Summary - pattern_v2

Name	pattern_v2
Version	v1.0
Release Date	August 2018
Component Library	ocpi.assets.util_comps
Workers	pattern_v2.hdl
Tested Platforms	xsim, isim, matchstiq_z1

Functionality

The pattern_v2 component provides the ability to output a pattern of messages by allowing the user to create a record of messages each having a configurable number of bytes and associated opcode. Through a set of properties, the component may send messages (data and opcode) up to the amount dictated by the build-time parameters.

The `messages` property defines the record of messages to send, as well as, defines the number of data bytes and an opcode for each message.

For example:

When `messages` = {4, 255}, one message will be sent having 4 bytes of data and an opcode of 255.
When `messages` = {8, 251}, {6, 250}, two messages will sent, the first having 8 bytes of data and an opcode of 251, and the second message having 6 bytes of data and an opcode of 250.

Data to be sent with a message is defined by the `data` property and is referred to as the data buffer. The number of data words in the data buffer is the number of data bytes for the messages.

The component offers an additional feature when there are multiple messages via the `dataRepeat` property which indicates whether the a message starts at the beginning of the data buffer, or continues from its current index within the buffer.

For example:

Given `messages` = {4, 251},{8, 252},{12, 253},{16, 254},{20, 255}

If `dataRepeat` = true, then `numDataWords` is 5. To calculate the `numDataWords` when `dataRepeat` is true, divide the largest message size (in bytes) by 4. Dividing by four required because the data is output as a 4 byte data word. Since the largest message size in the given messages assignment is 20, 20/4 = 5.

When `numDataWords` = 5, then a valid data assignment would be `data` = {0, 1, 2, 3, 4}, and the data within each message would look like: msg1 = {0}, msg2 = {0, 1}, msg3 = {0, 1, 2}, msg4 = {0, 1, 2, 3}, msg5 = {0, 1, 2, 3, 4}

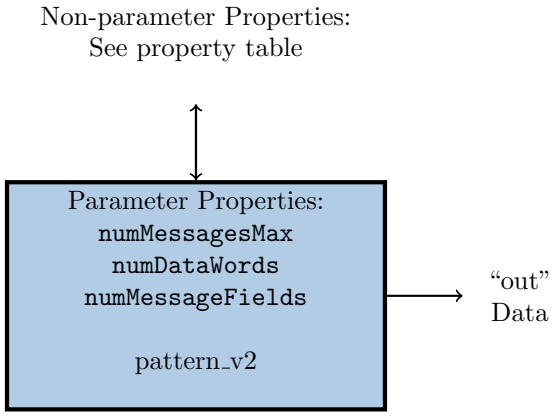
If `dataRepeat` = false, then `numDataWords` is 15. To calculate the `numDataWords` when `dataRepeat` is false, divide the sum of all the message sizes (in bytes) by 4. Dividing by four is required because the data is output as a 4 byte data word. Since the sum of all message sizes in the given messages assignment is (4+8+12+16+20)/4 = 15.

When `numDataWords` = 15, then a valid data assignment would be `data` = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}, and the data within each message would look like: msg1 = {0}, msg2 = {1, 2}, msg3 = {3, 4, 5}, msg4 = {6, 7, 8, 9}, msg5 = {10, 11, 12, 13, 14}

There is also a `messagesToSend` property that must be less than or equal to the the number of messages to send (`numMessagesMax`). The worker will check for this and report an error if `messagesToSend` is greater than `numMessagesMax`. This error reporting is for simulation only.

Block Diagrams

Top level



Source Dependencies

pattern_v2.hdl

- assets/components/util_comps/pattern_v2.hdl/pattern_v2.vhd

Component Spec Properties

Name	Type	Default	SequenceLength	ArrayLength	ArrayDimensions	Parameter	Accessibility	Usage
dataRepeat	bool	false	-	-	-	false	Initial	True - Multiple messages sent from the beginning of the data buffer. False - Multiple messages sent from the current position of the data buffer.
numMessagesMax	uLong	5	-	-	-	true	-	Max number of messages to send.
messagesToSend	uLong	5	-	-	-	false	Volatile, Writable	Counter of messages to send. This value must be less than or equal to numMessagesMax. Decrements as they are sent.
messagesSent	uLong	-	-	-	-	false	Volatile	Messages sent counter. Initialized to 0.
dataSent	uLong	-	-	-	-	false	Volatile	Words sent counter. Initialized to 0.
numDataWords	uLong	15	-	-	-	true	-	Max number of four byte data for the data buffer. To calculate the numDataWords when dataRepeat is true, divide the largest message size (in bytes) by 4. To calculate the numDataWords when dataRepeat is false, divide the sum of all the message sizes (in bytes) by 4. Dividing by four required because the data is output as a 4 byte data word.
numMessageFields	uLong	2	-	-	-	true	-	Due to a limitation, cannot use constrained elements in unconstrained array declarations, so cannot directly set the second dimension for the messages property to 2. The numMessageFields property must always be 2 since there are 2 message fields; the number of data bytes and opcode. So the default value must not be changed.
messages	uLong	-	-	-	numMessagesMax, numMessageFields	false	Initial	Multidimensional array that defines the record of messages to send, as well as, defines the number of data bytes and an opcode for each message.
data	uLong	-	-	numDataWords	-	false	Initial	Data buffer containing the data to be sent.

Component Ports

Name	Protocol	Producer	Optional	Usage
out	-	true	false	Data generated by the component

Worker Interfaces

pattern_v2.hdl

Type	Name	DataWidth (b)	Advanced	Usage
StreamInterface	out	32	DataValueWidth = 8, NumberOfOpcodes='256', ZeroLengthMessages = true	Data generated by the worker

Control Timing and Signals

The pattern_v2 worker uses the clock from the Control Plane and standard Control Plane signals.

Worker Configuration Parameters

pattern_v2.hdl

Table 1: Table of Worker Configurations for worker: pattern_v2

Configuration	numDataWords	numMessagesMax
0	15	5

Performance and Resource Utilization

pattern_v2.hdl

Table 2: Resource Utilization Table for worker: pattern_v2

Configuration	OCPI Target	Tool	Version	Device	Registers (Typ)	LUTs (Typ)	Fmax (MHz) (Typ)	Memory/Special Functions
0	zynq	Vivado	2017.1	xc7z020clg400-3	933	585	N/A	N/A
0	virtex6	ISE	14.7	6vcx75tff484-2	932	606	252.666	N/A
0	stratix4	Quartus	17.1.0	N/A	934	611	N/A	N/A

Test and Verification

The pattern_v2 worker is tested by generating data for the `messages` and `data` properties and verifying that final value of the volatile properties and the output data are correct. Since the pattern_v2 worker's `messages` and `data` property array sizes depend on parameters, they have to be generated via scripts(`gen_messages.py` and `gen_data.py`)

Applications

For an example of the pattern_v2 component used in an application, please reference the tb_bias_v2 application located in `assets/applications/tb_bias_v2`.