFTWARE IS SUBJECT TO LICENSE RESTRICTIONS. CAREFULLY READ THIS LICENSE AGREEMENT BEFORE USING THE PRODUCTS. USE OF SOFTWARE INDICATES CUSTOMER'S COMPLETE AND UNCONDITIONAL ACCEPTANCE OF THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT. ANY ADDITIONAL OR DIFFERENT PURCHASE ORDER TERMS AND CONDITIONS SHALL NOT APPLY.

END-USER LICENSE AGREEMENT ("Agreement")

This is a legal agreement concerning the use of Software (as defined in Section 2) and hardware (collectively "Products") between the company acquiring the Products ("Customer"), and the Mentor Graphics entity that issued the corresponding quotation or, if no quotation was issued, the applicable local Mentor Graphics entity ("Mentor Graphics"). Except for license agreements related to the subject matter of this license agreement which are physically signed by Customer and an authorized representative of Mentor Graphics, this Agreement and the applicable quotation contain the parties' entire understanding relating to the subject matter and supersede all prior or contemporaneous agreements. If Customer does not agree to these terms and conditions, promptly return or, in the case of Software received electronically, certify destruction of Software and all accompanying items within five days after receipt of Software and receive a full refund of any license fee paid.

1. **ORDERS, FEES AND PAYMENT.**

1.1. To the extent Customer (or if agreed by Mentor Graphics, Customer's appointed third party buying agent) places and Mentor Graphics accepts purchase orders pursuant to this Agreement ("Order(s)"), each Order will constitute a contract between Customer and Mentor Graphics, which shall be governed solely and exclusively by the terms and conditions of this Agreement, any applicable addenda and the applicable quotation, whether or not these documents are referenced on the Order. Any additional or conflicting terms and conditions appearing on an Order will not be effective unless agreed in writing by an authorized representative of Customer and Mentor Graphics.

1.2. Amounts invoiced will be paid, in the currency specified on the applicable invoice, within 30 days from the date of such invoice. Any past due invoices will be subject to the imposition of interest charges in the amount of one and one-half percent per month or the applicable legal rate currently in effect, whichever is lower. Prices do not include freight, insurance, customs duties, taxes or other similar charges, which Mentor Graphics will state separately in the applicable invoice(s). Unless timely provided with a valid certificate of exemption or other evidence that items are not taxable, Mentor Graphics will invoice Customer for all applicable taxes including, but not limited to, VAT, GST, sales tax and service tax. Customer will make all payments free and clear of, and without reduction for, any withholding or other taxes; any such taxes imposed on payments by Customer hereunder will be Customer's sole responsibility. If Customer appoints a third party to place purchase orders and/or make payments on Customer's behalf, Customer shall be liable for payment under Orders placed by such third party in the event of default.

1.3. All Products are delivered FCA factory (Incoterms 2000), freight prepaid and invoiced to Customer, except Software delivered electronically, which shall be deemed delivered when made available to Customer for download. Mentor Graphics retains a security interest in all Products delivered under this Agreement, to secure payment of the purchase price of such Products, and Customer agrees to sign any documents that Mentor Graphics determines to be necessary or convenient for use in filing or perfecting such security interest. Mentor Graphics' delivery of Software by electronic means is subject to Customer's provision of both a primary and an alternate e-mail address.

2. **GRANT OF LICENSE.** The software installed, downloaded, or otherwise acquired by Customer under this Agreement, including any updates, modifications, revisions, copies, documentation and design data ("Software") are copyrighted, trade secret and confidential information of Mentor Graphics or its licensors, who maintain exclusive title to all Software and retain all rights not expressly granted by this Agreement. Mentor Graphics grants to Customer, subject to payment of applicable license fees, a nontransferable, nonexclusive license to use Software solely: (a) in machine-readable, object-code form (except as provided in Subsection 5.2); (b) for Customer's internal business purposes; (c) for the term of the license; and (d) on the computer hardware and at the site authorized by Mentor Graphics. A site is restricted to a one-half mile (800 meter) radius. Customer may have Software temporarily used by an employee for telecommuting purposes from locations other than a Customer office, such as the employee's residence, an airport or hotel, provided that such employee's primary place of employment is the site where the Software is authorized for use. Mentor Graphics' standard policies and programs, which vary depending on Software, license fees paid or services purchased, apply to the following: (a) relocation of Software; (b) use of Software, which may be limited, for example, to execution of a single session by a single user on the authorized hardware or for a restricted period of time (such limitations may be technically implemented through the use of authorization codes or similar devices); and (c) support services provided, including eligibility to receive telephone support, updates, modifications, and revisions. For the avoidance of doubt, if Customer requests any change or enhancement to Software, whether in the course of receiving support or consulting services, evaluating Software, performing beta testing or otherwise, any inventions, product

improvements, modifications or developments made by Mentor Graphics (at Mentor Graphics' sole discretion) will be the exclusive property of Mentor Graphics.

3. **ESC SOFTWARE.** If Customer purchases a license to use development or prototyping tools of Mentor Graphics' Embedded Software Channel ("ESC"), Mentor Graphics grants to Customer a nontransferable, nonexclusive license to reproduce and distribute executable files created using ESC compilers, including the ESC run-time libraries distributed with ESC C and C++ compiler Software that are linked into a composite program as an integral part of Customer's compiled computer program, provided that Customer distributes these files only in conjunction with Customer's compiled computer program. Mentor Graphics does NOT grant Customer any right to duplicate, incorporate or embed copies of Mentor Graphics' real-time operating systems or other embedded software products into Customer's products or applications without first signing or otherwise agreeing to a separate agreement with Mentor Graphics for such purpose.

4. **BETA CODE.**

4.1. Portions or all of certain Software may contain code for experimental testing and evaluation ("Beta Code"), which may not be used without Mentor Graphics' explicit authorization. Upon Mentor Graphics' authorization, Mentor Graphics grants to Customer a temporary, nontransferable, nonexclusive license for experimental use to test and evaluate the Beta Code without charge for a limited period of time specified by Mentor Graphics. This grant and Customer's use of the Beta Code shall not be construed as marketing or offering to sell a license to the Beta Code, which Mentor Graphics may choose not to release commercially in any form.

4.2. If Mentor Graphics authorizes Customer to use the Beta Code, Customer agrees to evaluate and test the Beta Code under normal conditions as directed by Mentor Graphics. Customer will contact Mentor Graphics periodically during Customer's use of the Beta Code to discuss any malfunctions or suggested improvements. Upon completion of Customer's evaluation and testing, Customer will send to Mentor Graphics a written evaluation of the Beta Code, including its strengths, weaknesses and recommended improvements.

4.3. Customer agrees to maintain Beta Code in confidence and shall restrict access to the Beta Code, including the methods and concepts utilized therein, solely to those employees and Customer location(s) authorized by Mentor Graphics to perform beta testing. Customer agrees that any written evaluations and all inventions, product improvements, modifications or developments that Mentor Graphics conceived or made during or subsequent to this Agreement, including those based partly or wholly on Customer's feedback, will be the exclusive property of Mentor Graphics. Mentor Graphics will have exclusive rights, title and interest in all such property. The provisions of this Subsection 4.3 shall survive termination of this Agreement.

5. **RESTRICTIONS ON USE.**

5.1. Customer may copy Software only as reasonably necessary to support the authorized use. Each copy must include all notices and legends embedded in Software and affixed to its medium and container as received from Mentor Graphics. All copies shall remain the property of Mentor Graphics or its licensors. Customer shall maintain a record of the number and primary location of all copies of Software, including copies merged with other software, and shall make those records available to Mentor Graphics upon request. Customer shall not make Products available in any form to any person other than Customer's employees and on-site contractors, excluding Mentor Graphics competitors, whose job performance requires access and who are under obligations of confidentiality. Customer shall take appropriate action to protect the confidentiality of Products and ensure that any person permitted access does not disclose or use it except as permitted by this Agreement. Customer shall give Mentor Graphics written notice of any unauthorized disclosure or use of the Products as soon as Customer learns or becomes aware of such unauthorized disclosure or use. Except as otherwise permitted for purposes of interoperability as specified by applicable and mandatory local law, Customer shall not reverse-assemble, reverse-compile, reverse-engineer or in any way derive any source code from Software. Log files, data files, rule files and script files generated by or for the Software (collectively "Files"), including without limitation files containing Standard Verification Rule Format ("SVRF") and Tcl Verification Format ("TVF") which are Mentor Graphics' proprietary syntaxes for expressing process rules, constitute or include confidential information of Mentor Graphics. Customer may share Files with third parties, excluding Mentor Graphics competitors, provided that the confidentiality of such Files is protected by written agreement at least as well as Customer protects other information of a similar nature or importance, but in any case with at least reasonable care. Customer may use Files containing SVRF or TVF only with Mentor Graphics products. Under no circumstances shall Customer use Software or Files or allow their use for the purpose of developing, enhancing or marketing any product that is in any way competitive with Software, or disclose to any third party the results of, or information pertaining to, any benchmark.

5.2. If any Software or portions thereof are provided in source code form, Customer will use the source code only to correct software errors and enhance or modify the Software for the authorized use. Customer shall not disclose or permit disclosure of source code, in whole or in part, including any of its methods or concepts, to anyone except Customer's employees or contractors, excluding Mentor Graphics competitors, with a need to know. Customer shall not copy or compile source code in any manner except to support this authorized use.

5.3. Customer may not assign this Agreement or the rights and duties under it, or relocate, sublicense or otherwise transfer the Products, whether by operation of law or otherwise ("Attempted Transfer"), without Mentor Graphics' prior written consent and payment of Mentor Graphics' then-current applicable relocation and/or transfer fees. Any Attempted Transfer without Mentor Graphics' prior written consent shall be a material breach of this Agreement and may, at Mentor Graphics' option, result in the immediate termination of the Agreement and/or the licenses granted under this Agreement. The terms of this Agreement, including without limitation the licensing and assignment provisions, shall be binding upon Customer's permitted successors in interest and assigns.

5.4. The provisions of this Section 5 shall survive the termination of this Agreement.

6. **SUPPORT SERVICES.** To the extent Customer purchases support services, Mentor Graphics will provide Customer updates and technical support for the Products, at the Customer site(s) for which support is purchased, in accordance with Mentor Graphics' then current End-User Support Terms located at http://supportnet.mentor.com/about/legal/.

7. **AUTOMATIC CHECK FOR UPDATES; PRIVACY.** Technological measures in Software may communicate with servers of Mentor Graphics or its contractors for the purpose of checking for and notifying the user of updates and to ensure that the Software in use is licensed in compliance with this Agreement. Mentor Graphics will not collect any personally identifiable data in this process and will not disclose any data collected to any third party without the prior written consent of Customer, except to Mentor Graphics' outside attorneys or as may be required by a court of competent jurisdiction.

8. **LIMITED WARRANTY.**

8.1. Mentor Graphics warrants that during the warranty period its standard, generally supported Products, when properly installed, will substantially conform to the functional specifications set forth in the applicable user manual. Mentor Graphics does not warrant that Products will meet Customer's requirements or that operation of Products will be uninterrupted or error free. The warranty period is 90 days starting on the 15th day after delivery or upon installation, whichever first occurs. Customer must notify Mentor Graphics in writing of any nonconformity within the warranty period. For the avoidance of doubt, this warranty applies only to the initial shipment of Software under an Order and does not renew or reset, for example, with the delivery of (a) Software updates or (b) authorization codes or alternate Software under a transaction involving Software re-mix. This warranty shall not be valid if Products have been subject to misuse, unauthorized modification or improper installation. MENTOR GRAPHICS' ENTIRE LIABILITY AND CUSTOMER'S EXCLUSIVE REMEDY SHALL BE, AT MENTOR GRAPHICS' OPTION, EITHER (A) REFUND OF THE PRICE PAID UPON RETURN OF THE PRODUCTS TO MENTOR GRAPHICS OR (B) MODIFICATION OR REPLACEMENT OF THE PRODUCTS THAT DO NOT MEET THIS LIMITED WARRANTY, PROVIDED CUSTOMER HAS OTHERWISE COMPLIED WITH THIS AGREEMENT. MENTOR GRAPHICS MAKES NO WARRANTIES WITH RESPECT TO: (A) SERVICES; (B) PRODUCTS PROVIDED AT NO CHARGE; OR (C) BETA CODE; ALL OF WHICH ARE PROVIDED "AS IS."

8.2. THE WARRANTIES SET FORTH IN THIS SECTION 8 ARE EXCLUSIVE. NEITHER MENTOR GRAPHICS NOR ITS LICENSORS MAKE ANY OTHER WARRANTIES EXPRESS, IMPLIED OR STATUTORY, WITH RESPECT TO PRODUCTS PROVIDED UNDER THIS AGREEMENT. MENTOR GRAPHICS AND ITS LICENSORS SPECIFICALLY DISCLAIM ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY.

9. **LIMITATION OF LIABILITY.** EXCEPT WHERE THIS EXCLUSION OR RESTRICTION OF LIABILITY WOULD BE VOID OR INEFFECTIVE UNDER APPLICABLE LAW, IN NO EVENT SHALL MENTOR GRAPHICS OR ITS LICENSORS BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES (INCLUDING LOST PROFITS OR SAVINGS) WHETHER BASED ON CONTRACT, TORT OR ANY OTHER LEGAL THEORY, EVEN IF MENTOR GRAPHICS OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL MENTOR GRAPHICS' OR ITS LICENSORS' LIABILITY UNDER THIS AGREEMENT EXCEED THE AMOUNT RECEIVED FROM CUSTOMER FOR THE HARDWARE, SOFTWARE LICENSE OR SERVICE GIVING RISE TO THE CLAIM. IN THE CASE WHERE NO AMOUNT WAS PAID, MENTOR GRAPHICS AND ITS LICENSORS SHALL HAVE NO LIABILITY FOR ANY DAMAGES WHATSOEVER. THE PROVISIONS OF THIS SECTION 9 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

10. **HAZARDOUS APPLICATIONS.** CUSTOMER ACKNOWLEDGES IT IS SOLELY RESPONSIBLE FOR TESTING ITS PRODUCTS USED IN APPLICATIONS WHERE THE FAILURE OR INACCURACY OF ITS PRODUCTS MIGHT RESULT IN DEATH OR PERSONAL INJURY ("HAZARDOUS APPLICATIONS"). NEITHER MENTOR GRAPHICS NOR ITS LICENSORS SHALL BE LIABLE FOR ANY DAMAGES RESULTING FROM OR IN CONNECTION WITH THE USE OF MENTOR GRAPHICS PRODUCTS IN OR FOR HAZARDOUS APPLICATIONS. THE PROVISIONS OF THIS SECTION 10 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

11. **INDEMNIFICATION.** CUSTOMER AGREES TO INDEMNIFY AND HOLD HARMLESS MENTOR GRAPHICS AND ITS LICENSORS FROM ANY CLAIMS, LOSS, COST, DAMAGE, EXPENSE OR LIABILITY, INCLUDING ATTORNEYS' FEES, ARISING OUT OF OR IN CONNECTION WITH THE USE OF PRODUCTS AS DESCRIBED IN SECTION 10. THE PROVISIONS OF THIS SECTION 11 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

12. **INFRINGEMENT.**

12.1. Mentor Graphics will defend or settle, at its option and expense, any action brought against Customer in the United States, Canada, Japan, or member state of the European Union which alleges that any standard, generally supported Product acquired by Customer hereunder infringes a patent or copyright or misappropriates a trade secret in such jurisdiction. Mentor Graphics will pay costs and damages finally awarded against Customer that are attributable to the action. Customer understands and agrees that as conditions to Mentor Graphics' obligations under this section Customer must: (a) notify Mentor Graphics promptly in writing of the action; (b) provide Mentor Graphics all reasonable information and assistance to settle or defend the action; and (c) grant Mentor Graphics sole authority and control of the defense or settlement of the action.

12.2. If a claim is made under Subsection 12.1 Mentor Graphics may, at its option and expense, (a) replace or modify the Product so that it becomes noninfringing; (b) procure for Customer the right to continue using the Product; or (c) require the return of the Product and refund to Customer any purchase price or license fee paid, less a reasonable allowance for use.

12.3. Mentor Graphics has no liability to Customer if the action is based upon: (a) the combination of Software or hardware with any product not furnished by Mentor Graphics; (b) the modification of the Product other than by Mentor Graphics; (c) the use of other than a current unaltered release of Software; (d) the use of the Product as part of an infringing process; (e) a product that Customer makes, uses, or sells; (f) any Beta Code or Product provided at no charge; (g) any software provided by Mentor Graphics' licensors who do not provide such indemnification to Mentor Graphics' customers; or (h) infringement by Customer that is deemed willful. In the case of (h), Customer shall reimburse Mentor Graphics for its reasonable attorney fees and other costs related to the action.

12.4. THIS SECTION 12 IS SUBJECT TO SECTION 9 ABOVE AND STATES THE ENTIRE LIABILITY OF MENTOR GRAPHICS AND ITS LICENSORS FOR DEFENSE, SETTLEMENT AND DAMAGES, AND CUSTOMER'S SOLE AND EXCLUSIVE REMEDY, WITH RESPECT TO ANY ALLEGED PATENT OR COPYRIGHT INFRINGEMENT OR TRADE SECRET MISAPPROPRIATION BY ANY PRODUCT PROVIDED UNDER THIS AGREEMENT.

13. **TERMINATION AND EFFECT OF TERMINATION.** If a Software license was provided for limited term use, such license will automatically terminate at the end of the authorized term.

13.1. Mentor Graphics may terminate this Agreement and/or any license granted under this Agreement immediately upon written notice if Customer: (a) exceeds the scope of the license or otherwise fails to comply with the licensing or confidentiality provisions of this Agreement, or (b) becomes insolvent, files a bankruptcy petition, institutes proceedings for liquidation or winding up or enters into an agreement to assign its assets for the benefit of creditors. For any other material breach of any provision of this Agreement, Mentor Graphics may terminate this Agreement and/or any license granted under this Agreement upon 30 days written notice if Customer fails to cure the breach within the 30 day notice period. Termination of this Agreement or any license granted hereunder will not affect Customer's obligation to pay for Products shipped or licenses granted prior to the termination, which amounts shall be payable immediately upon the date of termination.

13.2. Upon termination of this Agreement, the rights and obligations of the parties shall cease except as expressly set forth in this Agreement. Upon termination, Customer shall ensure that all use of the affected Products ceases, and shall return hardware and either return to Mentor Graphics or destroy Software in Customer's possession, including all copies and documentation, and certify in writing to Mentor Graphics within ten business days of the termination date that Customer no longer possesses any of the affected Products or copies of Software in any form.

14. **EXPORT.** The Products provided hereunder are subject to regulation by local laws and United States government agencies, which prohibit export or diversion of certain products and information about the products to certain countries and certain persons. Customer agrees that it will not export Products in any manner without first obtaining all necessary approval from appropriate local and United States government agencies.

15. **U.S. GOVERNMENT LICENSE RIGHTS.** Software was developed entirely at private expense. All Software is commercial computer software within the meaning of the applicable acquisition regulations. Accordingly, pursuant to US FAR 48 CFR 12.212 and DFAR 48 CFR 227.7202, use, duplication and disclosure of the Software by or for the U.S. Government or a U.S. Government subcontractor is subject solely to the terms and conditions set forth in this Agreement, except for provisions which are contrary to applicable mandatory federal laws.

16. **THIRD PARTY BENEFICIARY.** Mentor Graphics Corporation, Mentor Graphics (Ireland) Limited, Microsoft Corporation and other licensors may be third party beneficiaries of this Agreement with the right to enforce the obligations set forth herein.

17. **REVIEW OF LICENSE USAGE.** Customer will monitor the access to and use of Software. With prior written notice and during Customer's normal business hours, Mentor Graphics may engage an internationally recognized accounting firm to review Customer's software monitoring system and records deemed relevant by the internationally recognized accounting firm to confirm Customer's compliance with the terms of this Agreement or U.S. or other local export laws. Such review may include FLEXlm or FLEXnet (or successor product) report log files that Customer shall capture and provide at Mentor Graphics' request. Customer shall make records available in electronic format and shall fully cooperate with data gathering to support the license review. Mentor Graphics shall bear the expense of any such review unless a material non-compliance is revealed. Mentor Graphics shall treat as confidential information all information gained as a result of any request or review and shall only use or disclose such information as required by law or to enforce its rights under this Agreement. The provisions of this Section 17 shall survive the termination of this Agreement.

18. **CONTROLLING LAW, JURISDICTION AND DISPUTE RESOLUTION.** The owners of certain Mentor Graphics intellectual property licensed under this Agreement are located in Ireland and the United States. To promote consistency around the world, disputes shall be resolved as follows: excluding conflict of laws rules, this Agreement shall be governed by and construed under the laws of the State of Oregon, USA, if Customer is located in North or South America, and the laws of Ireland if Customer is located outside of North or South America. All disputes arising out of or in relation to this Agreement shall be submitted to the exclusive jurisdiction of the courts of Portland, Oregon when the laws of Oregon apply, or Dublin, Ireland when the laws of Ireland apply. Notwithstanding the foregoing, all disputes in Asia arising out of or in relation to this Agreement shall be resolved by arbitration in Singapore before a single arbitrator to be appointed by the chairman of the Singapore International Arbitration Centre ("SIAC") to be conducted in the English language, in accordance with the Arbitration Rules of the SIAC in effect at the time of the dispute, which rules are deemed to be incorporated by reference in this section. This section shall not

restrict Mentor Graphics' right to bring an action against Customer in the jurisdiction where Customer's place of business is located. The United Nations Convention on Contracts for the International Sale of Goods does not apply to this Agreement.

19. **SEVERABILITY.** If any provision of this Agreement is held by a court of competent jurisdiction to be void, invalid, unenforceable or illegal, such provision shall be severed from this Agreement and the remaining provisions will remain in full force and effect.

20. **MISCELLANEOUS.** This Agreement contains the parties' entire understanding relating to its subject matter and supersedes all prior or contemporaneous agreements, including but not limited to any purchase order terms and conditions. Some Software may contain code distributed under a third party license agreement that may provide additional rights to Customer. Please see the applicable Software documentation for details. This Agreement may only be modified in writing by authorized representatives of the parties. Waiver of terms or excuse of breach must be in writing and shall not constitute subsequent consent, waiver or excuse.

Rev. 100615, Part No. 246066