Automated Configuration of Verification Environments using Specman Macros

Milos Mirosavljevic, Veriest Solutions, Belgrade, Serbia, (milosm@veriestS.com)

Ron Sela, Valens Semiconductors, Hod HaSharon, Israel (ron.sela@valens.com)

Dejan Janjic, Veriest Solutions, Belgrade, Serbia, (dejanj@veriestS.com)

Efrat Shneydor, Cadence Design Systems, Petach Tikva, Israel (efrat@cadence.com)

Introduction

- Problem: increasing ASIC complexity
- More lines of RTL code, more gates and more features
- Higher logic complexity and more configurations
- Goal: achieve highest verification efficiency
- Our solution: Specman macros

HDBaseT® At a Glance (1)

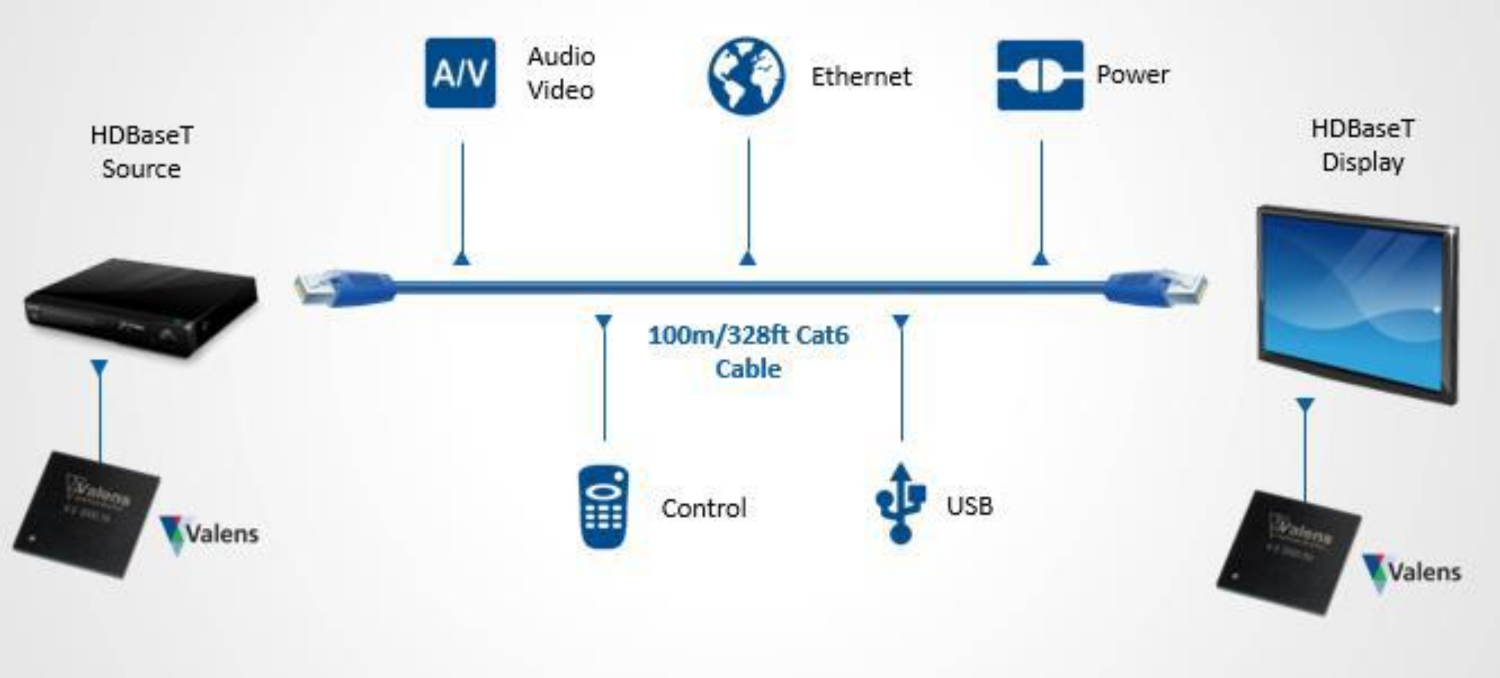
- Invented and developed by Valens
- Standard for the transmission of HDMI, Ethernet, controls, USB and up to 100W of power over a single, long-distance, cable.

HDBaseT® At a Glance (2)

Veriest

HDBaseT® At a Glance (3)

- Used in audiovisual, consumer electronics, medical and government applications, as well as automotive.



Veriest

HDBaseT® Switch (“T-Switch”) (1)

- 16 x ports with 16+2Gbps per port
- Each port supports HDCP 2.2 and HDCP 1.4
- HDCP: form of digital protection developed by Intel
- Prevention of copying of audio and video content

HDBaseT® Switch (“T-Switch”) (2)

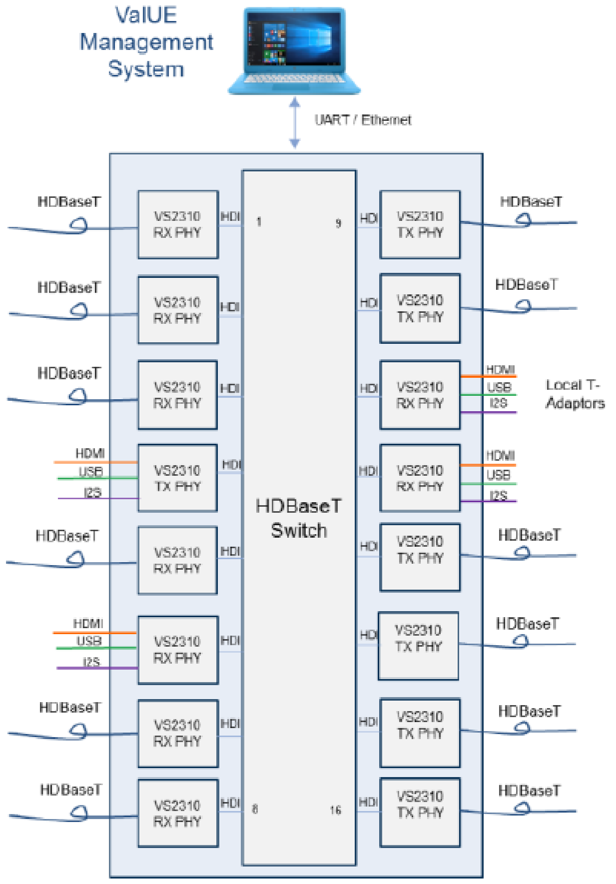


Figure 2: Native and HDBaseT switching

T-Switch – Verification challenges (1)

- Verification team is composed of members with different levels of experience
- How to ensure that tests can be written and run quickly
- Cover all configuration and stimuli options
- Typical project: lost of boolean

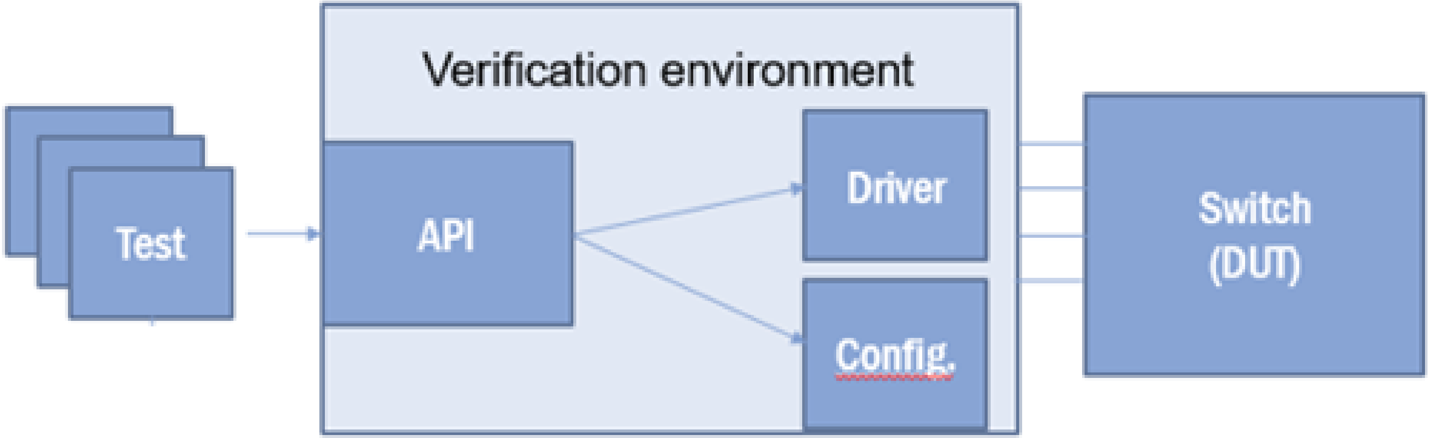
fields whom each engineer should

be familiar with in order to

activate the chip and generate the right stimuli

T-Switch – Verification challenges (2)

- Goal: develop an easy to use API which solves all configuration matters under-the-hood and allows straightforward test creation without too much environment background knowledge



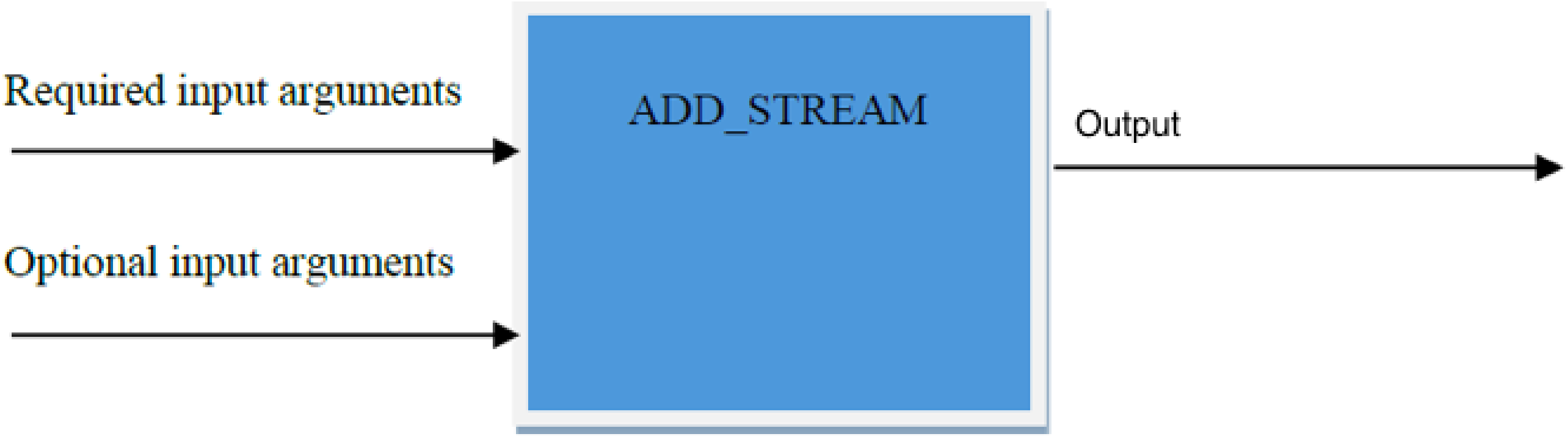
Veriest

T-Switch – Verification solution

- Specman macros – extending ***e*** and adding new constructs
- Main advantage – easy to use syntax for test writers
- *define as* vs *define as computed*
- *define as*: replacement code is written in the macro body
- *define as computed*: macro user writes procedural code which *computes* replacement code. Useful when replacement code is not fixed

ADD_STREAM macro (1)

- Used under the pre-defined run()
- Goal: Override old configuration generated automatically, just before the first Specman tick *(sequences need at least 1 tick to start their body)*
- Output: set of rules used by verification environment



ADD_STREAM macro (2)

- Simple usage:

```
run() is also {
   ADD_STREAM
   stream_type                     = UNICAST
   src_port                        = 0
   dst_ports                       = {1}
   pkt_types_category_in_strm      = {OTHER_P1}
   specific_p_type_in_category     = {PTYPE14}
   priority                        = {PRIORITY_1}
   pkt_type_bw                     = {16000}
   hdcp                            = FALSE
   burst_cycles                    = 0
   sid                             = 100
   ayalon_source                   = FALSE
   ayalon_dest                     = FALSE
   pkt_num                         = 500;
};
```

Veriest

ADD_STREAM macro (3)

- Quickly create desired scenarios:
  - *Inject HDMI data to the switch port 0 and send this data through the HDCP towards port 14 and also send it back from port 0 (multicast)*
  - *Inject data with very low bandwidth to the switch ports 0-7, and send this data towards ports 8-15*

- Macro: 15 lines of code in the test
- Macro instantiation translates to 180 lines of code under the hood: 12x code reduction

ADD_STREAM macro (4)

```
define <add_stream'action> "ADD_STREAM [ ]stream_type[ ]=[ ]<stream_type'type>\
                            src_port = <src_port'exp>\
                            dst_ports = {<dst_port'exp>;...}\
                            pkt_types_category_in_strm[ ]=[ ]{<pkt_types_category_in_strm'type>;...}|
                            specific_p_type_in_category[ ]=[ ]{<specific_p_type_in_category'type>;...
                            priority = {<priority'exp>;...}\
                            pkt_type_bw = {<pkt_type_bw'exp>;...}\
                            hdcp = <hdcp'exp>[ hdcp_bypass = <hdcp_bypass'exp>][ hdcp_bfg_en = <hdcp_
                            burst_cycles = <burst_cycles'exp>\
                            sid[ ]=[ ]<sid'exp>[ hdmic_cmd = <hdmic_cmd'type>]\
                            ayalon_source[ ]=[ ]<ayalon_source'exp>\
                            ayalon_dest[ ]=[ ]<ayalon_dest'exp>\
                            pkt_num = <pkt_num'exp>[ add_stream_on_the_fly = <add_stream_on_the_fly'e

   var ingress_stream              : stream_s;
   var tmp_packet_type_bw_st       : packet_type_bw_st;
   var tmp_packet_type_bw_st_l     : list of packet_type_bw_st;
   var num_of_pkt_types_in_strm    : uint;

   num_of_pkt_types_in_strm = {<pkt_type_bw'exps>}.size();

   for i from 0 to num_of_pkt_types_in_strm-1 do {
#ifndef REAL_CPU {
      set_hdmic_parameters({<pkt_types_category_in_strm'types>}[i],<hdmic_cmd'type|RXS>);   // milosm
};//#ifndef REAL_CPU
      gen tmp_packet_type_bw_st keeping {
         .pkt_type_category                                  == {<pkt_types_category_in_strm'types>}[i
         .pkt_type                                           == {<specific_p_type_in_category'types>}|
         .priority                                           == {<priority'exps>}[i];
         .pkt_type_bw                                        == {<pkt_type_bw'exps>}[i];
#ifndef REAL_CPU {
         .as_a(bist_en packet_type_bw_st).bist_port_mode     == <bist_port_mode'type|DHDI_SLAVE_2_2>;
};//#ifndef REAL_CPU
      };
      tmp_packet_type_bw_st_l.add(tmp_packet_type_bw_st);
   };
```

Veriest

Example scenario – HDMI + HDCP (1)

- Drive video (HDMI) stream from port 0 to ports 1 and 13
- The stream is going through HDCP block in bypass mode (no encryption)
- HDCP version in all HDCP blocks is 1.4

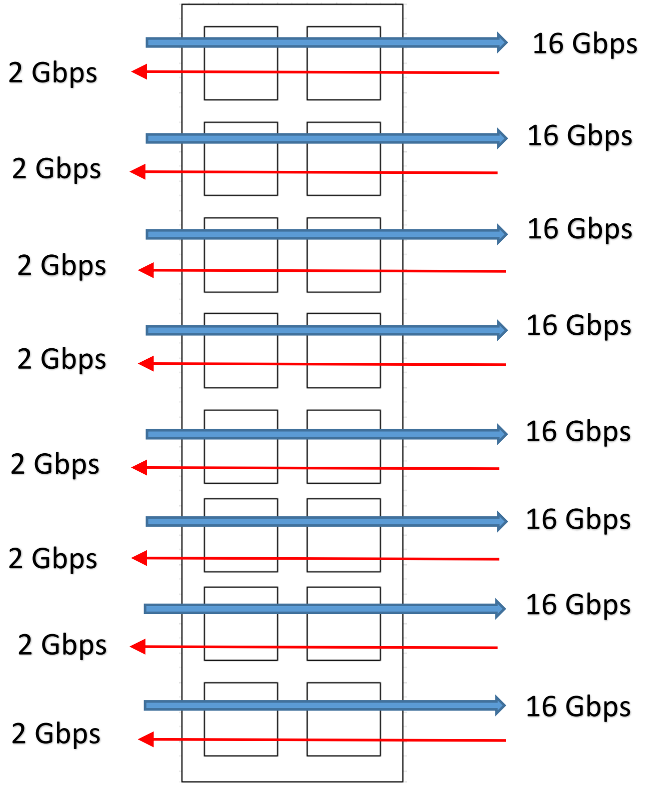
Example scenario – HDMI + HDCP (2)

```
ADD_STREAM
stream_type                     = MULTICAST
src_port                        = 0
dst_ports                       = {1;13}
pkt_types_category_in_strm      = {HDMI}
specific_p_type_in_category     = {PTYPE14}
priority                        = {PRIORITY_1}
pkt_type_bw                     = {7000} – 7 Gbps
hdcp                            = TRUE
hdcp_bypass                     = TRUE
hdcp_version                    = {VER_1_4; VER_1_4; VER_1_4; VER_1_4}
burst_cycles                    = 0
sid                             = 100
ayalon_source                   = FALSE
ayalon_dest                     = FALSE
pkt_num                         = 200;
```

Veriest

Example scenario – stress test (1)

- Stress test: maximum bandwidth through the chip
- Each port: 16G +2G = 18Gbps
- Total: 288 Gbps



Veriest

Example scenario – stress test (2)

- Total of 16 ADD_STREAM macros used

```
for i from 0 to 7 {
    ADD_STREAM
    stream_type                    = UNICAST
    src_port                       = i
    dst_ports                      = {i+8}
    pkt_types_category_in_strm     = {OTHER_P1;        OTHER_P2;       OTHER_P3}
    specific_p_type_in_category    = {PTYPE14;         SPDIF;          PTYPE13}
    priority                       = {PRIORITY_1;      PRIORITY_2;     PRIORITY_3}
    pkt_type_bw                    = {15750;           200;            50}
    hdcp                           = FALSE;
    burst_cycles                   = 0
    sid                            = 100+i
    ayalon_source                  = FALSE
    ayalon_dest                    = FALSE
    pkt_num                        = 30000;
};
```

16 Gbps

```
for i from 0 to 7 {
    ADD_STREAM
    stream_type                    = UNICAST
    src_port                       = i+8
    dst_ports                      = {i}
    pkt_types_category_in_strm     = {OTHER_P1; OTHER_P2; OTHER_P3}
    specific_p_type_in_category    = {PTYPE14; SPDIF; PTYPE13}
    priority                       = {PRIORITY_1; PRIORITY_2; PRIORITY_3}
    pkt_type_bw                    = {1750; 200; 50}
    hdcp                           = FALSE
    burst_cycles                   = 0
    sid                            = 200+i
    ayalon_source                  = FALSE
    ayalon_dest                    = FALSE
    pkt_num                        = 3750;
};
```

2 Gbps

Veriest

Specman macros advantages

- Easier to read – unlike other languages, macro developer defines syntax for end user
- Allows designers to create scenarios without Specman knowledge itself
- Macro parameters can be lists of unknown length:

```
priority = {<priority'exp>;...}
```

- Simplicity of usage: easier than functions when there are so many inputs
- Function:

```
Add_stream(UNICAST,0,1,OTHER_P1,PTYPE14,PRIORITY_1,16000,FALSE,0,100,FALSE,FALSE,500)
```

Veriest

Specman macro limitations

- Limitation of 14 input arguments
- Team utilized mechanism of optional arguments to increase number of inputs
- Our case: 14->14+10
- Optional arguments allowed much more versatility in stimuli generation
- Macro simplification: optional arguments default values

```
hdcp = <hdcp'exp>[ hdcp_bypass = <hdcp_bypass'exp>]
```

Potential improvements

- Make macro more generic – not dependent on the exact environment hierarchy
- Instead of assuming macro is called from the configuration unit, define as computed could, using reflection, find where a unit of specific type is instantiated, what fields has, and more.

Results

- Macro allowed accelerated verification – tape out on time
- Exhaustive coverage achieved
- Full leverage of the team: juniors and seniors, as well as designers
- Project life span ~ 2.5 years
- 14 different verification engineers
- 7 design engineers who sporadically joined verification
- 185000 registers in ASIC, 480 tests
- Macro used in ~75% of the tests

Veriest

Results

Thank you!

www.VeriestS.com

Veriest

Questions?

www.VeriestS.com