SystemC extension for power specification, simulation and verification

Mikhail Moiseev[1], Ilya Klotchkov[1],
Maxim Petrov[2], and Kirill Gagarski[2]
1) Intel Corporation, USA
2) St. Petersburg Polytechnic University, Russia

Motivation

- Low power design requires special techniques
  - Power gating
  - Memory power control
- These techniques are present in HAS and in RTL, intermediate SystemC codes also should has them
- Problems in SystemC simulation without power properties
  - Mismatches in SystemC and RTL simulation
  - No support of memory power modes
  - Generated RTL and manually created UPF
  - No verification of power rules, incorrect RTL possible

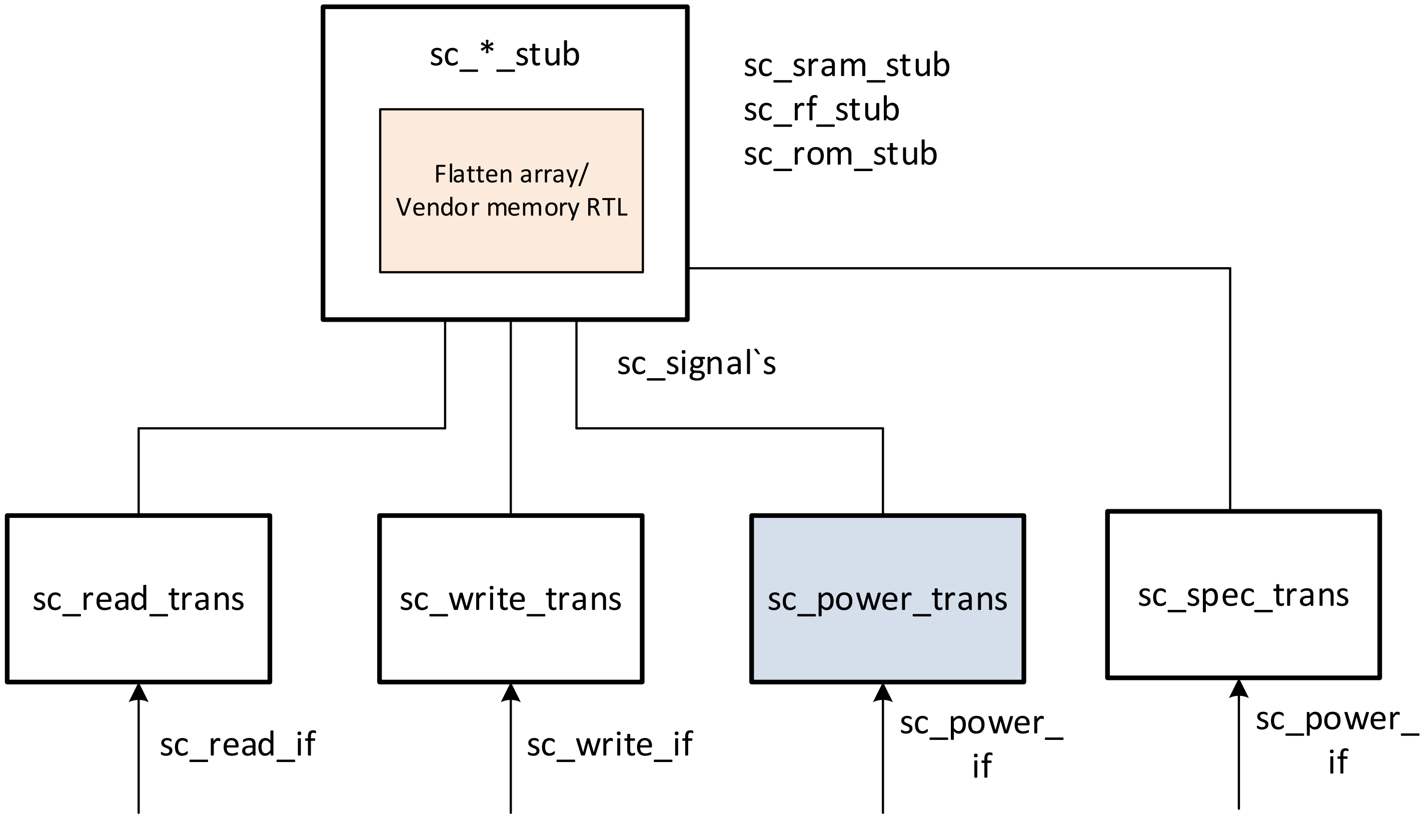
SC Power extension

- SC Power provides power management in synthesizable SystemC designs
  - Power specification in SystemC code
  - Power aware SystemC simulation
  - UPF file generation
  - Run-time checking of power related rules
- SC Power is an extension of OSCI SystemC kernel
  - Contains template classes
  - No SystemC kernel modification

Power specification primitives

- Design top module `sc_top_module`
  - Specifies module as top module for UPF
  - Stores all registered domains, memory modules and others
- Power domain `sc_domain`
  - Contains one or more modules included into power domain
  - Provides `power_on/power_off` methods
- Domain input/output isolation `sc_in_iso/sc_out_iso`
  - Used instead of `sc_in/sc_out`
- Isolation strategy `sc_isolation`
  - Specifies isolation values for elements given by UPF-style pattern
  - Provides `enable/disable/isEnabled`
- Power supply signal `sc_supply_sig`

Memory support



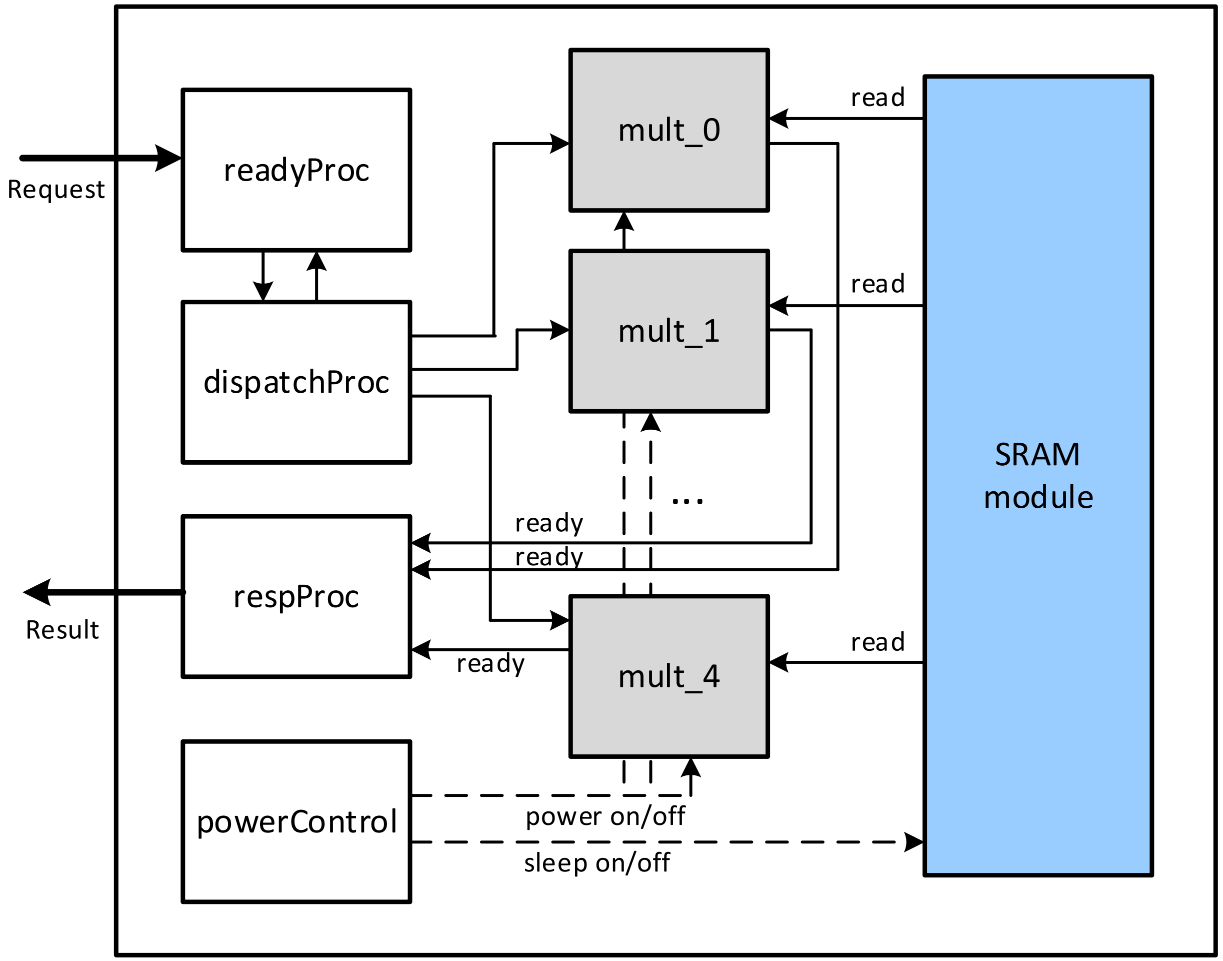
Memory stub and transactors

- Memory stubs, power FSM with delay specified
  - ROM/RF/SRAM memory with shutdown and sleep modes
  - Checking power control rules
- Memory transactors
  - Power transactor has `shutdwn_on/shutdwn_off` and `sleep_on/sleep_off` methods
    - Control power state of the memory
    - Checking memory state and waiting for power acknowledge
  - Read and write transactors
    - Function interface for access from `SC_METHOD/SC_CTHREAD`
  - Memory specific parameter transactor

SystemC Simulation

- Power-aware SystemC simulation
  - Disable processes in power off domains, process cannot corrupt retention state like memory
  - Apply isolation values for power off domain inputs/outputs, isolation values may differ from reset values
  - Check accesses to power off memory
    - warning provided
  - Check read before write, such access returns undefined value
    - memory cells are filled with `0xDEADBEEF`
  - Check reset state for power on/off domain modules, incorrect reset state may lead to bad state after power up
    - warning provided

Case study



Request
readyProc
dispatchProc
mult_0
mult_1
...
mult_4
read
read
read
SRAM module
ready
ready
ready
respProc
Result
powerControl
power on/off
sleep on/off

Power specification example

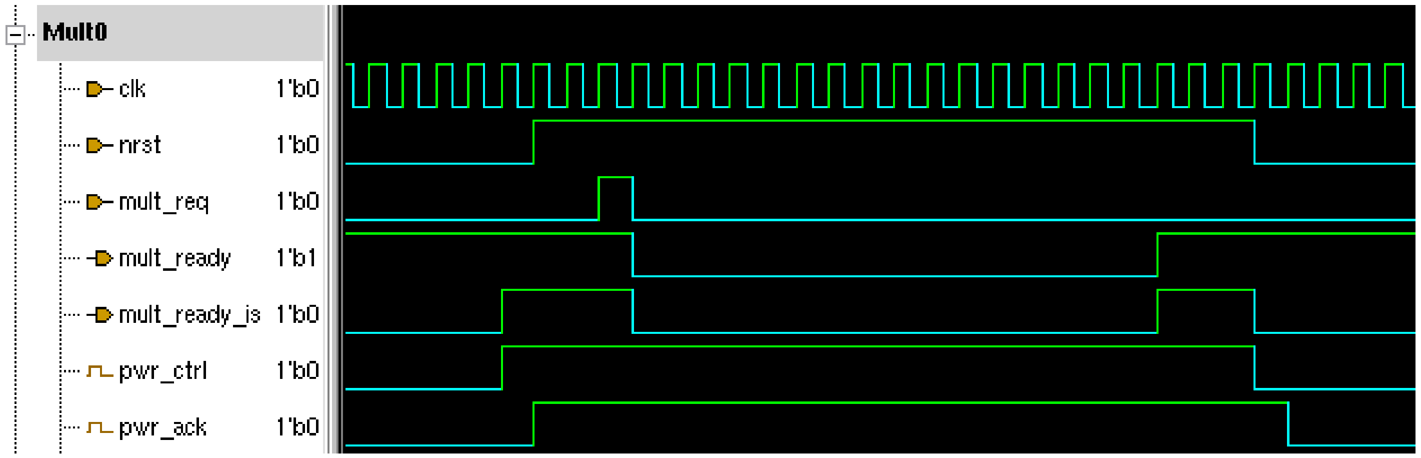
```
// Register supply signals
register_supply(vdd);
register_supply(vdd_core);
register_supply(vss);

// Domain creation
sc_power::sc_domain* d = new sc_power::sc_domain("mult_1",
                                      vdd, vss, "1 ns");
register_domain(d);

// Isolation creation, isolation value is 0
sc_power::sc_isolation* iso = new sc_power::sc_isolation
                                  ("mut_iso_1", 0);
d->register_isolation_strategy(iso);

// SRAM module instantiation
sram_stub = new sc_mem::sc_sram_stub<AWIDTH, DWIDTH, TRAITS>
                ("sram", vdd, vdd_core, vss);
```

SystemC Simulation example

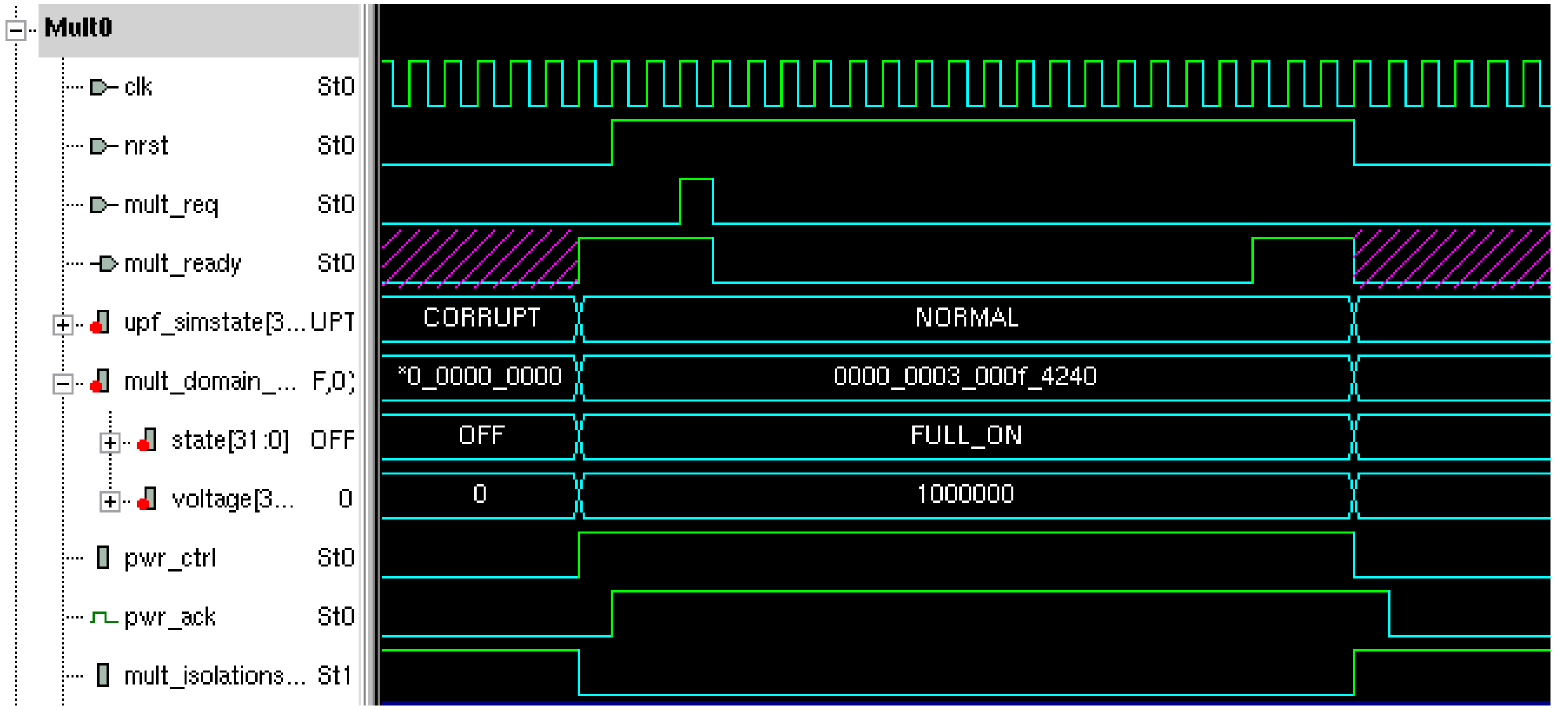


SystemC simulation

`pwr_ctrl` – domain power control signal
`mult_ready` – ready output before isolation
`mult_ready_is` – ready output after isolation

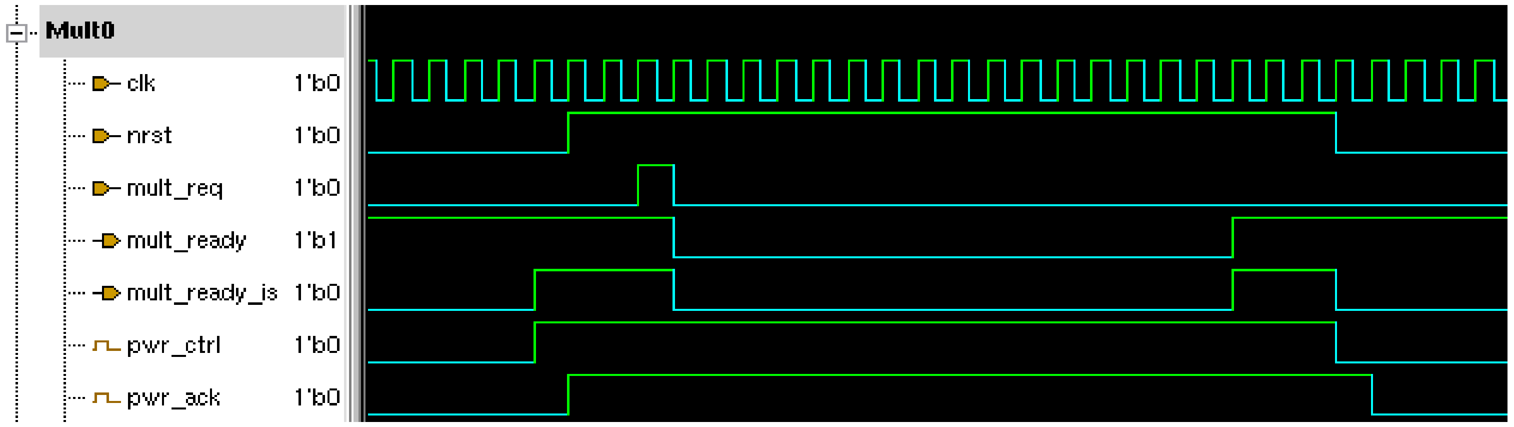
RTL+UPF Simulation example

- UPF file with isolation for `mult_ready` output

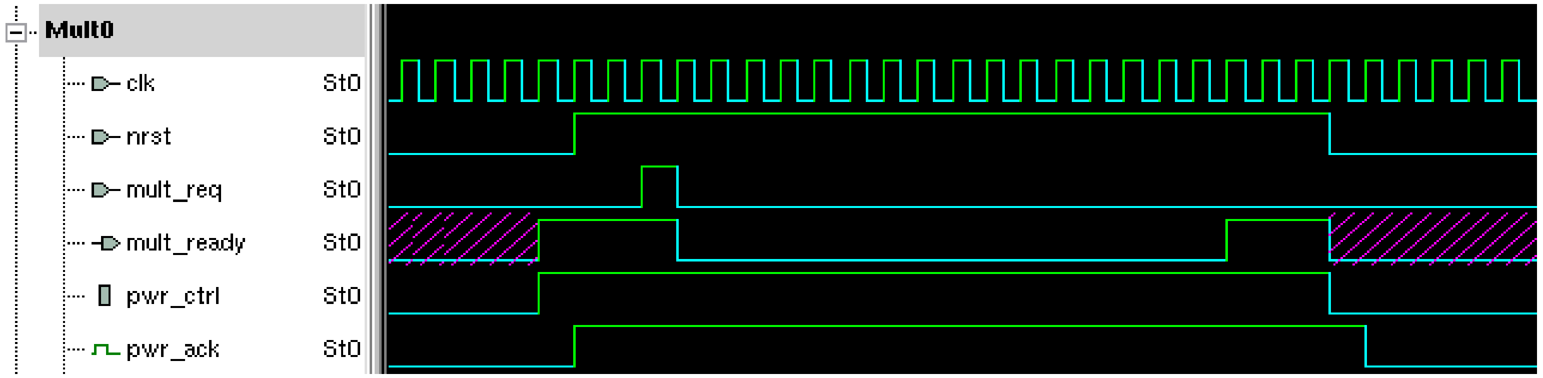


RTL simulation with UPF

Simulation equivalence



SystemC simulation

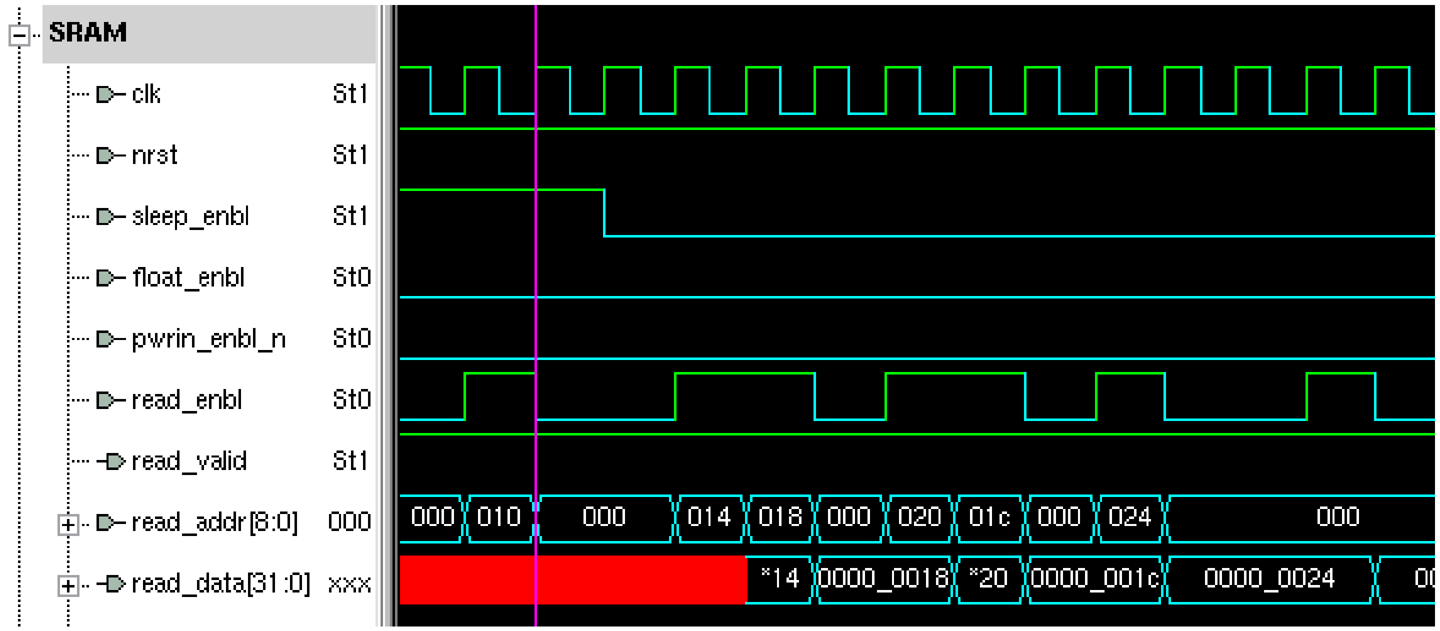


RTL simulation with UPF

Incorrect memory access example

- Access to SRAM in sleep mode warning

```
245 ns, WARNING: soc.sram_stub access SRAM in SLEEP
```

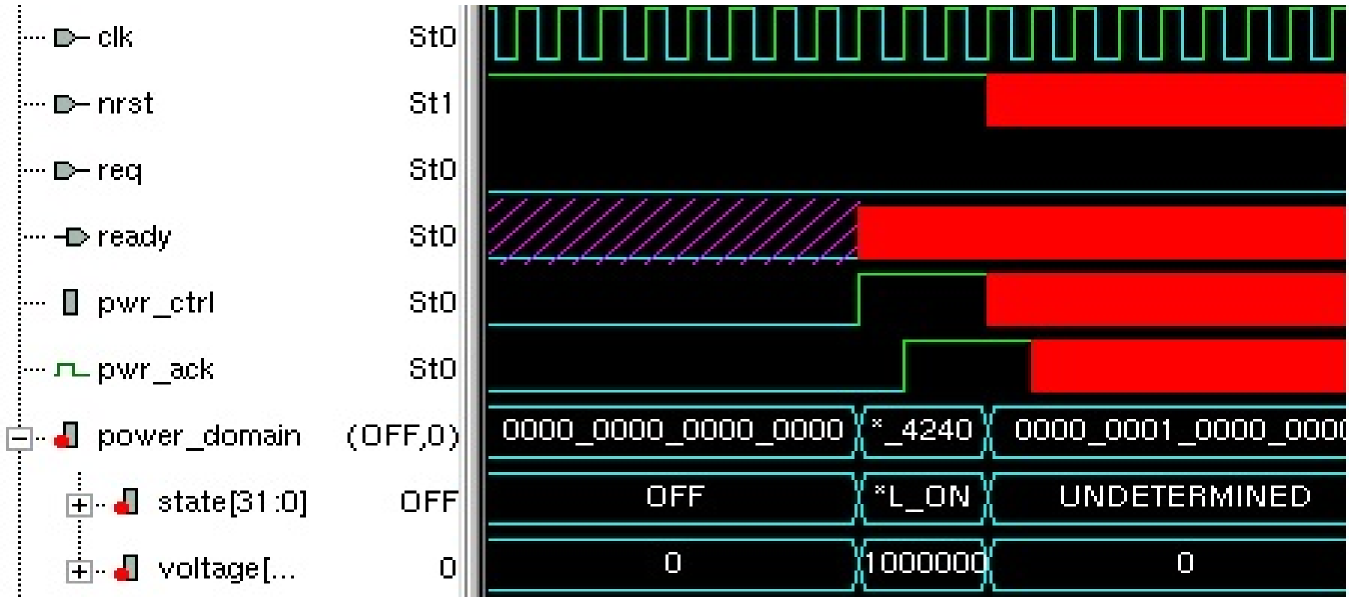


RTL simulation

Incorrect reset example

- No reset asserted in domain power up

```
150 ns, WARNING: No reset for domain module
mult_domain_1 in power up
```



RTL simulation

Power debugging API

- Iterate over domains/memories/power supplies in the top module
- Get domain/memory/power supply by name from the top module
- Get domain/memory state string representation
- Iterate over isolation strategies in the domain
- Iterate over ports in the isolation strategy

UPF generation

- SCPower provides automatic generation of UPF file for the power specification injected
  - Set design top module and create always-on domain
  - Create power domains and power switch per domain
  - Create power supply for domains and memories
  - Connect power and logic nets
  - Create isolation strategies for domains
    - Several strategies per domain supported

UPF example, power domain

```
create_power_domain mult_domain_0
    -elements {mults_0}
create_supply_net vdd_top
    -reuse
    -domain mult_domain_0
create_supply_net vss_top
    -reuse
    -domain mult_domain_0
set_domain_supply_net mult_domain_0
    -primary_ground_net vss_top
    -primary_power_net mult_domain_0_from_switch
```

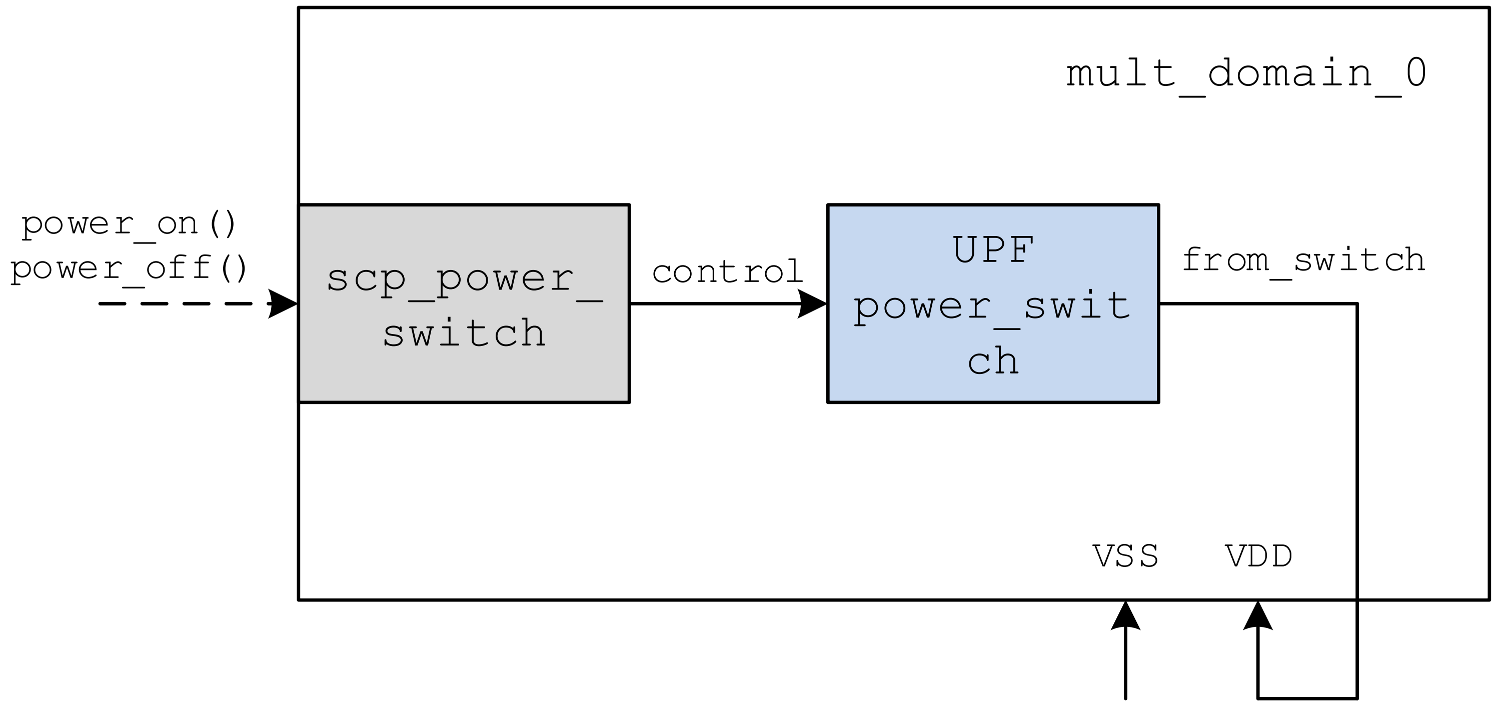
UPF example, power switch

```
create_logic_net mult_domain_0_control
connect_logic_net mult_domain_0_control
    -ports mult_domain_0_scp_power_switch/control_
create_power_switch mult_domain_0_power_switch
    -off_state {state_off !a}
    -on_state {state_on vcc_in a}
    -ack_delay {o 1ns}
    -ack_port {o mult_domain_0_ack_signal a}
    -control_port {a mult_domain_0_control}
    -input_supply_port {vcc_in vdd_top}
    -output_supply_port {gtdout mult_domain_0_from_switch}
    -domain mult_domain_0
```

Power switch inside of SCPower

```
class sc_domain_ctl : public sc_module, public sc_domain_ctl_if {
    sc_out<bool> power_control;
    sc_domain_ctl(const sc_module_name& name) : sc_module(name) {}
    void power_on() {power_control = 1;}
    void power_off() {power_control = 0;}
};
class sc_domain : public sc_module, public sc_domain_ctl_if {
    sc_domain_ctl ctl;
    sc_domain(...) : ... {
        ctl.power_control(control_signal);
        ctl.ack(ack_signal);
        scp_power_switch.control(control_signal);
        scp_power_switch.ack(ack_signal);
    }
    void power_on() {ctl.power_on();}
    void power_off() {ctl.power_off();}
};
```

UPF example picture



Implementation issues

- SC Power is developed w/o SystemC kernel modification
- Correct constructor names (name equals instance name) is required to provide correct UPF
  - No name checking
  - Issues with names for array elements
- SystemC to Verilog name translation cannot be strictly specified – causes to possible issues with flattened modules
  - "module/element" and "module_element" options

Conclusion

- Power management of separate modules and whole SoC
  - Efficient for design with big amount of SystemC code and power control features
- Future plans
  - Global power FSM specification and verification
  - Distributed power management unit support
  - Software power control support

Questions