

SBI BFM – Quick Reference

sbi_write (addr_value, data_value, msg, clk, sbi_if, [scope, [msg_id_panel, [config]]])

Example: sbi_write(x"1000", x"40", "Set baud rate to 9600", clk, sbi_if);

Suggested usage: sbi_write(C_ADDR_UART_TX, C_BAUD_9600, "Set baud rate to 9600"); -- Suggested usage requires local overload (see section 5)

sbi_read (addr_value, data_value, msg, clk, sbi_if, [scope, [msg_id_panel, [config, [proc_name]]]])

Example: sbi_read(x"1000", v_data_out, "Read UART baud rate", clk, sbi_if);

Suggested usage: sbi_read(C_ADDR_UART_BAUD, v_data_out, "Read UART baud rate"); -- Suggested usage requires local overload (see section 5)

sbi_check (addr_value, data_exp, msg, clk, sbi_if, [alert_level, [scope, [msg_id_panel, [config]]]])

Example: sbi_check(x"1155", x"3B", "Check data from UART RX", clk, sbi_if);

Suggested usage: sbi_check(C_ADDR_UART_RX, x"3B", "Check data from UART RX"); -- Suggested usage requires local overload (see section 5)

sbi_poll_until (addr_value, data_exp, max_polls, timeout, msg, clk, sbi_if, terminate_loop, [alert_level, [scope, [msg_id_panel, [config]]]])

Example: sbi_poll_until(x"1155", x"0D", 10, 100 ns, "Read UART until CR is found", clk, sbi_if, terminate_loop);

Suggested usage: sbi_poll_until(C_ADDR_UART_RX, x"0D", "Read UART until CR is found"); -- Suggested usage requires local overload (see section 5)

init_sbi_if_signals (addr_width, data_width)

Example: sbi_if <= init_sbi_if_signals(addr_width, data_width);

BFM Configuration record 't_sbi_bfm_config'

| Record element | Type | C_SBI_BFM_CONFIG_DEFAULT |
|----------------------------|---------------|--------------------------|
| max_wait_cycles | integer | 10 |
| max_wait_cycles_severity | t_alert_level | FAILURE |
| use_fixed_wait_cycles_read | boolean | false |
| fixed_wait_cycles_read | natural | 0 |
| clock_period | time | 10 ns |
| clock_period_margin | time | 0 ns |
| clock_margin_severity | t_alert_level | TB_ERROR |
| setup_time | time | 2.5 ns |
| hold_time | time | 2.5 ns |
| id_for_bfm | t_msg_id | ID_BFM |
| id_for_bfm_wait | t_msg_id | ID_BFM_WAIT |
| id_for_bfm_poll | t_msg_id | ID_BFM_POLL |
| use_ready_signal | boolean | true |
| setup_time | time | 0 ns |
| hold_time | time | 0 ns |

Signal record 't_sbi_if'

| Record element | Type |
|----------------|------------------|
| cs | std_logic |
| addr | unsigned |
| wena | std_logic |
| rena | std_logic |
| wdata | std_logic_vector |
| ready | std_logic |
| rdata | std_logic_vector |

BFM



sbi_bfm_pkg.vhd



Note: BFM calls can also be made with listing of single signals rather than t_sbi_if.

BFM non-signal parameters

| Name | Type | Example(s) | Description |
|--------------|------------------|--------------------------|--|
| addr_value | unsigned | x"5A" | The address of a software accessible register. |
| data_value | std_logic_vector | x"D3" | The data value to be written to the addressed register |
| data_exp | std_logic_vector | x"0D" | The data value to expect when reading the addressed register. A mismatch results in an alert with severity 'alert_level' |
| max_polls | integer | 1 | The maximum number of polls (reads) before the expected data must be found. Exceeding this limit results in an alert with severity 'alert_level'. |
| timeout | time | 100 ns | The maximum time to pass before the expected data must be found. Exceeding this limit results in an alert with severity 'alert_level'. |
| alert_level | t_alert_level | ERROR or TB_WARNING | Set the severity for the alert that may be asserted by the BFM procedure. |
| msg | string | "Write to Peripheral 1" | A custom message to be appended in the log/alert. |
| scope | string | "SBI BFM" | A string describing the scope from which the log/alert originates. In a simple single sequencer typically "SBI BFM". In a verification component typically "SBI_VVC". |
| msg_id_panel | t_msg_id_panel | shared_msg_id_panel | Optional msg_id_panel, controlling verbosity within a specified scope. Defaults to a common ID panel defined in the adaptations package. |
| config | t_sbi_bfm_config | C_SBI_BFM_CONFIG_DEFAULT | Configuration of BFM behaviour and restrictions. See section 2 for details. |

BFM signal parameters

| Name | Type | Description |
|----------------|-----------|---|
| clk | std_logic | The clock signal used to read and write data in/out of SBI BFM. |
| sbi_if | t_sbi_if | See table "Signal record 't_sbi_if'" |
| terminate_loop | std_logic | External control of loop termination to e.g. stop polling prematurely |

Note 1: All signals are active high.

Note 2: Record sbi_if can be replaced with the signals listed in said record.

BFM details

1 BFM procedure details and examples

| Procedure | Description |
|--------------------|---|
| sbi_write() | <p>sbi_write(addr_value, data_value, msg, clk, sbi_if, [scope, [msg_id_panel, [config]]])</p> <p>The sbi_write() procedure writes the given data to the given address on the DUT, using the SBI protocol:</p> <ol style="list-style-type: none"> At 'config.clock_period'/4 before the first rising clock edge the bus lines are set: <ol style="list-style-type: none"> cs and wena are set to '1' rena is set to '0' addr is set to addr_value wdata is set to data_value With ready-signalling: <ol style="list-style-type: none"> on the first rising edge the DUT ready signal is evaluated: <ul style="list-style-type: none"> If ready is '1', cs and wena are set to '0' again 'config.clock_period'/4 after the last rising edge and the write procedure was successful If ready is '0', the procedure will wait one clock cycle and evaluate the ready signal again. This will repeat until ready is set to '1', or invoke an error if the process has repeated 'config.max_wait_cycles' times. A log message with ID config.id_for_bfm_wait is logged at the first wait. Without ready-signalling: <ol style="list-style-type: none"> cs and wena are set to '0' again 'config.clock_period'/4 after the first rising edge <ul style="list-style-type: none"> The default value of scope is C_SCOPE ("SBI BFM") The default value of msg_id_panel is shared_msg_id_panel, defined in UVVM_Util. The default value of config is C_SBI_BFM_CONFIG_DEFAULT, see table on the first page. A log message is written if message ID 'config.id_for_bfm' is enabled for the specified message ID panel. <p>The procedure reports an alert if:</p> <ul style="list-style-type: none"> ready signal is not set to '1' within 'config.max_wait_cycles' after cs and wena are set to '1' (alert_level: 'config.max_wait_cycles_severity'). <p>Examples:</p> <pre>sbi_write(x"1000", x"55", "Write data to Peripheral 1", clk, sbi_if); sbi_write(x"1000", x"55", "Write data to Peripheral 1", clk, sbi_if, C_SCOPE, shared_msg_id_panel, C_SBI_BFM_CONFIG_DEFAULT);</pre> <p>Suggested usage (requires local overload, see section 5):</p> <pre>sbi_write(C_ADDR_UART_TX, x"40", "Set baud rate to 9600");</pre> <p>Note: Record sbi_if can be replaced with the signals cs, addr, rena, wena, ready, wdata.</p> |
| sbi_read() | <p>sbi_read(addr_value, data_value, msg, clk, sbi_if, [scope, [msg_id_panel, [config, [proc_name]]]])</p> <p>The sbi_read() procedure reads data from the DUT at the given address, using the SBI protocol:</p> <ol style="list-style-type: none"> At 'config.clock_period'/4 before the first rising clock edge the bus lines are set: <ol style="list-style-type: none"> cs and rena are set to '1' wena is set to '0' addr is set to addr_value |

2. With ready-signalling:
 - a. On the first rising edge the DUT ready signal is evaluated:
 - If ready is '1', the data on the rdata line is returned to the reader in 'data_value'.
 - If ready is '0', the procedure will wait one clock cycle and evaluate the ready signal again. This will repeat until ready is set to '1', or invoke an error if the process has repeated 'config.max_wait_cycles' times. A log message with ID config.id_for_bfm_wait is logged at the first wait.
 2. Without ready-signalling:
 - a. On the first rising edge the data on the rdata line is returned to the reader in 'data_value'.
 3. After 'config.clock_period/4' cs and rena are set to '0' again
- The default value of scope is C_SCOPE ("SBI BFM")
 - The default value of msg_id_panel is shared_msg_id_panel, defined in UVVM_Util.
 - The default value of config is C_SBI_BFM_CONFIG_DEFAULT, see table on the first page.
 - The default value of proc_name is "sbi_read". This argument is intended to be used internally, when procedure is called by sbi_check() or sbi_poll_until().
 - A log message is written if 'config.id_for_bfm' ID is enabled for the specified message ID panel. This will only occur if the argument proc_name is left unchanged.

The procedure reports an alert if:

- ready signal is not set to '1' within 'config.max_wait_cycles' after cs and wena are set to '1' (alert_level: 'config.max_wait_cycles_severity')

Examples:

```
sbi_read(x"1000", v_data_out, "Read from Peripheral 1", clk, sbi_if);
sbi_read(x"1000", v_data_out, "Read from Peripheral 1", clk, sbi_if,, C_SCOPE, shared_msg_id_panel, C_SBI_BFM_CONFIG_DEFAULT);
```

Suggested usage (requires local overload, see section 5):

```
sbi_read(C_ADDR_UART_BAUD, v_data_out, "Read UART baud rate");
```

Note: Record sbi_if can be replaced with the signals cs, addr, rena, wena, ready, rdata.

sbi_check()

sbi_check(addr_value, data_exp, msg, clk, sbi_if, [alert_level, [scope, [msg_id_panel, [config]]]])

The sbi_check() procedure reads data from the DUT at the given address, using the SBI protocol described under sbi_read(). After reading data from the SBI bus, the read data is compared with the expected data, 'data_exp'.

- The default value of alert_level is ERROR
- The default value of scope is C_SCOPE ("SBI BFM")
- The default value of msg_id_panel is shared_msg_id_panel, defined in UVVM_Util.
- The default value of config is C_SBI_BFM_CONFIG_DEFAULT, see table on the first page.
- If the check was successful, and the read data matches the expected data, a log message is written with ID 'config.id_for_bfm' (if this ID has been enabled).
- If the read data did not match the expected data, an alert with severity 'alert_level' will be reported.

The procedure will also report alerts for the same conditions as the sbi_read() procedure.

Examples:

```
sbi_check(x"1155", x"3B", "Check data from Peripheral 1", clk, sbi_if);
sbi_check(x"1155", x"3B", "Check data from Peripheral 1", clk, sbi_if, ERROR, C_SCOPE, shared_msg_id_panel,
C_SBI_BFM_CONFIG_DEFAULT);
```

Suggested usage (requires local overload, see section 5):

```
sbi_check(C_ADDR_UART_RX, x"3B", "Check data from UART RX buffer");
```

Note: Record sbi_if can be replaced with the signals cs, addr, rena, wena, ready, rdata.

sbi_poll_until()

sbi_poll_until(addr_value, data_exp, max_polls, timeout, msg, clk, sbi_if, terminate_loop, [alert_level, [scope, [msg_id_panel, [config]]]])

The `sbi_poll_until()` procedure reads data from the DUT at the given address, using the SBI protocol described under `sbi_read()`. After reading data from the DUT, the read data is compared with the expected data, 'data_exp'. If the read data does not match the expected data, the process is repeated until one or more of the following occurs:

1. The read data matches the expected data, 'data_exp'
2. The number of read retries is equal to 'max_polls'
3. The time between start of `sbi_poll_until` procedure and now is greater than 'timeout'
4. 'terminate_loop' signal is set to '1'

If the procedure exits because of 2. or 3. an alert with severity 'alert_level' is issued. If either 'max_polls' or 'timeout' is set to 0 (ns), this constraint will be ignored and interpreted as no limit.

- The default value of alert_level is ERROR
- The default value of scope is C_SCOPE ("SBI BFM")
- The default value of msg_id_panel is shared_msg_id_panel, defined in UVVM_Util.
- The default value of config is C_SBI_BFM_CONFIG_DEFAULT, see table on the first page.
- If the check was successful, and the read data matches the expected data, a log message is written with ID 'config.id_for_bfm' (if this ID has been enabled).
- If the procedure is terminated using 'terminate_loop' a log message with ID ID_TERMINATE_CMD will be issued.
- If the read data did not match the expected data, an alert with severity 'alert_level' will be reported.

The procedure will also report alerts for the same conditions as the `sbi_read()` procedure.

Examples:

```
sbi_poll_until(x"1155", x"0D", 10, 100 ns, "Poll for data from Peripheral 1", clk, sbi_if, terminate_loop);
sbi_poll_until(x"1155", x"0D", 10, 100 ns, "Poll for data from Peripheral 1", clk, sbi_if, terminate_loop, ERROR, C_SCOPE,
shared_msg_id_panel, C_SBI_BFM_CONFIG_DEFAULT);
```

Suggested usage (requires local overload, see section 5):

```
sbi_poll_until(C_ADDR_UART_RX, x"0D", "Poll UART RX buffer until CR is found");
sbi_poll_until(C_ADDR_UART_RX, x"0D", C_MAX_POLLS, C_TIMEOUT, "Poll UART RX buffer until CR is found");
```

Note: Record `sbi_if` can be replaced with the signals `cs`, `addr`, `rena`, `wena`, `ready`, `rdata`.

init_sbi_if_signals()

init_sbi_if_signals(addr_width, data_width)

This function initializes the SBI interface. All the BFM outputs are set to zeros ('0'), and BFM inputs are set to 'Z'.

Example:

```
sbi_if <= init_sbi_if_signals(addr_width, data_width)
```

2 BFM Configuration record

Type name: t_sbi_bfm_config

| Record element | Type | C_SBI_BFM_CONFIG_DEFAULT | Description |
|----------------------------|---------------|--------------------------|--|
| max_wait_cycles | integer | 10 | The maximum number of clock cycles to wait for the DUT ready signal before reporting a timeout alert. |
| max_wait_cycles_severity | t_alert_level | failure | The above timeout will have this severity |
| use_fixed_wait_cycles_read | boolean | false | When true, wait 'fixed_wait_cycles_read' after asserting 'rena' signal, before sampling 'rdata' from DUT |
| fixed_wait_cycles_read | natural | 0 | Number of clock cycles to wait after asserting 'rena' signal, before sampling 'rdata' from DUT. |
| clock_period | time | 10 ns | Period of the clock signal. |
| clock_period_margin | time | 0 ns | Input clock period margin to specified clock_period. Will check T/2 if input clock is low when BFM is called and T if input clock is high |
| clock_margin_severity | t_alert_level | TB_ERROR | The above margin will have this severity |
| setup_time | time | 2.5 ns | Generated signals setup time. Suggested value is clock_period/2. An alert is reported if setup_time exceed clock_period/2. |
| hold_time | time | 2.5 ns | Generated signals hold time. Suggested value is clock_period/2. An alert is reported if hold_time exceed clock_period/2. |
| id_for_bfm | t_msg_id | ID_BFM | The message ID used as a general message ID in the SBI BFM |
| id_for_bfm_wait | t_msg_id | ID_BFM_WAIT | The message ID used for logging waits in the SBI BFM |
| id_for_bfm_poll | t_msg_id | ID_BFM_POLL | The message ID used for logging polling in the SBI BFM |
| use_ready_signal | boolean | true | Whether or not to use the interface 'ready' signal |
| setup_time | time | 0 ns | Generated signals setup time. |
| hold_time | time | 0 ns | Generated signals hold time. |

3 Additional Documentation

The SBI BFM is used in the IRQC example provided with the UVVM Utility Library (available from Bitvis.no Downloads page). Thus you can find info under:

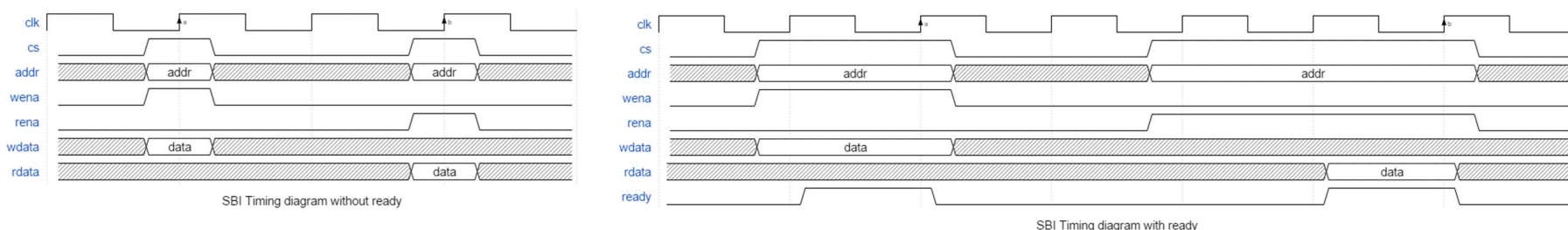
- 'Making a simple, structured and efficient VHDL testbench – Step-by-step' (PPT)

There is also a webinar available on 'Making a simple, structured and efficient VHDL testbench – Step-by-step' (via Aldec). Link is available on our Downloads page.

3.1 SBI protocol

SBI is our name for the simplest bus interface possible, one that has been used for decades in the electronics industry. Some think of it as a simple SRAM interface, but that is not a standard, and is probably understood and used in many different ways. Thus we have defined a name and an exact behaviour, with some flexibility.

SBI is a single cycle bus with an optional ready-signalling. The protocol for SBI with and without ready-signalling is given below. Data is sampled on rising edge **a** and **b**.



As can be seen from the figure all required signals including data input must be ready on the rising edge of the clock. This also applies for a read access, but the actual data output is provided combinatorial as soon as the combinational logic allows

Note that an active 'cs', a valid 'addr' and an active 'wena' or 'rena' is needed on the same active clock edge to be registered as a valid read or write. (Being active on two consecutive rising clocks will result in two consecutive accesses - with or without side-effects depending on the module's internal functional logic.) 'rdata' will just ripple out for the right combination of 'cs', 'addr' and 'rena'.

With this simple version, the designer has the option to provide input and/or output registers externally to allow a higher frequency (with added latency).

SBI has optional ready-signalling. When 'ready' is used it applies to both read and write accesses. For both read and write accesses all input signals must be held until 'ready' is active. For a read access, the output data may not be used (sampled) until 'ready' is active, but must do so on the first rising edge of the clock after 'ready' active.

4 Compilation

The SBI BFM may only be compiled with VHDL 2008. It is dependent on the UVVM Utility Library (UVVM-Util), which is only compatible with VHDL 2008.

See the separate UVVM-Util documentation for more info. After UVVM-Util has been compiled, the sbi_bfm_pkg.vhd BFM can be compiled into any desired library.

4.1 Simulator compatibility and setup

This BFM has been compiled and tested with Modelsim version 10.3d and Riviera-PRO version 2015.10.85.

For required simulator setup see UVVM-Util Quick reference.

5 Local BFM overloads

A good approach for better readability and maintainability is to make simple, local overloads for the BFM procedures in the TB process. This allows calling the BFM procedures with the key parameters only

e.g.

```
sbi_write(C_ADDR_UART_BAUDRATE, C_BAUDRATE_9600, "Set Baudrate to 9600");
```

rather than

```
sbi_write(C_ADDR_UART_BAUDRATE, C_BAUDRATE_9600, "Set Baudrate to 9600", clk, sbi_if,  
          C_CLK_PERIOD, C_SCOPE, shared_msg_id_panel, C_SBI_CONFIG_DEFAULT);
```

By defining the local overload as e.g.:

```
procedure sbi_write(  
  constant addr_value : in unsigned;  
  constant data_value : in std_logic_vector;  
  constant msg        : in string) is  
begin  
  sbi_write(addr_value,          -- keep as is  
            data_value,         -- keep as is  
            msg,                -- keep as is  
            sbi_if,             -- Signal must be visible in local process scope  
            C_CLK_PERIOD,       -- Just use the default  
            C_SCOPE,           -- Just use the default  
            shared_msg_id_panel, -- Use global, shared msg_id_panel  
            C_SBI_CONFIG_LOCAL); -- Use locally defined configuration or C_SBI_CONFIG_DEFAULT  
end;
```

Using a local overload like this also allows the following – if wanted:

- Have address value as natural – and convert in the overload
- Set up defaults for constants. May be different for two overloads of the same BFM
- Apply dedicated message ID panel to allow dedicated verbosity control

IMPORTANT

This is a simplified Bus Functional Model (BFM) for SBI.

The given BFM complies with the basic SBI protocol and thus allows a normal access towards a SBI interface. This BFM is not a SBI protocol checker.

For a more advanced BFM please contact Bitvis AS at support@bitvis.no

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.