Error Injection VIP - Quick Reference

UVVM Verification IP

The Error Injection VIP consists of two VHDL entities - a std_logic entity and a std_logic_vector entity, and is used for inducing non protocol-dependent errors on single and vector signals.
See full introduction to functionality on page 2.

Error Injection – Generics

Generic element	Type	Default	Description
GC_START_TIME	time	0 ns	The error injector will wait the specified time after initialization before starting to monitor signal events, and injecting error if configured to. Note that default is 0 ns, i.e. not waiting after initialization.
GC_INSTANCE_IDX	natural	1	Error injection instance number.

Error Injection – Configuration record ´**t_error_injection_config**´

Record element	Type	Example	C_EI_CONFIG_DEFAULT	Description
error_type	t_error_injection	JITTER	BYPASS	Type of error to be injected to signal. No error is injected when error_type is set to BYPASS. [1]
initial_delay_min	time	5 ns	0 ns	Error injection start relative to initial signal event. [1]
initial_delay_max	time	0 ns	0 ns	Setting the max parameter will generate a randomized initial timing parameter.[2]
return_delay_min	time	3 ns	0 ns	Error injection end relative to next signal event. [1, 4]
return_delay_max	time	0 ns	0 ns	Setting the max parameter will generate a randomized return timing parameter.[2, 4]
width_min	time	0 ns	0 ns	The width of an error injected pulse if error type PULSE is selected.
width_max	time	0 ns	0 ns	Setting the max parameter will generate a randomized timing parameter.[2]
interval	positive	3	1	Errors will be injected in this interval, e.g. 1 for every signal event, 2 for every second etc.[3, 5] Interval = 1: SL will experience error injection on every second signal event, e.g. every rising edge. SLV will experience error injection on every signal event. Interval = 2: SL will experience error injection on every fourth signal event, e.g. every second rising edge. SLV will experience error injection on every second signal event.
base_value	std_logic	'1'	'0'	The initial edge in an SL error injection will start on the transition from `base_value` to another value, e.g. from '0' to '1'. The return edge in an SL error injection will be the transition back to `base_value`. [6] Note that setting base_value = '-'will start SL error injection on first upcoming edge after configuration, regardless of input value
randomization_seed1	positive	12	1	Global seed 1 for randomized timing parameters.
randomization_seed2	positive	14	2	Global seed 2 for randomized timing parameters.

Note 1: See Error injection types parameters on page 2 for required configuration parameters for each error type.
Note 2: Randomization is selected by setting the max parameter higher than the min parameter and will be in the range of min parameter to max parameter.
Note 3: SL signal is handled with respect to a start and an end transition, and an error injection is activated at the start transition.
vector signal is handled with respect to a start transition, and the following transition will be treated as a new start transition.
Note 4: Vector signal does not have a return_delay_min/max parameter [3]
Note 5: Error injection interval will always start injection on the first signal event for any interval setting, followed by the configured interval, e.g. every second for interval = 2.
Note 6: For initial and return edge definitions, see BYPASS example on page 3.

VHDL 2008 only

Error Injection Types Parameters – Configure error injection type ´**t_error_injection_types**´

Parameters marked with **R** is required, **O** is optional, and **X** is ignored.

Error	Intial_delay_min	Intial_delay_max	Return_delay_min	Return_delay_max	Width_min	Width_max
BYPASS	X	X	X	X	X	X
PULSE	**R**	**O**	X	X	**R**	**O**
DELAY	**R**	**O**	X	X	X	X
JITTER	**R**	**O**	**R**	**O**	X	X
INVERT	X	X	X	X	X	X
STUCK_AT_OLD	X	X	X	X	**R**	**O**
STUCK_AT_NEW	X	X	X	X	**R**	**O**

Note that a randomized error injection is selected by setting optional parameter "_max" ≠ 0 ns, and by setting "_max" > "_min". The randomisation interval will be in the range of required parameter time to optional parameter time.

Error injection functional details

Error	Std_logic	Std_logic_vector
BYPASS	Signal is preserved and no error is injected.	Signal is preserved and no error is injected.
PULSE	Signal will experience a pulse of width **`width`** after a time of **`initial_delay`**. [1] Note that the pulse value will be the value prior to initial event.	As for std_logic. Note that the pulse value will be the value prior to initial event.
DELAY	Signal will be skewed for a time of **`initial_delay`** on initial edge and on return edge. [1] Any signal activity while error injection is active will override the error injection.	As for std_logic. .
JITTER	As for DELAY, but with different values on edge delays. Note that only positive jitter is supported, and that reordering of positive and negative edges yield an invalid configuration and will not work.	Not applicable for vector signals.
INVERT	The signal is inverted as long as the error injection configuration is set to INVERT. Note that when error injection configuration is changed from INVERT, a new signal event is required for the new configuration to take effect.	As for std_logic.
STUCK_AT_OLD	The signal will be stuck at the value it had prior to initial signal event, and released to the current signal value after the specified **`width_min`** time. [1] Note that any signal event during STUCK_AT_OLD is ignored.	As for std_logic.
STUCK_AT_NEW	The signal will be stuck at the value it had prior to initial signal event, and released to the current signal value after a specified **`width_min`** time. [1] Note that any signal event during STUCK_AT_NEW is ignored.	As for std_logic

Note 1: initial event is when a SL signal is at `base_value` (see Configuration Record details on page 1) and changes signal level. The return event is when a SL signal is returning to its `base_value`. Note that vector signals only have initial events. See Figure in BYPASS example on page 3.

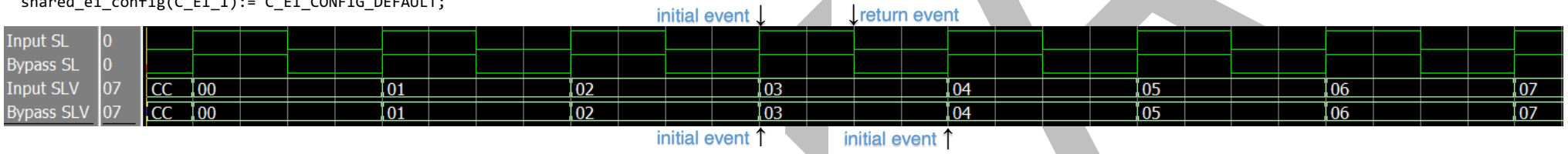
Error Injection examples for std_logic and std_logic_vector signals

The section “Error injection type parameters” on page 2 show how error types are configured. Below follows a series of examples showing error configuration of a std_logic and a std_logic_vector type signal, respectively.

Note in these examples. the std_logic_signal is active high for 20 ns and active low for 20 ns, while the std_logic_vector signal is updated to a new value every 40 ns.
Note that `JITTER` is an invalid error injection type for vector signals and that the signal therefore is not error injected.

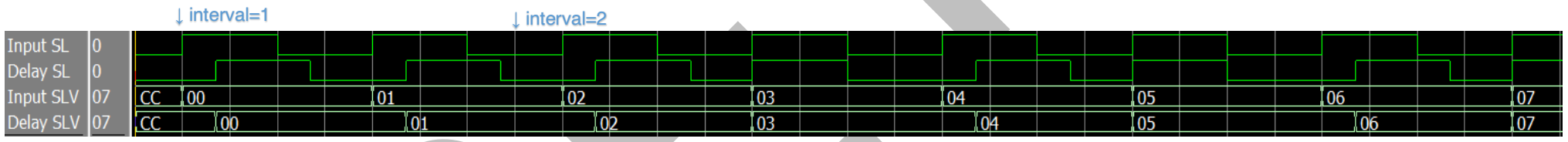
BYPASS

```
shared_ei_config(C_EI_1):= C_EI_CONFIG_DEFAULT;
```



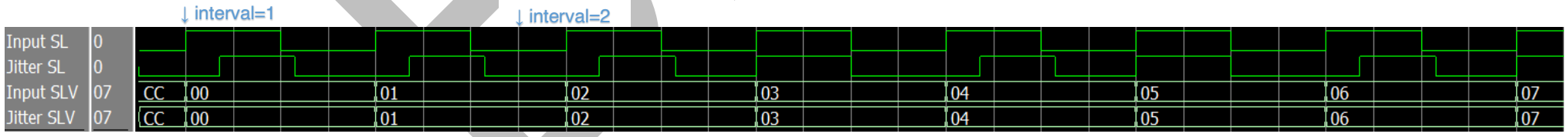
DELAY

```
shared_ei_config(C_EI_1).error_type        := DELAY;
shared_ei_config(C_EI_1).initial_delay_min := 7 ns;
```



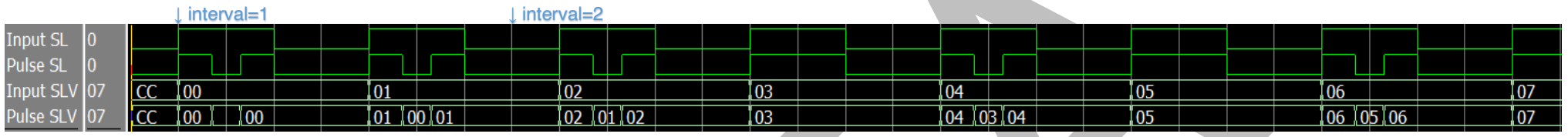
JITTER

```
shared_ei_config(C_EI_1).error_type        := JITTER;
shared_ei_config(C_EI_1).initial_delay_min := 7 ns;
shared_ei_config(C_EI_1).return_delay      := 3 ns;
```



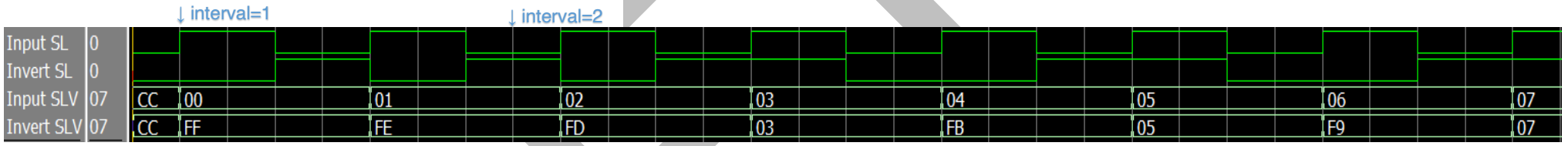
PULSE

```
shared_ei_config(C_EI_1).error_type        := PULSE;
shared_ei_config(C_EI_1).initial_delay_min := 7 ns;
shared_ei_config(C_EI_1).width_min         := 6 ns;
```



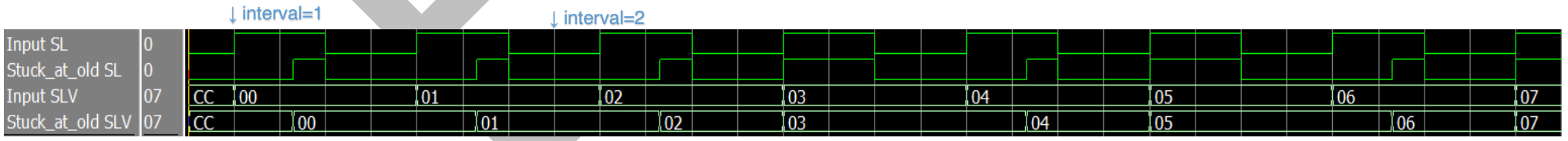
INVERT

```
shared_ei_config(C_EI_1).error_type := INVERT;
```



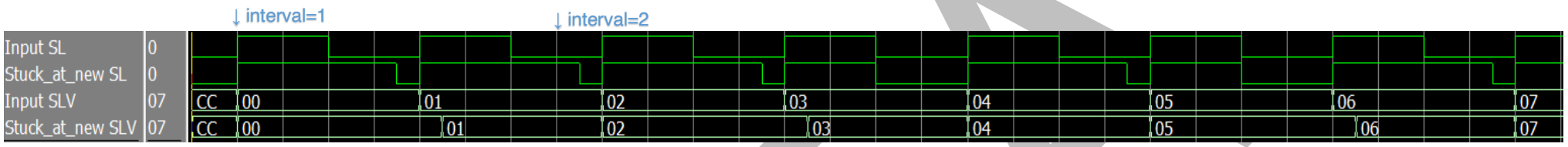
STUCK_AT_OLD

```
shared_ei_config(C_EI_1).error_type := STUCK_AT_OLD;
shared_ei_config(C_EI_1).width_min  := 13 ns;
```



STUCK_AT_NEW

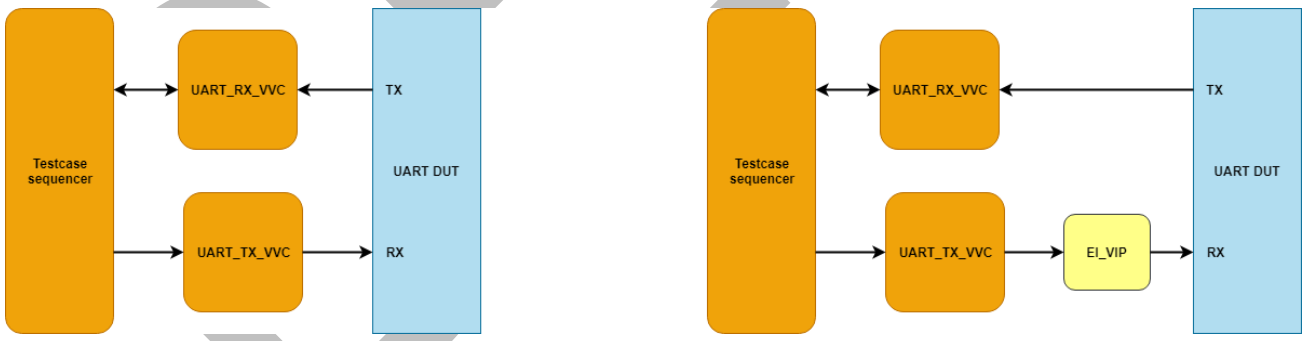
```
shared_ei_config(C_EI_1).error_type := STUCK_AT_NEW;
shared_ei_config(C_EI_1).width_min  := 35 ns; -- SL
shared_ei_config(C_EI_1).width_min  := 45 ns; -- SLV
```



Error Injection VIP details

The error injection VIP can be used as a single instance or with multiple instances and can be used with std_logic or vector signals. [1] A generic GC_WIDTH is needed for vector signals to constrain the vector width in the error injector port map. The default error injector behaviour is in `BYPASS` mode, i.e. without any error injected to the signal. The error injection is controlled using a dedicated shared variable array, `shared_ei_config(index)`, of type t_error_injection_config, where each error injector has its own dedicated array index. Error injection is started by setting the error injection VIPs index in the `shared_ei_config` array as listed in the Error Injection configuration record table on page 1 and Error Injection Types Parameters on page 2. The error injection is disabled and set to bypass mode by setting the configuration to `C_EI_CONFIG_DEFAULT`.

The provided example below shows how a typical UART error injection setup could be done, using an `error_injection_sl` entity for the UART RX signal line.



Note 1: Std_logic signals require the `error_injection_sl` entity, while vector signals require the `error_injection_slv` entity.

1 Example usage

The error injection VIP comes with two entities, a std_logic version for single signals and a std_logic_vector version for vector signals. Setting up an error injector VIP is fast and only has a few mandatory steps:

- Include the error injection package.
- Define the needed error injection configuration signal(s).
- Instantiate the error injector(s) needed.
- Set the error injection configurations.

In testbench:

```
library bitvis_vip_error_injection;
use bitvis_vip_error_injection.error_injection_pkg.all;

...
constant C_UART_TX_EI : natural := 1;
...

Uart_tx_error_injector: entity work.error_injection_sl
  generic map (
    GC_START_TIME    => 10 ns,
    GC_INSTANCE_IDX  => C_UART_TX_EI
  )
  port map (
    ei_in   => uart_tx,
    ei_out  => ei_uart_tx
  );

....
uart_transmit(UART_VVCT, 1, TX, x"AA", "Transmitting data");

shared_ei_config(C_UART_TX_EI).error_type := DELAY;
shared_ei_config(C_UART_TX_EI).delay_min  := 2 ns;
shared_ei_config(C_UART_TX_EI).base_value := '0';
wait for 10 ns;
shared_ei_config(C_UART_TX_EI) := C_EI_CONFIG_DEFAULT;
```

Include the Bitvis VIP Error Injection package

Define the Error Injection instance index.

Instantiate the Error Injector VIP(s).
Note that component instances normally would be located in the test harness and not in the testbench.

Use the UART VVC to send data to DUT.

Configure the Error Injector VIP using the shared_ei_config from the error_injection_pkg.

Set the Error Injection config back to signal BYPASS (default) when done.

For more detailed examples see VHDL example `ei_demo_tb.vhd`, located in the `tb` folder.

2 Additional documentation

Additional documentation about UVVM and its features can be found under “/uvvm_vvc_framework/doc/”.

3 Compilation

The Error Injection VIP must be compiled with VHDL 2008.
It is dependent on the following libraries

- ***UVVM Utility Library (UVVM-Util), version 2.10.0 and up***

Compile order for the Error Injection VIP:

Compile to library	File	Comment
bitvis_vip_error_injection	error_injection_pkg.	Configuration declaration.
bitvis_vip_error_injection	error_injection_slv	Vector entity and error injection functionality.
bitvis_vip_error_injection	error_injection_sl	Std_logic entity, only needed when using error injection with std_logic signals.

4 Simulator compatibility and setup

See README.md for a list of supported simulators.
For required simulator setup see UVVM-Util Quick reference.