CodeWarrior Development Studio for Power Architecture Processors Simulator User Guide

Document Number: CWPASIMUG
Rev. 10.5.1, 01/2016

Contents

Section number **Title** **Page**

Chapter 1
Introduction

This manual describes the software simulation models released with CodeWarrior Development Studio for Power Architecture® Processors.

This chapter explains:

- Overview
- Accompanying documentation
- CCSSIM

1.1 Overview

CodeWarrior Development Studio for Power Architecture® Processors supports the following simulator models:

- B4xxx ISS simulators: Functional, unified simulator based on SC3900 core ISS simulator and PA e6500 JIT-ISS simulator. Currently, B4860iss and B4420iss simulator models are supported.
- T4xxx ISS simulators: Functional simulator based on PA e6500 JIT-ISS simulator. Currently, T4240iss simulator model is supported, but only for Linux64 systems.

1.2 Accompanying documentation

The Documentation page describes the documentation included in this version of CodeWarrior Development Studio for Power Architecture® Processors. You can access the Documentation page by:

- Clicking a shortcut link on the Desktop that the installer creates by default
- Opening `START_HERE.html` in the `CWInstallDir\PA\Help` folder

1.3 CCSSIM

CCSSIM enables remote access to simulators through the CCS protocol. To run the simulator on a remote computer, execute `ccssim2` on the remote machine using the command prompt.

To start the generic CCS Remote Connection executable, start `ccssim2` from the `CWInstall/PA/ccs/bin` folder. The simulator can interact with the hardware I/O through CCSSIM.

1.3.1 Running a simulator remotely

To run your application on the simulator:

1. Build your source code using the Power Architecture Build Tools (compiler, assembler, and linker).
2. Run `ccssim2` to execute the server. The ccssim2 server is located at the path `CWInstall/PA/ccs/bin`.
3. Use CodeWarrior debugger to debug the application using ccssim through remote connection.

NOTE

For more information, see *CodeWarrior Development Studio for Power Architecture® Processors Version 10.x Targeting Manual*.

Chapter 2
B4xxx System-on-Chip Simulators

This chapter explains all the simulator functions and models that are supported by the B4xxx System-on-Chip (SoC) instruction set simulators, which are B4860 and B4420 functional simulators. These simulators are intended for development of full chip applications. For more information, refer to *PSC9164 SOC Architecture Specification*.

Following features and components are supported by B4xxx SoC instruction set simulators:

- Supported simulator functions
- Peripherals and components
- B4xxx ISS MAPLE B3 support
- Configuring B4xxx simulator
- Working with B4xxx simulator
- Using FM TIO with B4xxx simulator models

2.1 Supported simulator functions

The following table lists the functions supported by the B4xxx instruction set simulator.

Table 2-1. Functions supported by B4xxx instruction set simulator

Function	B4xxx ISS	
	Planned	Current status
Windows 7	Yes	Fully tested
Windows 8	Yes	Smoke tested
Linux64-CentOS 6.3	Yes	Smoke tested
Linux64-Debian 7.1	Yes	Smoke tested
Linux64-Red Hat 5	Yes	Fully tested
Linux64-Red Hat 6.3	Yes	Smoke tested

Table continues on the next page...

Table 2-1. Functions supported by B4xxx instruction set simulator (continued)

Function	B4xxx ISS	
	Planned	Current status
Linux64-OpenSUSE 12.3	Yes	Smoke tested
Linux64-Ubuntu 12.10	Yes	Smoke tested
SC3900 Platform ISS	Yes	Yes
MPIC	Yes	Yes
MAPLE3	Yes	Yes
Virtual Interrupt (VI)	Yes	Yes
e6500 cores	No	Yes
CCM	Yes	Yes
CPC	Yes	Yes
BMAN	Yes	Yes
QMAN	Yes	Yes
FMAN	Yes	Yes
CAAM	Yes	Yes
I2C	Yes	Yes
PBL	Yes	Yes
Connection with CodeWarrior for StarCore and Power Architecture®	Yes	Yes
Traffic IO Support	Yes	Yes
Hardware memory mapped ports for fast IO with files	Yes	Yes
Check Points	Yes	Not supported

2.2 Peripherals and components

The following table lists aspects that you may need while using the B4xxx instruction set simulators.

Table 2-2. B4xxx instruction set simulator aspects

Aspect	Details
T4	Following Blocks are not modeled: • GPIO, SATA, PME, DMA • LA_1, LA1_RUNTIME, TMU • Security fuse processor • eSPI, SDMMC, Security Island • USB and USB PHYs
BMAN	• Error interrupts are not supported.
CCM	• Error Handling not supported. • Stashing not supported. • Configuration and control registers not directly supported.

Table continues on the next page...

Table 2-2. B4xxx instruction set simulator aspects (continued)

Aspect	Details
	• Out of order completion not supported. • Secure boot not supported. • By default CCM creates a dummy law window (scratch space representing cache locking space). Window size is 128 GB and start address is 0x0.
CONFIGUNIT	• Functional only. No timing. • RCPM registers (except Core time base enable register) functionality not supported. • RCPM block is compliant with Blockguide4.0 v0.71 • DCFG block is compliant with Device Configuration 3.0 Block Guide, Rev 1.0.3 • DCFG_COREDISR register functionality is not modeled.
CPC	• Write back cache and I/O stash are not supported. • ATQ priority arbitration (CSSA) is not supported. • Interrupts are not supported.
DUART	Duart does not model Character Timeout condition for Receive FIFO.
DCE	• Functional Model only, No timing • For compression/decompression, zlib-1.2.5 software library is used. It maintains its own history context and residue. Hence following fields in SCR or SCF are not used or set: • HISTORY_PTR • HISTORY_LEN • LIVB • RBC • RESIDUE_DATA • CRC16 • DECOMP_CTXT_PTR • PREVIOUS_CODE_LEN • BFINAL • BTYPE • FRAME_PARSE_STATE • NUM_CODE_LEN • HEADER_REMAINING • NCBB_REMAINING • HLIT, HDIST, HCLEN • HUFFMAN_RBC • HUFFMAN_RESIDUE
DCE	• The flush parameter of Z_TREES is not supported for compression. • Chunked encoding/decoding not supported. Input frame is encoded/decoded all at once - not small by small chunk. There are two side effects due to this restriction: • Bytes_processed would remain 0 until all of output data is written to system memory. Hence in recyling mode, if output blocked suspend exception occurs, bytes_processed will be zero. • If software combines multiple separately compressed+encoded frame into one frame and sends it back to DCE for decompression, the decoding may fail to remove padding that may be there in between the frames. • Single/Double bit error interrupts are never generated automatically by the model; they can only be forced through DCE_IFR register. • ECC error/status functionality is not supported. • The following registers have limitations: • DCE_CFG register - Field DMD is a dummy field with no side effects. • DCE_SMCACR register - Fields CHWC, CHRC, DHWC and DHRC are dummy fields with no side effects. • All of ECC error/status registers are dummy with no side effect.
e6500 Cores	• Functional Model only, Timing not supported. • No Support for External Debug Mode

Table continues on the next page...

Table 2-2. B4xxx instruction set simulator aspects (continued)

Aspect	Details
	• No Support for IAC, DAC, dni, unconditional debug events • No Support for Nexus Registers/Trace • No Support for Performance Management Registers/Interrupts/mtpmr/mfpmr • Timers are approximate since functional model does not model time • DCache is not modeled. DCBZ/DCBA clear memory. Rest of DCache instructions are nopped. Cache locking instructions do not lock cache lines. • ICache is not modeled. ICache instructions are nopped. Cache lock instructions do not lock Icache • wait, miso instructions are not modeled • Only one watchpoint supported per instruction • Machine Check interrupt not supported except for multi hit TLBs in fixed size TLB 4K array.
I2C	• Functional Model only, Timing not supported. • I2C Master Mode is supported, Slave mode is NOT supported. • I2C Interrupts are not modeled. • EEPROM Memory dump is not supported.
IFC	• Functional model only. Timing not supported. • GPCM not supported. • Software reset functionality not supported. • NOR Flash command sequence timeout error and interrupt generation not supported. • Default NOR Flash is 128MB, 1024 sectors with 128KB size and 8x8 port size • NOR Flash Device supports following commands (Remaining are NOT supported): • Flash read array command • Flash Program command • Flash Write to Buffer command • Flash Program Buffer to Flash command • Flash CFI Query command • Flash read ID command • Flash Sector Erase command
KIBO	• Support for greater than 8 byte reservation is only for single threaded mode. • Multi-threaded mode supports only semaphore actions for 4 and 8 bytes.
MPIC	• Functional Model only, No timing • Shared Message Signaling Interrupt Not supported • Timers not supported • No behavioral difference between edge and level sensitive interrupts. Level behavior is not modeled. • Registers not modeled : Shared Message Signaled Coalescing Registers (MSICRx0 - MSICRx15) [x= A,B,C,D] • Registers not modeled : Shared Message Signaled Interrupt Registers (MSIDRx8 - MSIDRx15) [x= A,B,C,D]
PAMU	• PAMU Error Handling not supported.
PMAN	• Algorithms are not supported. • Only register models are supported.
PBL	• Functional Model only, Timing not supported. • Only IFC-NOR interface is supported as RCW source, PBL source and Boot location. • PBL Interface with IFC-NAND not supported. • PBL Interface with I2C not supported. • PBL Interface with SPI not supported. • PBL Interface with SDHC not supported. • PBL Hard coded RCWs not supported.
QMAN	• CEETM's congestion management and associated functional interrupts not yet modeled.

Table continues on the next page...

Table 2-2. B4xxx instruction set simulator aspects (continued)

Aspect	Details
	• CEETM's Shaper not yet modeled. • CEETM's dequeue availability indication doesn't take care of Shaper's credit into account. • Basic Dequeue flow modeled in CEETM mode. Class/ Channel schedulers are much simplified such that each successive dequeue command selects CQs within channels in a strict priority order (favoring lower numbered CQ first) and then selects a CQ channel in a simple round-robin manner. • Real Class/Channel schedulers are yet to be modeled. • CS tail drop rejections (part of congestion management) are not modeled • SFDR_CFG[RM] bit and functionality not supported. • Support for FQD_AR[SO] field is not modeled. • FQD format changes for Cache Status bit, and Query FQ Non-Programmable Fields Response Format change is not implemented. • Avoid Blocking FQD control bit is not supported • Query WQ length command is not supported • CGR Test Write commands are not supported • Congestion state change notification (functional) interrupts are not supported • Error interrupts are not modeled • Debug interface is not modeled • Performance monitor interface is not supported • Dynamic debug interface is not supported • QCSP0-9_EQCR_CI_CENA/CINH: PB field is not supported • QCSPi_DQRR_DCAP[PK] - Park bit not supported
PCIExpress	• Functional Model only, No Timing. Retry, Timeout Register logic not supported. • In Byte Ordering, Data invariance policy not supported. • Error capture feature not supported. • LIODN permission table, PBA table not supported. • VF Migration not supported. • Dependencies (FDL) between PFs not supported. • MSI Generation not tested.
RMAN	• Functional model, Timing not supported. • Type 5 Outbound NWrite not Supported. • Type 6 Outbound SWrite not Supported. • Type 8 Maintenance not Supported. • Flow control management not supported in type-9. • Algorithmic FQ mode not supported.
SERDES	• Only register models are supported.
SRIO	• Functional Model only, No timing • Operations crossing outbound/inbound window borders not modeled. • Retry/priority is not modeled, since this is purely functional model
MPIC	• Functional Model only, timing not supported. • Shared Message Signaling Interrupt is not supported. • Timers are not supported. • No behavioral difference between edge and level sensitive interrupts. Level behavior is not modeled. • Registers not modeled : Shared Message Signaled Coalescing Registers (MSICRx0 - MSICRx15) [x= A,B,C,D] • Registers not modeled : Shared Message Signaled Interrupt Registers (MSIDRx8 - MSIDRx15) [x= A,B,C,D]
Frame Manager	Frame Manager SWSIM is a software simulators package which currently includes a simulation module for P4080/P3040/P5020/P1023/P2040/B4860/T4240/T1040 Frame Manager hardware block. For more information, refer *Frame Manager Software Simulator Release Notes*.

Table continues on the next page...

Table 2-2. B4xxx instruction set simulator aspects (continued)

Aspect	Details
Virtual Interrupt	Virtual Interrupts are implemented, but not really tested with sc3900 interrupts: It gets interrupts from the program and transmits them to the 6 sc3900 cores.
MAPLE3 Support	Refer the B4xxx ISS MAPLE B3 support section for details.

2.3 B4xxx ISS MAPLE B3 support

The MAPLE B3 support by the B4xxx ISS models provides the following features:

Memory map

Three MAPLE 3 elements (two LW and one W) are implemented.

- SBus memory is implemented at address 0x[F]fe800000 (partial logic, see unimplemented features below) can be configured by CCSRBAR
- MBus memory is implemented at address 0x[F]e0000000 (partial logic, see unimplemented features below) including:
  - DRAM
  - eTVPE2 registers
  - DEPE2 registers
  - EQPE2 registers
  - PDPE2 registers
  - PUPE2 registers
  - eFTPE2 registers (both in L&W) (LTE WCDMA)
- In MAPLE3 only
  - DL2 registers
  - ULF2 registers
  - ULB2 registers
  - CGPE registers (W)
  - TCPE registers (W)

PSIF3 registers

- 4 RISC cores are simulated, including registers on the SBus memory.
- Internal DMA is simulated partially (including MMU).
- DTU is simulated.
- PIC is simulated.
- Scheduler is simulated.

eTVPE2

- Registers memory map is implemented and reset values.
- Pipeline mode not supported.
- Passes basic tests for 3GLTE and UMTS.
- Some modes do not support. eTVPE2::UMTS::EDCH Separated vector (Not implemented in model).
- Failed on full SDOS test (Multicore)

eFTPE2

- Registers memory map is implemented and reset values.
- Passes basic tests.

DEPE2

- Registers memory map is implemented and reset values.
- Passes basic tests.
- Failed on full SDOS test (iterations).

EQPE2

- Registers memory map is implemented and reset values.
- Testing in progress.

PDPE2

- Registers memory map is implemented and reset values.
- Passes basic tests.

PUPE2

- Registers memory map is implemented and reset values.
- Testing in progress.

DL2

- Registers memory map is implemented and reset values.
- Testing in progress.

ULF2

- Registers memory map is implemented and reset values.
- Testing in progress.

ULB2

- Registers memory map is implemented and reset values.
- Testing in progress.

CGPE registers

- Registers memory map is implemented and reset values.
- Still at testing.

TCPE registers

- Registers memory map is implemented and reset values.
- Still at testing.

Buffer Descriptor (BD) programming model

Up to 8 High Priority BD rings and 8 Low Priority BD rings for each processing element for multiple master support.

Flexible interrupt scheme

Supports BD interrupts mapped from IRQ_242 and up.

CRPE

- Full CPRI data path does not get simulated. Currently, CPRI-CRPE interactions are supported.
- Part of the registers implemented in order to perform WCDMA demo.
- Only CPRI 0 registers are supported at the moment (although there are six CPRI modules)
- The CRPE put and get data to and from files in predefined format:
    - For UpLink up to 24 antenna files

      data structure:

      ant 0 file for OVS 2 :

      ```
      chip 0 phaze 0 <8I,8Q>

      chip 0 phaze 1 <8I,8Q>

      chip 1 phaze 0 <8I,8Q>

      chip 1 phaze 1 <8I,8Q>
      ```

      ant 0 file for no OVS :

      ```
      chip 0 phaze 0 <8I,8Q>

      chip 1 phaze 0 <8I,8Q>
      ```

    - DownLink: (only in CPRI mode in CRPE-DL) 16 antenna output files (or less)<16I, 16Q>

      data structure:

      ```
      chip 0 <16I,16Q>

      chip 1 <16I,16Q>
      ```

NOTE

DL writes out to file every 2083 system clocks. UL reads from file every 2083*16 system clocks.

2.4 Configuring B4xxx simulator

Following sections discuss the various files and options supported by the B4xxx instruction set simulators:

- Initialization files
- Simulator options
- Examples

2.4.1 Initialization files

B4xxx ISS supports the following initialization files:

- `b4xxxiss.ini` - Required for Power Architecture components initialization. This file is loaded at startup.
- `b4xxxiss_sim_init_params.cfg` - Provides default or configurable simulator initialization parameters. This file is loaded at startup.

2.4.2 Simulator options

B4xxx ISS can be executed on both `runsim` and `ccssim2`, using the following configuration parameters:

```
./runsim -imodel "option=value -option... " -smodel "option=value -
option... " -d b4xxxiss -nc 1 test.eld
```

```
./ccssim2 -imodel "option=value -option... " -smodel "option=value -
option..." -port 41475
```

Tip

`-imodel` options are used during the initialization of the simulator (Init Model). Whereas, the `-smodel` options are used after the initialization of the simulator, before starting the execution.

NOTE

`runsim` and `ccssim2` can run without `-imodel` and `-smodel` options. In this case the default values are used. For example: `./runsim -d b4xxxiss -nc 1 test.eld`

The configuration options supported by `imodel` and `smodel`, to run B4xxx ISS with runsim and ccssim2, are discussed in the following tables:

Table 2-3. Running B4xxx ISS with runsim or ccssim2 - Main options

Option	Supported by		Description	Default	Comments
	imodel	smodel			
`sim_pa_run_vcores=xxxxxxxx`	Yes	No	Enables specific PA cores.	00000000	Same as runsim option -pnc
`sc_pa_maple_run=n1:n2:n3`	Yes	Yes	Enable execution phase at startup for StarCore, Power Architecture and MAPLE cores	0:0:1	
`sc_pa_maple_ratios=n1:n2:n3`	Yes	Yes	The execution phase cycle ratios for StarCore, Power Architecture and Maple	20:1:10	Use this option for speed/ synchronization/ execution tunings. Zero means no exec cycles for the corresponding component
`sc_pa_maple_sync_exec=n1:n2:n3`	Yes	Yes	This option may be used to enable/ disable synchronization of MAPLE execution with StarCore and PA cores execution.	1:0:1	Currently, only the StarCore cores are taken into account. This means that MAPLE doesn't get cycles when all StarCore cores are stopped.
`sim_tio_hub=tio_server_name:tio_server_port`	Yes	No	Enables TIO support in B4xxx simulator and configures TIO server name and port	Values are set in *b4xxxiss_sim_init_params. cfg* file.	
`sim_pa_run_vcores=<enable_mask>`	Yes	No	Enables/disables (1/0) run at reset for a PA virtual cores (PA core thread0 or thread1)	10000000 Enable run at reset for the 1'st virtual core (PA core0 / thread0).	10101010 To enable run at reset for all main threads (thread0 of all PA cores).

Table continues on the next page...

Table 2-3. Running B4xxx ISS with runsim or ccssim2 - Main options (continued)

Option	Supported by		Description	Default	Comments
	imodel	smodel			
`-MULTITHREAD`	Yes	No	Enable multithreading: 1 thread for StarCore, 1 thread for PA cores, 1 thread for each MAPLE3 (A, B, C).	Disabled (single-core)/ Enabled (multi- core)	
`-NOMULTITHREAD`	Yes	No	Disables multithreading	Disabled (multi-core)/ Enabled (single-core)	

NOTE

For all options like `sc_pa_maple_[*]=n1:n2:n3`; `n1` stands for StarCore, `n2` stands for Power Architecture and `n3` stands for MAPLE.

Table 2-4. Running B4xxx ISS with runsim or ccssim2 - Advanced options

Option	Supported by		Description	Default	Comments
	imodel	smodel			
`sim_log_level=<level>`	Yes	Yes	Sets log level (NONE=0, ERROR=1, WARNING=2, INFO=3, DEBUG=4, TRACE=5). Level >=0.	1	
`sim_log_file=<sim.log>`	Yes	Yes	Writes log into a log file		
`ipmodels_log_level=<level>`	Yes	Yes	Sets log level for ipmodels components. Level >=0.	0	
`pa_sim_config_file=<test.cfg>`	Yes	Yes	Set configuration file loaded at startup.	No configuration file	It can be used to load binary images, write memory mapped registers (see linux_ir0/ test_uboot_linux.cfg)
`mapletrace_file <maple.trc>`	No	Yes	Maple trace file	No trace file	
`mapletrace start/stop`	No	Yes	Start/stop maple ucode trace	stop	

Table continues on the next page...

Table 2-4. Running B4xxx ISS with runsim or ccssim2 - Advanced options (continued)

Option	Supported by		Description	Default	Comments
	imodel	smodel			
`-cpricfg <cpri_config.xml>`	Yes	No	Load CPRI configuration file		
use_fm_risc_exec_one	No	Yes	Sets the FM_RISC exec mode to one cycle or one instructions	off	

Table 2-5. Running B4xxx ISS with runsim or ccssim2 - Internal options

Option	Supported by		Description	Default	Comments
	imodel	smodel			
`pa_sim_jit_trace=<level>`	Yes	Yes	Enable trace for the e6500 JIT cores. Level >=0	0	

Table 2-6. Running B4xxx ISS with runsim or ccssim2 - Configuration files

File Name	Description	Default	Comments
`sim.enable_fman_tio`	Enables FMAN TIO support. When this option is enabled you could use fm_tio_inject and fm_tio_capture applications to inject pcap files.	false	
`sim.jit_run_quanta`	Maximum number of instructions to be executed on the exec phase for PA cores.	1000	
`sim.clock_dpa`	Enables DPAA clock	false	
`sim.dpa_clock_divisor`	Configures the DPAA clock ratio (sim.jit_run_quanta/ sim.dpa_clock_divisor)	100	
`fman().log_filename`	Setlog filename for FMAN().	fman.log	
`fman().log_level`	Setlog level for FMAN()	0	
`sim.jit_trace`	Enable trace for PA cores.	false	
`sim.jit_log_level`	Sets log level for PA cores	0	
`sim.dpa_bman_log_level sim.dpa_qman_log_level sim.dpa_caam_log_level sim.dpa_fman0_log_levelsim.dpa_fman1_log_level`	Sets log level for BMAN, QMAN, CAAM, FMAN wrapper.	0 for all	
`configunit.model_p5020`	SET configunit model_p5020=1	0	

Table continues on the next page...

Table 2-6. Running B4xxx ISS with runsim or ccssim2 - Configuration files (continued)

File Name	Description	Default	Comments
`ccm.laws`	Configures custom LAW entries.	Is set in b4xxxiss_sim_init_params.cfg	
`sim.vipr_run_pa_vcores`	Vipr: enable run for PA virtual cores. By default, enables run for main thread for each PA core. Used when running with vipr frontend.	10101010	
`sim.sc_run_at_reset`	True if the simulator puts StarCore cores in run mode at reset.	false	
`sim.pa_run_at_reset`	True if the simulator puts PA cores in run mode at reset.	false	
`sim.maple_run_at_reset`	True if the simulator puts MAPLE cores in run mode at reset.	false	
`sim.starcore_dmi`	True if the simulator uses the dmi memory management for StarCore memory accesses.	false	
`sim.mth_mng_sync_exec`	True if the simulator initialize the threads manager to synchronize execution of starcore, pa and maple threads by ratios.	true	
`sim.enable_dnh_halt`	True if the simulator should stop on DNH instructions.	true	
`sim.mpic_mixed_mode`	True if the simulator should Set Mixed Mode for interrupts to go to CPUs.	true	
`sim.uninit_mem_fill_word`	Return value from uninitialized memory.	0x0	
`sim.pa_num_cores`	Number of Power Architecture cores.	4	
`sim.sync_timers`	Sync Multi-core timers after each quanta-iteration.	false	
`sim.enable_global_timer`	Enable Global Timer for maintaining time for all cores.	true	
`sim.jit_run_mode`	False if the simulator uses step mode.	true	
`sim.jit_print_speed_info`	True if the simulator reports speed info for jit cores.	false	
`sim.tio`	True if the simulator uses a TIO server -- output goes to the console.	false	
`sim.tio_server_name`	The port that the TIO server should be listening for consoles.	localhost	

Table continues on the next page...

Table 2-6. Running B4xxx ISS with runsim or ccssim2 - Configuration files (continued)

File Name	Description	Default	Comments
`sim.tio_server_port`	The port that the TIO server should be listening for consoles.	41969	
`sim.fm_risc_debug`	Activates FMAN RISCs debug support.	false	
`sim.use_functional_qbfman`	Use pure functional qman/ bman/fman components.	true	
`sim.dpa_quantum`	No. dpa clock ticks given to CAAM and FMan. This parameter is valid only if sim.use_functional_dpa is set to true	1	
`sim.clock_dpa`	Should the DPA (FMan, BMan, QMan, DCE, and so on) be clocked? Used for MSS simulations.	false	
`sim.dpa_clock_divisor`	Ratio of no. of per-core instructions and dpa clock ticks (=jit_run_quanta/ dpa_clock_divisor, 1 if jit_run_quanta is 0).	100	
`sim.disable_caam`	Disables CAAM. This parameter is valid only if sim.use_functional_dpa is set to true.	false	

2.4.3 Examples

Listing below shows the configuration to enable execution only on StarCore.

Listing 2-1. Enable execution only on StarCore

```
./runsim -smodel "sc_pa_maple_run=0:0:0 sc_pa_maple_ratios=1:0:0" -d
b4xxxiss -nc 1 test.eld
./ccssim2 -smodel "sc_pa_maple_run=0:0:0 sc_pa_maple_ratios=1:0:0" -
port 41475
```

Listing below shows the configuration to enable execution only on StarCore and Maple.

Listing 2-2. Enable execution only on StarCore and Maple

```
./runsim -smodel "sc_pa_maple_run=0:0:1 sc_pa_maple_ratios=20:0:10" -d
b4xxxiss -nc 1 test.eld
./ccssim2 -smodel "sc_pa_maple_run=0:0:1 sc_pa_maple_ratios=20:0:10" -
port 41475
```

Listing below shows the configuration to enable execution only on Power Architecture.

Listing 2-3. Enable execution only on Power Architecture

```
./ccssim2 -smodel "sc_pa_maple_ratios=0:1:0" -port 41475
```

Listing below shows the configuration to enable TIO support.

Listing 2-4. Enable TIO support

```
./runsim -imodel "sim_tio_hub=10.1.171.5:42476" -d b4xxxiss -nc 1
test.eld
./ccssim2 -imodel "sim_tio_hub=10.1.171.5:42476" -port 41475
```

Listing below shows the configuration to boot Linux with runsim.

Listing 2-5. Boot Linux with runsim

```
./runsim -smodel " sc_pa_maple_run=0:1:0 sc_pa_maple_ratios=0:1:0
pa_sim_config_file=./linux_ir0/test_uboot_linux.cfg" -d b4xxxiss -nc 1
./linux_ir0/starcore_endless_loop.eld
```

Listing below shows the configuration to set log level and start Maple trace.

Listing 2-6. Set log level and start Maple trace

```
./ccssim2 -imodel "sim_log_level=5 -MULTITHREAD" -smodel
"sim_log_level=4 sim_log_file=sim.log sc_pa_maple_ratios=1000:1:1
mapletrace start mapletrace_file maple.trc"
```

The -smodel option can be configured at runtime, from a CCS server console (if the connection with the simulator is done through CCS server)

Listing below shows the configuration of -smodel option at runtime.

Listing 2-7. Configure -smodel at runtime

```
::ccs::strcmd 0 "MODEL_MESSAGE sim_log_level=5 " 0

::ccs::strcmd 0 "MODEL_MESSAGE sim_log_level=0 " 0

::ccs::strcmd 0 "MODEL_MESSAGE pa_sim_jit_trace =1 " 0

::ccs::strcmd 0 "MODEL_MESSAGE pa_sim_jit_trace =0 " 0

::ccs::strcmd 0 "MODEL_MESSAGE sc_pa_maple_ratios =100:1:0 " 0

::ccs::strcmd 0 "MODEL_MESSAGE sc_pa_maple_ratios =20:1:10 " 0
```

2.5 Working with B4xxx simulator

For details on how to boot Linux and run U-boot on the B4xxx simulator, refer *readme.txt* file, available in the release package. For example, *dtsim_release \linux64\linux_ir0\readme.txt*.

2.6 Using FM TIO with B4xxx simulator models

In order to use FM TIO support with B4xxx simulator models, the following parameters need to be configured in the *b4xxxiss_sim_init_params.cfg* file:

- Enable fman tio support `sim.enable_fman_tio=true`
- Enable DPA clock `sim.clock_dpa=true`
- Set quanta for PA cores (optional) `sim.jit_run_quanta=5000`
- Set DPA clock divisor (optional) `sim.dpa_clock_divisor=2500`
- Set log level for fman (optional) `fman0.log_level=6`
- Set configunit model_p5020 (if working with older applications/tests) `configunit.model_p5020=1`

Following sections discuss the FM TIO support with B4xxx simulator models in detail:

- Injecting and capturing frames with fm_tio_inject and fm_tio_capture applications
- Connecting a virtual network interface with B4xxx simulator models for packet injection (ping)

2.6.1 Injecting and capturing frames with fm_tio_inject and fm_tio_capture applications

To inject and capture frames using fm_tio_inject and fm_tio_capture applications, follow these steps:

1. Run ccssim2 by setting 'sim_tio_hub' imodel. For example: `./ccssim2 -imodel "sim_tio_hub=10.171.71.105:42475" -port 41475`
2. Load and run an external traffic use-case with CCS or CodeWarrior. This step should configure the FM MAC(s) ports to work with external traffic.

Listing 2-8. Example CCS

```
delete all
config networktimeout 6000

config cc ccs:10.171.71.105:41475

ccs::config_chain 197
```

```
::ccs::strcmd 0 "MODEL_MESSAGE pa_sim_config_file=path_to_your_test/
test.cfg" 0

# run core(s)

::ccs::run_core 8
```

Wait, until the initialization has finished, to inject the traffic.

3. Run `fm_tio_capture` application in a new terminal window. For example: `./fm_tio_capture -hub 10.171.71.105:42475 -ser f0_m0 f0_m1 -verbose_level 2`

   Where `f0_m0` stands for FMAN0, MAC0 and `f0_m1` stands for FMAN0, MAC1.

4. Run `fm_tio_inject` application in a new terminal window. For example, inject traffic on FMAN0, MAC0: `./fm_tio_inject -hub 10.171.71.105:42475 -ser f0_m0 -file path_to_your_test/test.pcap -delay 1000 -verbose_level 2`

NOTE

If the simulator doesn't start because it cannot start tio server, another TIO server port number should be used (for example, instead of 42475 use 42476). The same port must be used for all TIO applications. Listing below shows an example of same port used for all TIO applications.

Listing 2-9. Same port used for all TIO applications

```
./ccssim2 -imodel "sim_tio_hub=10.171.71.105:42476" -port 41475
./fm_tio_capture -hub 10.171.71.105:42476 -ser f0_m0 f0_m1 -
verbose_level 2

./fm_tio_inject -hub 10.171.71.105:42476 -ser f0_m0 -file
path_to_your_test/test.pcap -delay 1000 -verbose_level 2
```

2.6.2 Connecting a virtual network interface with B4xxx simulator models for packet injection (ping)

To connect a virtual network interface with B4xxx simulator, for packet injection (ping), follow these steps:

1. Create the virtual interface(s) on the Linux64 machine by using `fm.sh` script. `./fm.sh -s -u user_name -f 0 -i 0 up`

NOTE

You need to have root/sudo permissions to execute the above instruction.

2. Use `ifconfig` to see the interfaces. For example: `user_name-f0e0 Link encap:Ethernet HWaddr 02:00:C0:A8:0A:01 inet addr:192.168.10.1 Bcast:192.168.11.255 Mask: 255.255.254.0UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1RX packets:0 errors:0 dropped:0 overruns:0 frame:0TX packets:0 errors:0 dropped:54 overruns:0 carrier: 0collisions:0 txqueuelen:500 RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)`
3. Start the simulator with TIO support enabled.
   a. Run ccssim2 in a terminal window on the Linux64 machine, by setting 'sim_tio_hub' imodel option. For example: `./ccssim2 -imodel "sim_tio_hub=10.171.71.105:42475" -port 41475`
   b. Enable TIO support in the configuration file. For example: `sim.tio=truesim.tio_server_name=10.171.71.105sim.tio_server_port=42475`
4. Start an external-traffic application on the simulator. The application should support frames receive/transmit logics (ping receive / transmit). Wait until the initialization is done, before injecting the traffic.
5. Start `tio_bridge` in a terminal window on the Linux64 machine. For example: `sudo ./tio_bridge -hub 10.171.71.105:42475 -ser f0_m0 -dev user_name-f0e0`

NOTE

You need to have root privileges or sudo permissions to run `tio_bridge` application.

Listing 2-10. Expected output

```
[user_name@aaa-bbb-105 linux64]$ sudo ./tio_bridge -hub
10.171.71.105:42475 -ser f0_m0 -dev user_name-f0e0
..

=============================

TIO hub      : 10.171.71.105:42475

Serials      : f0_m0

Net card     :user_name-f0e0

Flags        : none

=============================

Bridge is operational
```

6. Run ping in a terminal window on the Linux64 machine. Run ping on different IP address than the one set for the virtual interface (192.168.10.1). The ping should reach `FMAN MAC0`.

NOTE

The ping address should match the network mask for the virtual interfaces (the mask is 255.255.254.0) but not the ip address itself. For instance pinging 192.168.10.2 that matches `user_name_f0e0` interface should reach `fman-mac0`.

Index

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number CWPASIMUG
Revision 10.5.1, 01/2016