XM-Q: Performance Analysis

Xtratum 1.0.8 - Release

Fent Innovative Software Solutions

Reference:	`011.svalr.perf.003`
Status:	Review
Date:	13/07/2017
Last page:	22

This page is intentionally left blank

Document control page

Title:	XM-Q: Performance Analysis
Project:	Xtratum 1.0.8 - Release
Reference:	011.svalr.perf.003
Status:	Review
Date:	13/07/2017
Last page:	22

Written	13/07/2017	A. Esquinas
Reviewed	-	(none)
Approved	-	(none)

Summary: This document details the performance tests and report

Referencing this document:

```
@techreport {011.svalr.perf,
      title = {Xtratum 1.0.8 - Release --- XM-Q: Performance
      Analysis},
      author = { Fent Innovative Software Solutions},
      institution = {Fent Innovative Software Solutions},
      number = {011.svalr.perf.003},
      year={13/07/2017},
}
```

Change record:

Version	Date	Author	Comments
001	10/05/2017	A. Esquinas	Performance Results of XM-Q-1.0.7 for MMB project. Document based in “XM-Q Software Validation Report.”
002	05/06/2017	A. Esquinas	LEON3FT B2BST Errata workaround applied to XM source code. Change compiler to BCC 1.0.46 - GCC 4.4.2.
003	13/07/2017	A. Esquinas	Performance Results of XM-Q-1.0.8.

Contents

This page is intentionally left blank

Chapter 1

Introduction

This document contains the performance metrics report of XtratuM 1.0.8.

The document is structured in two parts: Performance test definition and Performance results. Section 1.1 defines the metric to be used. Section 1.2 specified the performance test defined to cover the metric. Section 2 of the document provides the results of the evaluation. Section 3 of the document provides the results of the analysis of footprint of XtratuM depending on the configuration. Finally, section 4 of the document provides the result of the analysis of the *LEON3FT Stale Cache Entry After Store with Data Tag Parity Error*.

This Report documents the results of the validation tests on GR-712 Development Board

1.1 Performance Metrics

In this section, we propose a metric that cover similar aspects to the defined for RTOS and can give a complete idea about the performance of the hypervisor.

The metric proposes to evaluate the following parameters:

1. Maximum and average Partition Context switch time: maximum and average value for switching between two partitions in the scheduling plan.
2. Effective slot time of partition execution: time used by the partition code in a slot.
3. Service cost: evaluation of the maximum and average cost of the hypervisor services.
4. Latency to interrupts: evaluation of the latency time when an interrupt is raised and the time it is handled at partition level.

The detailed specification of the test cases for the collection of the above metrics is presented in the following section.

1.2 Performance Tests

1.2.1 Partition Context Switch (PCS)

The partition context switch (PCS) refers to the time needed by the hypervisor to switch form one partition to the next one as specified in the plan. The operations performed by the hypervisor are:

1. Detection of the clock interrupt
2. Save the context of the Pi
3. Perform some internal verification test related to the time and the coherence with respect to the configuration vector
4. Select the next partition to be executed from the scheduling plan defined in the configuration vector
5. Set the interrupt mask and memory maps
6. Recover the state of the next partition
7. Jump to next partition

Table 1.1: T-XM-PERFO-050-000

Test: T-XM-PERFO-050-000-000
TEST PURPOSE: To measure the time needed by the hypervisor to switch from one partition in the scheduling plan and the next partition to be executed.
Test class: Performance
Environment scenario: Partitions: 2 partitions: P0 is user partition and P1 is user partition

TEST DESCRIPTION:
`The internal code of XtratuM has been instrumented to get the exact instant times of the PCS. These exact times are:`
`1. A clock interrupt signalling the end of the slot is received by the hypervisor (t1)`
`2. The return of the hypervisor code to the partition code (t2)`
 	
`The measurements are obtained by forcing breakpoints in the code instructions that initiate and finish the process. In debugging mode , the execution is halted each time the breakpoint is reached and the register that contains the time is logged.`

Expected results: Direct measurement of the PCS
Assumptions, Constraints and Comments:

1.2.2 Effective slot time of partition execution

The effective slot time used by a partition is the time when the partition executes its own code. The effective slot time is measured by analysing the activity performed by a partition under different slot durations. The effective slot time is affected by the PCS. So, it provides an inderect measurement of the PCS.

The effective time can be modelled taking into account the PCS that is part of the partition execution, $EffectTime = SlotTime – PCS$

In general if the partition is executed in a MAF N times with slots of different size, the total effective time in the MAF can be computed as:

$$EffectTimeMAF_{Pi}) = \sum (SlotTime_{Pi}) – N * PCS$$

Table 1.2: T-XM-PERFO-050-0XY

Test: T-XM-PERFO-050-0XY
TEST PURPOSE: To measure experimentally the effective time of the partition.
Test class: Performance
Environment scenario: Partitions: 4 partitions 3 Counter partitions 1 Reader partition (system)

TEST DESCRIPTION:
`The scenario is composed by 3 counter partitions (P1,P2 and P3) and a reader partition (P0). The counter partitions perform a loop incrementing a counter value that is used to see the effects of the experiment.`
`The duration of the slot is set to 1000, 500, 100, 50, 10, 5 and 1 milliseconds in different execution of the scenario. The number of slots is incremented depending on the duration slot (1, 10, 100 and 1000) in order to complete 1 second of execution. The reader partition reads the counter values after the execution of 1 second.`

Expected results: Indirect measurement of the PCS
Assumptions, Constraints and Comments:

- PERFO-050-040: Perform the estimation of the PCS with different Slot sizes using shared memory between counters and reader partition. Counters are allocated in shared memory. All partitions share the memory area. All slot duration are defined in different scheduling plans.

1.2.3 Service Cost

Service cost is evaluated by executing each service and measuring the time spent.

Table 1.3: T-XM-PERFO-060-0XY

Test: T-XM-PERFO-060-0XY
TEST PURPOSE: Invoke the services and measure the time spent to complete the service
Test class: Performance
Environment scenario: Partitions: 2 partitions Both partitions with the same code except those services related to send/receive messages. P0 is the sender partition and P1 is the receiver.
TEST DESCRIPTION: `The scenario is composed by 2 invoking the services and measuring individually the cost.`
Expected results: Cost of the services
Assumptions, Constraints and Comments:

This basic test is derived in several tests:

PERFO-060-010 Each service is invoked one by one.

PERFO-060-020 Each service is invoked after a partition context switch.

When XtratuM is configured to flush the cache after context switch, the test PERFO-060-020 provides the cost of the services with a cache flushed.

1.2.4 Interrupt latency Cost

Interrupt latency is the elapsed time from the interrupt occurs and the partition starts the execution of the interrupt handler.

Table 1.4: T-XM-PERFO-070-02X

Test: T-XM-PERFO-070-010
TEST PURPOSE: To measure the elapsed time from an interrupt occurrence and the time that the interrupt handler is ready to execute it.
Test class: Performance
Environment scenario: Partitions: 1 partition

TEST DESCRIPTION:
A partition sets a periodic timer at a specified time with different periods generating several interrupts in 1 second. When the IRQ arrives, the time is read.
Expected results: Direct estimation of the IRQ Latency
Assumptions, Constraints and Comments:

This basic test is derived in several tests:

PERFO-070-020 10 Irq in a second

PERFO-070-021 100 Irq in 1 second

1.3 Environment

This section describes the environemnt used to execute the tests. It is composed by:

- GR712 Development Board
- GRMON2 Version 2.0.75
- XtratuM 1.0.8
- BCC 1.0.46 - GCC 4.4.2

The environemnt also includes a host computer where the GRMON is executed and where the board is connected. This computer is based in a Ubuntu 14.04 32-bits.

Prior loading the test in the GR712 board, the platform is reset, the memory is zeroed, and the memory controller is configured using the next GRMON2 commands:

reset The board is reset.

wash The memory is zeroed.

mcfg1, mcfg2, mcfg3 The memory controller is configured.

The values set in the memory controller registers are presented in 1.3.2

1.3.1 XtratuM configuration

The XtratuM version used is 1.0.8 with the following configuration:

```
#
# Automatically generated make config: don't edit
# XM version: 1.0.8
# Tue Jul 11 16:11:48 2017
#
CONFIG_SPARCv8=y
CONFIG_HWIRQ_PRIO_LBS=y
CONFIG_ARCH_MMU_BYPASS=y
CONFIG_CPU_NO_IRQS=16
CONFIG_TARGET_BIG_ENDIAN=y

#
# Processor
#
# CONFIG_LEON2 is not set
# CONFIG_LEON3 is not set
CONFIG_LEON3FT=y
# CONFIG_LEON4 is not set
# CONFIG_TSIM is not set
# CONFIG_GR_CPCI_XC4V is not set
# CONFIG_GR_PCI_XC2V is not set
CONFIG_GR_712_RC=y
# CONFIG_CPU_ITAR_FREE is not set
# CONFIG_SPW_RTC is not set
# CONFIG_SIMLEON is not set
# CONFIG_GR_CPCI_XC4VLX200 is not set
CONFIG_NONE=y
# CONFIG_AMP_SUPPORT is not set
# CONFIG_SMP_SUPPORT is not set
# CONFIG_MULTICORE_SUPPORT is not set
CONFIG_NO_HWIRQS=16
# CONFIG_MPU is not set
CONFIG_MMU=y
CONFIG_UART_TIMEOUT=500
CONFIG_LEON_EDAC_SUPPORT=y
CONFIG_ENABLE_CACHE=y
CONFIG_CACHE_SNOOP=y
CONFIG_CACHE_IBURST_FETCH=y
CONFIG_FLUSH_CACHE_AFTER_CS=y
CONFIG_ENABLE_POWERDOWN=y

#
# Physical memory layout
#
CONFIG_XM_LOAD_ADDR=0x61000000
CONFIG_XM_OFFSET=0xFC000000
# CONFIG_EXPERIMENTAL is not set
CONFIG_ASSERT=y
CONFIG_DEBUG=y
CONFIG_VERBOSE_TRAP=y
# CONFIG_NO_GCC_OPT is not set
CONFIG_ID_STRING_LENGTH=16
```

```
CONFIG_MAX_NO_CUSTOMFILES=3

#
# Hypervisor
#
CONFIG_KSTACK_KB=8
CONFIG_NO_VCPUS=1
# CONFIG_IPVI_SUPPORT is not set
CONFIG_MAX_NO_MAREAS=8
CONFIG_PLAN_EXTSYNC=y
# CONFIG_AUDIT_EVENTS is not set
# CONFIG_CORE_COMPRESSION is not set
# CONFIG_JMP_USR_FUNC_COLD_RESET_SYSTEM is not set
CONFIG_ARCH="sparcv8"
CONFIG_KERNELVERSION="1.0.8"
CONFIG_XM_VERSION=1
CONFIG_XM_SUBVERSION=0
CONFIG_XM_REVISION=8

#
# MMU
#

#
# Drivers
#
CONFIG_DEV_UART=y
CONFIG_DEV_UART_1=y
# CONFIG_DEV_UART_2 is not set
# CONFIG_EARLY_OUTPUT is not set
# CONFIG_DEV_UART_FLOWCONTROL is not set
CONFIG_DEV_NO_UARTS=2
# CONFIG_UART_THROUGH_DSU is not set
CONFIG_DEV_MEMBLOCK=y
# CONFIG_DEV_MIL_STD is not set
# CONFIG_MIL_STD_1553_INTERNAL_CLOCK is not set
# CONFIG_MIL_STD_1553_ODD_PARITY is not set
# CONFIG_MIL_STD_1553_CLKFREQ_12 is not set
# CONFIG_MIL_STD_1553_CLKFREQ_16 is not set
# CONFIG_MIL_STD_1553_CLKFREQ_24 is not set
# CONFIG_LICE is not set
# CONFIG_DEV_LICE_INTERFACE is not set
# CONFIG_DEV_LICE_SCHEDULING is not set
# CONFIG_LICE_ADDRESS is not set
# CONFIG_GR712_WATCHDOG is not set
#
# Objects
#

# CONFIG_OBJ_HM_VERBOSE is not set
# CONFIG_OBJ_STATUS_ACC is not set
CONFIG_OBJ_TRACE_LOG_NO_ELEM=1000
CONFIG_OBJ_HM_LOG_NO_ELEM=20
CONFIG_OBJ_MAX_NO_COMMPORTS=256
```

1.3.2 Hardware configuration

The platform used is a Gaisler GR712 Evaluation Board with serial number 020. The configuration is summarized below:

- The frequency used is 80MHz
- XtratuM and partition codes have been loaded and executed in SDRAM
- Cache has been configured with the value: cctrl = 008b000f. Which corresponds to:
  - Instruction Cache=Enabled,
  - Data Cache=Enabled
  - Data cache snoop=Enabled
  - Instruction burst fetch=Enabled
- MCFG Registers

MCFG1	0x10F8800F
MCFG2	0x9A20546F
MCFG3	0x08260000

The cache is configured by XtratuM depending in its configuration (see 1.3.1).

The memory map has been divided as described below:

- 16MB at address 0x60000000 for partitions.
- 16MB at address 0x61000000 for XtratuM.

Although the memory areas described above, the XtratuM and partitions memory areas depends on the XtratuM configuration file of each test. For instance, the XtratuM memory area is configured to 512KB in almost all tests.

Chapter 2

Performance Test Results

Performance tests have been executed in GR712RC platform.

All partitions and shared memories used in the tests have been allocated in SDRAM memory. The memory layout in the configuration file used is:

```
<MemoryLayout>
  <Region type="sdram" start="0x60000000" size="32MB"/>
</MemoryLayout>
```

2.1 Partition Context Switch (PCS)

The partition context switch (PCS) refers to the time needed by the hypervisor to switch form one partition to the next one as specified in the plan. Test PERFO-050-000 performs the the extration of the time values when the symbols *begin_trap* and *end_trap* are reached. The simbol *begin_trap* corresponds with the first instruction of the trap handler for trap 0x19 (IRQ 9). The *end_trap* corresponds with the instruction that returns to the partition, this point is marked in the source with the symbol *.Tend_trap*.

2.1.1 GR712 board results

Next table summarises the first measures obtained by this test:

PCS measurements Board GR712

begin_trap	end_trap	difference
4291834483	4291834364	119
4291824483	4291824380	103
4290824483	4290824363	120
4289824483	4289824363	120
4288824483	4288824364	119
4288814483	4288814380	103
4287814483	4287814363	120
4286814483	4286814363	120
4285814483	4285814364	119
4285804483	4285804380	103
4284804483	4284804364	119
4283804483	4283804363	120
4282804483	4282804364	119
4282794483	4282794380	103
4281794483	4281794363	120
4280794483	4280794363	120
4279794483	4279794364	119
4279784483	4279784382	101

This interval corresponds to the partition context switch from the instant it is detected by the timer interrupt to the return to the next partition. This maximum observed value is $120\mu seconds$ and the minimum is $103\mu seconds$. The difference between the observed values depends in the number of window registers saved in hypervisor stack when a supervisor service is requested.

2.1.2 PCS observation

The worst case observed time for the Partition Context Switch are:

Worst case time observed for PCS.

PCS (usec) BOARD GR712
119

2.2 Effective slot time of partition execution

The effective slot time used by a partition is the time when the partition executes its own code. The effective slot time is measured by analysing the activity performed by a partition under different slot durations. The effective slot time is affected by the PCS. So, it provides an inderect measurement of the PCS.

The effective time can be modelled taking into account the PCS that is part of the partition execution, $EffectTime = SlotTime–PCS$

In general if the partition is executed in a MAF N times with slots of different size, the total effective time in the MAF can be computed as:

$$EffectTimeMAF_{Pi}) = \sum(SlotTime_{Pi}) – N * PCS$$

2.2.1 TEST-050-040

TEST-050-040 results

Slot(ms)	1000	500	100	50	10	5	1
Avg:	11371313	11369881	11360229	11348223	11251850	11131333	10168225
Max:	11371316	11369883	11360246	11348232	11251868	11131390	10168316
Min:	11371313	11369843	11360211	11348153	11251833	11131276	10168135
Diff:	0	1432	11084	23090	119463	239980	1203088
Loss:(%)	0.000	0.013	0.097	0.203	1.051	2.110	10.580
PCS1:	0.00	1432.00	1231.56	1215.26	1206.70	1205.93	1204.29
PCS2:	0.00	125.93	108.30	106.87	106.12	106.05	105.91

Where:

PCS1 Performance loss measured in number of operations not performed.

PCS2 Performance loss measured in useconds, approximately the PCS.

These results show that the estimated performance loss of slots can be modelled by $126\mu secondss$. Which corresponds to the worst case observed PCS.

The same test is exectued with ASSERTS deactivated (CONFIG_ASSERTS). The results obtained are shown in the next table.

Slot(ms)	1000	500	100	50	10	5	1
Avg:	11371423	11370170	11361777	11351264	11267252	11162278	10322293
Max:	11371425	11370171	11361780	11351284	11267268	11162292	10322353
Min:	11371421	11370169	11361762	11351261	11267236	11162226	10322232
Diff:	0	1253	9646	20159	104171	209145	1049130
Loss:(%)	0.000	0.011	0.085	0.177	0.916	1.839	9.226
PCS1:	0.00	1253.00	1071.78	1061.00	1052.23	1050.98	1050.18
PCS2:	0.00	110.19	94.25	93.30	92.53	92.42	92.35

2.3 Service Cost

Service cost is evaluated by executing each service and measuring the time spent.

Next table shows the results of PERFO-060 tests

service	PERFO-060-010	PERFO-060-020
XM_clear_irqmask	8	19
XM_create_queuing_port	25	43
XM_create_sampling_port	33	45
XM_get_partition_status	22	39
XM_get_plan_status	11	27
XM_get_system_status	22	26
XM_get_time(XM_EXEC_CLOCK)	9	18
XM_get_time(XM_HW_CLOCK)	8	17
XM_hm_read	14	25
XM_hm_status	12	24
XM_set_timer(XM_EXEC_CLOCK)	16	34
XM_set_timer(XM_HW_CLOCK)	23	25
XM_sparc_clear_pil	6	16
XM_sparc_inport	12	23
XM_sparc_outport	8	21
XM_sparc_set_pil	3	15
XM_trace_event	23	37
XM_trace_read	21	38
XM_trace_status	9	29

service	PERFO-060-010	PERFO-060-020
XM_memory_copy(64)	28	44
XM_memory_copy(128)	25	51
XM_memory_copy(256)	39	64
XM_memory_copy(512)	65	90
XM_memory_copy(1024)	118	143
XM_memory_copy(2048)	223	249
XM_memory_copy(4096)	436	459

service	PERFO-060-010	PERFO-060-020
XM_send_queuing_message(32)	46	46
XM_send_queuing_message(64)	22	48
XM_send_queuing_message(128)	18	50
XM_send_queuing_message(256)	22	54
XM_send_queuing_message(512)	28	61
XM_send_queuing_message(1024)	76	77
XM_send_queuing_message(2048)	62	107
XM_send_queuing_message(4096)	107	169
XM_write_sampling_message(32)	20	44
XM_write_sampling_message(64)	45	45
XM_write_sampling_message(128)	47	47
XM_write_sampling_message(256)	51	51
XM_write_sampling_message(512)	59	58
XM_write_sampling_message(1024)	74	74
XM_write_sampling_message(2048)	105	104
XM_write_sampling_message(4096)	166	166
XM_read_sampling_message(32)	27	41
XM_read_sampling_message(64)	42	43
XM_read_sampling_message(128)	45	45
XM_read_sampling_message(256)	48	48
XM_read_sampling_message(512)	55	56
XM_read_sampling_message(1024)	71	71
XM_read_sampling_message(2048)	102	102
XM_read_sampling_message(4096)	162	164
XM_receive_queuing_message(32)	41	42
XM_receive_queuing_message(64)	20	43
XM_receive_queuing_message(128)	18	45
XM_receive_queuing_message(256)	21	49
XM_receive_queuing_message(512)	34	56
XM_receive_queuing_message(1024)	72	71
XM_receive_queuing_message(2048)	75	101
XM_receive_queuing_message(4096)	142	168

2.4 Interrupt latency Cost

Latency to interrupts is measured by the performance tests PERF0-070-02X.

Next table shows the results of PERFO-070-020 with 10 IRQ occurrences in 1 second.

The obtained results are:

```
[0] ****************************************************
[0] TEST PERFO-070-020 Interrupts: 10 per second
[0] XAL Partition Measuring: latencies to IRQ: HW_CLOCK
[0] ****************************************************
```

```
Parameters 100000  * 9  + 100000
   Avg  Max      Min      NIrqs    Clock
1: 34   34       34  (10)
2: 34   34       34  (10)
3: 34   34       34  (10)
4: 33   34       32  (10)
5: 34   34       34  (10)
6: 34   34       34  (10)
7: 34   34       34  (10)
8: 33   34       32  (10)
9: 34   34       34  (10)
```

Next table shows the results of PERFO-070-020 with 100 IRQ occurrences in 1 second.

The obtained results are:

```
[0] ***************************************************
[0] TEST PERFO−070−020 Interrupts: 100 per second
[0] XAL Partition Measuring: latencies to IRQ: HW_CLOCK
[0] ***************************************************
Parameters 10000  * 99  + 10000
   Avg  Max      Min      NIrqs    Clock
1: 34   34       34  (100)
2: 34   34       34  (100)
3: 34   34       34  (100)
4: 34   34       34  (100)
5: 34   34       34  (100)
6: 34   34       34  (100)
7: 34   34       34  (100)
8: 34   34       34  (100)
9: 34   34       34  (100)
10: 34  34       34  (100)
```

The observed results show that the impact of the detection and management of the clock interrupt at partition level has a latency of $34\mu seconds$.

Chapter 3

XM sizes

3.1 XtratuM footprint

To achieve the size of XtratuM we use the *size* command.

```
   text    data     bss     dec     hex filename
 100188     196   11752  112136   1b608 xm_core
```

3.2 XtratuM configuration (XMCF) footprint

The data is statically reserved by the xmcparser tool. The footprint of the data depends in the number of elements (partitions, queuing ports, sampling ports) configured.

3.2.1 Number of Partitions

In order to analyse the impact of the number of partitions, several systems with different number of partitions have been defined. The results are:

```
Part    text    data    bss     dec                     Diferences
1       752     204     40960   41916   |
2       1024    288     69632   70944   |       272     84      28672   29028
3       1296    372     94208   95876   |       272     84      24576   24932
4       1576    456     122880  124912  |       280     84      28672   29036
5       1848    540     151552  153940  |       272     84      28672   29028
6       2120    624     176128  178872  |       272     84      24576   24932
7       2392    708     204800  207900  |       272     84      28672   29028
8       2672    792     229376  232840  |       280     84      24576   24940
9       2944    876     258048  261868  |       272     84      28672   29028
10      3216    960     286720  290896  |       272     84      28672   29028
11      3496    1044    311296  315836  |       280     84      24576   24940
12      3768    1128    339968  344864  |       272     84      28672   29028
13      4040    1212    364544  369796  |       272     84      24576   24932
14      4320    1296    393216  398832  |       280     84      28672   29036
15      4592    1380    421888  427860  |       272     84      28672   29028
16      4864    1464    446464  452792  |       272     84      24576   24932
```

17	5144	1548	475136	481828	280	84	28672	29036
18	5416	1632	499712	506760	272	84	24576	24932
19	5696	1716	528384	535796	280	84	28672	29036
20	5968	1800	557056	564824	272	84	28672	29028

The size of the text can be modeled as 752B + (272B x NoPartitions) + padding. The size of the data can be modeled as 204B + (84B x NoPartitions). The size of the bss can be modeled as 40960B + (24576B x NoPartitions) + padding.

3.2.2 Sampling Channels footprint

The configuration consists in 2 partitions with several sampling ports of 1024B.

SP+1	text	data	bss	dec	Diferences			
1	1024	288	69632	70944				
2	1104	348	69632	71084	80	60	0	140
3	1192	384	69632	71208	88	36	0	124
4	1272	420	69632	71324	80	36	0	116
5	1352	456	69632	71440	80	36	0	116
6	1432	492	69632	71556	80	36	0	116
7	1520	528	69632	71680	88	36	0	124
8	1600	564	69632	71796	80	36	0	116
9	1680	600	69632	71912	80	36	0	116
10	1760	636	69632	72028	80	36	0	116
11	1848	672	69632	72152	88	36	0	124
12	1928	708	69632	72268	80	36	0	116
13	2008	744	69632	72384	80	36	0	116
14	2088	780	69632	72500	80	36	0	116
15	2176	816	69632	72624	88	36	0	124
16	2256	852	69632	72740	80	36	0	116
17	2344	888	69632	72864	88	36	0	124
18	2416	924	69632	72972	72	36	0	108
19	2504	960	69632	73096	88	36	0	124
20	2584	996	69632	73212	80	36	0	116

The memory needs for sampling ports is linear with the number of sampling ports and corresponds to 80B + padding for text and 36B for data.

The next shows the footprint when the size of messages is changed. The size is configured from 32 bytes to 16384 bytes per message. The number of partition is 2 each with 5 sampling ports.

Msize	text	data	bss	dec	Diferences			
32	1432	492	69632	71556				
64	1432	492	69632	71556	0	0	0	0
128	1432	492	69632	71556	0	0	0	0
256	1432	492	69632	71556	0	0	0	0
512	1432	492	73728	75652	0	0	4096	4096
1024	1432	492	73728	75652	0	0	0	0
2048	1432	492	77824	79748	0	0	4096	4096
4096	1432	492	90112	92036	0	0	12288	12288
8192	1432	492	110592	112516	0	0	20480	20480
16384	1432	492	151552	153476	0	0	40960	40960

The memory is reserved in the BSS section. The size of BSS increase in pages of 4KB.

3.2.3 Queuing Channels footprint

The configuration consists in 2 partitions with a configuration from 0 to 19 queing ports, each with a capacity of 1 message of 32 bytes.

QP+1	text	data	bss	dec	Diferences			
1	1024	288	69632	70944				
2	1104	336	69632	71072	80	48	0	128
3	1192	360	69632	71184	88	24	0	112
4	1272	384	69632	71288	80	24	0	104
5	1352	408	69632	71392	80	24	0	104
6	1432	432	69632	71496	80	24	0	104
7	1520	456	69632	71608	88	24	0	112
8	1600	480	69632	71712	80	24	0	104
9	1680	504	69632	71816	80	24	0	104
10	1760	528	69632	71920	80	24	0	104
11	1848	552	69632	72032	88	24	0	112
12	1928	576	69632	72136	80	24	0	104
13	2008	600	69632	72240	80	24	0	104
14	2088	624	69632	72344	80	24	0	104
15	2176	648	69632	72456	88	24	0	112
16	2256	672	69632	72560	80	24	0	104
17	2336	696	69632	72664	80	24	0	104
18	2416	720	69632	72768	80	24	0	104
19	2504	744	69632	72880	88	24	0	112
20	2584	768	73728	77080	80	24	4096	4200

The memory needs for queuing ports is linear with the number of sampling ports and corresponds to 80B + padding for text and 24B for data.

Port size. 1 Port wiht 1 message capacity.

QPort	text	data	bss	dec	Diferences			
32	1104	336	69632	71072				
64	1104	336	69632	71072	0	0	0	0
128	1104	336	69632	71072	0	0	0	0
256	1104	336	69632	71072	0	0	0	0
512	1104	336	69632	71072	0	0	0	0
1024	1104	336	69632	71072	0	0	0	0
2048	1104	336	69632	71072	0	0	0	0
4096	1104	336	73728	75168	0	0	4096	4096
8192	1104	336	77824	79264	0	0	4096	4096
16384	1104	336	86016	87456	0	0	8192	8192

Port size. 5 Port wiht 1 message capacity.

QPort	text	data	bss	dec	Diferences			
32	1432	432	69632	71496				
64	1432	432	69632	71496	0	0	0	0
128	1432	432	69632	71496	0	0	0	0
256	1432	432	69632	71496	0	0	0	0
512	1432	432	73728	75592	0	0	4096	4096
1024	1432	432	73728	75592	0	0	0	0
2048	1432	432	77824	79688	0	0	4096	4096
4096	1432	432	90112	91976	0	0	12288	12288
8192	1432	432	110592	112456	0	0	20480	20480
16384	1432	432	151552	153416	0	0	40960	40960

The memory needs are increased in blocks of pages.

This page is intentionally left blank

Chapter 4

B2BST Analysis

This section includes the output provides by the B2BST script provide by Gailser to identify potential error locations. The analysis has been executed with the next command:

```
$objdump -d xm_core | leon3ft-b2bst-scan.tcl
```

The output is provided in the next listing. The analysis results provides that no potential errors locations exists. In addition, the potential error locations that the script clasify with "check manually" has been checked manually, and no potential erros has been found.

```
objdump -d xm_core | /home/lithosdev/Downloads/leon3ft-b2bst-scan.tcl

LEON3FT Stale Cache Entry After Store with Data Tag Parity Error errata
    scanning utility, rev 4 (20170215)
Searching objdump -d output on standard input for:
 - Sequence A
 - Sequence B

INFO: _start: Unable to trace jmpl at 0x610011f4 - check manually
NOTE: WindowOverflowTrap: Execution leaves function without return
NOTE: WindowOverflowTrap: Execution leaves function without return
NOTE: EWindowOverflowTrap: More restore than save instructions
NOTE: EWindowOverflowTrap: More restore than save instructions
NOTE: EWindowOverflowTrap: Execution leaves function without return
NOTE: EWindowOverflowTrap: Execution leaves function without return
NOTE: EWindowUnderflowTrap: Execution leaves function without return
NOTE: EWindowUnderflowTrap: Execution leaves function without return
INFO: .Tbegin_kpreempt: Unable to trace jmpl at 0xfc00178c - check
    manually
INFO: FromIRet: Unable to trace jmpl at 0xfc001a34 - check manually
NOTE: FromIRet: More restore than save instructions
INFO: FromIRet: Unable to trace jmpl at 0xfc001a34 - check manually
NOTE: FromIRet: More restore than save instructions
INFO: DoHypercall: Unable to trace jmpl at 0xfc001cb4 - check manually
NOTE: AsmHypercallHandler: Execution leaves function without return
NOTE: AsmHypercallHandler: Execution leaves function without return
```

```
INFO: AsmHypercallHandler: Unable to trace jmpl at 0xfc001d24 - check
    manually
NOTE: EmulateTrapSv: More restore than save instructions
NOTE: EmulateTrapSv: Execution leaves function without return
NOTE: EIRetCheckRetAddr: Execution leaves function without return
NOTE: SparcFlushRegWinSys: More than 10 deep saves (possibly regfile clear
     loop)
NOTE: SparcFlushRegWinSys: More than 10 deep saves (possibly regfile clear
     loop)
NOTE: SparcFlushRegWinSys: More than 10 deep saves (possibly regfile clear
     loop)
INFO: SparcFlushRegWinSys: Unable to trace jmpl at 0xfc0024c8 - check
    manually
INFO: SparcFlushRegWinSys: Unable to trace jmpl at 0xfc0024c8 - check
    manually
INFO: SparcFlushRegWinSys: Unable to trace jmpl at 0xfc0024c8 - check
    manually
INFO: SparcFlushRegWinSys: Unable to trace jmpl at 0xfc0024c8 - check
    manually
INFO: SparcFlushRegWinSys: Unable to trace jmpl at 0xfc0024c8 - check
    manually
INFO: SparcFlushRegWinSys: Unable to trace jmpl at 0xfc0024c8 - check
    manually
INFO: SparcFlushRegWinSys: Unable to trace jmpl at 0xfc0024c8 - check
    manually
INFO: SparcFlushRegWinSys: Unable to trace jmpl at 0xfc0024c8 - check
    manually
NOTE: SparcFlushRegWinSys: More than 10 deep saves (possibly regfile clear
     loop)
NOTE: SparcFlushRegWinSys: More than 10 deep saves (possibly regfile clear
     loop)
NOTE: SparcFlushRegWinSys: More than 10 deep saves (possibly regfile clear
     loop)
NOTE: SparcFlushRegWinSys: More than 10 deep saves (possibly regfile clear
     loop)
INFO: StartUpGuest: Unable to trace jmpl at 0xfc00be64 - check manually
NOTE: Schedule: Execution leaves function without return
NOTE: Schedule: Execution leaves function without return
NOTE: Schedule: Execution leaves function without return
NOTE: Schedule: Execution leaves function without return
NOTE: Schedule: Execution leaves function without return
NOTE: Schedule: Execution leaves function without return
NOTE: Schedule: Execution leaves function without return
NOTE: Schedule: Execution leaves function without return
NOTE: Schedule: Execution leaves function without return
NOTE: Schedule: Execution leaves function without return
NOTE: .Tend_cs: More restore than save instructions
INFO: .Tend_cs: Unable to trace jmpl at 0xfc00dc18 - check manually
INFO: .Tend_cs: Unable to trace jmpl at 0xfc00dbb8 - check manually
INFO: vprintf: Unable to trace jmpl at 0xfc0103fc - check manually
INFO: vprintf: Unable to trace jmpl at 0xfc0103fc - check manually
INFO: HmRaiseEvent: Unable to trace jmpl at 0xfc0117a0 - check manually
INFO: HmRaiseEvent: Unable to trace jmpl at 0xfc0116f8 - check manually

Objdump lines processed: 24962, lines skipped: 393
Functions scanned: 177, reachable instruction count: 18183
```

```
Potential error locations found: 0
```

This page is intentionally left blank