1 Theory of Operation

The csco_clkgen package is a verification component designed for emulating the function of a digital clock oscillator. This package contains components written in both *e* and HDL. The user configuration interface is written in *e* to take advantage of its random generation facility, while the actual clock implementation is written in HDL for efficiency.

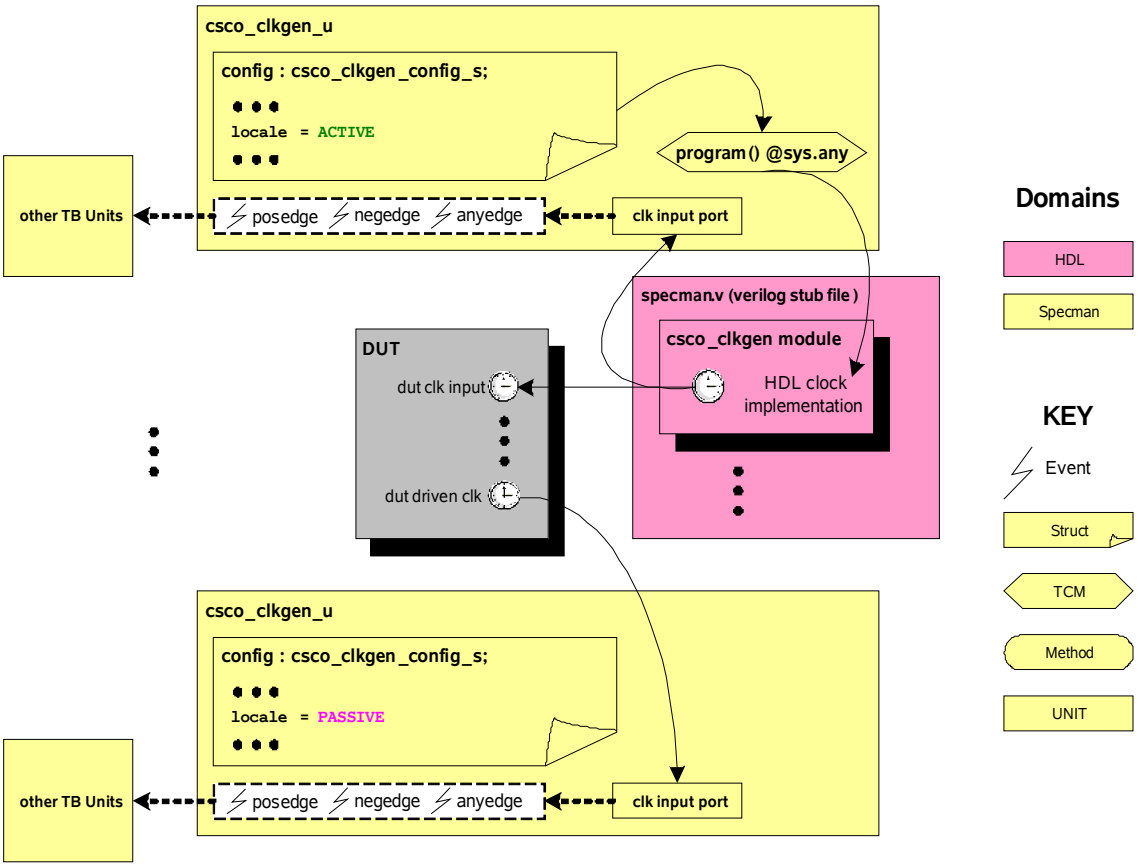
- **HDL Clock Generator module** – This HDL module is actually responsible for generating all the clock signals the package provides. This part is written in HDL to remain as efficient as possible.
- ***'e'* control logic for HDL Clock Generator Module** – This piece controls all the parameters of the HDL model as well as other aspects such as distributing clocks in the Specman realm.

This package can generate different clock configurations ranging from fully directed to fully random. This package is designed to provide clock events to both the Specman and HDL environments. All efforts were made to ensure this package complies with the spirit and intent of the *e* Re-use Methodology (*e*RM).

Each csco_clkgen_u unit instance can operate in either the `ACTIVE` or the `PASSIVE` mode:

- **`ACTIVE` mode** – The unit provides a clock to both the HDL and the *e* environment. The unit will instantiate the clock in the Specman verilog stub file, and will connect it to both the HDL and the *e* domains. The user may configure clock attributes such as duty-cycle and skew via the *e* configuration interface of the unit.
- **`PASSIVE` mode** – The unit provides a clock to the *e* environment by sampling a DUT driven clock in the HDL domain.

Figure 1: eVC Architecture



2 Quick Setup

2.1 Sampling Clock from DUT

Sometimes, especially when moving between a unit-level testbench and a higher level testbench, the clock is driven by the DUT itself, but we would still like to use it as the sampling event for TCMs in the *e* environment. In this case, the user should instantiate a PASSIVE instance of the csco_clkgen_u unit. On the *e* side, the instantiation would be done as follows:

```
extend csco_clkgen_name_t : [ MY_PCI ];

extend csco_pci_env_u {
  pci_clk : MY_PCI csco_clkgen_u is instance;
    keep pci_clk.sig_clock.hdl_path() == "chip.pci_core.clk";

  a_tcm_method() @pci_clk.posedge is {
    …
  };
};

// remember to create the parallel config instance
INSTANCE_CFG begin
  config_instance_name = pci_clk,
  child_config_type    = MY_PCI csco_clkgen_config_s,
  parent_config_type   = csco_pci_config_s,
  parent_unit_type     = csco_pci_env_u,
  child_unit_path      = pci_clk
end;
```

Note that any TCM declared at the unit-level will not need to change when we move up the testbench hierarchy since the clock interface still looks the same.

2.2 Generating Clock to Drive DUT

To generate a clock that will be used by both HDL logic and e units, the user should switch the csco_clkgen_u unit into ACTIVE mode.

```
extend PCI_CLK csco_clkgen_config_s {
  keep soft active_passive == ACTIVE;
};
```

On the HDL side, the generated clock will automatically be injected to the DUT at the specified hdl_path() of sig_clock. The verilog target type defaults to a reg, which is most efficient, but if the target signal is a verilog wire (eg, internal to the DUT) the driver type can be reconfigured:

```
extend PCI_CLK csco_clkgen_config_s {
  keep soft auto_drive_type == WIRE;
};
```

For ACTIVE csco_clkgen_u instances, the user may change a variety of attributes associated with the clock via the configuration interface:

```
extend PCI_CLK csco_clkgen_config_s {
  keep period == 10_000;   // assuming DUT timescale is 1ns
  keep duty_cycle == 55;
  keep skew == 1;
};
```

3 User Interface

Structs

Structs	Description
csco_clkgen_config_s	clock configuration struct
csco_clkgen_u	clock generator unit

3.1.1 csco_clkgen_config_s like csco_config_base_struct_s (top)

Fields of csco_clkgen_config_s

Field	Description
active_passive: erm_active_passive_t	ACTIVE driver or PASSIVE monitor:ACTIVE mode generates clocks using HDLmodel PASSIVE mode expects some HDL source (DUT) to provide a clock which this model samples and distributes to its clients
auto_drive_type: csco_clkgen_hdl_type_t	target verilog signal type to drive
auto_instance: bool	control whether HDLclock driver module is automatically instanced in specman stubs
auto_model: csco_clkgen_model_t	model to instance
duty_cycle: uint	percentage of period that clock will be high
enable: bool	model can be enabled/disabled and re-programmed to start/stop clock
enable_anyedge: bool	enable event at both edges of clock
enable_negedge: bool	enable event at falling edge of clock
enable_posedge: bool	enable event at rising edge of clock
inv_phase: bool	clock "phase" FALSE => start clk at 1'b0, first edge is rising after "skew" delayTRUE => start clk at 1'b1, first edge is falling after "skew" delay
jitter: uint	here we define "jitter" to be the size of the window in which the clock is allowed to transition this window will _start_ from "skew" so if you need it centered you better do a "skew" -= "jitter" / 2
name: csco_clkgen_name_t	Name of the clock type, usually refers to a clock domain
period: uint	all clk params in units of 1/1000 of timescale (ps?)
skew: uint	delay from programming time to first edge transition

Events of csco_clkgen_config_s

Event	Description
generated	configuration coverage

3.1.2 csco_clkgen_u like any_env (top)

Fields of csco_clkgen_u

Field	Description
config: csco_clkgen_config_s	clock configuration struct
name: csco_clkgen_name_t	clock type name for clkgen instance
sig_clock: in simple_port of bit	clock signal to sample or drive

Methods of csco_clkgen_u

Method	Description
set_enable(new_state: bool)	method interface to update enabled state and reprogram the verilog HDL driver. ignored for PASSIVE clock monitors. Useful for OIR.

Events of csco_clkgen_u

Event	Description
anyedge	both positive and negative edge of generated clock
negedge	positive edge of generated clock
posedge	positive edge of generated clock

3.1.3 Types (top)

Types	Description
csco_clkgen_hdl_type_t: [REG, WIRE]	HDL signal type to drive
csco_clkgen_model_t: [SIMPLE, SHAGGY]	type of model to auto-instance
csco_clkgen_name_t: [DEFAULT_CLK]	clock type name for clkgen unit

4 Application Notes