Summary - Metadata Stressor

Name	metadata_stressor
Worker Type	Application
Version	v1.4
Release Date	September 2018
Component Library	ocpi.core
Workers	metadata_stressor.hdl, metadata_stressor.rcc
Tested Platforms	isim, xsim, modelsim, xilinx13_3, centos6, centos7, alst4, ml605, ZedBoard(PL), Matchstiq-Z1(PL)

Functionality

The *metadata_stressor* component tests an HDL worker's robustness during development as part of the unit test suite. An HDL worker is expected to accept all valid combinations of metadata without failure, though some are unlikely to be encountered in normal operation. It also may starve the unit under test of data and insert delays between messages. The data starvation may be random or based on a duty cycle. The worker is automatically built into the HDL test assemblies generated by the framework. It can also add zero length messages between messages. It passes through the data it receives without change.

Worker Implementation Details

metadata_stressor.hdl

The *metadata_stressor* test worker has four modes controlling its primary operation: bypass, data, metadata, and full. In bypass mode, this worker passes through the data and metadata is receives without change. In data mode, the worker passes through the metadata associated with a message unchanged, but the data will be held based on the duty cycle or lfsr, imitating data starvation for the unit under test. (If enable_take_lsfr is not set to true or take_duty is not set to greater than 1, then the worker will set the duty cycle to 5.) In metadata mode, the worker passes through the data (not intentionally withholding any data) but manipulates the metadata in the following ways:

- early start of message (SOM), data, late end of message (EOM)
- early SOM, data, EOM with data
- SOM with data, data, late EOM
- SOM with data, data, EOM with data, (single word message if that is what is received)
- zero length message (if allow_zlms is true)

It repeats those patterns so long as there is data. In full mode, the worker combines data and metadata modes, manipulating both the metadata and starving the unit under test of data. Do not use data and allow_zlms in combination. Data mode does not allow for manipulation of metadata, so zero length messages cannot be inserted. Enabling both does not cause a failure, but one behavior will preclude the expression of the other.

metadata_stressor.rcc

The RCC version of this component is just a placeholder to fulfill the requirements of unit test framework. It passes through data without change and shouldn't be included in normal applications, as it provides no real functionality.

Theory

There are some combinations of metadata that are valid but not often encountered that a worker should be designed to handle.

Block Diagrams

Top level

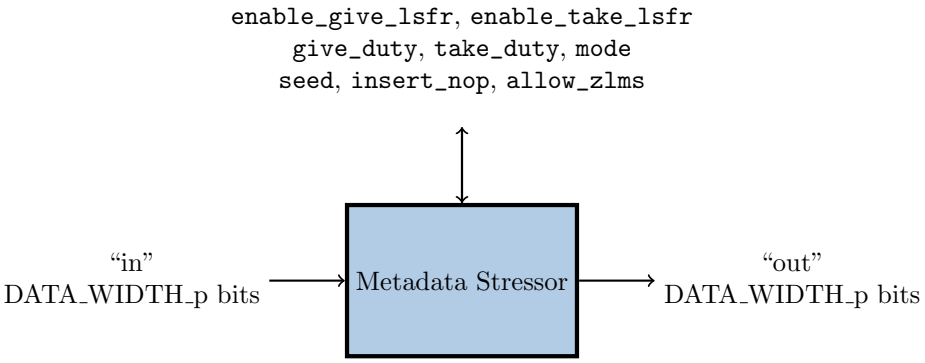
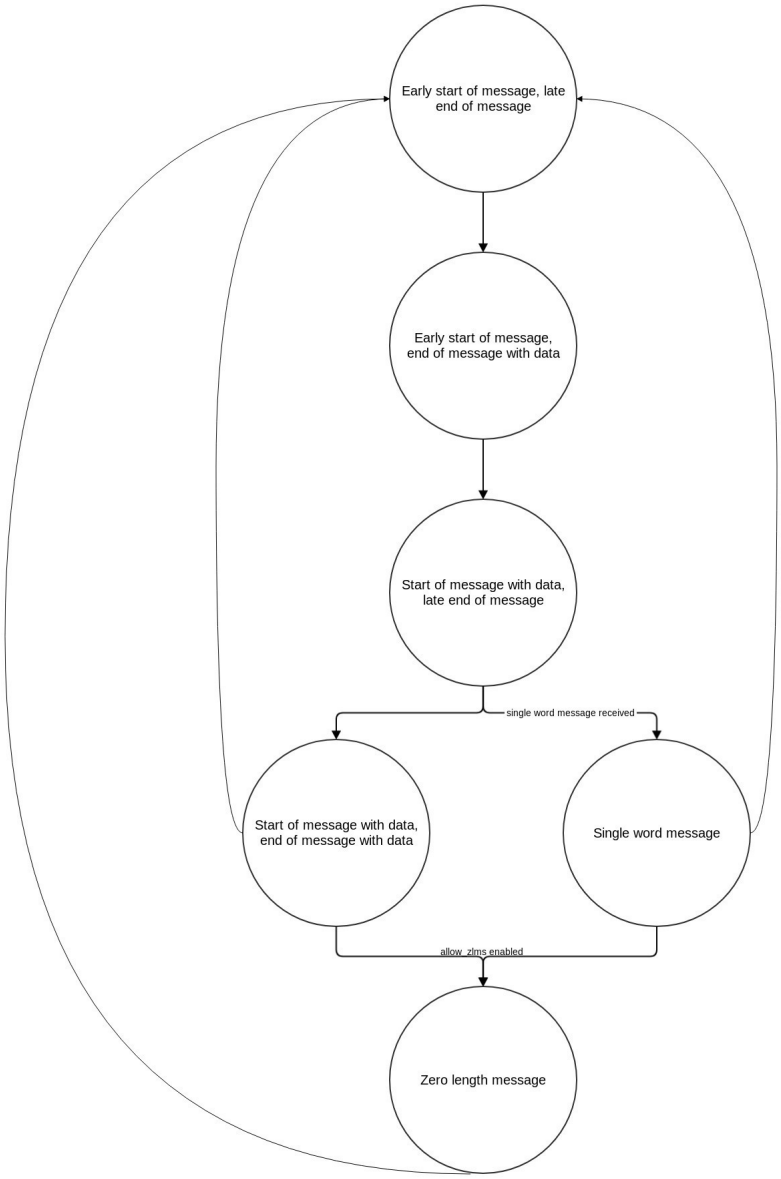


Figure 1: Top Level Block Diagram

State Machine

Below is an abbreviated representation of the primary finite state machine implemented in the HDL version of this component.



Source Dependencies

metadata_stressor.hdl

- projects/core/components//metadata_stressor.hdl/metadata_stressor.vhd
- core/hdl/primitives/util/util_pkg.vhd
  projects/core/hdl/primitives/util/zlm_detector.vhd

metadata_stressor.rcc

- projects/core/components/metadata_stressor.rcc/metadata_stressor.cc

Component Spec Properties

Name	Type	SequenceLength	ArrayDimensions	Accessibility	Valid Range	Default	Usage
enable_give_lsfr	bool	-	-	Readable, Writable	Standard	False	True: MSB of lsfr drives give, False: give_duty drives give
enable_take_lsfr	bool	-	-	Readable, Writable	Standard	False	True: 7th bit of lsfr drives take, False: take_duty drives take
give_duty	ushort	-	-	Readable, Writable	Standard	1	Set 'give' duty cycle if enable_give_lsfr is false
take_duty	ushort	-	-	Readable, Writable	Standard	1	Set 'take' duty cycle if enable_take_lsfr is false
mode	enum	-	-	Readable, Writable	Standard	bypass	bypass: worker passes through data and metadata, data: worker varies data, but passes through metadata, metadata: vary metadata, keep data steady, full: vary all metadata and data
seed	ushort	-	-	Readable, Writable	Standard	1	seed for lsfr
allow_zlms	bool	-	-	Readable, Writable	Standard	False	Insert ZLMs between some messages
insert_nop	bool	-	-	Readable, Writable	Standard	False	Insert delays between messages

Worker Properties

metadata_stressor.hdl

Type	Name	Type	SequenceLength	ArrayDimensions	Accessibility	Valid Range	Default	Usage
Property	DATA_WIDTH_p	UChar	-	-	Readable, Parameter	8/16/32/64	12	I/O data width

Component Ports

Name	Producer	Protocol	Optional	Advanced	Usage
in	false	None	False	-	32 bits
out	true	None	False	-	32 bits

Worker Interfaces

metadata_stressor.hdl

Type	Name	DataWidth	Advanced	Usage
StreamInterface	in	DATA_WIDTH_p	-	Size defined by DATA_WIDTH_p
StreamInterface	out	DATA_WIDTH_p	-	Size defined by DATA_WIDTH_p

Control Timing and Signals

metadata_stressor.hdl

This worker implementation uses the clock from the Control Plane and standard Control Plane signals.

Worker Configuration Parameters

metadata_stressor.hdl

Table 1: Table of Worker Configurations for worker: metadata_stressor

Configuration
0

Performance and Resource Utilization

metadata_stressor.hdl

Table 2: Resource Utilization Table for worker: metadata_stressor

Configuration	OCPI Target	Tool	Version	Device	Registers (Typ)	LUTs (Typ)	Fmax (MHz) (Typ)	Memory/Special Functions
0	zynq	Vivado	2017.1	xc7z020clg400-3	364	449	N/A	N/A
0	stratix4	Quartus	17.1.0	N/A	388	557	N/A	N/A
0	virtex6	ISE	14.7	6vcx75tff484-2	362	802	294.583	N/A

Test and Verification

This component is tested via the unit test automation feature of the framework. The component's .test/ contains XML files that describe the combinations of tests.

The test cases exercise changes in every property across three cases, though not every property in every case, as that would take a prohibitively long time.

- Case 1 - Message size set to 4, tests component with single word messages, duration set to 60 seconds
    1. give_duty = 1: constant
    2. give_duty = 4: 1 on 4 off
    3. take_duty = 1: constant
    4. take_duty = 5: 1 on 5 off
    5. insert_nop = True: insert delay between messages
    6. insert_nop = False: no delay between messages
    7. mode = data: only vary data
    8. mode = metadata: only vary metadata
    9. mode = full: vary data and metadata
- Case 2 - Message size set to 128, this tests most of the functionality, duration set to 100 seconds
    1. enable_give_lsfr = True: use lsfr to vary give
    2. enable_give_lsfr = False: use duty cycle to vary give
    3. enable_take_lsfr = True: use lsfr to vary take
    4. enable_take_lsfr = False: use duty cycle to vary take
    5. give_duty = 1: constant
    6. give_duty = 4: 1 on 3 off
    7. take_duty = 1: constant
    8. take_duty = 5: 1 on 4 off
    9. mode = data: only vary data
    10. mode = metadata: only vary metadata
    11. mode = full: vary data and metadata
    12. insert_nop = True: insert delay between messages
    13. insert_nop = False: no delay between messages
    14. seed = 35: seed for lsfr
- Case 3 - Tests using the component to generate zero length messages, duration set to 60 seconds
    1. allow_zlms = True: zero length messages inserted between some messages
    2. mode = metadata: only vary metadata
    3. mode = full: vary data and metadata
    4. insert_nop = True: insert delay between messages
    5. insert_nop = False: no delay between messages
- Case 4 - Tests the RCC version of this component, which is nothing but a placeholder, duration set to 60 seconds

In all test cases, the data is simply passed through the component and the tests are determined to be successful by comparing the input and output files.