axi_config_image_frame_sequence

METHODS

new

```
function new(
 string name= "axi_config_image_frame_sequence"
 )
```

SetStatus

```
function void SetStatus(
 input int Y
 )
```

body

```
virtual task body()
```

axi_write_channel

```
virtual task axi_write_channel (
 bit[7:0]  addr,
 bit[31:0] data
 )
```

d5m_camera_agent

uvm_agent

d5m_camera_agent

Heading:This is **bold text**. This is underlined text with spaces.

- Bullet one.
- Bullet two. Bullet two continued.
- Bullet three.

Some text after the bullet list.

- Level one.
- Level two, first paragraph. Level two, first paragraph continued.

  Level two, second paragraph.

  - Level three.

```
int x = 12;
int y = 0;
```

```
my $x = 12;
my $y = 0;
```

```
This is plain text.
```

The scoreboard will receive the transactions from the Monitors implemented and connected inside this agent.

METHODS

new

```
function new (
 string        name,
 uvm_component parent
 )
```

Parameters

name Description of x.
string

parent Description of y.
uvm_component

build_phase

```
function void build_phase(
 uvm_phase phase
 )
```

Use build() method to create agents's subcomponents.

connect_phase

```
function void connect_phase(
 uvm_phase phase
 )
```

Use connect() method to connect the component TLM ports

d5m_camera_base_seq

METHODS

new

```
function new (
  string name= "d5m_camera_base_seq"
  )
```

SetStatus

```
function void SetStatus(
  input int a
  )
```

Set1Status

```
virtual function void Set1Status(
  input int a
  )
```

d5m_camera_configuration

METHODS

new

```
function new(
  string name = ""
  )
```

d5m_camera_driver

This class which extended from uvm driver pull data items generated by a sequencer and drive it to the DUT. In run phase methods are used for read and write operation to dut through dut interface handle.

VARIABLES

d5m_camera_vif

```
protected virtual d5m_camera_if d5m_camera_vif
```

dut interface

METHODS

new

```
function new (
  string        name,
  uvm_component parent
  )
```

build_phase

```
function void build_phase (
  uvm_phase phase
  )
```

In this phase, config the dut interface handle through get method.

run_phase

```
virtual task run_phase (
  uvm_phase phase
  )
```

In this method, fork join constructs are used to separate threads that drive each of the channels.

- reset_signals : this method reset the signals to dut
- d5m_frame : this method drive data to dut and get response from dut

reset_signals

```
virtual protected task reset_signals()
```

Reset the dut axi4 lite and d5m_cam_mod input signals when system reset is asserted from low to high.

d5m_frame

```
virtual protected task d5m_frame()
```

In this method, drive the signals from defined seq in uvm_sequence.

- seq_item_port.get_next_item : Get next item from the sequencer
- drive_transfer : this method drive data to dut and get response from dut
- seq_item_port.item_done : Tell the sequence that driver has finished current item

drive_transfer

```
virtual protected task drive_transfer (
 d5m_camera_transaction d5m_tx
 )
```

This method which is master to dut axi4lite interface read or write data at given address using bus handshaking protocol.First valid address is transmit and then wait for valid response in given time of 61 clock cycles in address_channel method. Timeout accord on 62 clock cycle, if no response is asserted high on bvalid signal from dut and timeout is flagged using uvm_error macro. If valid response is asserted then data_channel method write or read data depending on case statement.

- address_channel : this method drive the axi4lite write/read address channels.
- data_channel : this method read or write the the data at given address.

address_channel

```
virtual protected task address_channel (
 d5m_camera_transaction d5m_tx
 )
```

In this method, drive axi4 read/write addresses | d5m read/write.Note need to rewite this method.

- case : AXI4_WRITE AXI4_READ D5M_WRITE IMAGE_READ
- drive_write_address_channel : axi4 write address
- drive_read_address_channel : axi4 read address
- d5m_data_phase : write data to d5m mod
- read_d5m_phase : read data from d5m mod

data_channel

```
virtual protected task data_channel (
 d5m_camera_transaction d5m_tx
 )
```

In this method, drive axi4 address channel | d5m read/write

- case : AXI4_WRITE AXI4_READ D5M_WRITE IMAGE_READ
- drive_write_data_channel : axi4 write data
- drive_read_data_channel : axi4 read data
- d5m_data_phase : write data to d5m mod
- read_d5m_phase : read data from d5m mod

read_d5m_phase

```
virtual protected task read_d5m_phase(
 d5m_camera_transaction d5m_tx
 )
```

In this method, config test type during read operation and wait for end of frame pulse

- iImageTypeTest : axi4 write address
- iReadyToRead : axi4 read address

- eof : write data to d5m mod

d5m_data_phase

```
virtual protected task d5m_data_phase (
 d5m_camera_transaction d5m_tx
 )
```

In this method, write data to d5m camera mod

- case : AXI4_WRITE AXI4_READ D5M_WRITE IMAGE_READ
- drive_write_address_channel : axi4 write address
- drive_read_address_channel : axi4 read address
- d5m_data_phase : write data to d5m mod
- read_d5m_phase : read data from d5m mod

d5m_write_idata

```
virtual protected task d5m_write_idata (
        bit[23:0] iRgb,
        bit       ilvalid,
        bit       ifvalid,
        bit       iImageTypeTest,
 output bit       error
 )
```

In this method, drive axi4 address channel | d5m read/write

- iReadyToRead : axi4 write address
- iImageTypeTest : axi4 read address
- rgb : write data to d5m mod
- fvalid : read data from d5m mod
- lvalid : read data from d5m mod

drive_write_address_channel

```
virtual protected task drive_write_address_channel (
 d5m_camera_transaction d5m_tx
 )
```

In this method, axi4lite write address

- drive_write_address_channel : axi4 write address
- drive_read_address_channel : axi4 read address
- d5m_data_phase : write data to d5m mod
- read_d5m_phase : read data from d5m mod

drive_write_data_channel

```
virtual protected task drive_write_data_channel (
 d5m_camera_transaction d5m_tx
 )
```

In this method, axi4lite write data

- WDATA : axi4 write address
- WVALID : axi4 read address
- WREADY : write data to d5m mod
- AWVALID : read data from d5m mod

drive_read_address_channel

```
virtual protected task drive_read_address_channel (
 d5m_camera_transaction d5m_tx
 )
```

In this method, axi4lite write read address

- ARADDR : axi4 write address
- ARVALID : axi4 read address
- ARREADY : write data to d5m mod
- ARPROT : read data from d5m mod

drive_read_data_channel

```
virtual protected task drive_read_data_channel (
 output bit [31:0] data,
     output bit    error
 )
```

In this method, axi4lite read data

- RDATA : axi4 write address
- RVALID : axi4 read address
- RREADY : write data to d5m mod
- RRESP : read data from d5m mod

d5m_camera_env

uvm_env

d5m_camera_env

This Class,

VARIABLES

aL_agt

```
d5m_camera_agent aL_agt
```

this method reset the signals to dut

aL_fc_sub

```
d5m_camera_fc_subscriber aL_fc_sub
```

this method reset the signals to dut

aL_sb

```
d5m_scoreboard aL_sb
```

this method reset the signals to dut

METHODS

new

```
function new(
  string        name,
  uvm_component parent
  )
```

build_phase

```
function void build_phase(
  uvm_phase phase
  )
```

connect_phase

```
function void connect_phase(
  uvm_phase phase
  )
```

d5m_camera_fc_subscriber

```
#(d5m_camera_transaction)
uvm_subscriber
```

d5m_camera_fc_subscriber

This class provides an analysis export for receiving transactions from a connected analysis export. Making such a connection this component to any transactions emitted by the connected analysis port

METHODS

new

```
function new(
  string        name,
  uvm_component parent
  )
```

write

```
function void write(
  d5m_camera_transaction t
  )
```

The d5m_camera_transaction from the monitor is sampled by write function.

d5m_camera_image_cgain_hsl_sequence

METHODS

new

```
function new(
  string name = "d5m_camera_image_cgain_hsl_sequence"
  )
```

d5m_camera_image_file_blur_test

uvm_test

d5m_camera_image_file_blur_test

d5m_camera_image_file_cgain_cgain_test

uvm_test

d5m_camera_image_file_cgain_cgain_test

d5m_camera_image_file_cgain_hsl_test

uvm_test

d5m_camera_image_file_cgain_hsl_test

d5m_camera_image_file_cgain_sharp_test

uvm_test

d5m_camera_image_file_cgain_sharp_test

d5m_camera_image_file_cgain_test

uvm_test

d5m_camera_image_file_cgain_test

d5m_camera_image_file_emboss_test

uvm_test

d5m_camera_image_file_emboss_test

d5m_camera_image_file_hsl_test

uvm_test

d5m_camera_image_file_hsl_test

d5m_camera_image_file_hsv_test

uvm_test

d5m_camera_image_file_hsv_test

d5m_camera_image_file_rgb_test

uvm_test

d5m_camera_image_file_rgb_test

d5m_camera_image_file_sharp_cgain_test

uvm_test

d5m_camera_image_file_sharp_cgain_test

d5m_camera_image_file_sharp_test

uvm_test

d5m_camera_image_file_sharp_test

METHODS

new

```
function new(
 string        name,
 uvm_component parent
 )
```

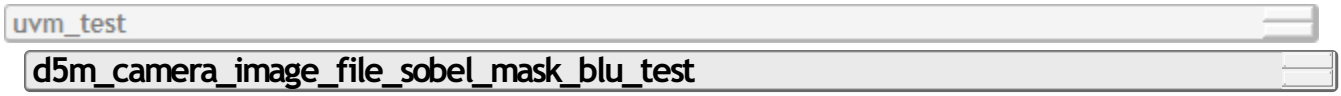
build_phase

```
function void build_phase(
 uvm_phase phase
 )
```

end_of_elaboration_phase

```
function void end_of_elaboration_phase(
 uvm_phase phase
 )
```

d5m_camera_image_file_sobel_mask_blu_test



d5m_camera_image_file_sobel_mask_cga_test

uvm_test

d5m_camera_image_file_sobel_mask_cga_test

d5m_camera_image_file_sobel_mask_hsl_test

uvm_test

d5m_camera_image_file_sobel_mask_hsl_test

d5m_camera_image_file_sobel_mask_hsv_test

uvm_test

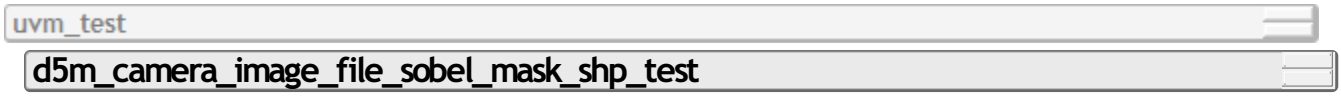
d5m_camera_image_file_sobel_mask_hsv_test

d5m_camera_image_file_sobel_mask_rgb_test

uvm_test

d5m_camera_image_file_sobel_mask_rgb_test

d5m_camera_image_file_sobel_mask_shp_test



d5m_camera_image_file_sobel_test

uvm_test

d5m_camera_image_file_sobel_test

d5m_camera_image_file_test

uvm_test

d5m_camera_image_file_test

METHODS

new

```
function new(
  string        name,
  uvm_component parent
  )
```

build_phase

```
function void build_phase(
  uvm_phase phase
  )
```

end_of_elaboration_phase

```
function void end_of_elaboration_phase(
  uvm_phase phase
  )
```

run_phase

```
task run_phase(
  uvm_phase phase
  )
```

d5m_camera_image_pattern_test

uvm_test

d5m_camera_image_pattern_test

METHODS

new

```
function new(
  string        name,
  uvm_component parent
  )
```

build_phase

```
function void build_phase(
  uvm_phase phase
  )
```

end_of_elaboration_phase

```
function void end_of_elaboration_phase(
  uvm_phase phase
  )
```

run_phase

```
task run_phase(
  uvm_phase phase
  )
```

d5m_camera_monitor

uvm_monitor

d5m_camera_monitor

METHODS

new

```
function new (
  string        name,
  uvm_component parent
  )
```

build_phase

```
function void build_phase (
  uvm_phase phase
  )
```

run_phase

```
virtual task run_phase (
  uvm_phase phase
  )
```

collect_transactions

```
virtual protected task collect_transactions()
```

d5m_camera_sequence

METHODS

new

```
function new(
  string name= "d5m_camera_sequence"
  )
```

d5m_camera_sequencer

METHODS

new

```
function new (
  string        name,
  uvm_component parent
  )
```

d5m_camera_test

uvm_test

d5m_camera_test

METHODS

new

```
function new(
  string        name,
  uvm_component parent
  )
```

build_phase

```
function void build_phase(
  uvm_phase phase
  )
```

end_of_elaboration_phase

```
function void end_of_elaboration_phase(
  uvm_phase phase
  )
```

run_phase

```
task run_phase(
  uvm_phase phase
  )
```

d5m_rgb_patten_test

uvm_test

d5m_rgb_patten_test

METHODS

new

```
function new(
  string        name,
  uvm_component parent
  )
```

build_phase

```
function void build_phase(
  uvm_phase phase
  )
```

end_of_elaboration_phase

```
function void end_of_elaboration_phase(
  uvm_phase phase
  )
```

run_phase

```
task run_phase(
  uvm_phase phase
  )
```

d5m_scoreboard

uvm_scoreboard

d5m_scoreboard

d5m_scoreboard This class compare the dut output values with prediction or golden ref values. Using uvm fifo sychronize write transaction from dut and predict monitor. Get method access data from fifo which are connected to write method input.

METHODS

new

```
function new(
  string        name,
  uvm_component parent
  )
```

build_phase

```
function void build_phase(
  uvm_phase phase
  )
```

connect_phase

```
function void connect_phase(
  uvm_phase phase
  )
```

run

```
task run()
```

compare

```
virtual function void compare()
```

```
trans_d5m_dut.d5m.valid is bla v bla
```

```
(trans_d5m_dut.d5m.red == trans_d5m_prd.vfp.red)
```

```
This is plain text
```

vfp_registers

METHODS

new

```
function new(
  string name = "vfp_registers"
  )
```

post_randomize

```
function void post_randomize()
```

TYPES

e_bool

FALSE Causes the counter to only increment in odd numbers.

TRUE Causes the counter to only increment in even numbers.

d5m_camera_agent

uvm_agent

d5m_camera_agent

Heading:This is **bold text**. This is underlined text with spaces.

- Bullet one.
- Bullet two. Bullet two continued.
- Bullet three.

Some text after the bullet list.

- Level one.
- Level two, first paragraph. Level two, first paragraph continued.

  Level two, second paragraph.

  - Level three.

```
int x = 12;
int y = 0;
```

```
my $x = 12;
my $y = 0;
```

```
This is plain text.
```

The scoreboard will receive the transactions from the Monitors implemented and connected inside this agent.

METHODS

new

```
function new (
 string        name,
 uvm_component parent
 )
```

Parameters

name Description of x.
string

parent Description of y.
uvm_component

build_phase

```
function void build_phase(
 uvm_phase phase
 )
```

Use build() method to create agents's subcomponents.

connect_phase

```
function void connect_phase(
 uvm_phase phase
 )
```

Use connect() method to connect the component TLM ports

d5m_camera_configuration

METHODS

new

```
function new(
  string name = ""
  )
```

d5m_camera_fc_subscriber

```
#(d5m_camera_transaction)
uvm_subscriber
```

d5m_camera_fc_subscriber

This class provides an analysis export for receiving transactions from a connected analysis export. Making such a connection this component to any transactions emitted by the connected analysis port

METHODS

new

```
function new(
  string        name,
  uvm_component parent
  )
```

write

```
function void write(
  d5m_camera_transaction t
  )
```

The d5m_camera_transaction from the monitor is sampled by write function.

d5m_scoreboard

uvm_scoreboard

d5m_scoreboard

d5m_scoreboard This class compare the dut output values with prediction or golden ref values. Using uvm fifo sychronize write transaction from dut and predict monitor. Get method access data from fifo which are connected to write method input.

METHODS

new

```
function new(
  string        name,
  uvm_component parent
  )
```

build_phase

```
function void build_phase(
  uvm_phase phase
  )
```

connect_phase

```
function void connect_phase(
  uvm_phase phase
  )
```

run

```
task run()
```

compare

```
virtual function void compare()
```

```
trans_d5m_dut.d5m.valid is bla v bla
```

```
(trans_d5m_dut.d5m.red == trans_d5m_prd.vfp.red)
```

```
This is plain text
```

d5m_camera_driver

This class which extended from uvm driver pull data items generated by a sequencer and drive it to the DUT. In run phase methods are used for read and write operation to dut through dut interface handle.

VARIABLES

d5m_camera_vif

```
protected virtual d5m_camera_if d5m_camera_vif
```

dut interface

METHODS

new

```
function new (
  string        name,
  uvm_component parent
  )
```

build_phase

```
function void build_phase (
  uvm_phase phase
  )
```

In this phase, config the dut interface handle through get method.

run_phase

```
virtual task run_phase (
  uvm_phase phase
  )
```

In this method, fork join constructs are used to separate threads that drive each of the channels.

- reset_signals : this method reset the signals to dut
- d5m_frame : this method drive data to dut and get response from dut

reset_signals

```
virtual protected task reset_signals()
```

Reset the dut axi4 lite and d5m_cam_mod input signals when system reset is asserted from low to high.

d5m_frame

```
virtual protected task d5m_frame()
```

In this method, drive the signals from defined seq in uvm_sequence.

- seq_item_port.get_next_item : Get next item from the sequencer
- drive_transfer : this method drive data to dut and get response from dut
- seq_item_port.item_done : Tell the sequence that driver has finished current item

drive_transfer

```
virtual protected task drive_transfer (
 d5m_camera_transaction d5m_tx
 )
```

This method which is master to dut axi4lite interface read or write data at given address using bus handshaking protocol.First valid address is transmit and then wait for valid response in given time of 61 clock cycles in address_channel method. Timeout accord on 62 clock cycle, if no response is asserted high on bvalid signal from dut and timeout is flagged using uvm_error macro. If valid response is asserted then data_channel method write or read data depending on case statement.

- address_channel : this method drive the axi4lite write/read address channels.
- data_channel : this method read or write the the data at given address.

address_channel

```
virtual protected task address_channel (
 d5m_camera_transaction d5m_tx
 )
```

In this method, drive axi4 read/write addresses | d5m read/write.Note need to rewite this method.

- case : AXI4_WRITE AXI4_READ D5M_WRITE IMAGE_READ
- drive_write_address_channel : axi4 write address
- drive_read_address_channel : axi4 read address
- d5m_data_phase : write data to d5m mod
- read_d5m_phase : read data from d5m mod

data_channel

```
virtual protected task data_channel (
 d5m_camera_transaction d5m_tx
 )
```

In this method, drive axi4 address channel | d5m read/write

- case : AXI4_WRITE AXI4_READ D5M_WRITE IMAGE_READ
- drive_write_data_channel : axi4 write data
- drive_read_data_channel : axi4 read data
- d5m_data_phase : write data to d5m mod
- read_d5m_phase : read data from d5m mod

read_d5m_phase

```
virtual protected task read_d5m_phase(
 d5m_camera_transaction d5m_tx
 )
```

In this method, config test type during read operation and wait for end of frame pulse

- iImageTypeTest : axi4 write address
- iReadyToRead : axi4 read address

- eof : write data to d5m mod

d5m_data_phase

```
virtual protected task d5m_data_phase (
 d5m_camera_transaction d5m_tx
 )
```

In this method, write data to d5m camera mod

- case : AXI4_WRITE AXI4_READ D5M_WRITE IMAGE_READ
- drive_write_address_channel : axi4 write address
- drive_read_address_channel : axi4 read address
- d5m_data_phase : write data to d5m mod
- read_d5m_phase : read data from d5m mod

d5m_write_idata

```
virtual protected task d5m_write_idata (
        bit[23:0] iRgb,
        bit       ilvalid,
        bit       ifvalid,
        bit       iImageTypeTest,
 output bit       error
 )
```

In this method, drive axi4 address channel | d5m read/write

- iReadyToRead : axi4 write address
- iImageTypeTest : axi4 read address
- rgb : write data to d5m mod
- fvalid : read data from d5m mod
- lvalid : read data from d5m mod

drive_write_address_channel

```
virtual protected task drive_write_address_channel (
 d5m_camera_transaction d5m_tx
 )
```

In this method, axi4lite write address

- drive_write_address_channel : axi4 write address
- drive_read_address_channel : axi4 read address
- d5m_data_phase : write data to d5m mod
- read_d5m_phase : read data from d5m mod

drive_write_data_channel

```
virtual protected task drive_write_data_channel (
 d5m_camera_transaction d5m_tx
 )
```

In this method, axi4lite write data

- WDATA : axi4 write address
- WVALID : axi4 read address
- WREADY : write data to d5m mod
- AWVALID : read data from d5m mod

drive_read_address_channel

```
virtual protected task drive_read_address_channel (
 d5m_camera_transaction d5m_tx
 )
```

In this method, axi4lite write read address

- ARADDR : axi4 write address
- ARVALID : axi4 read address
- ARREADY : write data to d5m mod
- ARPROT : read data from d5m mod

drive_read_data_channel

```
virtual protected task drive_read_data_channel (
 output bit [31:0] data,
     output bit    error
 )
```

In this method, axi4lite read data

- RDATA : axi4 write address
- RVALID : axi4 read address
- RREADY : write data to d5m mod
- RRESP : read data from d5m mod

d5m_camera_monitor

uvm_monitor

d5m_camera_monitor

METHODS

new

```
function new (
  string        name,
  uvm_component parent
  )
```

build_phase

```
function void build_phase (
  uvm_phase phase
  )
```

run_phase

```
virtual task run_phase (
  uvm_phase phase
  )
```

collect_transactions

```
virtual protected task collect_transactions()
```

axi_config_image_frame_sequence

METHODS

new

```
function new(
 string name= "axi_config_image_frame_sequence"
 )
```

SetStatus

```
function void SetStatus(
 input int Y
 )
```

body

```
virtual task body()
```

axi_write_channel

```
virtual task axi_write_channel (
 bit[7:0]  addr,
 bit[31:0] data
 )
```

d5m_camera_base_seq

METHODS

new

```
function new (
  string name= "d5m_camera_base_seq"
  )
```

SetStatus

```
function void SetStatus(
  input int a
  )
```

Set1Status

```
virtual function void Set1Status(
  input int a
  )
```

Packet

METHODS

convert2string

```
virtual function string convert2string()
```

Function is used to return contents of this class in a string format

do_copy

```
virtual function void do_copy(
  uvm_object rhs
  )
```

Implementation of "do_copy". A generic uvm_object called "rhs" is received and type casted into Packet called "_pkt".

Then m_addr is copied from _pkt to the variable in current class

new

```
function new(
  string name = "Packet"
  )
```

Object

METHODS

new

```
function new(
  string name = "Object"
  )
```

convert2string

```
virtual function string convert2string()
```

Function used to return contents of this class in a string format

do_copy

```
virtual function void do_copy(
  uvm_object rhs
  )
```

"rhs" does not contain m_bool, m_mode, etc since its a parent handle. So cast into child data type and access using child handle Copy each field from the casted handle into local variables

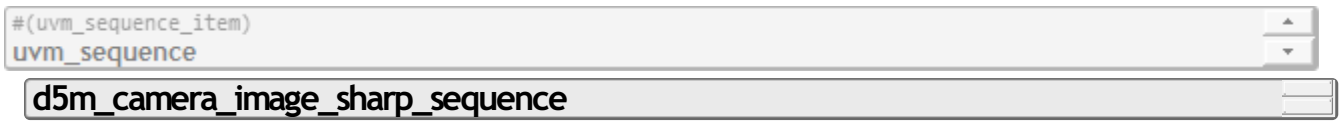
d5m_camera_image_cgain_hsl_sequence

METHODS

new

```
function new(
  string name = "d5m_camera_image_cgain_hsl_sequence"
  )
```

d5m_camera_image_sharp_sequence



METHODS

new

```
function new(
  string name = "d5m_camera_image_sharp_sequence"
  )
```

d5m_camera_sequence

METHODS

new

```
function new(
  string name= "d5m_camera_sequence"
  )
```

d5m_camera_sequencer

METHODS

new

```
function new (
  string        name,
  uvm_component parent
  )
```

d5m_image_generator_sequence

METHODS

new

```
function new(
  string name= "d5m_image_generator_sequence"
  )
```

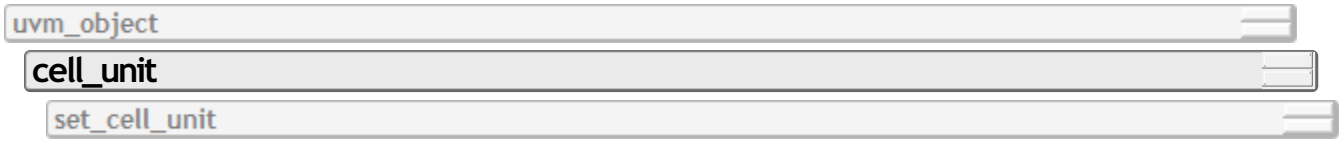
d5m_image_random_sequence

METHODS

new

```
function new(
  string name = "d5m_image_random_sequence"
  )
```

cell_unit



METHODS

new

```
function new(
 string name = "cell_unit"
 )
```

pre_call

```
function void pre_call(
 input cell_set selected_box,
       int      incre,
                set_cell_r,
                set_cell_g,
                set_cell_b
 )
```

set_cell_unit

METHODS

new

```
function new(
 string name = "set_cell_unit"
 )
```

pre_call_set

```
function void pre_call_set(
 int set_increment,
     set_cell_red,
     set_cell_gre,
     set_cell_blu
 )
```

per_cell

METHODS

new

```
function new(
 string name = "per_cell"
 )
```

pre_randomize_call

```
function void pre_randomize_call(
 input cell_set choices,
       int      set_increment,
                set_cell_red,
                set_cell_gre,
                set_cell_blu
 )
```

per_set_rows

METHODS

new

```
function new(
 string name = "per_set_rows"
 )
```

create_c_block_arrays

```
virtual function create_c_block_arrays(
 int c_size = 5
 )
```

per_rows_call

```
function void per_rows_call(
 input cell_set choices,
       int      set_increment,
                set_cell_per_r_red,
                set_cell_per_r_gre,
                set_cell_per_r_blu,
                ar_size
 )
```

convert2string

```
virtual function string convert2string()
```

per_set_cols

METHODS

new

```
function new(
  string name = "per_set_cols"
  )
```

create_arrays

```
virtual function create_arrays(
  int c_size = 5
  )
```

per_cols_call

```
function void per_cols_call(
  input cell_set choices,
        int      set_increment,
                 set_cell_per_r_red,
                 set_cell_per_r_gre,
                 set_cell_per_r_blu,
                 ar_size
  )
```

convert2string

```
virtual function string convert2string()
```

cell_box

METHODS

new

```
function new(
  string name = "cell_box"
  )
```

re_gen_cell_box

```
function void re_gen_cell_box(
  input int      outter_size,
                 inner_size,
                 set_cell_red,
                 set_cell_gre,
                 set_cell_blu,
                 set_incr,
        cell_set choices
  )
```

d5m_rgb_sequence

METHODS

new

```
function new(
  string name= "d5m_rgb_sequence"
  )
```

body

```
virtual task body()
```

axi_write_channel_test

```
virtual protected task axi_write_channel_test()
```

axi_read_channel_test

```
virtual protected task axi_read_channel_test()
```

axi_multi_writes_to_address

```
virtual protected task axi_multi_writes_to_address (
  bit[7:0]  waddr,
  bit[31:0] max_value
  )
```

axi_write_channel

```
virtual protected task axi_write_channel (
  bit[7:0]  addr,
  bit[31:0] data
  )
```

axi_read_channel

```
virtual protected task axi_read_channel()
```

d5m_write_pre_set_ifval

```
virtual protected task d5m_write_pre_set_ifval()
```

create_rgb_frames

```
virtual protected task create_rgb_frames(
  int nframes,
  int l_lines,
  int l_offset,
  int image_width
  )
```

write_per_pixel

```
virtual protected task write_per_pixel(
  bit       ImTyTest,
  bit       rImage,
  bit       fval,
  bit       lval,
  bit[23:0] rgb
  )
```

axi_write_aBusSelect_channel

```
virtual protected task axi_write_aBusSelect_channel (
  bit[7:0]  addr,
  bit[31:0] data
  )
```

vfp_configuration

METHODS

new

```
function new(
 string name = "vfp_configuration"
 )
```

convert2string

```
virtual function string convert2string()
```

unpack_packets

```
function void unpack_packets()
```

valid

```
virtual function int valid(
 input int a,
           b
 )
```

check_packets

```
function void check_packets()
```

pack_packets

```
function void pack_packets()
```

do_copy

```
virtual function void do_copy(
 uvm_object rhs
 )
```

vfp_registers

METHODS

new

```
function new(
  string name = "vfp_registers"
  )
```

post_randomize

```
function void post_randomize()
```

d5m_camera_env

uvm_env

d5m_camera_env

This Class,

VARIABLES

aL_agt

```
d5m_camera_agent aL_agt
```

this method reset the signals to dut

aL_fc_sub

```
d5m_camera_fc_subscriber aL_fc_sub
```

this method reset the signals to dut

aL_sb

```
d5m_scoreboard aL_sb
```

this method reset the signals to dut

METHODS

new

```
function new(
  string        name,
  uvm_component parent
  )
```

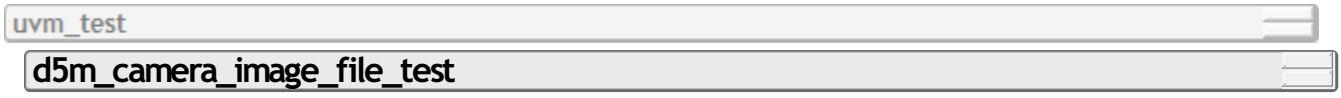
build_phase

```
function void build_phase(
  uvm_phase phase
  )
```

connect_phase

```
function void connect_phase(
  uvm_phase phase
  )
```

d5m_camera_image_file_test



METHODS

new

```
function new(
  string        name,
  uvm_component parent
  )
```

build_phase

```
function void build_phase(
  uvm_phase phase
  )
```

end_of_elaboration_phase

```
function void end_of_elaboration_phase(
  uvm_phase phase
  )
```

run_phase

```
task run_phase(
  uvm_phase phase
  )
```

d5m_camera_image_file_sharp_test

METHODS

new

```
function new(
  string        name,
  uvm_component parent
  )
```

build_phase

```
function void build_phase(
 uvm_phase phase
 )
```

end_of_elaboration_phase

```
function void end_of_elaboration_phase(
 uvm_phase phase
 )
```

d5m_camera_image_file_cgain_test

uvm_test

d5m_camera_image_file_cgain_test

d5m_camera_image_file_cgain_hsl_test

uvm_test

d5m_camera_image_file_cgain_hsl_test

d5m_camera_image_file_cgain_sharp_test

uvm_test

d5m_camera_image_file_cgain_sharp_test

d5m_camera_image_file_sharp_cgain_test

uvm_test

d5m_camera_image_file_sharp_cgain_test

d5m_camera_image_file_cgain_cgain_test

uvm_test

d5m_camera_image_file_cgain_cgain_test

d5m_camera_image_file_sobel_mask_hsl_test

uvm_test

d5m_camera_image_file_sobel_mask_hsl_test

d5m_camera_image_file_sobel_mask_blu_test

uvm_test

d5m_camera_image_file_sobel_mask_blu_test

d5m_camera_image_file_sobel_mask_cga_test

uvm_test

d5m_camera_image_file_sobel_mask_cga_test

d5m_camera_image_file_sobel_mask_hsv_test

uvm_test

d5m_camera_image_file_sobel_mask_hsv_test

d5m_camera_image_file_sobel_mask_rgb_test

uvm_test

d5m_camera_image_file_sobel_mask_rgb_test

d5m_camera_image_file_sobel_mask_shp_test

uvm_test

d5m_camera_image_file_sobel_mask_shp_test

d5m_camera_image_file_blur_test

uvm_test

d5m_camera_image_file_blur_test

d5m_camera_image_file_emboss_test

uvm_test

d5m_camera_image_file_emboss_test

d5m_camera_image_file_sobel_test

uvm_test

d5m_camera_image_file_sobel_test

d5m_camera_image_file_rgb_test

uvm_test

d5m_camera_image_file_rgb_test

d5m_camera_image_file_hsl_test

uvm_test

d5m_camera_image_file_hsl_test

d5m_camera_image_file_hsv_test

uvm_test

d5m_camera_image_file_hsv_test

d5m_camera_image_pattern_test

uvm_test

d5m_camera_image_pattern_test

METHODS

new

```
function new(
  string        name,
  uvm_component parent
  )
```

build_phase

```
function void build_phase(
  uvm_phase phase
  )
```

end_of_elaboration_phase

```
function void end_of_elaboration_phase(
  uvm_phase phase
  )
```

run_phase

```
task run_phase(
  uvm_phase phase
  )
```

d5m_camera_test

uvm_test

d5m_camera_test

METHODS

new

```
function new(
  string        name,
  uvm_component parent
  )
```

build_phase

```
function void build_phase(
  uvm_phase phase
  )
```

end_of_elaboration_phase

```
function void end_of_elaboration_phase(
  uvm_phase phase
  )
```

run_phase

```
task run_phase(
  uvm_phase phase
  )
```

d5m_rgb_patten_test

uvm_test

d5m_rgb_patten_test

METHODS

new

```
function new(
  string        name,
  uvm_component parent
  )
```

build_phase

```
function void build_phase(
  uvm_phase phase
  )
```

end_of_elaboration_phase

```
function void end_of_elaboration_phase(
  uvm_phase phase
  )
```

run_phase

```
task run_phase(
  uvm_phase phase
  )
```