Eccelerators Library IP specification

Table of Contents

List of Figures

List of Tables

1. InterruptCollectorIfc

1.1. Interrupt Collector

1.1.1. Contents

This README is a partial view of the final document InterruptCollector.pdf

1.1.2. Introduction

Interrupts connect software (SW) and hardware (HW) very closely. They require a carefully planned concept that starts from the HW source bit, which causes an interrupt event, through different HW controllers/collectors located both outside and inside a CPU system, and extends to the SW operating system and its drivers, up to the application software. The concept is dependent on the specific nature of all components in this chain and dramatically affects the real-time capabilities and overall performance of a system.

A designer of a peripheral logic block for a CPU system is often required to issue an interrupt to the system if their logic needs attention from the CPU. Typically, there is more than one event source in the peripheral logic block that can trigger an interrupt. The *Inter-*

rupt-Collector presented here is a proven solution for managing these tasks. It does not impose excessive overhead on a peripheral logic block for a tiny microcontroller and is suitable for the most complex systems on a chip (SOC) with multiple cores running in symmetric or asymmetric multiprocessing mode. Thus, it and its driver software can be reused across a wide range of designs.

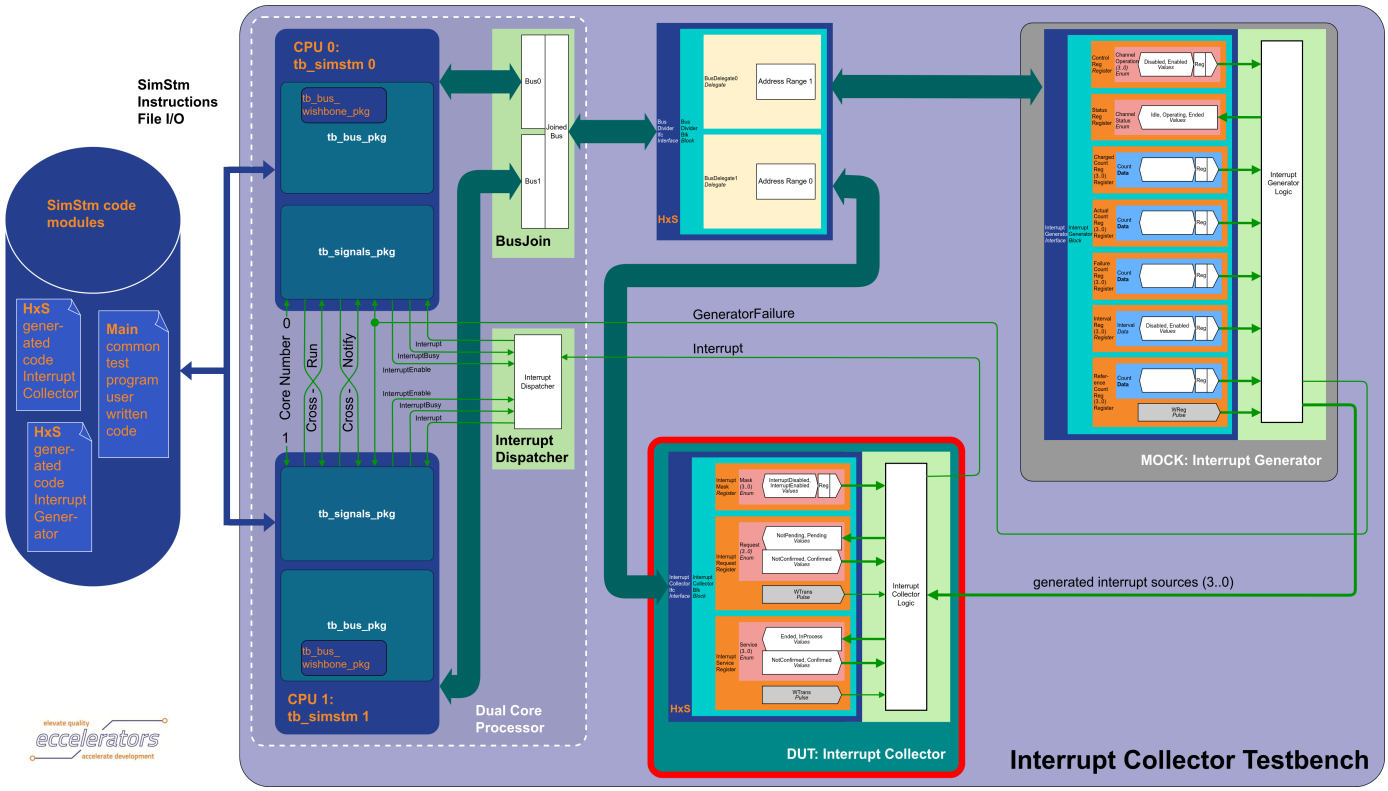
The description of the *Interrupt-Collector* in the subsequent chapters is structured in a top-down manner, starting from a testbench that includes it, and progressing to its internal functions.

The testbench simulation can be run immediately with either the GHDL or ModelSim simulator using the included ANT scripts. For more information, see the QuickStartSimlation Quick Start Simulation section.

The VHDL source code, along with other source code in this repository, can be used liberally under the MIT license for any design.

1.1.3. Testbench

Figure 1.1. Interrupt Collector Testbench



The “Interrupt Collector Testbench” illustrates the Device Under Test (DUT)—our Interrupt Collector—within a red-bordered block. On the left, there are two processors, simulating a multiprocessor system. Each processor is an instance of the Eccelerators SimStm Testbench. These processors are augmented by a BusJoin unit, which forms the common bus connection for the Dual Core Processor system, and an InterruptDispatcher unit that dispatches arriving interrupts to one of the cores. Both cores execute the same SimStm code, as shown on the left. The main routine is differentiated based on the core number; Processor 0 is responsible for managing initial processes and cross-run signals for Processor 1. It is only Processor 0 that performs the initialization of the DUT and MOCK.

A short glance at the SimStm program “Main” routine snippet, the corresponding source file is testMain.stm .

Figure 1.2. SimStm Main Routine Snippet

```
102
103 testMain:
104 proc
105     bus pointer copy ReadModifyWriteBus32 wishbone
106     bus pointer copy ReadModifyWriteBus16 wishbone
107     bus pointer copy ReadModifyWriteBus8 wishbone
108     bus pointer copy InterruptCollectorIfcBus wishbone
109     bus pointer copy InterruptGeneratorIfcBus wishbone
110
111     signal read InCoreNumber CoreNumber
112     signal write OutCrossNotifyInterruptToOtherCore 0
113     signal write OutCrossNotifyInterruptIsInServiceToOtherCore 0
114
115     call $InterruptCollectorIfcInit
116
117     equ InterruptGeneratorIfcAddress 0x100
118     call $InterruptGeneratorIfcInit
119
120     verbosity $INFO_4
121     trace 0
122     wait 1000
123
124     if $CoreNumber = 0
125         log message $INFO "Core0: Main test started"
126
127         -- test cross notify interrupt
128         signal write OutCrossNotifyInterruptToOtherCore 1
129         call $WaitForCrossNotifyInterruptIsInServiceFromOtherCore
130         signal write OutCrossNotifyInterruptToOtherCore 0
131
132         -- test generator interrupts
133         bus write InterruptGeneratorIfcBus 32 $ChargedCountReg0Address $ChargedNumberOfInterrupts0
134         bus write InterruptGeneratorIfcBus 32 $ChargedCountReg1Address $ChargedNumberOfInterrupts1
135         bus write InterruptGeneratorIfcBus 32 $ChargedCountReg2Address $ChargedNumberOfInterrupts2
136         bus write InterruptGeneratorIfcBus 32 $ChargedCountReg3Address $ChargedNumberOfInterrupts3
137
138         bus write InterruptGeneratorIfcBus 32 $IntervalReg0Address $IntervalNsOfInterrupts0
139         bus write InterruptGeneratorIfcBus 32 $IntervalReg1Address $IntervalNsOfInterrupts1
140         bus write InterruptGeneratorIfcBus 32 $IntervalReg2Address $IntervalNsOfInterrupts2
141         bus write InterruptGeneratorIfcBus 32 $IntervalReg3Address $IntervalNsOfInterrupts3
142
143         signal write OutCrossSignalToOtherCore 1
144
145         wait 100
146         bus write InterruptCollectorIfcBus 32 $InterruptMaskRegAddress 0b1111 -- enable all interrupts
147         bus write InterruptGeneratorIfcBus 32 $ControlRegAddress 0b1111 -- start generation for all interrupts
148
149         call $WaitForGeneratorStatusAllEnded
150         call $WaitForNotRunCrossFromOtherCore
151
152         wait 1000
153
154         log message $INFO_4 " "
155         log message $INFO_4 "Core0 finally: Interrupts in total: {:d}, Source0: {:d}, Source1: {:d}, Source2: {:d}, Source3: {
156
157         bus read InterruptGeneratorIfcBus 32 $ActualCountReg0Address ActualCount0
158         bus read InterruptGeneratorIfcBus 32 $ActualCountReg1Address ActualCount1
159         bus read InterruptGeneratorIfcBus 32 $ActualCountReg2Address ActualCount2
160         bus read InterruptGeneratorIfcBus 32 $ActualCountReg3Address ActualCount3
161
162         equ ActualCount $ActualCount0
163         add ActualCount $ActualCount1
164         add ActualCount $ActualCount2
165         add ActualCount $ActualCount3
166
167         log message $INFO_4 " "
168         log message $INFO_4 "Total counts finally: ActualSum: {:d}, Actual0: {:d}, Actual1: {:d}, Actual2: {:d}, Actual3: {:d}
169
170         bus read InterruptGeneratorIfcBus 32 $FailureCountReg0Address FailureCount0
171         bus read InterruptGeneratorIfcBus 32 $FailureCountReg1Address FailureCount1
172         bus read InterruptGeneratorIfcBus 32 $FailureCountReg2Address FailureCount2
173         bus read InterruptGeneratorIfcBus 32 $FailureCountReg3Address FailureCount3
174
175         equ FailureCount $FailureCount0
176         add FailureCount $FailureCount1
177         add FailureCount $FailureCount2
```

The SimStm source code is very compact and easy readable. It starts with the declaration and definition of constants, variables signals and buses. Following is the ‘testMain’ procedure, then the Interrupt procedure and helper procedures.

A BusDivider unit, generated by HxS, features two delegates. It splits the bus into two separate buses, each designated for different address areas: one for the Device Under Test (DUT) and another for the MOCK.

The MOCK includes an interrupt generator with four channels. These channels are designed to generate a charged number of interrupts at programmable intervals repeatedly.

The interrupt generator is maintained in its own repository Eccelerators Interrupt Generator. The documentation of its HW/SW interface is InterruptGeneratorIfc.pdf.

For each interrupt, acknowledgment is required from the software interrupt service routine within the corresponding channel of the generator. This acknowledgment process involves reading the actual count from a register and then writing this count to a reference count register. Any missed or incorrect acknowledgments are logged for each issued interrupt in a failure count register. Should any channel experience a fault, the generator signals this failure through its “GeneratorFailure” output at the first occurrence of such a fault.

The four interrupts generated by the Interrupt Generator are processed by the Device Under Test (DUT), the Interrupt Collector. They are delivered to the Dual Core Processor system via its “OutUpInterrupt” output.

The running simulation of the testbench should demonstrate that all generated interrupts are serviced in parallel, with the services being evenly distributed between both cores.

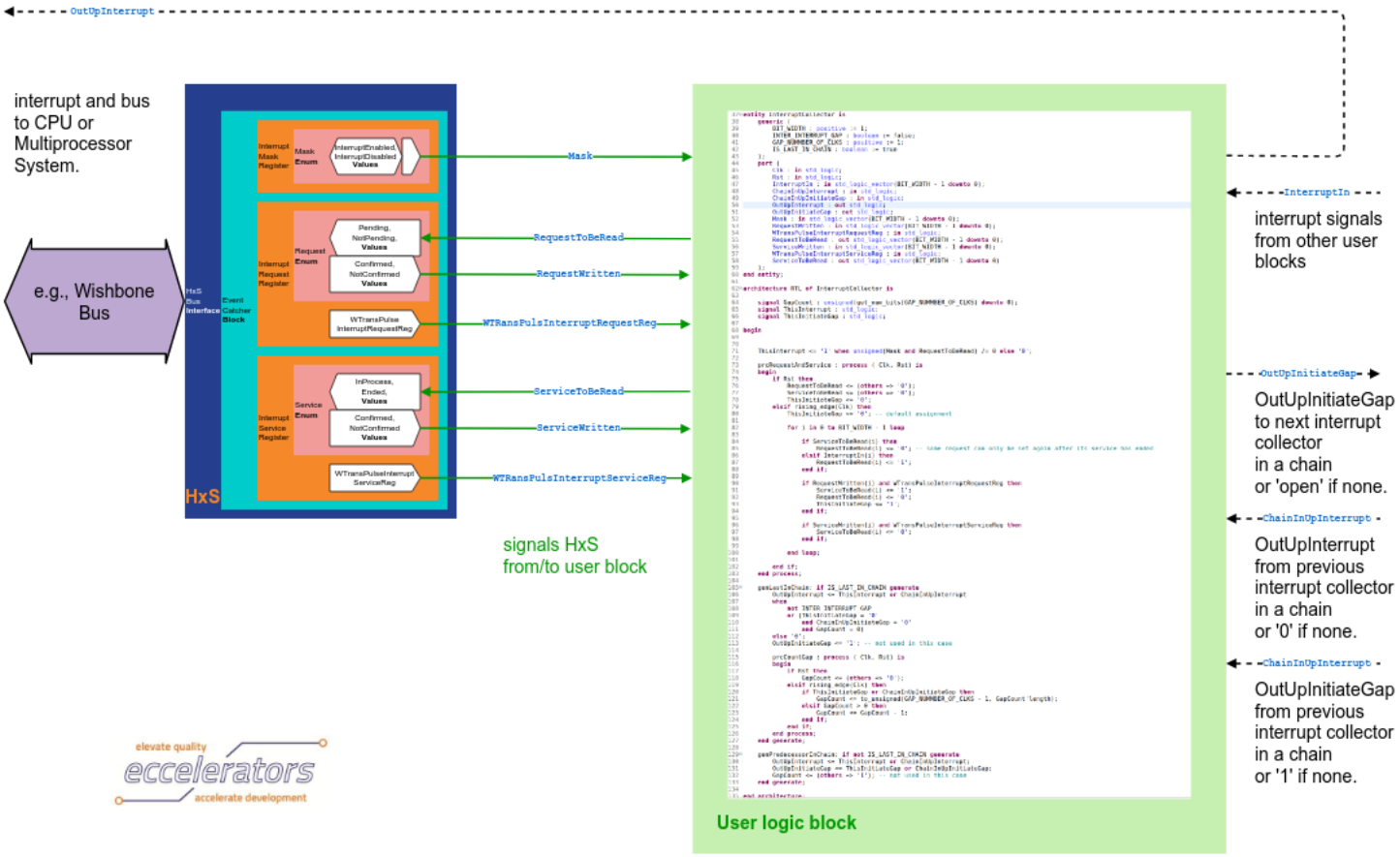
1.1.4. Function (DUT)

The interrupt collector consists of two parts. The first part, the hardware-software interface, is generated by the Eccelerators Tool HxS. It uses the description of the Mask-, Service- and Request, Registers in the HxS language as input. The output is the complete implementation of these registers in VHDL connected to a Wishbone Bus, with the necessary inputs and outputs to a user logic block. The second part contains the logic of the interrupt collector function, coded by the user Eccelerators.

The design allows for the simultaneous processing of different interrupts within the same interrupt collector by multiple processors of a multiprocessor system, without the need for additional synchronization measures such as Spin-Locks.

The accompanying diagram illustrates the implementation, including the connection of the HxS block to a Wishbone bus leading to the CPU or the multiprocessor system. The User Block demonstrates the realization of the interrupt logic. Both blocks are scalable in terms of the number of serviced interrupt inputs.

Figure 1.3. Interrupt Collector Overview



Upon the arrival of an interrupt event at an input, an interrupt is triggered to the CPU system. This leads to the execution of an interrupt routine by software on exactly one CPU of the system. The routine reads the *Request-Register* in the interrupt collector, selects one

of the reported interrupt requests for processing, and acknowledges this by setting the corresponding bit in the Request-Register. This interrupt request is then reset in the Request-Register and appears in the *Service-Register*. After completing the tasks in the interrupt service routine, the CPU sets to the corresponding bit in the Service-Register, thereby signaling the end of the interrupt routine to the interrupt collector. Only from this point can the corresponding Request-Register bit be set again by the same interrupt input.

The actual triggering of an interrupt to the CPU system can be enabled or disabled in the *Mask-Register* for each input. After each acknowledged request, the interrupt line to the CPU system is briefly deactivated to allow for the processing of further interrupts by other CPUs of the system.

The sources for the Interrupt-Collector inputs must have a level-triggering nature. The source logic block issuing an interrupt event must keep its signal active until it is acknowledged by the interrupt service routine by some SW access to the source logic block.

Edge-triggering sources e.g., timers must be converted to level-triggering sources. A solution for this conversion is the the EventCatcher IP.

1.1.5. Quick Start Simulation

1.1.6. GHDL Linux

We use Ubuntu 22.04 for demonstration.

At least a java runtime >= 17 is required. If not already present e.g., install it by:

```
sudo apt install openjdk-17-jre
```

All builds are run by means of APACHE ANT and respective build.xml files. If not already present e.g., install it by:

```
sudo apt install ant
```

Git is required to clone the interrupt-collector repository. If not already present e.g., install it by:

```
sudo apt install git
```

Next, we clone the actual interrupt collector repository repository:

```
git clone --recursive https://github.com/eccelerators/interrupt-collector.git
```

The `--recursive` parameter is mandatory because many resources e.g. child build.xml files are residing there.

In an unconventional approach, all necessary artifacts generated by previously executed build steps in the Ecclerators IP project workflow are already present in the cloned working copy. This setup facilitates easy use and progress with this starter IP.

To run the simulation with GHDL its version 4.0.0 must be present.

If not jump to Install GHDL Ubuntu22.04 section, if not callable as ‘ghdl’ in path, usually wit Windows, jump to *AdaptAntBuild* Adapt Ant Build .

Then simulation can be run, assuming the cloned repository is located in ‘git/interrupt-collector’:

```
cd git/interrupt-collector
ant ghdl-wishbone-all
```

This should result in:

```
[exec] Core0: Main test started
[exec] Core1: Cross notify interrupt happened
[exec] Core1: test started
[exec] Core0: Cross notify interrupt happened
[exec] Core1 currently: Interrupts in total: 32, Source0: 12, Source1: 6, Source2: 6, Source3: 8
[exec] Core0 currently: Interrupts in total: 32, Source0: 13, Source1: 11, Source2: 6, Source3: 2
[exec] Core1 currently: Interrupts in total: 64, Source0: 22, Source1: 14, Source2: 12, Source3: 16
[exec] Core0 currently: Interrupts in total: 64, Source0: 28, Source1: 21, Source2: 12, Source3: 3
[exec] Core1 currently: Interrupts in total: 96, Source0: 34, Source1: 19, Source2: 20, Source3: 23
[exec] Core0 currently: Interrupts in total: 96, Source0: 43, Source1: 30, Source2: 18, Source3: 5
[exec] Core1 currently: Interrupts in total: 128, Source0: 46, Source1: 25, Source2: 25, Source3: 32
[exec] Core0 currently: Interrupts in total: 128, Source0: 56, Source1: 41, Source2: 24, Source3: 7
[exec] Core1 currently: Interrupts in total: 160, Source0: 56, Source1: 32, Source2: 32, Source3: 40
[exec] Core0 currently: Interrupts in total: 160, Source0: 69, Source1: 52, Source2: 30, Source3: 9
[exec] Core1 currently: Interrupts in total: 192, Source0: 68, Source1: 38, Source2: 38, Source3: 48
[exec] Core0 currently: Interrupts in total: 192, Source0: 84, Source1: 62, Source2: 36, Source3: 10
[exec] Core1 currently: Interrupts in total: 224, Source0: 78, Source1: 46, Source2: 44, Source3: 56
[exec] Core0 currently: Interrupts in total: 224, Source0: 98, Source1: 72, Source2: 42, Source3: 12
[exec] Core1 currently: Interrupts in total: 256, Source0: 90, Source1: 51, Source2: 52, Source3: 63
[exec] Core0 currently: Interrupts in total: 256, Source0: 110, Source1: 84, Source2: 48, Source3: 14
[exec] Core1 currently: Interrupts in total: 288, Source0: 102, Source1: 57, Source2: 57, Source3: 72
[exec] Core0 currently: Interrupts in total: 288, Source0: 125, Source1: 93, Source2: 54, Source3: 16
[exec] Core1 currently: Interrupts in total: 320, Source0: 112, Source1: 64, Source2: 64, Source3: 80
[exec] Core0 currently: Interrupts in total: 320, Source0: 139, Source1: 103, Source2: 61, Source3: 17
[exec] Core1 currently: Interrupts in total: 352, Source0: 124, Source1: 70, Source2: 70, Source3: 88
[exec] Core0 currently: Interrupts in total: 352, Source0: 153, Source1: 113, Source2: 67, Source3: 19
[exec] Core1 currently: Interrupts in total: 384, Source0: 134, Source1: 78, Source2: 76, Source3: 96
[exec] Core0 currently: Interrupts in total: 384, Source0: 166, Source1: 125, Source2: 72, Source3: 21
[exec] Core1 currently: Interrupts in total: 416, Source0: 146, Source1: 83, Source2: 84, Source3: 103
[exec] Core0 currently: Interrupts in total: 416, Source0: 180, Source1: 135, Source2: 79, Source3: 22
[exec] Core1 currently: Interrupts in total: 448, Source0: 158, Source1: 89, Source2: 89, Source3: 112
[exec] Core1 currently: Interrupts in total: 480, Source0: 168, Source1: 96, Source2: 96, Source3: 120
[exec] Core0 currently: Interrupts in total: 448, Source0: 195, Source1: 144, Source2: 85, Source3: 24
[exec] Core1 currently: Interrupts in total: 512, Source0: 180, Source1: 102, Source2: 102, Source3: 128
[exec] Core0 currently: Interrupts in total: 480, Source0: 207, Source1: 156, Source2: 91, Source3: 26
[exec] Core1 currently: Interrupts in total: 544, Source0: 190, Source1: 110, Source2: 108, Source3: 136
[exec] Core0 currently: Interrupts in total: 512, Source0: 221, Source1: 166, Source2: 97, Source3: 28
[exec] Core1 currently: Interrupts in total: 576, Source0: 202, Source1: 115, Source2: 116, Source3: 143
[exec] Core0 currently: Interrupts in total: 544, Source0: 236, Source1: 176, Source2: 103, Source3: 29
[exec] Core1 currently: Interrupts in total: 608, Source0: 214, Source1: 121, Source2: 121, Source3: 152
[exec] Core0 currently: Interrupts in total: 576, Source0: 251, Source1: 185, Source2: 109, Source3: 31
[exec] Core1 currently: Interrupts in total: 640, Source0: 224, Source1: 128, Source2: 128, Source3: 160
[exec] Core0 currently: Interrupts in total: 608, Source0: 264, Source1: 197, Source2: 114, Source3: 33
[exec] Core1 currently: Interrupts in total: 672, Source0: 236, Source1: 134, Source2: 134, Source3: 168
[exec] Core0 currently: Interrupts in total: 640, Source0: 276, Source1: 208, Source2: 121, Source3: 35
[exec] Core1 currently: Interrupts in total: 704, Source0: 246, Source1: 142, Source2: 140, Source3: 176
[exec] Core0 currently: Interrupts in total: 672, Source0: 291, Source1: 218, Source2: 127, Source3: 36
[exec] Core1 currently: Interrupts in total: 736, Source0: 258, Source1: 147, Source2: 148, Source3: 183
[exec] Core0 currently: Interrupts in total: 704, Source0: 305, Source1: 228, Source2: 133, Source3: 38
[exec] Core1 currently: Interrupts in total: 768, Source0: 270, Source1: 153, Source2: 153, Source3: 192
[exec] Core0 currently: Interrupts in total: 736, Source0: 318, Source1: 239, Source2: 139, Source3: 40
[exec] Core1 currently: Interrupts in total: 800, Source0: 280, Source1: 160, Source2: 160, Source3: 200
[exec] Core0 currently: Interrupts in total: 768, Source0: 332, Source1: 249, Source2: 145, Source3: 42
[exec] Core1 currently: Interrupts in total: 832, Source0: 292, Source1: 166, Source2: 166, Source3: 208
[exec] Core0 currently: Interrupts in total: 800, Source0: 341, Source1: 260, Source2: 156, Source3: 43
[exec] Core1 currently: Interrupts in total: 864, Source0: 299, Source1: 181, Source2: 171, Source3: 213
[exec] Core0 currently: Interrupts in total: 832, Source0: 341, Source1: 268, Source2: 180, Source3: 43
[exec] Core1 currently: Interrupts in total: 896, Source0: 299, Source1: 213, Source2: 171, Source3: 213
[exec] Core0 currently: Interrupts in total: 864, Source0: 341, Source1: 276, Source2: 204, Source3: 43
[exec]
[exec] Core1 finally: Interrupts in total: 916, Source0: 299, Source1: 233, Source2: 171, Source3: 213
[exec] Core1: test finished
[exec]
[exec] Core0 finally: Interrupts in total: 876, Source0: 341, Source1: 279, Source2: 213, Source3: 43
[exec]
[exec] Total counts finally: ActualSum: 1792, Actual0: 640, Actual1: 512, Actual2: 384, Actual3: 256
[exec] Total failure count finally: FailureSum: 0, Failures0: 0, Failures1: 0, Failures2: 0, Failures3: 0
[exec]
[exec] Core0: Main test finished
[exec]
[exec] /home/heinrich/git/interrupt-collector/submodules/simstm/src/tb_simstm.vhd:1308:21:@773696100ps:(assertion note): test finished with no errors!!
[exec] /home/heinrich/git/interrupt-collector/simulation/ghdl-wishbone/work/tb_top_wishbone:info: simulation stopped by --stop-time @99992130300ps
```

Then simulation can be re-run then, since compilation and elaboration has already been done by the target ‘ghdl-wishbone-all’:

```
cd git/interrupt-collector
ant ghdl-wishbone-simulate
```

This is very useful since the SimStm (.stm) stimuli files can be changed to do a new simulation WITHOUT recompilation.

Then simulation can be re-run with GUI:

```
cd git/interrupt-collector
ant ghdl-wishbone-simulate-gui
```

If the script complains about ‘gtkwave’ missing, see respective install section in Install GHDL Ubuntu 2204 to install it.

1.1.7. GHDL Windows

We use Windows 10 for demonstration.

At least a java runtime >= 17 is required. If not already present e.g., install it by downloading:

JAVA

Git is required to clone the interrupt-collector repository. If not already present e.g., by having already installed MSYS2 for GHDL install it in your favorite way.

Next, we clone the actual interrupt collector repository repository:

```
git clone --recursive https://github.com/eccelerators/interrupt-collector.git
```

The `--recursive` parameter is mandatory because many resources e.g. child build.xml files are residing there.

In an unconventional approach, all necessary artifacts generated by previously executed build steps in the Ecclerators IP project workflow are already present in the cloned working copy. This setup facilitates easy use and progress with this starter IP.

All builds are run by means of APACHE ANT and respective build.xml files. If not already present e.g., install it by downloading:

ANT

Unzip it e.g., to C:/apache-ant-1.10.14-bin and add the bin folder to your path environment.

```
c:\Data\git\interrupt-collector>ant -p build.xml
```

should produce then:

```
Buildfile: c:\Data\git\interrupt-collector\build.xml

Main targets:

 _helper-add-submodules                    initially add all submodules given in the helper add_submodules.py list
 _helper-clean-project-totally             remove all generated folders
 _helper-generate-ghdl-wishbone-ant-build-xml      generate ant build file for ghdl wishbone case
 _helper-generate-modelsim-wishbone-ant-build-xml  generate ant build file for modelsim wishbone case
 _helper-generate-proposal-for-setup-py    generate a proposal for setup.py
 _helper-plausibility_check_of_setup_py    check plausibility of setup.py
 _helper-remove-ghdl                       remove ghdl
 _helper-remove-modelsim                   remove modelsim
 _helper-remove-submodules                 remove all submodules given in the helper remove_submodules.py list
 ghdl-wishbone-all                         all from scratch until interactive simulation
 ghdl-wishbone-clean                       delete work folder
 ghdl-wishbone-compile                     compile all
 ghdl-wishbone-elaborate                   elaborate
 ghdl-wishbone-prepare                     make work folder
 ghdl-wishbone-simulate                    run simulation
 ghdl-wishbone-simulate-gui                simulate and write trace.vcd
 hxs-all                                   compile all
 hxs-clean                                 Delete all previously generated result files
 hxs-docbook2html                          Generate a HTML5 file
 hxs-docbook2pdf                           Generate a PDF file
 hxs-hxs2c                                 Build c files
 hxs-hxs2python                            Build python files
 hxs-hxs2rst                               Build rst text description
 hxs-hxs2simstm                            Build simstm files
 hxs-hxs2vhdl                              Build vhdl files
 hxs-rst2docbook                           Build docbook description from rst with Sphinx
 hxs-rst2html-sphinx                       Build html description from rst with Sphinx
 modelsim-wishbone-all                     all from scratch until interactive simulation
 modelsim-wishbone-all-gui                 all from scratch until interactive simulation
 modelsim-wishbone-clean                   delete work folder
 modelsim-wishbone-compile                 compile all
 modelsim-wishbone-prepare                 make work folder
 modelsim-wishbone-simulate                simulate
 modelsim-wishbone-simulate-gui            simulate start gui
```

To run the simulation with GHDL its version 4.0.0 must be present.

If not jump to Install GHDL Windows section, if not callable as ‘ghdl’ in path, usually with Windows, jump to *AdaptAntBuild* Adapt Ant Build.

Then simulation can be run, assuming the cloned repository is located in ‘git/interrupt-collector’:

```
cd git/interrupt-collector
ant ghdl-wishbone-all
```

This should result in:

```
[exec] Core0: Main test started
[exec] Core1: Cross notify interrupt happened
[exec] Core1: test started
[exec] Core0: Cross notify interrupt happened
[exec] Core1 currently: Interrupts in total: 32, Source0: 12, Source1: 6, Source2: 6, Source3: 8
[exec] Core0 currently: Interrupts in total: 32, Source0: 13, Source1: 11, Source2: 6, Source3: 2
[exec] Core1 currently: Interrupts in total: 64, Source0: 22, Source1: 14, Source2: 12, Source3: 16
[exec] Core0 currently: Interrupts in total: 64, Source0: 28, Source1: 21, Source2: 12, Source3: 3
[exec] Core1 currently: Interrupts in total: 96, Source0: 34, Source1: 19, Source2: 20, Source3: 23
[exec] Core0 currently: Interrupts in total: 96, Source0: 43, Source1: 30, Source2: 18, Source3: 5
[exec] Core1 currently: Interrupts in total: 128, Source0: 46, Source1: 25, Source2: 25, Source3: 32
[exec] Core0 currently: Interrupts in total: 128, Source0: 56, Source1: 41, Source2: 24, Source3: 7
[exec] Core1 currently: Interrupts in total: 160, Source0: 56, Source1: 32, Source2: 32, Source3: 40
[exec] Core0 currently: Interrupts in total: 160, Source0: 69, Source1: 52, Source2: 30, Source3: 9
[exec] Core1 currently: Interrupts in total: 192, Source0: 68, Source1: 38, Source2: 38, Source3: 48
[exec] Core0 currently: Interrupts in total: 192, Source0: 84, Source1: 62, Source2: 36, Source3: 10
[exec] Core1 currently: Interrupts in total: 224, Source0: 78, Source1: 46, Source2: 44, Source3: 56
[exec] Core0 currently: Interrupts in total: 224, Source0: 98, Source1: 72, Source2: 42, Source3: 12
[exec] Core1 currently: Interrupts in total: 256, Source0: 90, Source1: 51, Source2: 52, Source3: 63
[exec] Core0 currently: Interrupts in total: 256, Source0: 110, Source1: 84, Source2: 48, Source3: 14
[exec] Core1 currently: Interrupts in total: 288, Source0: 102, Source1: 57, Source2: 57, Source3: 72
[exec] Core0 currently: Interrupts in total: 288, Source0: 125, Source1: 93, Source2: 54, Source3: 16
[exec] Core1 currently: Interrupts in total: 320, Source0: 112, Source1: 64, Source2: 64, Source3: 80
[exec] Core0 currently: Interrupts in total: 320, Source0: 139, Source1: 103, Source2: 61, Source3: 17
[exec] Core1 currently: Interrupts in total: 352, Source0: 124, Source1: 70, Source2: 70, Source3: 88
[exec] Core0 currently: Interrupts in total: 352, Source0: 153, Source1: 113, Source2: 67, Source3: 19
[exec] Core1 currently: Interrupts in total: 384, Source0: 134, Source1: 78, Source2: 76, Source3: 96
[exec] Core0 currently: Interrupts in total: 384, Source0: 166, Source1: 125, Source2: 72, Source3: 21
[exec] Core1 currently: Interrupts in total: 416, Source0: 146, Source1: 83, Source2: 84, Source3: 103
[exec] Core0 currently: Interrupts in total: 416, Source0: 180, Source1: 135, Source2: 79, Source3: 22
[exec] Core1 currently: Interrupts in total: 448, Source0: 158, Source1: 89, Source2: 89, Source3: 112
[exec] Core1 currently: Interrupts in total: 480, Source0: 168, Source1: 96, Source2: 96, Source3: 120
[exec] Core0 currently: Interrupts in total: 448, Source0: 195, Source1: 144, Source2: 85, Source3: 24
[exec] Core1 currently: Interrupts in total: 512, Source0: 180, Source1: 102, Source2: 102, Source3: 128
[exec] Core0 currently: Interrupts in total: 480, Source0: 207, Source1: 156, Source2: 91, Source3: 26
[exec] Core1 currently: Interrupts in total: 544, Source0: 190, Source1: 110, Source2: 108, Source3: 136
[exec] Core0 currently: Interrupts in total: 512, Source0: 221, Source1: 166, Source2: 97, Source3: 28
[exec] Core1 currently: Interrupts in total: 576, Source0: 202, Source1: 115, Source2: 116, Source3: 143
[exec] Core0 currently: Interrupts in total: 544, Source0: 236, Source1: 176, Source2: 103, Source3: 29
[exec] Core1 currently: Interrupts in total: 608, Source0: 214, Source1: 121, Source2: 121, Source3: 152
[exec] Core0 currently: Interrupts in total: 576, Source0: 251, Source1: 185, Source2: 109, Source3: 31
[exec] Core1 currently: Interrupts in total: 640, Source0: 224, Source1: 128, Source2: 128, Source3: 160
[exec] Core0 currently: Interrupts in total: 608, Source0: 264, Source1: 197, Source2: 114, Source3: 33
[exec] Core1 currently: Interrupts in total: 672, Source0: 236, Source1: 134, Source2: 134, Source3: 168
[exec] Core0 currently: Interrupts in total: 640, Source0: 276, Source1: 208, Source2: 121, Source3: 35
[exec] Core1 currently: Interrupts in total: 704, Source0: 246, Source1: 142, Source2: 140, Source3: 176
[exec] Core0 currently: Interrupts in total: 672, Source0: 291, Source1: 218, Source2: 127, Source3: 36
[exec] Core1 currently: Interrupts in total: 736, Source0: 258, Source1: 147, Source2: 148, Source3: 183
[exec] Core0 currently: Interrupts in total: 704, Source0: 305, Source1: 228, Source2: 133, Source3: 38
[exec] Core1 currently: Interrupts in total: 768, Source0: 270, Source1: 153, Source2: 153, Source3: 192
[exec] Core0 currently: Interrupts in total: 736, Source0: 318, Source1: 239, Source2: 139, Source3: 40
[exec] Core1 currently: Interrupts in total: 800, Source0: 280, Source1: 160, Source2: 160, Source3: 200
[exec] Core0 currently: Interrupts in total: 768, Source0: 332, Source1: 249, Source2: 145, Source3: 42
[exec] Core1 currently: Interrupts in total: 832, Source0: 292, Source1: 166, Source2: 166, Source3: 208
[exec] Core0 currently: Interrupts in total: 800, Source0: 341, Source1: 260, Source2: 156, Source3: 43
[exec] Core1 currently: Interrupts in total: 864, Source0: 299, Source1: 181, Source2: 171, Source3: 213
[exec] Core0 currently: Interrupts in total: 832, Source0: 341, Source1: 268, Source2: 180, Source3: 43
[exec] Core1 currently: Interrupts in total: 896, Source0: 299, Source1: 213, Source2: 171, Source3: 213
[exec] Core0 currently: Interrupts in total: 864, Source0: 341, Source1: 276, Source2: 204, Source3: 43
[exec]
[exec] Core1 finally: Interrupts in total: 916, Source0: 299, Source1: 233, Source2: 171, Source3: 213
[exec] Core1: test finished
[exec]
[exec] Core0 finally: Interrupts in total: 876, Source0: 341, Source1: 279, Source2: 213, Source3: 43
[exec]
[exec] Total counts finally: ActualSum: 1792, Actual0: 640, Actual1: 512, Actual2: 384, Actual3: 256
[exec] Total failure count finally: FailureSum: 0, Failures0: 0, Failures1: 0, Failures2: 0, Failures3: 0
[exec]
[exec] Core0: Main test finished
[exec]
[exec] /home/heinrich/git/interrupt-collector/submodules/simstm/src/tb_simstm.vhd:1308:21:@773696100ps:(assertion note): test finished with no errors!!
[exec] /home/heinrich/git/interrupt-collector/simulation/ghdl-wishbone/work/tb_top_wishbone:info: simulation stopped by --stop-time @99992130300ps
```

Then simulation can be re-run then, since compilation and elaboration has already been done by the target ‘ghdl-wishbone-all’:

```
cd git/interrupt-collector
ant ghdl-wishbone-simulate
```

This is very useful since the SimStm (.stm) stimuli files can be changed to do a new simulation WITHOUT recompilation.

Then simulation can be re-run with GUI:

```
cd git/interrupt-collector
ant ghdl-wishbone-simulate-gui
```

If the script complains about ‘gtkwave’ missing, see respective install section in Install GHDL Windows to install it.

1.1.8. ModelSim

Assuming ModelSim is already installed, to simulate the design we run:

```
cd git/interrupt-collector
ant modelsim-wishbone-all
```

Possibly the path to the ModelSim executable may have to be adapted in the ant build.xml file.

Using ModelSim Starter edition it may take up to 5 minutes until the output of a successful simulation will yield like this:

(ModelSim Starter edition will take already a very long time cause of design size)

```
...

# Core0 currently: Interrupts in total: 0x0380, Source0: 0x013E, Source1: 0xFF, Source2: 0xC0, Source3: 0x83
# Core1 currently: Interrupts in total: 0x0380, Source0: 0x0142, Source1: 0x0101, Source2: 0xC0, Source3: 0x7D
#
# Core1 finally: Interrupts in total: 0x0380, Source0: 0x0142, Source1: 0x0101, Source2: 0xC0, Source3: 0x7D
# Core1: test finished
#
# Core0 finally: Interrupts in total: 0x0380, Source0: 0x013E, Source1: 0xFF, Source2: 0xC0, Source3: 0x83
#
# Total counts finally: ActualSum: 0x0700, Actual0: 0x0280, Actual1: 0x0200, Actual2: 0x0180, Actual3: 0x0100
# Total failure count finally: FailureSum: 0x00, Failures0: 0x00, Failures1: 0x00, Failures2: 0x00, Failures3: 0x00
#
# Core0: Main test finished
#
# ** Note: test finished with no errors!!
#    Time: 773216100 ps  Iteration: 0  Instance: /tb_top_wishbone/i0_tb_simstm
# ** Note: Leaving proc Main and halt at line 195 end_proc file ../../tb/simstm/TestMainWishbone.stm
#    Time: 1000790207100 ps  Iteration: 0  Instance: /tb_top_wishbone/i1_tb_simstm
```

Then simulation can be re-run, since compilation and elaboration has already been done by the target ‘ghdl-wishbone-all’:

```
cd git/interrupt-collector
ant modelsim-wishbone-simulate
```

This is very useful since the SimStm (.stm) stimuli files can be changed to do a new simulation WITHOUT recompilation.

Then simulation can be re-run with GUI:

```
cd git/interrupt-collector
ant ghdl-wishbone-simulate-gui
```

1.1.9. HxS - Hardware/Software interface

The Hardware/Software interface description of the Interrupt-Collector generated by HxS is: file.

The respective HxS sources are found in the hxs folder of the interrupt-collector clone.

Further generated artifacts Vhdl, SimStm, C, Python, HTML-Documentation, and PDF-Documentation is placed in hxs artifacts folder.

The same applies for the Interrupt-Generator of the Mock and can be found the respective submodule folder.

A glance at the HxS source snippet of the Interrupt Request Register:

Figure 1.4. HxS Interrupt Request Register Snippet

```
        register InterruptRequestReg
        {
            Name = "Interrupt Request Register";
            Width = 32;
            WriteTransparentPulse = true;
            Bits = [
                Request(Id="Request0"),
                Request(Id="Request1"),
                Request(Id="Request2"),
                Request(Id="Request3")
            ];
            enum Request
            {
                Behaviour = BitBehaviour.Transparent;
                Width = 1;
                Values = [
                    NotPending,
                    Pending,
                    NotConfirmed,
                    Confirmed
                ];

                value NotPending
                {
                    Value = 0x0;
                    Behaviour = ValueBehaviour.Read;
                    Description = "An Interrupt is not pending.";
                }

                value Pending
                {
                    Value = 0x1;
                    Behaviour = ValueBehaviour.Read;
                    Description = "An Interrupt is pending.";
                }

                value NotConfirmed
                {
                    Name = "Writing this value has no effect.";
                    Behaviour = ValueBehaviour.Write;
                    Value = 0x0;
                }

                value Confirmed
                {
                    Value = 0x1;
                    Behaviour = ValueBehaviour.Write;
                    Name = "Notifies HW that a pending interrupt has been recognized by SW.";
                    Description = "
                        SW confirms that a respective interrupt service routine has been entered.
                        The value isn't stored, thus there is no need to reset it to '0' again.
                        Solely the write action is sufficient.
                        ";
                }

                Resets = {
                0b0:  BusReset.Async (Behaviour = ResetBehaviour.Read,
                        Description = "
                            The bit Request is set to NotPending after reset in the usually attached InterruptCollector HW block.")
                };
            }
        }
```

Then HxS artifacts can be generated by calling the following ANT target. A precondition for this step is having installed the HxS Tool and Python, see Install Python. However since the artifacts are already present, since they are unusually under version control in this repository too, it is not necessary to have the tool to run the simulation.

```
cd git/interrupt-collector
ant hxs-all
```

The target calls further targets in different levels of the complete workflow:

1. • hxs-vhdl to generate the vhdl files in the src-gen/vhdl folder referenced by the user code files in the src/vhdl folder.
   - hxs-hxs2c to generate the C-header files in the src-gen/c folder.
   - hxs-hxs2python to generate the Python class files in the src-gen/python folder.
   - hxs-hxs2simstm to generate the SimStm files in the src-gen/simstm folder included by the testbench.
   - hxs-hxs2rst to generate the restructured text files in the src-gen/rst folder referenced by the user code files in the src/rst folder.

     A '.drawio' drawing draw.io is generated there, waiting to be included in documentation by the user

or used for presentations.

A preliminary ‘.docx’ Microsoft Word is generated there, if enabled by annotation in the HxS source. The user restructured text entered in

HxS descriptions is not yet transformed but flows through as it is, it will be presented in following realses of HxS.

2. • hxs-rst2html-sphinx to generate a final Sphinx style HTML document in src-gen/html-sphinx

   The composition of the resulting document is determined by user source code in the folder src/rst.The generated

   files are included there to determine the place where they are located in the final user document with e.g., additional user sections.

   • hxs-rst2docbook to generate the necessary interim input files for further DocBook transformations .

   The composition of the resulting document by further steps is determined by user source code in the folder src/docbook. The generated

   files are included there to determine the place where they are located in the final user document with e.g., additional user sections.

   Resources like the company logo can be adapted there. For further customization hints see Docbook Customizations section.

3. • hxs-docbook2pdf to generate the final PDF document im src-gen/docbook-pdf

   • hxs-docbook2html to generate a final HTML document im src-gen/docbook-html

   This html output is an alternative to the Sphinx html, it is much closer to the form of the PDF output.

1.1.10. DocBook customizations

The docbook tranformation is done by the submodule eccelerators-docbook. Further customizations can be done by taking this as a base for a own ‘user-docbook’ submodule. The adaptions are to be done in the folder customization similar to the eccelerators-book found there in our submodule. Detailed explanations can be found at DocBook.

1.1.11. Further steps

- Simply use it for your design.
- Have a template for Eclipse and VsCode to easily enter the Mask, Request and Service bits with common name stem.
- Adapt tb_signals_pkg.vhd

  to different interrupt priority schemes or nested interrupts.

  Extend SimStm code with respective test cases.
- Adapt InterruptDispatcher.vhd

  to respect disabled interrupts in a core or a core already busy with an interrupt in dispatachin scheme.

  Extend SimStm code with respective test cases.

- Model more cores and user specific behaviour

1.1.12. Install GHDL Ubuntu 22.04

Install it by downloading:

https://github.com/ghdl/ghdl/releases/tag/v4.0.0/ghdl-gha-ubuntu-22.04-llvm.tgz.

Copy the downloaded file to the a local folder e.g, ‘ghdl_download’ and unpack it there e.g., with

```
cd ghdl_download
tar -xzf ghdl-gha-ubuntu-22.04-llvm.tgz -C ./usr
```

It is sufficient to copy the contents of the subfolders of the unpacked user folder to their respective pendants in the system root ‘/usr’ after their owner has been set to root.

```
sudo chown -R root:root ./usr
sudo cp -r ./usr/bin/* /usr/bin
sudo cp -r ./usr/include/* /usr/include
sudo cp -r ./usr/lib/* /usr/lib
```

Then issuing:

```
ghdl --version
```

should show:

```
GHDL 4.0.0 (3.0.0.r912.gc0e7e1483) [Dunoon edition]
 Compiled with GNAT Version: 10.5.0
 llvm 14.0.0 code generator
Written by Tristan Gingold.

Copyright (C) 2003 - 2024 Tristan Gingold.
GHDL is free software, covered by the GNU General Public License.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

It may complain about missing libraries, then install them with:

```
sudo apt update
sudo apt install build-essential
sudo apt install llvm-14
sudo apt install gnat
```

Install ‘gtkwave’ to use the GUI variant for simulation:

```
sudo apt update
sudo apt install gtkwave
```

1.1.13. Install GHDL Windows

There are many options beneath the following, we describe the MSYS way.

MSYS2 is required, install it by downloading:

MSYS2

Add `C:\msys64\ucrt64\bin` to your environment path variable.

Download GHDL:

GHDL

Execute in the ‘MSYS2 MSYS’ console:

```
cd /c/Users/<your-user>/Downloads
pacman -U mingw-w64-ucrt-x86_64-ghdl-llvm-ci-1-any.pkg.tar.zst
```

Then issuing:

```
ghdl --version
```

should show:

```
GHDL 4.0.0 (3.0.0.r912.gc0e7e1483) [Dunoon edition]
 Compiled with GNAT Version: 10.5.0
 llvm 14.0.0 code generator
Written by Tristan Gingold.

Copyright (C) 2003 - 2024 Tristan Gingold.
GHDL is free software, covered by the GNU General Public License.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Install ‘gtkwave’ to use the GUI variant for simulation in the ‘MSYS2 MSYS’ console:

```
pacman -S mingw-w64-x86_64-gtkwave
```

1.1.14. Install Python

Python must be present to run the helper- and hxs- generator targets of the ANT build file. To get all necessary dependencies you can run:

```
cd git/interrupt-collector
pip install -r requirements.txt
```

1.1.15. Adapt ANT build

If it is present its path can be adapted in the top ‘build.xml’ file in the repository root once for all builds and child builds. Especially the pathes to the python , ghdl and modelsim executables usually have to be adapted for windows.

```
``<!-- properties for local purposes, should be overriden by ci ant call e.g,      ``
``     with -Dpython-executable argument for ci purposes                          ``
``                                                                                ``
``defaults :                                                                      ``
``                                                                                ``
``<property name="ghdl-executable" value="ghdl"/>                                 ``
``<property name="gtkwave-executable" value="gtkwave"/>                           ``
``                                                                                ``
``<property name="vlib-executable" value="vlib"/>                                 ``
``<property name="vmap-executable" value="vmap"/>                                 ``
``<property name="vcom-executable" value="vcom"/>                                 ``
``<property name="vsim-executable" value="vsim"/>                                 ``
``                                                                                ``
``override respectively by uncommenting e.g, for python the following line:       ``
``-->                                                                             ``
``<!-- <property name="python-executable"                                         ``
``     value="C:Users\user\AppData\Local\Programs\Python\Python38\python.exe"/> --> ``
```

1.2. Interrupt Collector Interface (InterruptCollectorIfc)

Interface containing a basic Interrupt Collector block.

Interrupt Collector details:

Figure 1.5. Interrupt Collector details slice0

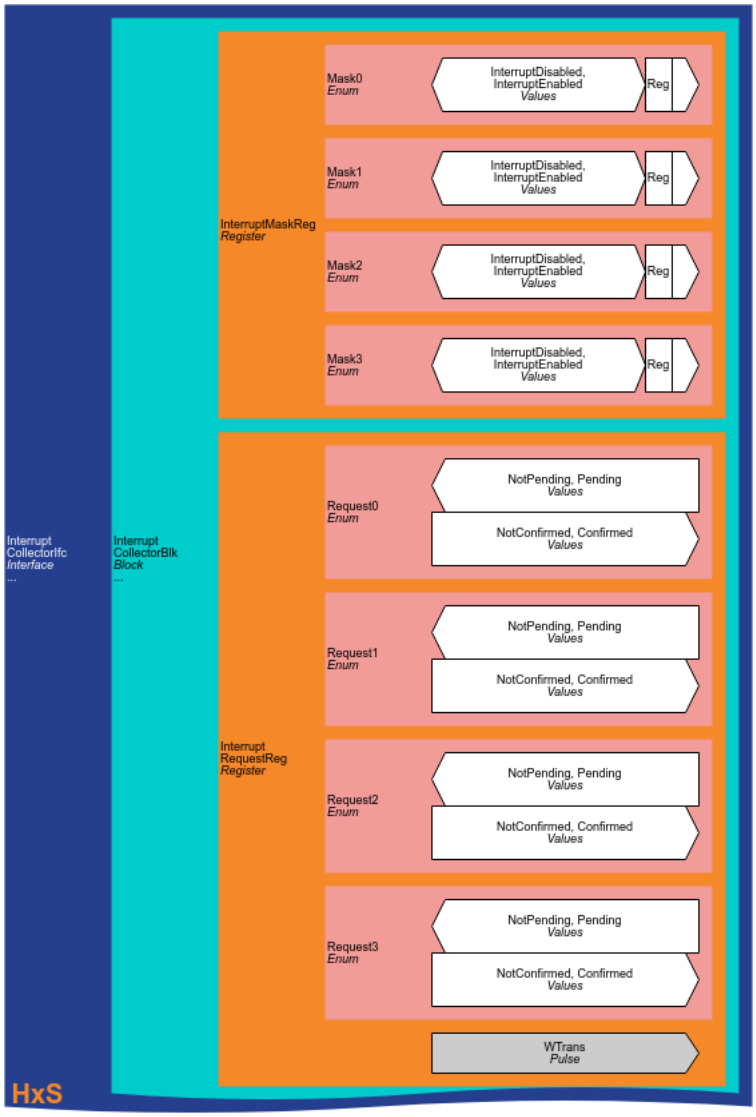


Figure 1.6. Interrupt Collector details slice1

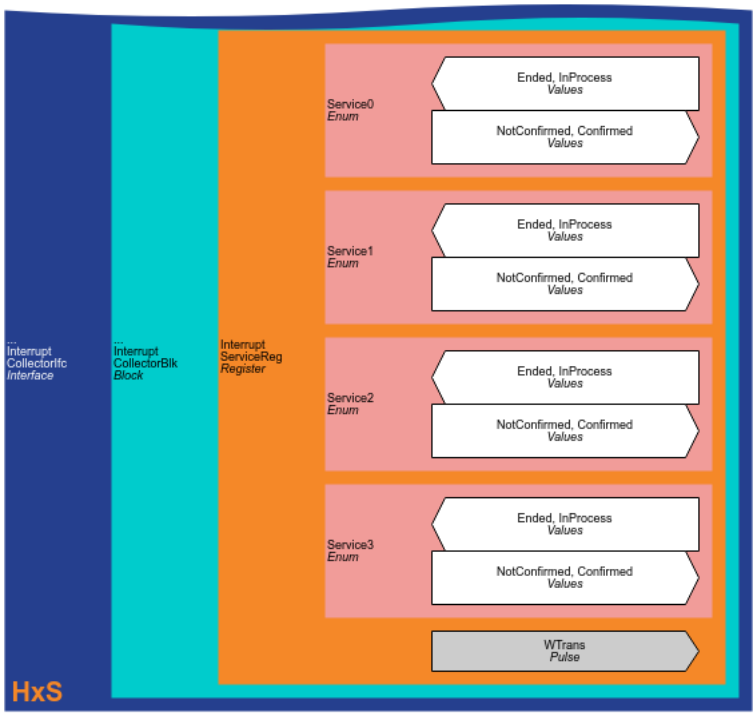


Table 1.1. Blocks of Interrupt Collector Interface

Blocks of Interrupt Collector Interface		
Block Address	ID	Block Name
0x00	InterruptCollectorBlk	Interrupt Collector Block

Table 1.2. Resets of Interrupt Collector Interface

Resets of Registers of Interrupt Collector Interface	
ID	Reset Name
Async	BusReset: Asynchronous Bus Reset

1.2.1. Interrupt Collector Block (InterruptCollectorBlk)

This block defines a basic interrupt collector for **level triggered** interrupt sources. Usually edge triggered sources e.g., timer pulses can be converted to level triggered ones by catching them in the user logic.

Constraints:

1. Allow interrupt processing by multiple CPUs without need for spinlocks.
2. Enable forwarding an interrupt to the CPU(s) by a mask for each source.
3. Provide control pulses to notify the user logic when a interrupt service for an interrupt request has been started and has been ended.
4. Use the control pulses to reset the interrupt request fo a source or do it by a write or read access directly to the user logic e.g. reading the receive data register of an UART.

Table 1.3. Registers or Delegates of Interrupt Collector Block

Registers or Delegates of Interrupt Collector Block		
0x00 .. 0x0b		Interrupt Collector Block
Address	ID	Name
0x00	InterruptMaskReg	Interrupt Mask Register
0x04	InterruptRequestReg	Interrupt Request Register
0x08	InterruptServiceReg	Interrupt Service Register

1.2.1.1. Interrupt Mask Register (InterruptMaskReg)

Table 1.4. Bits of Interrupt Mask Register

Bits of Interrupt Mask Register			
0x00			Interrupt Mask Register (InterruptMaskReg)
Bits	ID	Type	Description
03	Mask3	RW	**Table 1.5. Values of Mask3** Value / ID / Type / Description 0x0 / InterruptDisabled / RW / Interrupt is not forwarded to CPU(s). 0x1 / InterruptEnabled / RW / Interrupt is forwarded to CPU(s). **Table 1.6. Resets of Mask3** 0x0 / BusReset / RW / Default Bus Reset
02	Mask2	RW	**Table 1.7. Values of Mask2** Value / ID / Type / Description 0x0 / InterruptDisabled / RW / Interrupt is not forwarded to CPU(s). 0x1 / InterruptEnabled / RW / Interrupt is forwarded to CPU(s). **Table 1.8. Resets of Mask2** 0x0 / BusReset / RW / Default Bus Reset
01	Mask1	RW	**Table 1.9. Values of Mask1** Value / ID / Type / Description 0x0 / InterruptDisabled / RW / Interrupt is not forwarded to CPU(s). 0x1 / InterruptEnabled / RW / Interrupt is forwarded to CPU(s). **Table 1.10. Resets of Mask1** 0x0 / BusReset / RW / Default Bus Reset
00	Mask0	RW	**Table 1.11. Values of Mask0** Value / ID / Type / Description 0x0 / InterruptDisabled / RW / Interrupt is not forwarded to CPU(s). 0x1 / InterruptEnabled / RW / Interrupt is forwarded to CPU(s). **Table 1.12. Resets of Mask0** 0x0 / BusReset / RW / Default Bus Reset

1.2.1.2. Interrupt Request Register (InterruptRequestReg)

Table 1.13. Bits of Interrupt Request Register

Bits of Interrupt Request Register			
0x04			Interrupt Request Register (InterruptRequestReg)
Bits	ID	Type	Description
03	Request3	R/W	**Table 1.14. Values of Request3** Value / ID / Type / Description 0x0 / NotPending / R / An Interrupt is not pending.

0x1	Pending	R	An Interrupt is pending.
0x0	NotConfirmed	W	Writing this value has no effect.
0x1	Confirmed	W	Notifies HW that a pending interrupt has been recognized by SW. SW confirms that a respective interrupt service routine has been entered. The value isn't stored, thus there is no need to reset it to '0' again. Solely the write action is sufficient.

Table 1.15. Resets of Request3

Init Value	ID	Impact	Description
0b0	Async	R	BusReset: Asynchronous Bus Reset The bit Request is set to NotPending after reset in the usually attached InterruptCollector HW block.

02	Request2	R/W	**Table 1.16. Values of Request2**

Value	ID	Type	Description
0x0	NotPending	R	An Interrupt is not pending.
0x1	Pending	R	An Interrupt is pending.
0x0	NotConfirmed	W	Writing this value has no effect.
0x1	Confirmed	W	Notifies HW that a pending interrupt has been recognized by SW. SW confirms that a respective interrupt service routine has been entered. The value isn't stored, thus there is no need to reset it to '0' again. Solely the write action is sufficient.

Table 1.17. Resets of Request2

Init Value	ID	Impact	Description
0b0	Async	R	BusReset: Asynchronous Bus Reset The bit Request is set to NotPending after reset in the usually attached InterruptCollector HW block.

01	Request1	R/W	**Table 1.18. Values of Request1**

Value	ID	Type	Description
0x0	NotPending	R	An Interrupt is not pending.
0x1	Pending	R	An Interrupt is pending.
0x0	NotConfirmed	W	Writing this value has no effect.
0x1	Confirmed	W	Notifies HW that a pending interrupt has been recognized by SW. SW confirms that a respective interrupt service routine has been entered. The value isn't stored, thus there is no need to reset it to '0' again. Solely the write action is sufficient.

Table 1.19. Resets of Request1

Init Value	ID	Impact	Description
0b0	Async	R	BusReset: Asynchronous Bus Reset The bit Request is set to NotPending after reset in the usually attached InterruptCollector HW block.

00	Request0	R/W	**Table 1.20. Values of Request0**

Value	ID	Type	Description
0x0	NotPending	R	An Interrupt is not pending.
0x1	Pending	R	An Interrupt is pending.
0x0	NotConfirmed	W	Writing this value has no effect.
0x1	Confirmed	W	Notifies HW that a pending interrupt has been recognized by SW. SW confirms that a respective interrupt service routine has been entered. The value isn't stored, thus there is no need to reset it to '0' again. Solely the write action is sufficient.

Table 1.21. Resets of Request0

Init Value	ID	Impact	Description
0b0	Async	R	BusReset: Asynchronous Bus Reset The bit Request is set to NotPending after reset in the usually attached InterruptCollector HW block.

1.2.1.3. Interrupt Service Register (InterruptServiceReg)

Table 1.22. Bits of Interrupt Service Register

Bits of Interrupt Service Register			
0x08			Interrupt Service Register (InterruptServiceReg)
Bits	ID	Type	Description
03	Service3	R/W	**Table 1.23. Values of Service3**

Value	ID	Type	Description

			0x0	Ended	R	The Interrupt service has ended.
			0x1	InProcess	R	The Interrupt is in service.
			0x0	NotConfirmed	W	Writing this value has no effect.
			0x1	Confirmed	W	Notifies HW that a pending interrupt has been recognized by SW. SW confirms that a respective interrupt service routine has been processed completely. The value isn't stored, thus there is no need to reset it to '0' again. Solely the write action is sufficient.

Table 1.24. Resets of Service3

Init Value	ID	Impact	Description
0b0	Async	R	BusReset: Asynchronous Bus Reset The bit Service is set to Ended after reset in the usually attached InterruptCollector HW block.

02	Service2	R/W	**Table 1.25. Values of Service2**

Value	ID	Type	Description
0x0	Ended	R	The Interrupt service has ended.
0x1	InProcess	R	The Interrupt is in service.
0x0	NotConfirmed	W	Writing this value has no effect.
0x1	Confirmed	W	Notifies HW that a pending interrupt has been recognized by SW. SW confirms that a respective interrupt service routine has been processed completely. The value isn't stored, thus there is no need to reset it to '0' again. Solely the write action is sufficient.

Table 1.26. Resets of Service2

Init Value	ID	Impact	Description
0b0	Async	R	BusReset: Asynchronous Bus Reset The bit Service is set to Ended after reset in the usually attached InterruptCollector HW block.

01	Service1	R/W	**Table 1.27. Values of Service1**

Value	ID	Type	Description
0x0	Ended	R	The Interrupt service has ended.
0x1	InProcess	R	The Interrupt is in service.
0x0	NotConfirmed	W	Writing this value has no effect.
0x1	Confirmed	W	Notifies HW that a pending interrupt has been recognized by SW. SW confirms that a respective interrupt service routine has been processed completely. The value isn't stored, thus there is no need to reset it to '0' again. Solely the write action is sufficient.

Table 1.28. Resets of Service1

Init Value	ID	Impact	Description
0b0	Async	R	BusReset: Asynchronous Bus Reset The bit Service is set to Ended after reset in the usually attached InterruptCollector HW block.

00	Service0	R/W	**Table 1.29. Values of Service0**

Value	ID	Type	Description
0x0	Ended	R	The Interrupt service has ended.
0x1	InProcess	R	The Interrupt is in service.
0x0	NotConfirmed	W	Writing this value has no effect.
0x1	Confirmed	W	Notifies HW that a pending interrupt has been recognized by SW. SW confirms that a respective interrupt service routine has been processed completely. The value isn't stored, thus there is no need to reset it to '0' again. Solely the write action is sufficient.

Table 1.30. Resets of Service0

Init Value	ID	Impact	Description
0b0	Async	R	BusReset: Asynchronous Bus Reset The bit Service is set to Ended after reset in the usually attached InterruptCollector HW block.

1.3. Interrupt Collector C-Header preview

The original file can be found in the generated src-gen/c folder.

The defines in this file should be used for FW/SW development, thus bits, register. … can be moved in HxS congruently.

```
/* Copyright (C) 2024 Eccelerators GmbH

   This code was generated by:

   HxS Compiler v0.0.0-0000000
   C Extension for HxS 1.0.24-85d98215

   Further information at https://eccelerators.com/hxs

   Changes to this file may cause incorrect behavior and will be lost if the
   code is regenerated.
*/

#ifndef _InterruptCollectorIfc_H
#define _InterruptCollectorIfc_H

/* ---------------------------------------------------------- */
/* Addresses, widths, values and masks for direct access */
/* ---------------------------------------------------------- */
#define InterruptCollectorIfcAddressBusWidth 4
#define InterruptCollectorIfcDataBusWidth 32

#define InterruptCollectorBlkAddress 0x0
#define InterruptCollectorBlkSize 0xC

#define InterruptMaskRegAddress (0x0 + InterruptCollectorBlkAddress)
#define InterruptMaskRegWidth 32

#define Mask3Mask 0x00000008
#define Mask3Position 3
#define Mask3Width 1
#define Mask3_InterruptDisabledMVal 0x00000000
#define Mask3_InterruptEnabledMVal 0x00000008
#define Mask3BusResetMRstVal 0x00000000


#define Mask2Mask 0x00000004
#define Mask2Position 2
#define Mask2Width 1
#define Mask2_InterruptDisabledMVal 0x00000000
#define Mask2_InterruptEnabledMVal 0x00000004
#define Mask2BusResetMRstVal 0x00000000

#define Mask1Mask 0x00000002
#define Mask1Position 1
#define Mask1Width 1
#define Mask1_InterruptDisabledMVal 0x00000000
#define Mask1_InterruptEnabledMVal 0x00000002
#define Mask1BusResetMRstVal 0x00000000

#define Mask0Mask 0x00000001
#define Mask0Position 0
#define Mask0Width 1
#define Mask0_InterruptDisabledMVal 0x00000000
#define Mask0_InterruptEnabledMVal 0x00000001
#define Mask0BusResetMRstVal 0x00000000

#define InterruptRequestRegAddress (0x4 + InterruptCollectorBlkAddress)
#define InterruptRequestRegWidth 32

#define Request3Mask 0x00000008
#define Request3Position 3
#define Request3Width 1
#define Request3_NotPendingMVal 0x00000000
#define Request3_PendingMVal 0x00000008
#define Request3_NotConfirmedMVal 0x00000000
#define Request3_ConfirmedMVal 0x00000008
#define Request3_AsyncMRstVal 0x00000000

#define Request2Mask 0x00000004
#define Request2Position 2
#define Request2Width 1
#define Request2_NotPendingMVal 0x00000000
#define Request2_PendingMVal 0x00000004
#define Request2_NotConfirmedMVal 0x00000000
#define Request2_ConfirmedMVal 0x00000004
```

```
#define Request2_AsyncMRstVal 0x00000000

#define Request1Mask 0x00000002
#define Request1Position 1
#define Request1Width 1
#define Request1_NotPendingMVal 0x00000000
#define Request1_PendingMVal 0x00000002
#define Request1_NotConfirmedMVal 0x00000000
#define Request1_ConfirmedMVal 0x00000002
#define Request1_AsyncMRstVal 0x00000000

#define Request0Mask 0x00000001
#define Request0Position 0
#define Request0Width 1
#define Request0_NotPendingMVal 0x00000000
#define Request0_PendingMVal 0x00000001
#define Request0_NotConfirmedMVal 0x00000000
#define Request0_ConfirmedMVal 0x00000001
#define Request0_AsyncMRstVal 0x00000000

#define InterruptServiceRegAddress (0x8 + InterruptCollectorBlkAddress)
#define InterruptServiceRegWidth 32

#define Service3Mask 0x00000008
#define Service3Position 3
#define Service3Width 1
#define Service3_EndedMVal 0x00000000
#define Service3_InProcessMVal 0x00000008
#define Service3_NotConfirmedMVal 0x00000000
#define Service3_ConfirmedMVal 0x00000008
#define Service3_AsyncMRstVal 0x00000000

#define Service2Mask 0x00000004
#define Service2Position 2
#define Service2Width 1
#define Service2_EndedMVal 0x00000000
#define Service2_InProcessMVal 0x00000004
#define Service2_NotConfirmedMVal 0x00000000
#define Service2_ConfirmedMVal 0x00000004
#define Service2_AsyncMRstVal 0x00000000


#define Service1Mask 0x00000002
#define Service1Position 1
#define Service1Width 1
#define Service1_EndedMVal 0x00000000
#define Service1_InProcessMVal 0x00000002
#define Service1_NotConfirmedMVal 0x00000000
#define Service1_ConfirmedMVal 0x00000002
#define Service1_AsyncMRstVal 0x00000000

#define Service0Mask 0x00000001
#define Service0Position 0
#define Service0Width 1
#define Service0_EndedMVal 0x00000000
#define Service0_InProcessMVal 0x00000001
#define Service0_NotConfirmedMVal 0x00000000
#define Service0_ConfirmedMVal 0x00000001
#define Service0_AsyncMRstVal 0x00000000

#endif
```

1.4. Interrupt Collector Python code preview

The original file can be found in the generated src-gen/python folder.

Together with simstm2py it should be used to run the simstm test on the real target HW.

```
# Copyright (C) 2024 Eccelerators GmbH
#
# This code was generated by:
#
# HxS Compiler v0.0.0-0000000
# Python Extension for HxS 1.0.7-62bb9ef4
#
# Further information at https://eccelerators.com/hxs
#
# Changes to this file may cause incorrect behavior and will be lost if the
# code is regenerated.


# ------------------------------------------------------
# Addresses, widths, values and masks for direct access
# ------------------------------------------------------
InterruptCollectorIfcAddressBusWidth = 4
InterruptCollectorIfcDataBusWidth = 32

class InterruptCollectorBlk:
    InterruptCollectorBlkAddress = 0x0
    InterruptCollectorBlkSize = 0xC

    @property
    def InterruptMaskRegAddress(self):
        return (0x0 + self.InterruptCollectorBlkAddress)

    InterruptMaskRegWidth = 32

    Mask3Mask = 0x00000008
    Mask3Position = 3
    Mask3Width = 1
    Mask3_InterruptDisabledMVal = 0x00000000
    Mask3_InterruptEnabledMVal = 0x00000008
    Mask3BusResetMRstVal = 0x00000000


    Mask2Mask = 0x00000004
    Mask2Position = 2
    Mask2Width = 1
    Mask2_InterruptDisabledMVal = 0x00000000
    Mask2_InterruptEnabledMVal = 0x00000004
    Mask2BusResetMRstVal = 0x00000000

    Mask1Mask = 0x00000002
    Mask1Position = 1
    Mask1Width = 1
    Mask1_InterruptDisabledMVal = 0x00000000
    Mask1_InterruptEnabledMVal = 0x00000002
    Mask1BusResetMRstVal = 0x00000000

    Mask0Mask = 0x00000001
    Mask0Position = 0
    Mask0Width = 1
    Mask0_InterruptDisabledMVal = 0x00000000
    Mask0_InterruptEnabledMVal = 0x00000001
    Mask0BusResetMRstVal = 0x00000000


    @property
    def InterruptRequestRegAddress(self):
        return (0x4 + self.InterruptCollectorBlkAddress)

    InterruptRequestRegWidth = 32

    Request3Mask = 0x00000008
    Request3Position = 3
    Request3Width = 1
    Request3_NotPendingMVal = 0x00000000
    Request3_PendingMVal = 0x00000008
    Request3_NotConfirmedMVal = 0x00000000
    Request3_ConfirmedMVal = 0x00000008
    Request3_AsyncMRstVal = 0x00000000

    Request2Mask = 0x00000004
    Request2Position = 2
```

```
Request2Width = 1
Request2_NotPendingMVal = 0x00000000
Request2_PendingMVal = 0x00000004
Request2_NotConfirmedMVal = 0x00000000
Request2_ConfirmedMVal = 0x00000004
Request2_AsyncMRstVal = 0x00000000

Request1Mask = 0x00000002
Request1Position = 1
Request1Width = 1
Request1_NotPendingMVal = 0x00000000
Request1_PendingMVal = 0x00000002
Request1_NotConfirmedMVal = 0x00000000
Request1_ConfirmedMVal = 0x00000002
Request1_AsyncMRstVal = 0x00000000

Request0Mask = 0x00000001
Request0Position = 0
Request0Width = 1
Request0_NotPendingMVal = 0x00000000
Request0_PendingMVal = 0x00000001
Request0_NotConfirmedMVal = 0x00000000
Request0_ConfirmedMVal = 0x00000001
Request0_AsyncMRstVal = 0x00000000


@property
def InterruptServiceRegAddress(self):
   return (0x8 + self.InterruptCollectorBlkAddress)

InterruptServiceRegWidth = 32

Service3Mask = 0x00000008
Service3Position = 3
Service3Width = 1
Service3_EndedMVal = 0x00000000
Service3_InProcessMVal = 0x00000008
Service3_NotConfirmedMVal = 0x00000000
Service3_ConfirmedMVal = 0x00000008
Service3_AsyncMRstVal = 0x00000000


Service2Mask = 0x00000004
Service2Position = 2
Service2Width = 1
Service2_EndedMVal = 0x00000000
Service2_InProcessMVal = 0x00000004
Service2_NotConfirmedMVal = 0x00000000
Service2_ConfirmedMVal = 0x00000004
Service2_AsyncMRstVal = 0x00000000

Service1Mask = 0x00000002
Service1Position = 1
Service1Width = 1
Service1_EndedMVal = 0x00000000
Service1_InProcessMVal = 0x00000002
Service1_NotConfirmedMVal = 0x00000000
Service1_ConfirmedMVal = 0x00000002
Service1_AsyncMRstVal = 0x00000000

Service0Mask = 0x00000001
Service0Position = 0
Service0Width = 1
Service0_EndedMVal = 0x00000000
Service0_InProcessMVal = 0x00000001
Service0_NotConfirmedMVal = 0x00000000
Service0_ConfirmedMVal = 0x00000001
Service0_AsyncMRstVal = 0x00000000
```

1.5. Interrupt Collector Simulation code preview

It is complemented by the simstm code in the tb/simstm folder of the repository and should be used for HDL testbench development to operate the InterruptCollector.

The original file can be found in the generated src-gen/simstm folder.

The entry point for simulation is the testMain.stm file found in the tb/simstm folder..

```
-- Copyright (C) 2024 Eccelerators GmbH
--
-- This code was generated by:
--
-- HxS Compiler v0.0.0-0000000
-- SimStm Extension for HxS 1.0.12-d6fdbb9b
--
-- Further information at https://eccelerators.com/hxs
--
-- Changes to this file may cause incorrect behavior and will be lost if the
-- code is regenerated.


-- Eccelerators.Library.IP

const InterruptCollectorIfcAddressBusWidth 4
const InterruptCollectorIfcDataBusWidth 32

var InterruptCollectorIfcAddress 0
bus InterruptCollectorIfcBus 0

var InterruptCollectorBlkAddress 0x0
const InterruptCollectorBlkSize 0xC

var InterruptMaskRegAddress 0x0
const InterruptMaskRegWidth 32

const Mask3Mask 0x00000008
const Mask3Position 3
const Mask3Width 1
const Mask3_InterruptDisabledMVal 0x00000000
const Mask3_InterruptEnabledMVal 0x00000008
const Mask3BusResetMRstVal 0x00000000


const Mask2Mask 0x00000004
const Mask2Position 2
const Mask2Width 1
const Mask2_InterruptDisabledMVal 0x00000000
const Mask2_InterruptEnabledMVal 0x00000004
const Mask2BusResetMRstVal 0x00000000

const Mask1Mask 0x00000002
const Mask1Position 1
const Mask1Width 1
const Mask1_InterruptDisabledMVal 0x00000000
const Mask1_InterruptEnabledMVal 0x00000002
const Mask1BusResetMRstVal 0x00000000

const Mask0Mask 0x00000001
const Mask0Position 0
const Mask0Width 1
const Mask0_InterruptDisabledMVal 0x00000000
const Mask0_InterruptEnabledMVal 0x00000001
const Mask0BusResetMRstVal 0x00000000

var InterruptRequestRegAddress 0x0
const InterruptRequestRegWidth 32

const Request3Mask 0x00000008
const Request3Position 3
const Request3Width 1
const Request3_NotPendingMVal 0x00000000
const Request3_PendingMVal 0x00000008
const Request3_NotConfirmedMVal 0x00000000
const Request3_ConfirmedMVal 0x00000008
const Request3_AsyncMRstVal 0x00000000

const Request2Mask 0x00000004
const Request2Position 2
const Request2Width 1
const Request2_NotPendingMVal 0x00000000
const Request2_PendingMVal 0x00000004
const Request2_NotConfirmedMVal 0x00000000
```

```
const Request2_ConfirmedMVal 0x00000004
const Request2_AsyncMRstVal 0x00000000

const Request1Mask 0x00000002
const Request1Position 1
const Request1Width 1
const Request1_NotPendingMVal 0x00000000
const Request1_PendingMVal 0x00000002
const Request1_NotConfirmedMVal 0x00000000
const Request1_ConfirmedMVal 0x00000002
const Request1_AsyncMRstVal 0x00000000

const Request0Mask 0x00000001
const Request0Position 0
const Request0Width 1
const Request0_NotPendingMVal 0x00000000
const Request0_PendingMVal 0x00000001
const Request0_NotConfirmedMVal 0x00000000
const Request0_ConfirmedMVal 0x00000001
const Request0_AsyncMRstVal 0x00000000

var InterruptServiceRegAddress 0x0
const InterruptServiceRegWidth 32

const Service3Mask 0x00000008
const Service3Position 3
const Service3Width 1
const Service3_EndedMVal 0x00000000
const Service3_InProcessMVal 0x00000008
const Service3_NotConfirmedMVal 0x00000000
const Service3_ConfirmedMVal 0x00000008
const Service3_AsyncMRstVal 0x00000000

const Service2Mask 0x00000004
const Service2Position 2
const Service2Width 1
const Service2_EndedMVal 0x00000000
const Service2_InProcessMVal 0x00000004
const Service2_NotConfirmedMVal 0x00000000
const Service2_ConfirmedMVal 0x00000004

const Service2_AsyncMRstVal 0x00000000

const Service1Mask 0x00000002
const Service1Position 1
const Service1Width 1
const Service1_EndedMVal 0x00000000
const Service1_InProcessMVal 0x00000002
const Service1_NotConfirmedMVal 0x00000000
const Service1_ConfirmedMVal 0x00000002
const Service1_AsyncMRstVal 0x00000000

const Service0Mask 0x00000001
const Service0Position 0
const Service0Width 1
const Service0_EndedMVal 0x00000000
const Service0_InProcessMVal 0x00000001
const Service0_NotConfirmedMVal 0x00000000
const Service0_ConfirmedMVal 0x00000001
const Service0_AsyncMRstVal 0x00000000


InterruptCollectorIfcInit:
proc
    equ InterruptCollectorBlkAddress 0x0
    add InterruptCollectorBlkAddress $InterruptCollectorIfcAddress
    call $InterruptCollectorBlkInit
end proc

InterruptCollectorBlkInit:
proc
    equ InterruptMaskRegAddress 0x0
    add InterruptMaskRegAddress $InterruptCollectorBlkAddress
    equ InterruptRequestRegAddress 0x4
    add InterruptRequestRegAddress $InterruptCollectorBlkAddress
    equ InterruptServiceRegAddress 0x8
    add InterruptServiceRegAddress $InterruptCollectorBlkAddress
end proc

var RvalInterruptCollectorBlk_BusReset 0
```

```
ResetTestInterruptCollectorBlkByBusReset:
proc
   bus verify InterruptCollectorIfcBus 32 $InterruptMaskRegAddress RvalInterruptCollectorBlk_BusReset 0 $Mask3Mask
   bus verify InterruptCollectorIfcBus 32 $InterruptMaskRegAddress RvalInterruptCollectorBlk_BusReset 0 $Mask2Mask
   bus verify InterruptCollectorIfcBus 32 $InterruptMaskRegAddress RvalInterruptCollectorBlk_BusReset 0 $Mask1Mask
   bus verify InterruptCollectorIfcBus 32 $InterruptMaskRegAddress RvalInterruptCollectorBlk_BusReset 0 $Mask0Mask
end proc


var RbvlInterruptCollectorBlk 0

ReadBackTestInterruptCollectorBlk:
proc
   bus write InterruptCollectorIfcBus 32 $InterruptMaskRegAddress 0x00000000
   bus verify InterruptCollectorIfcBus 32 $InterruptMaskRegAddress RbvlInterruptCollectorBlk 0x00000000 $Mask3Mask
   bus write InterruptCollectorIfcBus 32 $InterruptMaskRegAddress 0x00000008
   bus verify InterruptCollectorIfcBus 32 $InterruptMaskRegAddress RbvlInterruptCollectorBlk 0x00000008 $Mask3Mask
   bus write InterruptCollectorIfcBus 32 $InterruptMaskRegAddress 0x00000000
   bus verify InterruptCollectorIfcBus 32 $InterruptMaskRegAddress RbvlInterruptCollectorBlk 0x00000000 $Mask2Mask
   bus write InterruptCollectorIfcBus 32 $InterruptMaskRegAddress 0x00000004
   bus verify InterruptCollectorIfcBus 32 $InterruptMaskRegAddress RbvlInterruptCollectorBlk 0x00000004 $Mask2Mask
   bus write InterruptCollectorIfcBus 32 $InterruptMaskRegAddress 0x00000000
   bus verify InterruptCollectorIfcBus 32 $InterruptMaskRegAddress RbvlInterruptCollectorBlk 0x00000000 $Mask1Mask
   bus write InterruptCollectorIfcBus 32 $InterruptMaskRegAddress 0x00000002
   bus verify InterruptCollectorIfcBus 32 $InterruptMaskRegAddress RbvlInterruptCollectorBlk 0x00000002 $Mask1Mask
   bus write InterruptCollectorIfcBus 32 $InterruptMaskRegAddress 0x00000000
   bus verify InterruptCollectorIfcBus 32 $InterruptMaskRegAddress RbvlInterruptCollectorBlk 0x00000000 $Mask0Mask
   bus write InterruptCollectorIfcBus 32 $InterruptMaskRegAddress 0x00000001
   bus verify InterruptCollectorIfcBus 32 $InterruptMaskRegAddress RbvlInterruptCollectorBlk 0x00000001 $Mask0Mask
end proc
```