Automated Configuration of Verification Environments using Specman Macros

Milos Mirosavljevic, Veriest Solutions, Belgrade, Serbia, (milosm@veriestS.com)
Ron Sela, Valens Semiconductors, Hod HaSharon, Isreal (ron.sela@valens.com)
Dejan Janjic, Veriest Solutions, Belgrade, Serbia, (dejanj@veriestS.com)
Efrat Shneydor, Cadence Design Systems, Petach Tikva, Isreal
(efrat@cadence.com)

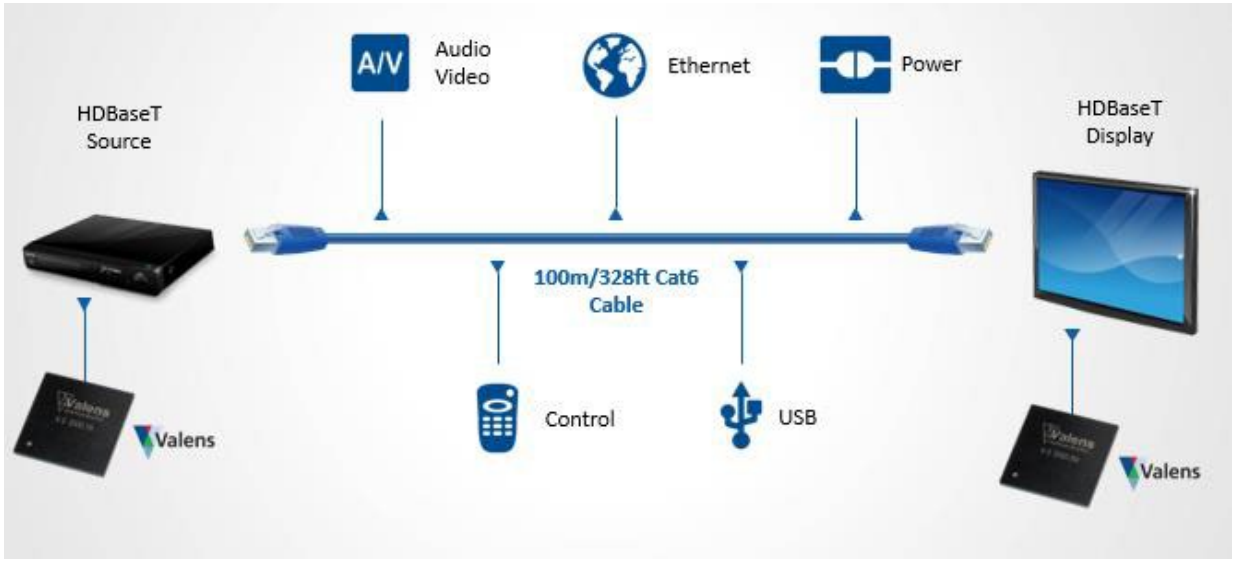
accellera SYSTEMS INITIATIVE
2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Introduction

- Problem: increasing ASIC complexity
- More lines of RTL code, more gates and more features
- Higher logic complexity and more configurations
- Verification point of view: how to plan it and organize it for the highest efficiency?

HDBaseT® At a Glance

- Standard for the transmission of ultra-high-definition video & audio, Ethernet, controls, USB and up to 100W of power over a single, long-distance, cable.
- Used in audiovisual, consumer electronics, medical and government applications, as well we automotive.



HDBaseT® Switch (“T-Switch”)

- 16 x ports with 16+2Gbps per port
- Each port supports HDCP 2.2 and HDCP 1.4

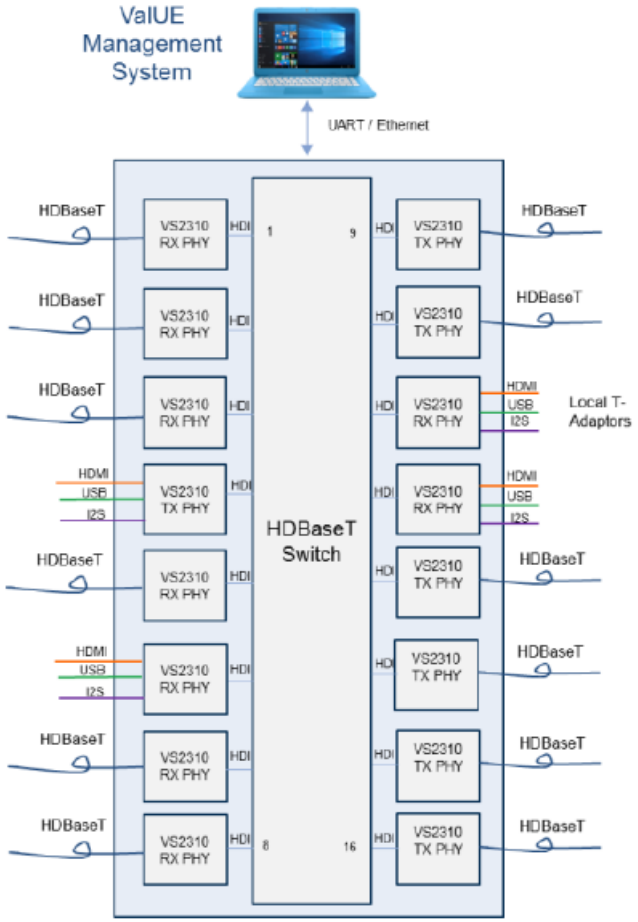


Figure 2: Native and HDBaseT switching

T-Switch – Verification challenges

- Cover all different configuration options
- Verification team size and different levels of expertise: need easy to use API which solves all configuration matters under-the-hood and allows straightforward test creation
- New engineers assigned to the project should be able to start working right away, without the need for excessive environment introduction
- Verification environment structure?

Typical verification solutions

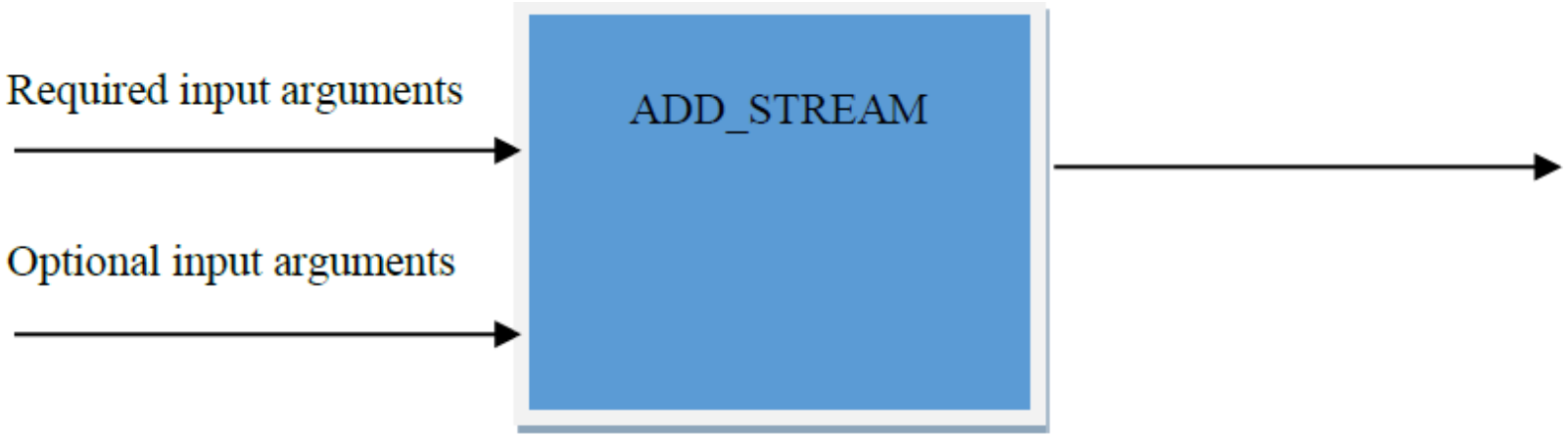
- Implementing new verification methodologies
- Using advanced code tecnhiques
- Power of reusability

T-Switch – Verification solution

- Specman macros – extending e and adding new constructcs
- Main advantage – easy to use syntax for test writers
- define as
- Macro output is called stream – set of rules used by verification environment
- Each stream conveys different protocol
- 16 ports to be configured
- 256 possibilities of communication between the ports

ADD_STREAM macro (1)

- *action* macro used under pre-defined run()
- Override necessary old configuration generated automatically, just before the first specman tick *(sequences need at least 1 tick to start their body)*



ADD_STREAM macro (2)

- 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 portd 8-15*

ADD_STREAM macro (3)

- 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;
};
```

ADD_STREAM macro (4)

- Allows designers to create scenarios without Specman knowledge itself
- Macro usage: 15 lines of code in the test
- Macro instantiation translates to 180 lines of code under the hood: 12x code reduction
- Required and optional arguments
- Required: stream type, source, destination, hdcp, packet types
- Optional: hdmi command, bist termination, hdcp version

Example scenario – stress test

- Stress test: maximum bandwidth throught the chip
- Each port: 16G +2G = 288Gbps
- 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;
};
```

```
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;
};
```

Specman macros advantages

- Easier to read – unlike other languages, macro developer defines syntax for end user
- 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)
```

- Macro parameters can be lists of unknown length:

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

Specman macro limitations

- Limitation of 14 input arguments
- Team utilized mechanism of optional arguments to increase number of inputs
- Improvement by ~70% (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>]
```

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, 279 tests
- Macro used in ~75% of the tests

Results

- Presenter:

Questions?