2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Random Directed Low Power Coverage Methodology:
(A Smart Approach to Power Aware Verification Closure)

Awashesh Kumar, Mentor Graphics
(awashesh_kumar@mentor.com)
Madhur Bhargava, Mentor Graphics
(madhur_bhargava@mentor.com)

Mentor Graphics®

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Agenda

- Introduction
- Basic concepts of UPF
- Motivation for low power coverage methodology
- IEEE 1801-2015 (UPF 3.0) Information model
- Proposed methodology
  - Case studies and examples
  - Benefits of proposed methodology
  - Current challenges and future work
- Conclusion

Introduction

Today's SoCs

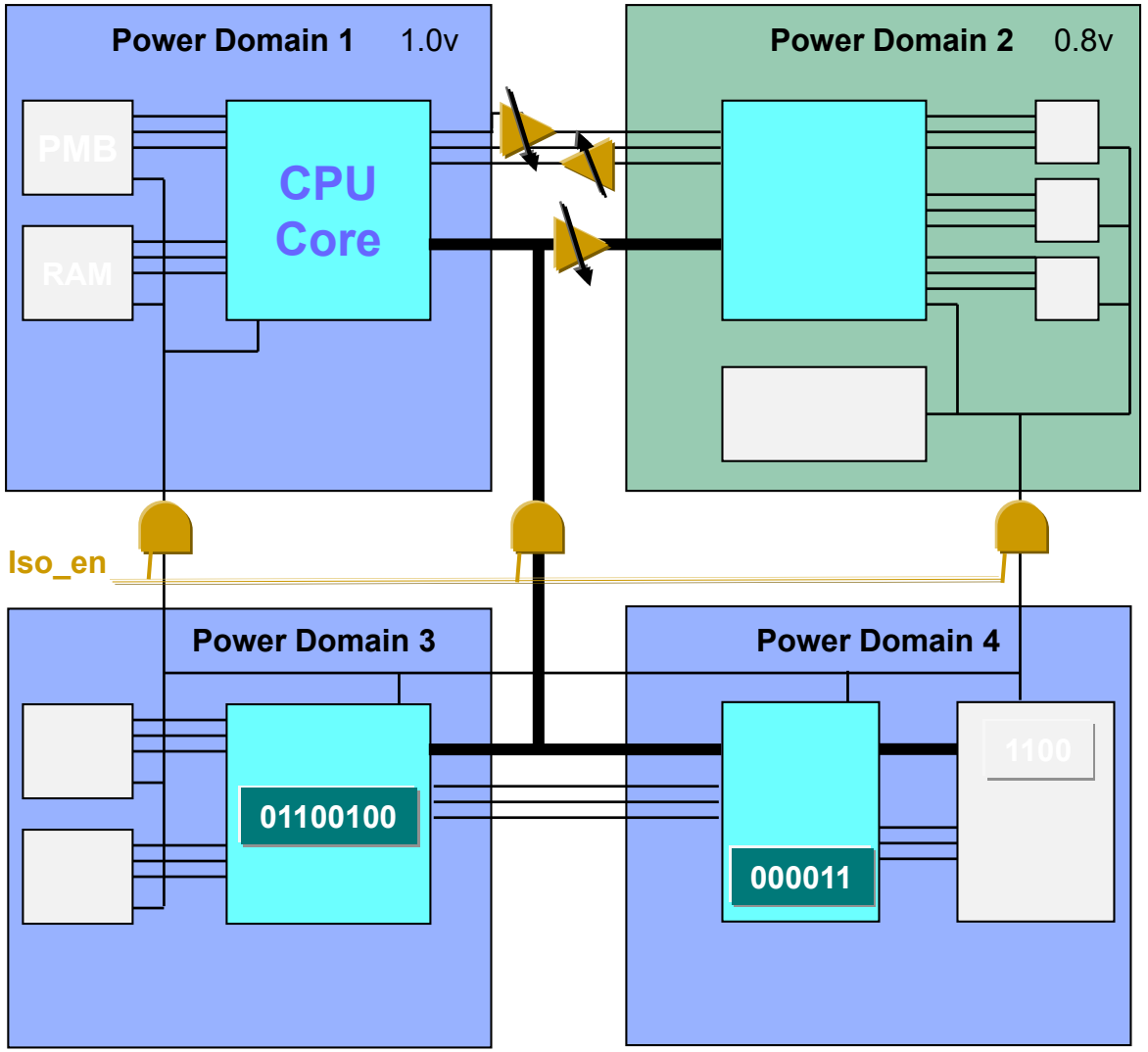
- Are incredibly Complex
- Have sophisticated power management strategies for highly power efficient design
- Integrate variety of implementation cells like isolation and retention
- Power states with various simstates

They Must

- Verify the power management
  - Early in the design flow

Basic Concepts of UPF

- Power Shut-off
- Isolation
- Retention
- Level Shifting
- Multi-Voltage
- Dynamic Voltage & Frequency Scaling



Challenges in LP verification

- Sign off low-power chip -> 100% LP coverage
  - Are all power management control sequences covered?
  - Are all complex interactions between power domains covered?
  - Are all the desired power states reached or not?
  - Are all desired power state transitions reached or not?
  - Is there any illegal power state reached?

No SMART way to achieve this !!

Motivation for Methodology

- No pre-defined coverage metric to capture power states and their transitions
- UCIS (Unified Coverage Interoperability Standard) does not provide any metric to capture low power intent (power states etc.)

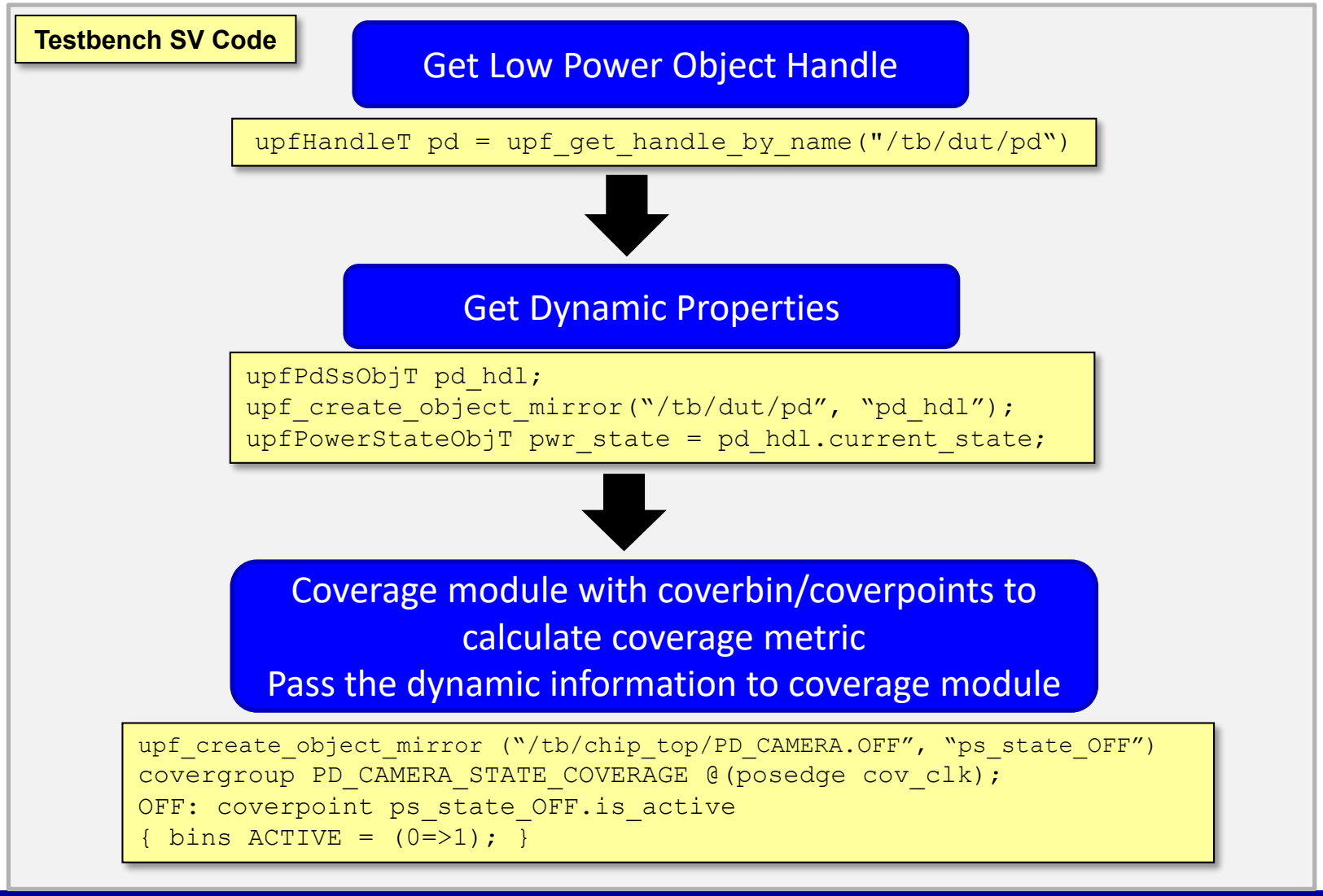
- Tool generated metrics
  - Not suitable for some user specific requirement
  - Users rely on tool for new coverage features related to low power verification

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Proposed Low Power Coverage Methodology

- Coverage methodology developed using
  - UPF 3.0 HDL package functions
    - Low power objects access
  - System Verilog coverage constructs
- Allows users to do random directed scenarios verification
  - Users can create
    - Random HDL verification scenarios
    - Directed HDL verification scenarios
      - Example: Mutual exclusivity of states of two power domains

Coverage Methodology



UPF 3.0 Information Model

- Introduced in UPF 1801-2015
- Abstract data model
- Represents low power objects created in UPF
  - E.g. Power Domain, Power State, Supply Set etc.
- Provides list of properties low power objects can have
  - Both static and dynamic information
- API interface; to allow access of objects and properties
  - **Tcl Interface**:
    - To access objects/properties in a Tcl script or UPF file
  - **HDL Interface:**
    - To access/manipulate objects/properties in a testbench or simulation model

UPF 3.0 Information Model – Native HDL Represenation

- Native HDL representation of LP objects
  - For object with dynamic properties
    - e.g. power domain or supply set
  - Represented by struct/record in HDL containing two fields
    - A value field – dynamic property value
    - A handle/reference to UPF object – to access other properties of the object

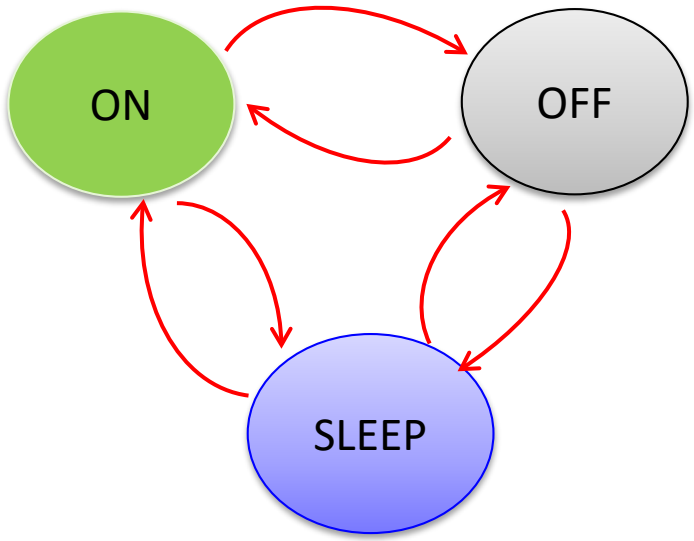
Type Name	SV Representation
upfPdSsObjT	struct { upfHandleT handle; upfPowerStateObjT current_state; } upfPdSsObjT

Accessing Low-Power object handles & properties in HDL

- HDL access functions
  - Basic functions to access the low power objects and properties
  - E.g. `upfHandleT pd = upf_get_handle_by_name("/top/dut/pd")`
    `upfHandleT ps_active_hndl = upf_query_object_properties(`
    `power_state, UPF_IS_ACTIVE )`
- Immediate read access HDL functions
  - E.g. `integer ps_on_value = upf_get_value_int(ps_active_hndl)`
- Continuous access HDL functions
  - Enables continuous monitoring of dynamic values
  - E.g. `upfSupplyObjT vdd_monitor`
    `upf_create_object_mirror("/top/dut/vdd", "vdd_monitor")`
- Utility functions
  - General utility function to assist users.
  - E.g. `upfClassIdE upf_query_object_type(upfHandleT handle)`

Case Study I – Coverage of Power States

- Camera domain (PD_CAMERA)
  - Three power states “ON, SLEEP, OFF”.
- Power states are defined in UPF




```
add_power_state PD_CAMERA \
-state ON \
{-logic_expr {primary == ON} }
{-supply_expr {VDDP == {FULL_ON,
                        1.2} }
 } ...
```

- Are all the desired power states reached or not?
- Are all desired power state transitions reached or not?

Case Study I – Coverage of Power States Cont.

HDL Code

```
// Native HDL objects for power states
upfPowerStateObjT ps_state_ON;
upfPowerStateObjT ps_state_OFF;
upfPowerStateObjT ps_state_SLEEP;

// Handle to hold active state
upfHandleT state_ON_active;
upfHandleT state_OFF_active;
upfHandleT state_SLEEP_active;

//integer values of active states, 1 indicates active
integer state_ON;
integer state_OFF;
integer state_SLEEP;
```

Case Study I – Coverage of Power States Cont.

```
//continuous access of power states

upf_create_object_mirror \
             ("/tb/chip_top/PD_CAMERA.ON", "ps_state_ON")
upf_create_object_mirror \
             ("/tb/chip_top/PD_CAMERA.OFF", "ps_state_OFF")
upf_create_object_mirror \
             ("/tb/chip_top/PD_CAMERA.SLEEP", "ps_state_SLEEP")

always @(ps_state_ON, ps_state_OFF, ps_state_SLEEP)
begin
     state_ON_active = upf_get_object_properties
                              (ps_state_ON.handle, UPF_IS_ACTIVE)
     state_OFF_active = upf_get_object_properties
                              (ps_state_OFF.handle,UPF_IS_ACTIVE)
     state_SLEEP_active = upf_get_object_properties
                              (ps_state_SLEEP.handle, UPF_IS_ACTIVE)
     state_ON = upf_get_value_int(state_ON_active);
     state_OFF = upf_get_value_int(state_OFF_active);
     state_SLEEP = upf_get_value_int(state_SLEEP_active);
end
```

Case Study I – Coverage of Power States Cont.

Coverage module

```
module covPowerStateModule (int state_ON, state_OFF,
state_Sleep, string StateName)
wire [2:0] curr_state;
reg cov_clk = 0;
assign curr_state = {state_OFF, state_ON, ……};

always @(state_OFF, state_ON, ……)
cov_clk = 1'b1;

always @(cov_clk)
cov_clk = 1'b0;
endmodule
```

Case Study I – Coverage of Power States Cont.

Covergroup modeling of state coverage

```
covergroup PD_CAMERA_STATE_COVERAGE @(posedge cov_clk);
    OFF: coverpoint state_OFF
        { bins ACTIVE = (0=>1); }
    ON: coverpoint state_ON
        { bins ACTIVE = (0=>1); }
    SLEEP: coverpoint state_SLEEP
        { bins ACTIVE = (0=>1); }
endgroup
PD_CAMERA_STATE_COVERAGE PS_primary = new;
```

Covergroup modeling of state transition coverage

```
covergroup primary_TRANSITION_COVERAGE @(posedge cov_clk);
PA_CAMERA_TRANSITION_COVERAGE:coverpoint curr_state
{
        wildcard bins OFF_to_ON = (3'b??1=> 3'b?1?);
        wildcard bins OFF_to_SLEEP = (3'b??1=> 3'b1??);
        wildcard bins ON_to_OFF = (3'b?1?=> 3'b??1);
}
endgroup
```

Case Study II – Coverage of Isolation Control Signals

HDL Code

```
// access all the isolation controls of a domain
upfHandleT iso_ctrls[];

upfHandleT iso_cnt = 0;
upfHandleT pd = upf_get_handle_by_name("/tb/pd");
upfHandleT iso_list = upf_query_object_properties(pd,
                                    UPF_ISOLATION_STRATEGIES);
upfHandleT ctrl;
upfHandleT iso = upf_iter_get_next(iso_list);
while (iso) begin
    ctrl =  upf_query_object_properities(iso,UPF_ISOLATION_CONTROLS);
    iso_ctrls[iso_cnt] = ctrl;
    iso_cnt++;
    iso = upf_iter_get_next(iso_list);
end
```

Case Study II – Coverage of Isolation Control Signals

Coverage Module

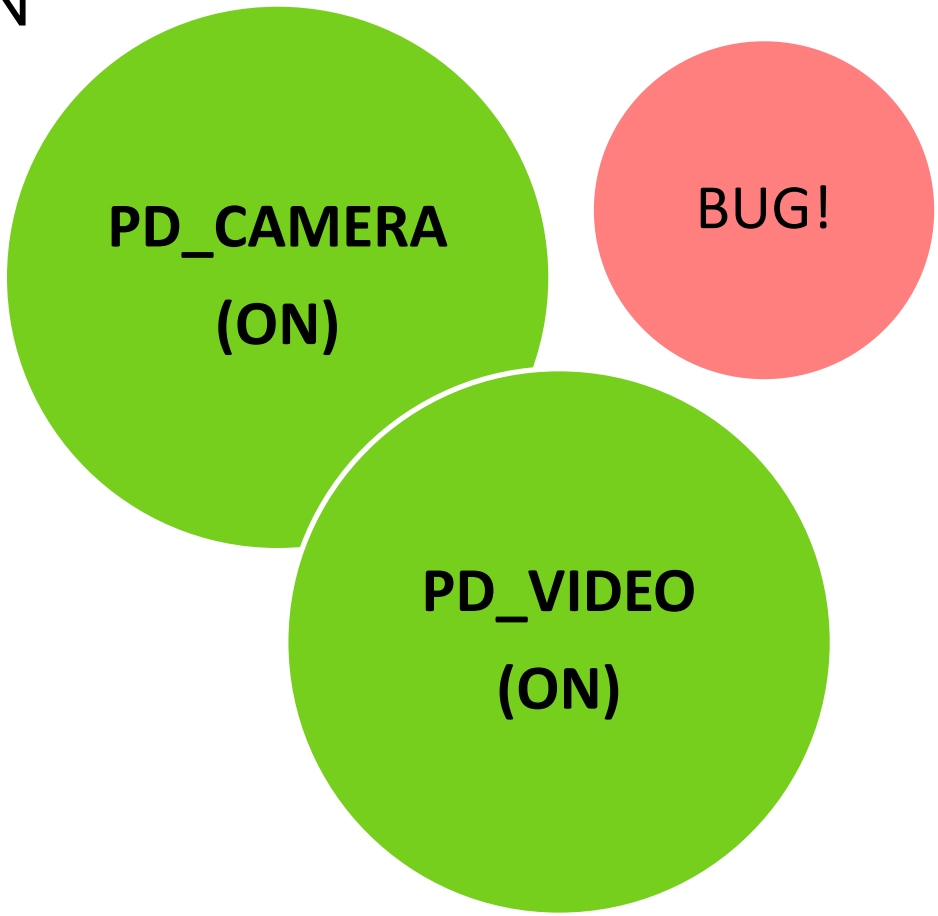
```
module covIsoModule (upfHandleT iso_ctrl);
    reg isovalue;
    initial begin
    upf_create_object_mirror(upf_query_object_pathname(iso_ctrl),
                                                    isovalue);
    end

    covergroup ISO_SIG_STATE_COVERAGE @(isovalue);
    ACTIVE_HIGH: coverpoint isovalue { bins ACTIVE = 1; }
    ACTIVE_LOW: coverpoint isovalue { bins ACTIVE = 0; }
    endgroup
endmodule
```

Coverage Module Instantiation

```
genvar i;
generate
for (i = 0; i<iso_cnt; i++) begin: cov_blk
     covIsoModule covInst(iso_ctrls[i]);
end
endgenerate
```

Case Study III – Assertions with UPF 3.0 Information Model

- A combination of state ON for PD_CAMERA and state ON for PD_VIDEO cannot be true at the same time.



Case Study III – Assertions with UPF 3.0 Information Model

Assertion Module

```
module assertionPowerState (int state_ON_CAMERA, int state_ON_VIDEO)
  reg cov_clk = 0;

  always @(state_ON_CAMERA, state_ON_VIDEO)
      cov_clk = 1'b1;
  always @(cov_clk)
      cov_clk = 1'b0;

  always@(posedge cov_clk)
      assert (state_ON_CAMERA != state_ON_VIDEO)
      else
              $error("Camera and Video both on at same time")
endmodule
```

Benefits of Proposed Methodology

- Early low power coverage closure
- Directed scenario testing
- Coverage code and scripts development can be automated easily
  - Can be used by tools to generated coverage models
- Unlike other coverage methodologies; low level details like supplies and logic values is not required.
- UPF based methodology
  - Consistent and usable across multiple vendors
- Easily scalable to more complex scenarios

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Unaddressed Low Power Verification Challenges

- Proposed approach robust at RTL level
- Lacks few things
  - Not addressing “Analog/AMS”
    - No ability to access analog/mixed-signal parameters from onboard regulators & voltage sources

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Conclusion

- Ad-hoc approaches of low power coverage not likely to succeed
- Proposed a methodology for low-power coverage
  - Using the UPF 3.0 HDL package functions
- Faster verification closure
- Case studies using covergroups and coverpoints
- Discussed the advantages of proposed methodology

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

References

- [1] IEEE Std 1801™-2015 for Design and Verification of Low Power Integrated Circuits. IEEE Computer Society, 05 Dec 2015.
- [2] “Amit Srivastava, Awashesh Kumar”, PA-APIs: Looking beyond power intent specification formats, DVCon USA 2015
- [3] “Veeresh Vikram Singh, Awashesh Kumar”, Cross Coverage of Power States, DVCon USA 2016
- [4] “Pankaj Kumar Dwivedi, Amit Srivastava, Veeresh Vikram Singh”, Let’s DisCOVER Power States, DVCon USA 2015

Thank You (Q & A)