

FIFO Collection - Quick Reference

UVVM Support Component

The FIFO Collection is a memory buffer that can be used to hold one or more FIFOs. Each FIFO will be allocated a chosen size and ID number. This allows a selectable number of FIFOs to operate individually and be independently accessed.

uvvm_fifo_init ([buffer_idx,] buffer_size_in_bits)

Example: v_buffer_idx := uvvm_fifo_init(C_BUFFER_SIZE-1); -- returns a buffer index on initialization

Example: uvvm_fifo_init(C_BUFFER_IDX_1, C_BUFFER_SIZE-1); -- buffer index is selected

uvvm_fifo_init ([buffer_idx,] buffer_size_in_bits)

Example: v_buffer_idx := uvvm_fifo_init(C_BUFFER_SIZE-1); -- returns a buffer index on initialization

Example: uvvm_fifo_init(C_BUFFER_IDX_1, C_BUFFER_SIZE-1); -- buffer index is selected

uvvm_fifo_get (buffer_idx, entry_size_in_bits)

Example: v_rx_data := uvvm_fifo_get (C_BUFFER_IDX_1, C_ENTRY_SIZE_1);

uvvm_fifo_flush (buffer_idx)

Example: uvvm_fifo_flush(C_BUFFER_IDX_1);

uvvm_fifo_peek (buffer_idx, entry_size_in_bits)

Example: v_rx_data := uvvm_fifo_peek(C_BUFFER_IDX_1, C_ENTRY_SIZE_1);

uvvm_fifo_get_count (buffer_idx)

Example: v_num_elements := uvvm_fifo_get_count(C_BUFFER_IDX_1);

uvvm_fifo_get_max_count (buffer_idx)

Example: v_max_fifo_elements := uvvm_fifo_get_max_count(C_BUFFER_IDX_1);

uvvm_fifo_is_full (buffer_idx)

Example: v_fifo_is_full := uvvm_fifo_is_full (C_BUFFER_IDX_1);

uvvm_fifo_deallocate (VOID)

Example: uvvm_fifo_deallocate(VOID);



FIFO Collection – Functional parameters

Name	Type	Example(s)	Description
buffer_idx	natural	1	The index of the FIFO that shall be initialized.
buffer_size_in_bits	natural	1024	The size of the FIFO.
data	SLV	v_rx_data	The data that shall be pushed to the FIFO.

FIFO Collection details

All FIFO functions and procedures are defined in the UVVM Data FIFO package, `ti_data_fifo_pkg.vhd`

1 FIFO Collection details and examples

Method	Description
uvvm_fifo_init()	<p>uvvm_fifo_init([buffer_idx,] buffer_size_in_bits)</p> <p>This UVVM FIFO call will allocate space in the FIFO buffer. If no <code>buffer_idx</code> is given, the call will return a buffer index for use when addressing the FIFO. Note that 0 will be returned on error. If a <code>buffer_idx</code> is given, the FIFO is initialized with this index.</p> <p>Example:</p> <pre>uvvm_fifo_init(C_BUFFER_IDX_1, C_BUFFER_SIZE-1); -- initialize buffer with index C_BUFFER_IDX_1 v_fifo_idx := uvvm_fifo_init(C_BUFFER_SIZE-1);</pre>
uvvm_fifo_put()	<p>uvvm_fifo_put(buffer_idx, data)</p> <p>This procedure puts data into a FIFO with index <code>buffer_idx</code>. The size of the data is unconstrained, meaning that it can be any size. Pushing data with a size that is larger than the FIFO size results in wrapping, i.e., that when reaching the end that data remaining will overwrite the data that was first written.</p> <p>Example:</p> <pre>uvvm_fifo_put(C_BUFFER_IDX_1, v_rx_data);</pre>
uvvm_fifo_get()	<p>uvvm_fifo_get(buffer_idx, entry_size_in_bits)</p> <p>This function returns the data from the FIFO and removes the returned data from the FIFO. Note that <code>buffer_idx</code> is the index of the FIFO that shall be read, and that <code>entry_size_in_bits</code> is the size of the returned data as SLV. Attempting to get data from an empty FIFO is allowed but triggers a <code>TB_WARNING</code> and returns garbage data. Attempting to get a larger value than the FIFO size is allowed but triggers a <code>TB_WARNING</code>.</p> <p>Example:</p> <pre>v_rx_data := uvvm_fifo_get(C_BUFFER_IDX_1, C_ENTRY_SIZE-1);</pre>

uvvm_fifo_flush()	uvvm_fifo_flush(buffer_idx) This procedure empties the FIFO given by buffer_idx. Example: <pre>uvvm_fifo_flush(C_BUFFER_IDX_1);</pre>
uvvm_fifo_peek()	uvvm_fifo_peek(buffer_idx, entry_size_in_bits) This function returns the data from the FIFO without removing it. Note that, apart from not removing the data, this function will behave in the same way as the uvvm_fifo_get() function. Example: <pre>v_rx_data := uvvm_fifo_peek(C_BUFFER_IDX_1, C_ENTRY_SIZE-1);</pre>
uvvm_fifo_get_count()	uvvm_fifo_get_count(buffer_idx) This function returns a natural indicating the number of elements currently occupying the FIFO given by buffer_idx. Example: <pre>v_num_elements := uvvm_fifo_get_count(C_BUFFER_IDX);</pre>
uvvm_fifo_get_max_count()	uvvm_fifo_get_max_count(buffer_idx) This function returns a natural indicating the maximum number of elements that can occupy the FIFO given by buffer_idx. Example: <pre>v_max_elements := uvvm_fifo_get_max_count(C_BUFFER_IDX);</pre>
uvvm_fifo_is_full()	uvvm_fifo_is_full(buffer_idx) This function returns a boolean indicating if the FIFO is full or not. Example: <pre>v_fifo_is_full := uvvm_fifo_is_full(C_BUFFER_IDX);</pre>
uvvm_fifo_deallocate()	uvvm_fifo_deallocate(VOID) This function deallocates the FIFO buffer, all the FIFO pointers are reset. Example: <pre>uvvm_fifo_deallocate(VOID);</pre>

INTELLECTUAL PROPERTY

Disclaimer: This IP and any part thereof are provided "as is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with this IP.