Low Power Verification with LDO

Shang-Wei Tu
Design Technology Division
MediaTek Inc., HsinChu, Taiwan

Amol Herlekar
Verification Group
Synopsys Inc., Mountain View, CA, USA

Introduction

Background
- Low-Dropout Regulator (LDO) is used to regulate output voltage
- LDO is commonly used for Dynamic Voltage and Frequency Scaling (DVFS)

Motivation
- Modeling LDO is a huge problem
- UPF still does not provide mechanism to model LDO
- Supply resolution mechanisms provided by UPF are not enough

Summary
- Existing SystemVerilog modeling solution is demonstrated with some drawbacks
- We propose extensions to UPF commands for easily modeling LDO and special supply net resolution
  - "*create_power_switch –output_voltage*"
  - "*create_supply_net –resolve weak/strong/either*"

Example LDO Design and Limitation of UPF for Modeling

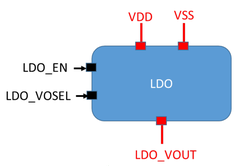

Figure 1

Pin Name	Type	Direction	Description
VDD	PG	Input	Power
VSS	PG	Input	Ground
LDO_VOUT	PG	Output	LDO output voltage
LDO_VOSEL[2:0]	Signal	Input	Output voltage level selection 3'h0: 0.600V 3'h4: 1.000V 3'h1: 0.700V 3'h5: 1.100V 3'h2: 0.800V 3'h6: 1.150V 3'h3: 0.900V 3'h7: 1.200V
LDO_EN	Signal	Input	Enable signal (Active high)

Figure 2

Given LDO design (Figure 1), I/O pins are illustrated in table and control sequence is in Figure 2.

Limitation of UPF:
- "*create_power_switch*" command cannot model voltage conversion

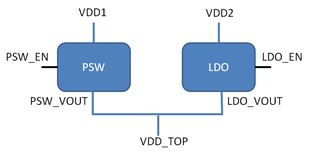

Figure 3

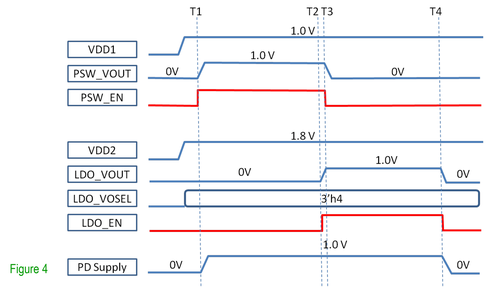

Figure 4

Power supply network in Figure 3 is used in real design. Although conceptually, either PSW or LDO will be ON, there will be some overlap between them (Figure 4).

Limitation of UPF:
- UPF supply resolutions "*one_hot, parallel, parallel_one_hot*" cannot model this design correctly

SystemVerilog Workaround Solution

SystemVerilog model for LDO Design:

```
1  import UPF::*;
2  module LDO (
3  `ifdef POWER_AWARE_SIMULATION
4    output supply_net_type VDD_to_SV,
5    output supply_net_type VSS_to_SV,
6    output supply_net_type LDO_VOUT,
7  `endif
8    input LDO_EN,
9    input [2:0] LDO_VOSEL
10 );
11   real LDO_VOUT_r;
12 `ifdef POWER_AWARE_SIMULATION
13   always @(*) begin
14       if(VDD_to_SV.state == UPF::FULL_ON && LDO_EN==1'b1) begin
15           LDO_VOUT.state = UPF::FULL_ON;
16           LDO_VOUT.voltage = LDO_VOUT_r*1000000;
17       end
18       else begin
19           LDO_VOUT.state = UPF::OFF;
20           LDO_VOUT.voltage = 0;
21       end
22   end
23 `endif
24
25   always @(*) begin
26       case (LDO_VOSEL)
27           3'h0: LDO_VOUT_r = 0.6;
28           3'h1: LDO_VOUT_r = 0.7;
29           3'h2: LDO_VOUT_r = 0.8;
30           3'h3: LDO_VOUT_r = 0.9;
31           3'h4: LDO_VOUT_r = 1.0;
32           3'h5: LDO_VOUT_r = 1.1;
33           3'h6: LDO_VOUT_r = 1.15;
34           3'h7: LDO_VOUT_r = 1.2;
35       endcase
36   end
37
38 endmodule
```

In/Out from/to UPF

Voltage conversion

Figure 5

SystemVerilog model for Special Supply Resolution:

```
1  import UPF::*;
2  module upf_sn_resolution(
3    input supply_net_type in_sn1,
4    input supply_net_type in_sn2,
5    output supply_net_type out_sn);
6
7    (* vcs_always_on *)
8    always @(in_sn1 or in_sn2) begin
9      if ((in_sn1.state==UPF::OFF) && (in_sn2.state==UPF::OFF)) begin
10       out_sn.state = UPF::OFF;
11       out_sn.voltage = 0.0;
12     end
13     else if((in_sn1.state!=UPF::OFF) && (in_sn2.state!=UPF::OFF)) begin
14       if (in_sn1.state != UPF::FULL_ON && in_sn2.state != UPF::FULL_ON) begin
15         out_sn.state = in_sn1.state;
16         out_sn.voltage = (in_sn1.voltage > in_sn2.voltage)? in_sn2.voltage:in_sn1.voltage;
17       end else if (in_sn1.state != UPF::FULL_ON) begin
18         out_sn.state = in_sn2.state;
19         out_sn.voltage = in_sn2.voltage;
20       end else if (in_sn2.state != UPF::FULL_ON) begin
21         out_sn.state = in_sn1.state;
22         out_sn.voltage = in_sn1.voltage;
23       end else begin
24         out_sn.state = in_sn1.state;
25         out_sn.voltage = (in_sn1.voltage > in_sn2.voltage)? in_sn2.voltage:in_sn1.voltage;
26       end
27     end
28     else if((in_sn1.state==UPF::OFF) && (in_sn2.state!=UPF::OFF)) begin
29       out_sn.state = in_sn2.state;
30       out_sn.voltage = in_sn2.voltage;
31     end
32     else if((in_sn1.state!=UPF::OFF) && (in_sn2.state==UPF::OFF)) begin
33       out_sn.state = in_sn1.state;
34       out_sn.voltage = in_sn1.voltage;
35     end
36     else begin
37       $display("Error! undefined resolution: in_sn1 %s in_sn2 %s %t",
38           in_sn1.state.name, in_sn2.state.name, $realtime);
39     end
40   end
41 endmodule
```

In/Out from/to UPF

Resolution "either"

Figure 6

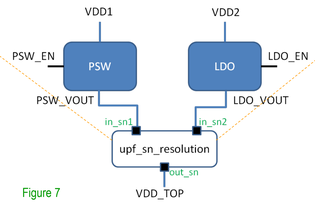

Figure 7

Proposed UPF Solution with Revised Syntax

Proposed UPF Syntax for Modeling LDO Design:

***create_power_switch** switch_name*
[-output_voltage {state_name voltage_value}]

```
1  create_power_domain PD_LDO
2
3  create_supply_port VDD -direction in
4  create_supply_port VSS -direction in
5  create_supply_port LDO_VOUT -direction out
6  create_supply_net VDD -domain PD_LDO
7  create_supply_net VSS -domain PD_LDO
8  create_supply_net LDO_VOUT -domain PD_LDO
9
10 connect_supply_net VDD -ports VDD
11 connect_supply_net VSS -ports VSS
12 connect_supply_net LDO_VOUT -ports LDO_VOUT
13
14 set_domain_supply_net PD_LDO -primary_power_net VDD -primary_ground_net VSS
15
16 create_power_switch PSW_LDO -domain PD_LDO \
17     -input_supply_port { vin VDD } \
18     -output_supply_port { vout LDO_VOUT } \
19     -control_port { en LDO_EN } \
20     -control_port { sl0 LDO_VOSEL[0] } \
21     -control_port { sl1 LDO_VOSEL[1] } \
22     -control_port { sl2 LDO_VOSEL[2] } \
23     -off_state { LDO_OFF {!en}} \
24     -on_state { LDO_ST0 vin {en && !sl0 && !sl1 && !sl2}} \
25     -on_state { LDO_ST1 vin {en && sl0  && !sl1 && !sl2}} \
26     -on_state { LDO_ST2 vin {en && !sl0 && sl1  && !sl2}} \
27     -on_state { LDO_ST3 vin {en && sl0  && sl1  && !sl2}} \
28     -on_state { LDO_ST4 vin {en && !sl0 && !sl1 &&  sl2}} \
29     -on_state { LDO_ST5 vin {en && sl0  && !sl1 &&  sl2}} \
30     -on_state { LDO_ST6 vin {en && !sl0 && sl1  &&  sl2}} \
31     -on_state { LDO_ST7 vin {en && sl0  && sl1  &&  sl2}} \
32     -output_voltage { LDO_ST0 0.6 } \
33     -output_voltage { LDO_ST1 0.7 } \
34     -output_voltage { LDO_ST2 0.8 } \
35     -output_voltage { LDO_ST3 0.9 } \
36     -output_voltage { LDO_ST4 1.0 } \
37     -output_voltage { LDO_ST5 1.1 } \
38     -output_voltage { LDO_ST6 1.15 } \
39     -output_voltage { LDO_ST7 1.2 }
```

LDO UPF model

Apply proposed syntax

Figure 8

Figure 9. Simulation waveform.

Proposed UPF Syntax for Special Supply Resolution:

***create_supply_net** net_name*
[-resolve <either | strong | weak>]

Worst-case scenario

Best-case scenario

Resolution mechanism for *–resolve weak*

Supply Source1 (sw1)	Supply Voltage (v1)	Supply Source2 (sw2)	Supply Voltage (v2)	Resolved Supply Net State	Resolved Supply Net Voltage
FULL_ON	1.5	FULL_ON	2.5	FULL_ON	1.5 (minimum of sw1 and sw2 voltage values)
FULL_ON	1.5	OFF	0	FULL_ON	1.5
OFF	0	FULL_ON	2.5	FULL_ON	2.5
OFF	0	OFF	0	OFF	0
PARTIAL_ON	0	FULL_ON	2.5	FULL_ON	0
FULL_ON	1.5	PARTIAL_ON	0	FULL_ON	0
PARTIAL_ON	0	OFF	0	PARTIAL_ON	0
OFF	0	PARTIAL_ON	0	PARTIAL_ON	0
PARTIAL_ON	0	PARTIAL_ON	0	PARTIAL_ON	0
UNDETERMINED	-	FULL_ON	2.5	FULL_ON	2.5
FULL_ON	1.5	UNDETERMINED	-	FULL_ON	1.5
UNDETERMINED	-	OFF	0	UNDETERMINED	0
OFF	0	UNDETERMINED	-	UNDETERMINED	0
UNDETERMINED	-	PARTIAL_ON	0	PARTIAL_ON	0
PARTIAL_ON	0	UNDETERMINED	-	PARTIAL_ON	0
UNDETERMINED	-	UNDETERMINED	-	UNDETERMINED	0

```
1  create_power_domain PD_TOP
2
3  create_supply_port VDD1
4  create_supply_port VDD2
5  create_supply_port VSS
6  create_supply_net VDD1 -domain PD_TOP
7  create_supply_net VDD2 -domain PD_TOP
8  create_supply_net VDD_TOP -domain PD_TOP -resolve weak
9  create_supply_net VSS -domain PD_TOP
10
11 connect_supply_net VDD1 -ports VDD1
12 load_upf LDO.upf -scope u_LDO
13 connect_supply_net VDD2 -ports { VDD2 u_LDO/VDD }
14 connect_supply_net VDD_TOP -ports u_LDO/LDO_VOUT
15 connect_supply_net VSS -ports { VSS u_LDO/VSS }
16
17 set_domain_supply_net PD_TOP -primary_power_net VDD_TOP \
18     -primary_ground_net VSS
19
20 create_power_switch PSW -domain PD_TOP \
21     -input_supply_port { vin VDD1 } \
22     -output_supply_port { vout VDD_TOP } \
23     -control_port { psw_en PSW_EN } \
24     -on_state { PSW_ON vin {psw_en} } \
25     -off_state { PSW_OFF {!psw_en} }
```

UPF for Figure 3

Apply proposed syntax

Connect output of LDO & PSW

Figure 10

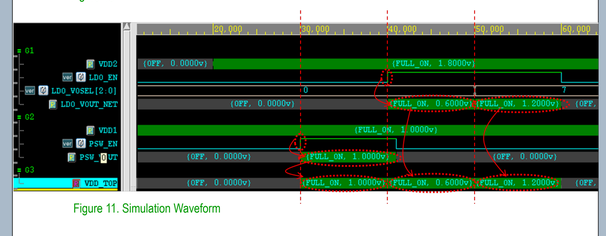

Figure 11. Simulation Waveform

Conclusion

- Low power verification is "must" to guarantee correct functionality after adding low power techniques
- Some low power techniques cannot be modeled using UPF like LDO and special supply net resolution
- Our existing solution using SystemVerilog modules, but …
  - Leave some verification holes related to coverage
  - Cumbersome to understand and maintain
- We propose extensions to UPF commands "*create_power_switch –output_voltage*" and "*create_supply_net –resolve weak/strong/either*"
  - Can model LDO and special supply net resolution
  - Make low power verification more robust
  - Have implemented these extensions into power-aware simulator MVSIN NLP
- We hope to donate these to the IEEE-1801 committee

Contact information

www.mediatek.com
Mediatek Inc.

No.1, Dusing 1st Rd.
Hsinchu Science Park
Hsinchu City 30078, Taiwan

T: +886-3-567-0766
Ext. 22794
E: maso.chen@mediatek.com