Unified Power Format

Power Design

Power Management

Fully-automated Synthesis of Power Management Controllers from UPF

The Long and Winding Road: How we can implement power management controllers in UPF…?

22.01.2019, Dustin Peterson | University of Tuebingen
Agenda

1. Concepts and Limitations of UPF
2. Extensions to the Unified Power Format
3. PMC Synthesis
4. Implementation + Results
5. Summary + Roadmap
Unified Power Format (UPF)

- Tcl-based power design specification language (IEEE-1801) supported by most commercial synthesis and simulation tools
- **Basic Paradigm:** Separation of Concerns (Power vs. Function)
• **Example**: How can we implement power gating by UPF?

We want this SLEEP signal to control the power gate!

```upf
cREATE_SUPPLY_PORT VDD
CREATE_SUPPLY_NET VDD_NET
CONNECT_SUPPLY_NET VDD_NET
  -PORTS {VDD}
CREATE_POWER_DOMAIN PD1
  -ELEMENTS {BIG_COMPONENT}
CREATE_SUPPLY_NET VVDD_NET
CREATE_POWER_SWITCH
  -INPUT_SUPPLY_PORT {VDD VDD_NET}
  -OUTPUT_SUPPLY_PORT {VDDG VVDD_NET}
  -CONTROL_PORT {SLEEP SLEEP}
  -ON_STATE {ON VDD !SLEEP}
  -OFF_STATE {OFF SLEEP}
SET_DOMAIN_SUPPLY_NET PD1
  -PRIMARY_POWER VVDD_NET
```
• But instead of just switching the sleep signal like this…

...you have to do a lot more stuff for each power domain!

Power Management Controller (PMC) undertakes this task!

Currently, this needs to be implemented in RTL code!
Goals of our Approach

• Keeping RTL clean from power management intent.
• Automatic synthesis of one or multiple power management controllers (PMC) from UPF + integration back into the SoC.
• PMC controlled by software (bus I/O) or hardware.

Extended UPF Script

1. Extended UPF Script

2. Power Management Synthesis

RTL Design

RTL Design with PMC
What UPF Extensions do we need?

• We need a way to specify…
  - …various power management controllers with several interfaces (AXI, APB, …) managing different devices!
  - …power state machines per power management controller!
What UPF Extensions do we need?

- We need a way to specify...
  - ...which actions we need to trigger during transition?
    
    "Disable SW_2, Enable Isolation, Enable Clock Gating, ..."

- ...and in what order do we need them to be triggered?
### Extended Unified Power Format

- **Starting Point:**
  Basic PMC I/O Specification  
  + List of Managed Devices

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Create a new power management controller in the current scope that implements the given power state machine.</th>
</tr>
</thead>
</table>
| Syntax  | **create_pmc** *pmc_name*  
  [-*devices* { *power_switch* | *voltage_source* }]*  
  [-*domain* *domain_name*]  
  [-*interface* < *plain* | *self* | *apb3* | *axi4lite* | *avalon* | *wishbone*>  
  [-*interface_param* { *name* *value* }]  
  [-*extend_interface* { *target_scope* }]  
  [-*clock* { *clock_signal* [ < *posedge* | *negedge*> ] }]  
  [-*reset* { *reset_signal* [ < *sync* | *async* > < *low* | *high*> ] }]  
  [-*scheduler* < *default* | *custom*> ] |
Extended Unified Power Format

- Existing commands to specify power state machine for each PMC:
  - `add_power_state`, `describe_state_transition`

- But: Some information is missing!
  - Initial State? Under which condition shall a transition be taken?

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Sets the initial state of a power management controller.</th>
</tr>
</thead>
</table>
| Syntax                      | `set_initial_state pmc_name
- state state_name`                                      |

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Enhanced version of IEEE-1801 <code>describe_state_transition</code> with support for transition conditions.</th>
</tr>
</thead>
</table>
| Syntax                      | `describe_state_transition transition_name
- object object_name
- from from_list
- to to_list
- paired (from_state to_state)*
[legal|illegal]
- condition (boolean_expression)`                                                               |

- `fifo_is_empty`
- `!fifo_is_empty`

- `PD_1=ON PD_2=ON`
- `PD_1=ON PD_2=OFF`
• Actions are triggered when switching between two power modes!
  - In what order do they need to be triggered?

  e.g. „enable clock gate before enabling isolation“

  ![Waveform Diagram]

• Three commands are intended for that:
  - `schedule_strategy`: Which strategy needs to be enabled/disabled in which device state / power state?
  - `update_strategy_schedule`: Strategy dependencies
  - `update_device_schedule`: Device dependencies

• Instead of an RTL signal, control handles (-control_port of isolation, retention, … shall reference a PMC object!)
A lot of more commands have been specified to:
- Specify clock gating strategies.
- Specify reset strategies.
- Specify voltage sources.
- Map voltage sources to IP cells.
- Specify (pipeline stall) signals and when to trigger them.
- Specify existing power gating implementations (memory IPs…)

Latest Specification is available here:
http://bit.do/xupf
• **Power Management Synthesis:** Transform an RTL design + xUPF specification into a fully power-managed SoC platform!

   - (1) Model Generation
     - Supply Network Model
     - Power State Machine Model
     - Power Management Controller Configuration

   - (2) Feature Scheduling
     - Annotated Power State Machine Model

   - (3) IP Generator
     - Power Management Controller IP (Verilog, VHDL, SystemC)

   - (4) IP Integrator
     - RTL Design (power-aware)
     - Standard-Compliant UPF

Fully-automated Synthesis of Power Management Controllers from UPF

**Power Management Synthesis**

1. **Model Generation**
   - Supply Network Model
   - Power State Machine Model
   - Power Management Controller Configuration

2. **Feature Scheduling**
   - Annotated Power State Machine Model

3. **IP Generator**
   - Power Management Controller IP (Verilog, VHDL, SystemC)

4. **IP Integrator**
   - RTL Design (power-aware)
   - Standard-Compliant UPF

**Basic Supply Network Model:** Specifies voltage sources (blue), power switches (green) and their connections (arrows).

**Voltage Source State Tables:** Specify the available voltages for each voltage source.

<table>
<thead>
<tr>
<th>VDD</th>
<th>Port</th>
<th>State 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT</td>
<td>1.0V</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VSS</th>
<th>Port</th>
<th>State 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT</td>
<td>0.0V</td>
<td></td>
</tr>
</tbody>
</table>

**Power Switch State Tables:** Specify to which input supply net and output supply net is connected in a dedicated state.

| SW0 | VDD Net → VVDD Net [VVDD Net is disconnected] |

**Supply Network Model**

**Power State Machine Model**

Power Management Synthesis

1. Model Generation
   - Supply Network Model
   - Power State Machine Model
   - Power Management Controller Configuration

2. Feature Scheduling
   - Annotated Power State Machine Model

3. IP Generator
   - Power Management Controller IP (Verilog, VHDL, SystemC)

4. IP Integrator
   - RTL Design (power-aware)
   - Standard-Compliant UPF

---

Scheduled Power State Machine Model

Latest Specification is available here: http://bit.do/xupf
Power Management Synthesis

(1) Model Generation
- Supply Network Model
- Power State Machine Model
- Power Management Controller Configuration

(2) Feature Scheduling
- Annotated Power State Machine Model

(3) IP Generator
- Power Management Controller IP (Verilog, VHDL, SystemC)

(4) IP Integrator
- RTL Design (power-aware)
- Standard-Compliant UPF

Connects to SoC!
Connects to UPF Objects!

Power Management Controller IP (Verilog/VHDL)

Latest Specification is available here:
http://bit.do/xupf
Implementation as Synopsys Design Compiler Plugin

- Implementation of a full-blown UPF library in Scala/Java including a power management synthesis tool!

- Integration into Design Compiler using a Tcl interface!

```
analyze -library ${library} -format verilog ${verilog_sources}
elaborate ${top_design}
source -echo ${sdc_input}
uniquify

# Synthesize power management
--------------------------------------------------------------------------------
#load_upf ${upf_file}
pmc_compile ${upf_file} -output ${pmc_output_directory} -write_sim_files
compile Ultra -no_autoungroup
```

Latest Specification is available here: http://bit.do/xupf
**Example:** RISC-V SoC with a PMC managing a counter. PMC managed by Software (APB bus).

```
PMC_switch_state(ALL_OFF);
```
Software-managed Power Management Example in UPF

- Example: VexRiscV SoC with a PMC managing a counter. PMC managed by Software (APB bus).

```c
create_pmc PMC -domain TOP_PD -devices {PG_SW}
  -clock {io_mainClk posedge} -reset {io_asyncReset async high}
  -interface apb3
  -interface_param {addr_width 20}
  -interface_param {data_width 32}
  -interface_param {base_address 0xf0030000}
  -interface_param {use_slave_error false}
```

```c
add_power_state TOP_PD/PMC \n  -state {ALL_ON -logic_expr {PG_SS == ON_STATE}} \n  -state {ALL_OFF -logic_expr {PG_SS == OFF_STATE}}
set_initial_state TOP_PD/PMC -state ALL_ON
describe_state_transition -object TOP_PD/PMC \n  PSM_transition0 -from ALL_ON -to ALL_OFF
describe_state_transition -object TOP_PD/PMC \n  PSM_transition1 -from ALL_OFF -to ALL_ON
```

Latest Specification is available here: http://bit.do/xupf
• **Example**: VexRiscV SoC with a PMC managing a counter. PMC managed by Software (APB bus).

```c
# Isolation managed by PMC
set_isolation PG_ISO -domain PG_PD -clamp_value 0 \\
   -isolation_sense high \\
   -isolation_signal TOP_PD/PMC

# Retention managed by PMC
set_retention PG_RET -domain PG_PD \\
   -save_signal TOP_PD/PMC low \\
   -restore_signal TOP_PD/PMC high

# Clocks managed by PMC
```

Controls driven by PMC!

```c
# Need to activate strategies when power switch is off!
schedule_strategy TOP_PD/PMC -state PG_SW.PG_SW_OFF \\
   -strategy PG_PD/PG_ISO \\
   -strategy PG_PD/PG_RET \\
   -strategy PG_PD/PG_CG \\
   -strategy PG_PD/PG_DR

Strategies to enable during power-off!
```

Latest Specification is available here: http://bit.do/xupf
Software-managed Power Management Example in UPF

• Results after PMC synthesis:

<table>
<thead>
<tr>
<th>Bare-Meta C Drivers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bare_Metal_drivers</td>
</tr>
<tr>
<td>documentation.pdf</td>
</tr>
<tr>
<td>little_soc.pmc.vcd</td>
</tr>
<tr>
<td>PMC.c</td>
</tr>
<tr>
<td>PMC.h</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UPF for MentorGraphics QuestaSim</th>
</tr>
</thead>
<tbody>
<tr>
<td>little_soc.upf</td>
</tr>
<tr>
<td>little_soc.v</td>
</tr>
<tr>
<td>little_soc.vsim_upf</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UPF for Synopsys Design Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>little_soc.upf</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bare-Meta C Drivers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bare_Metal_drivers</td>
</tr>
<tr>
<td>documentation.pdf</td>
</tr>
<tr>
<td>little_soc.pmc.vcd</td>
</tr>
<tr>
<td>PMC.c</td>
</tr>
<tr>
<td>PMC.h</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UPF for MentorGraphics QuestaSim</th>
</tr>
</thead>
<tbody>
<tr>
<td>little_soc.upf</td>
</tr>
<tr>
<td>little_soc.v</td>
</tr>
<tr>
<td>little_soc.vsim_upf</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UPF for Synopsys Design Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>little_soc.upf</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>The following power management controllers have been created:</td>
</tr>
<tr>
<td>• /PMC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location: /PMC</td>
</tr>
<tr>
<td>Bus Interface: APB3</td>
</tr>
<tr>
<td>Base Address: 0x00300000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory Map</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>0x003000</td>
</tr>
<tr>
<td>0x0030004</td>
</tr>
<tr>
<td>0x0030008</td>
</tr>
<tr>
<td>0x003000c</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Power States</th>
</tr>
</thead>
<tbody>
<tr>
<td>State Bit Width: 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNDEFINED</td>
<td>0x0</td>
</tr>
<tr>
<td>ALL_OFF</td>
<td>0x1</td>
</tr>
<tr>
<td>ALL_ON</td>
<td>0x2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software Driver</th>
</tr>
</thead>
<tbody>
<tr>
<td>The software drivers are available here:</td>
</tr>
</tbody>
</table>

Latest Specification is available here: http://bit.do/xupf
Software-managed Power Management Example in UPF

Software uses automatically generated bare-metal drivers!

```c
#include <stdint.h>
#include <muran.h>
#include "././././syn/output/bare_metal_drivers/PMC.h"

void main() {
    int i = 0;
    PMC_switch_state_b(PMC_ALL_OFF);
    for(i=0; i < 100; i++){
        PMC_switch_state_b(PMC_ALL_ON);
        for(i=0; i < 100; i++){
            PMC_switch_state_b(PMC_ALL_OFF);
        }
        // Caution: Please don't use Timer B in any code and do not remove this code.
        // The following code is for the testbench which checks that timer B will reach the limit (io_full will go to high).
        prescaler_init(TIMER_PRESCALER);
        TIMER_PRESCALER->LIMIT = 1;
        timer_init(TIMER_B);
        TIMER_B->LIMIT = 8;
        TIMER_B->CLEAR_S = 0x00010002;
    }
    while(1);
}
```

Latest Specification is available here: http://bit.do/xupf
Summary:
• UPF is currently not able to capture all aspects of dynamic power management techniques.
• xUPF might be a good hint what we need in future UPF revisions!
• Synthesis tool for xUPF is already available.

Roadmap:
• ASIC-readyness shall be proven in a tapeout in Q1/2020!
• Scala framework planned to be published open-source next year!
• Proposals to IEEE-1801 committee…

Latest Specification is available here:
http://bit.do/xupf
Thank you.
Dustin Peterson
Eberhard Karls Universität Tübingen
Lehrstuhl für Eingebettete Systeme
Fon: +49 7071 - 29 – 75458
Fax: +49 7071 - 29 – 5062
E-Mail: dustin.peterson@uni-tuebingen.de
Power Management Synthesis

- Implementation evaluated with Synopsys 32nm EDK and GlobalFoundries 22FDX + Invecas 8-Track RVT Library.

![Area Results for 22FDX and differently sized power designs with multiple fully-meshed power state machines](chart.png)

- Software-controlled PMC has been integrated into Pulpino, managing a timing tracer and multiple SRAM blocks.
- Our implementation supports dynamic body biasing, so we plan to manage the body bias by our PMC in the tape-out, too.
- Integration of xUPF into power design exploration under development!
Example: VexRiscV SoC with a PMC managing a counter. PMC managed by Software (APB bus).

Basic Power Management Specification

```plaintext
create_pmc PMC -domain TOP_PD -devices {PG_SW} -clock {io mainClk posedge} -reset {io asyncReset async high} 
    interface apb3 -interface_param {io addr_width 20} -interface_param {data_width 32} 
    interface_param {use slave_error false} -interface_param {base address 0xf0030000}
```

Power State Machine

```plaintext
add_power_state TOP_PD/PMC
    -state {ALL_ON -logic_expr {PG_SS == ON_STATE}}
    -state {ALL_OFF -logic_expr {PG_SS == OFF_STATE}}
set_initial_state TOP_PD/PMC -state ALL ON
describe_state_transition -object TOP_PD/PMC
    PSM transition0 -from ALL_ON -to ALL_OFF
```

Managed Strategies

```plaintext
# Isolation managed by PMC
set_isolation PG_TISO -domain PG_PD -clamp_value 0
    -isolation_reduced
    -isolation_signal {TOP_PD/PMC}
# Retention managed by PMC
set_retention PG_RET -domain PG_PD
    -save_signal {TOP_PD/PMC low}
    -restore_signal {TOP_PD/PMC high}
# Clock Gating managed by PMC
set_clock_gating PG_CG -domain PG_PD
    -type and -location inbound -clock sense posedge -control {TOP_PD/PMC}
# Power Domain Reset managed by PMC
set_domain_reset PG_DR -domain PG_PD
    - resets {u_big_block/resetCtrl_systemReset}
    -location inbound -reset_type {async high}
    -control {TOP_PD/PMC}
```

Connect PMC to Bus Interconnect

```plaintext
connect_logic_net apb3Router_1/io outputs_3 PADDR
    -ports TOP_PD/PMC/1 apb_paddr reconnect
connect_logic_net apb3Router_1/io outputs_3 PSEL
    -ports TOP_PD/PMC/1 apb_psel reconnect
```

Latest Specification is available here: http://bit.do/xupf
**Power Management Synthesis**

1. **Model Generation**
   - Supply Network Model
   - Power State Machine Model
   - Power Management Controller Configuration

2. **Feature Scheduling**
   - Annotated Power State Machine Model

3. **IP Generator**
   - Power Management Controller IP (Verilog, VHDL, SystemC)

4. **IP Integrator**
   - RTL Design (power-aware)
   - Standard-Compliant UPF

- Instantiation and Connection of PMC IP
- Synthesis of Clock Gating, Domain Reset and Stall Propagation Logic
- Rewriting of UPF Script

---

**Power Managed SoC Design**

**Interconnected! automatically**

Configured in UPF, generated automatically in Verilog***

*** Currently supported and generated interfaces: AXI4-Lite, APB3, Avalon-MM, Wishbone

**Latest Specification is available here:**
http://bit.do/xupf