RTL-Agent Switch

Implementation and Applications

Aman Arora, Nathan Wooster, Pavan Mula, Rob Porter
NVIDIA

September 18, 2015
Austin

Agenda

Problem Statement

RTL-Agent Switch

Applications

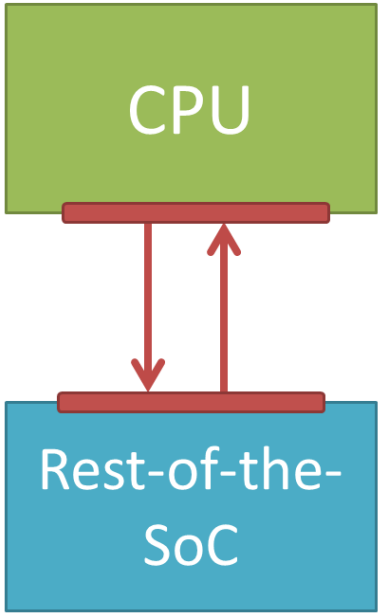
Discussion

Problem Statement

Problem Statement

What are we trying to solve?

- Common challenges for SOC verification
  - Time consuming simulations
  - Multiple IP's delivered at different times
  - Difficult to control and observe deeply embedded interfaces
- Stimulate/respond to internal interface in a DUT
  - Stub out pieces of RTL
  - Expose interfaces to chip boundary
- This implies
  - Creating new testbench variants
  - Modifying the DUT
  - Static configuration
- There is a better solution…



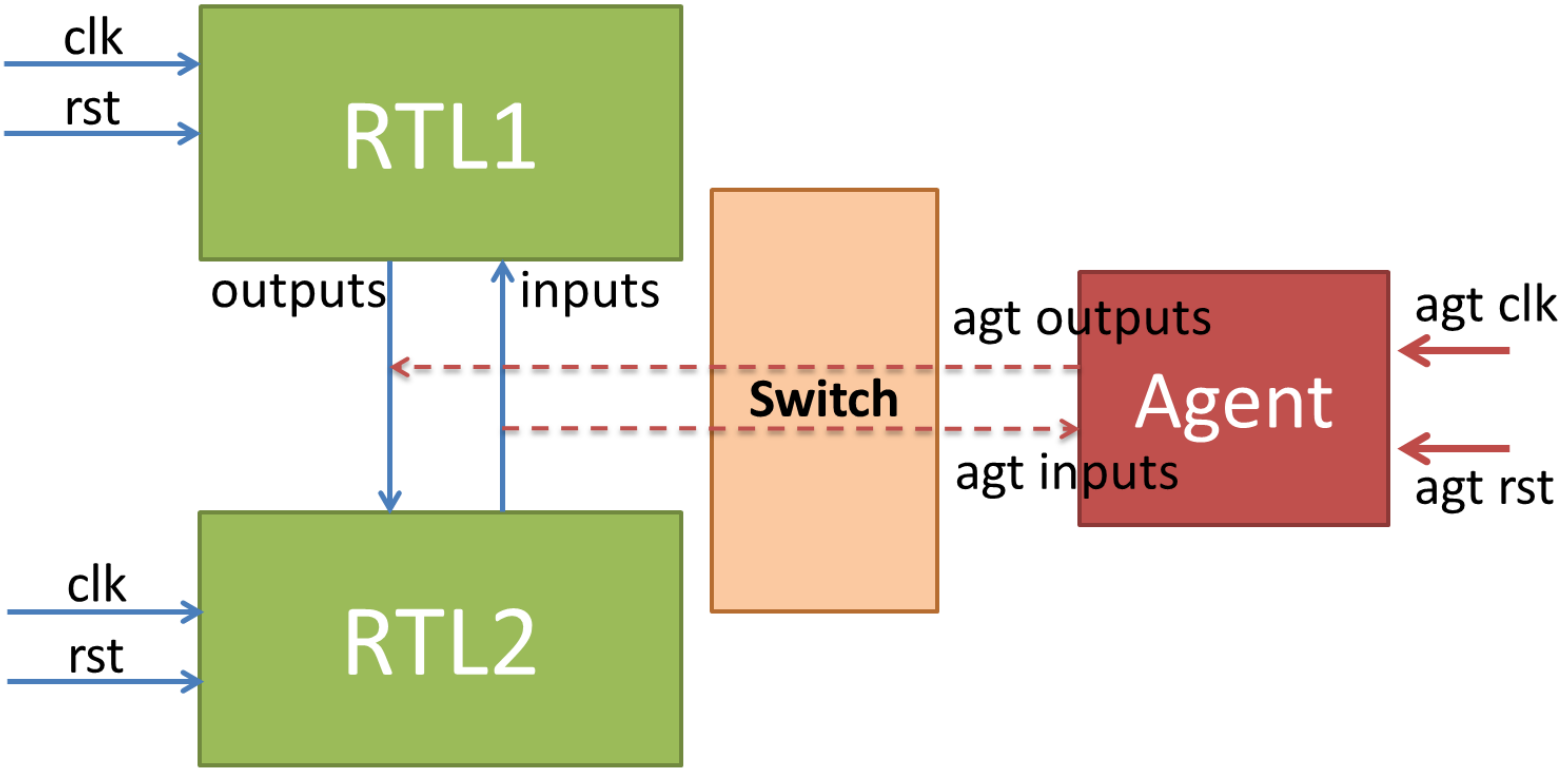
RTL-Agent Switch

What is it and how does it work?

RTL-Agent Switch

What is it?

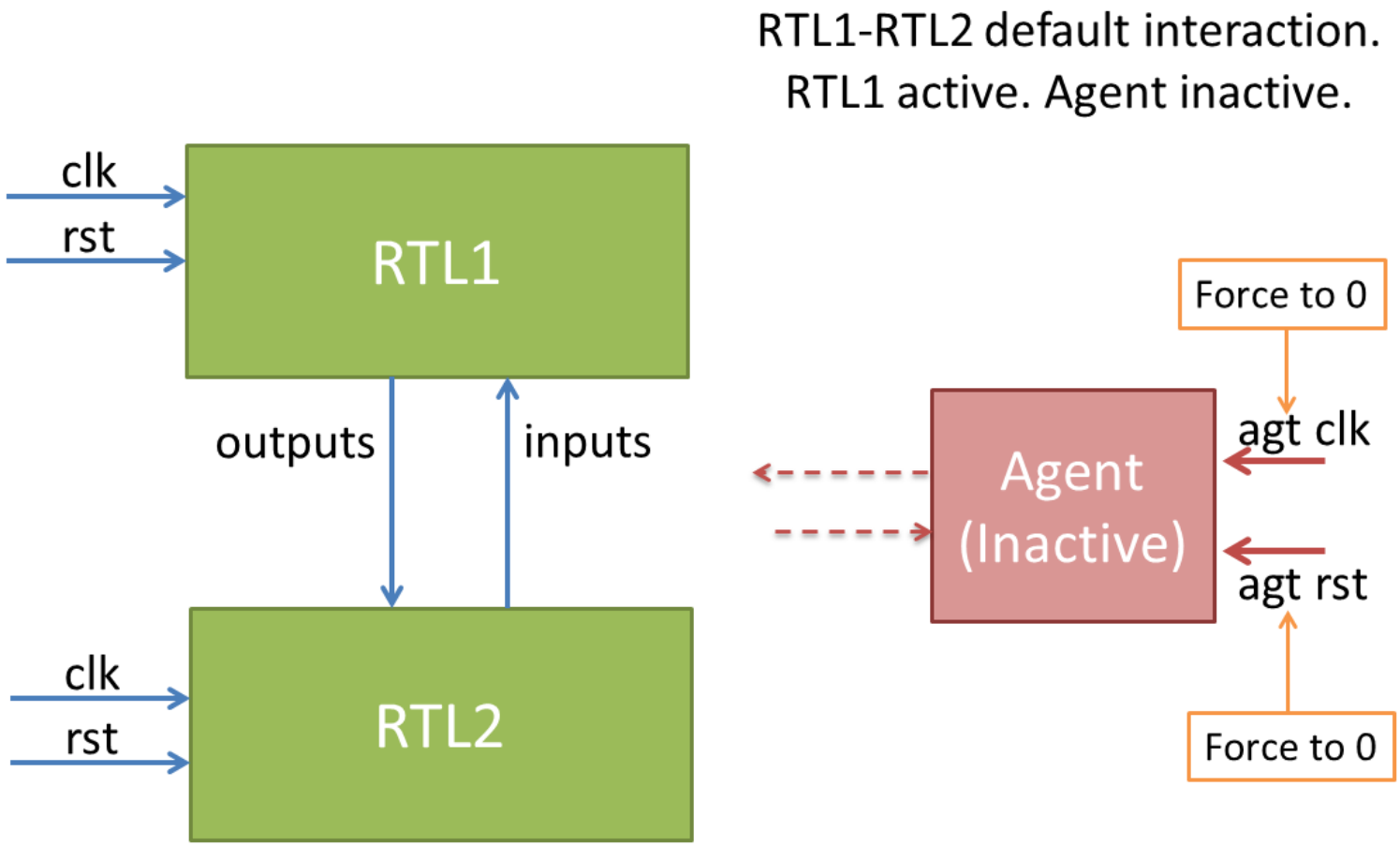
- A verification component that allows us to attach an agent to an interface in a DUT without modifying the design



RTL-Agent Switch

How does it work?

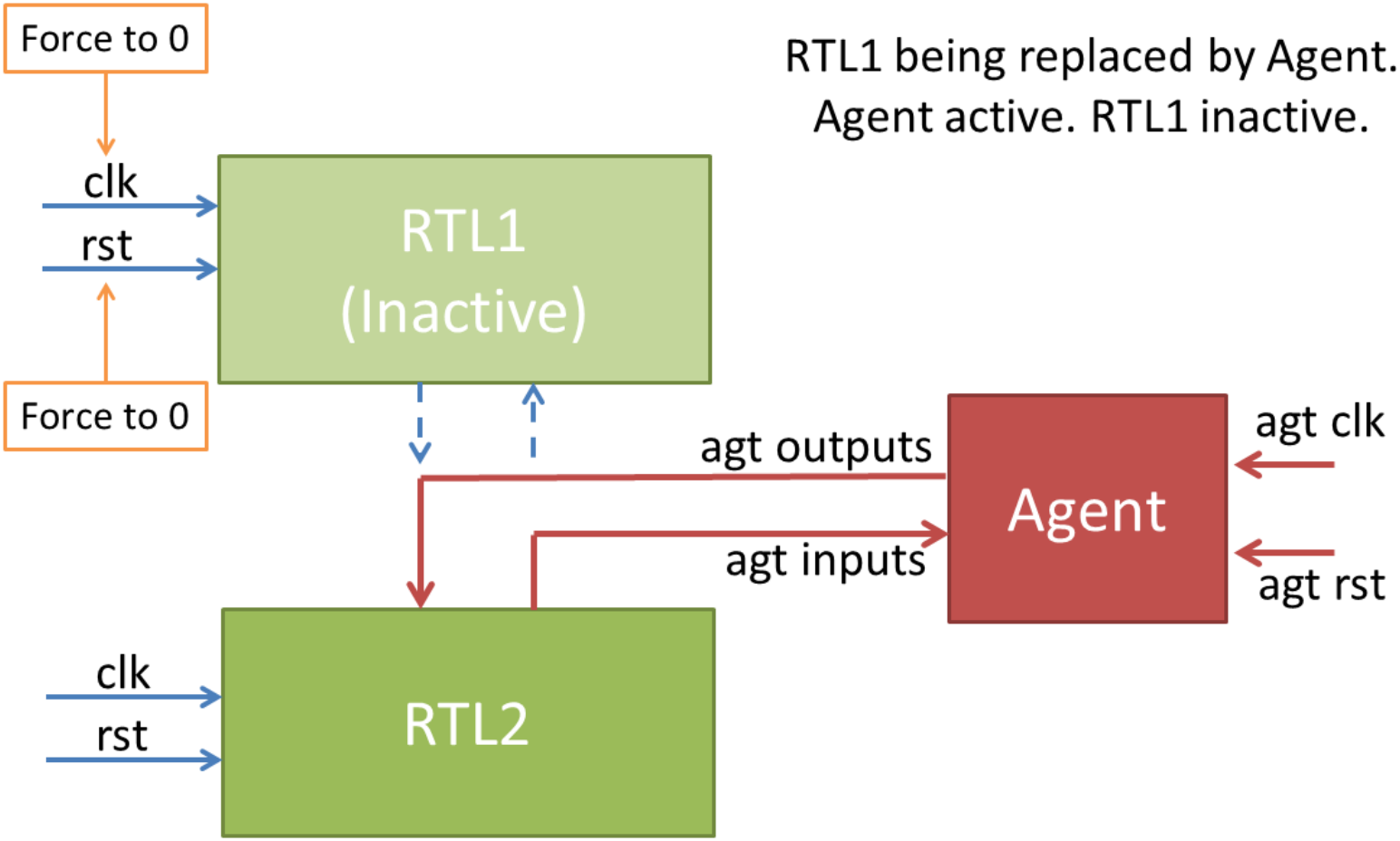
- RTL only mode



RTL-Agent Switch

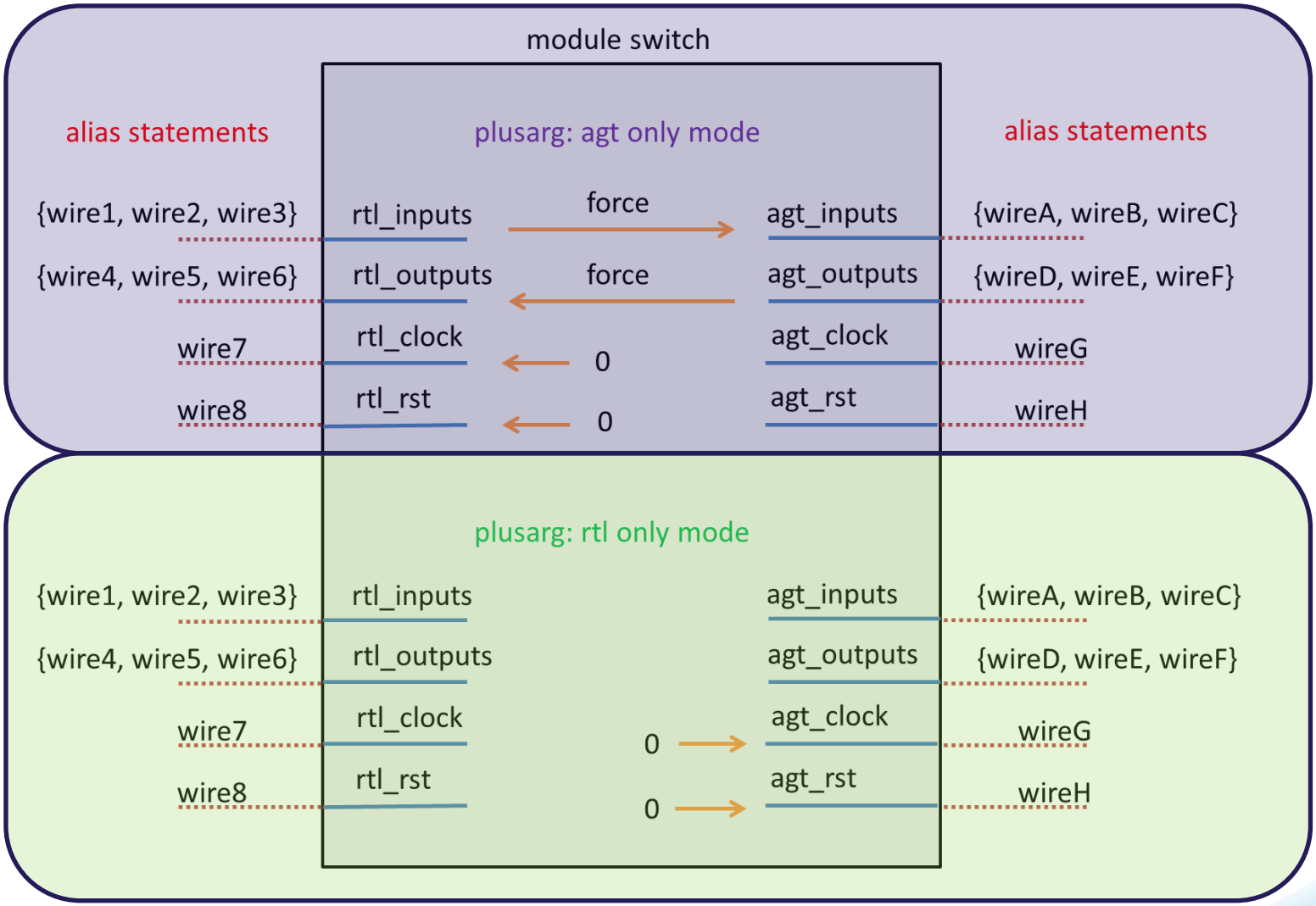
How does it work?

- Agent only mode



RTL-Agent Switch

How is it implemented?



RTL-Agent Switch

How do I use it?

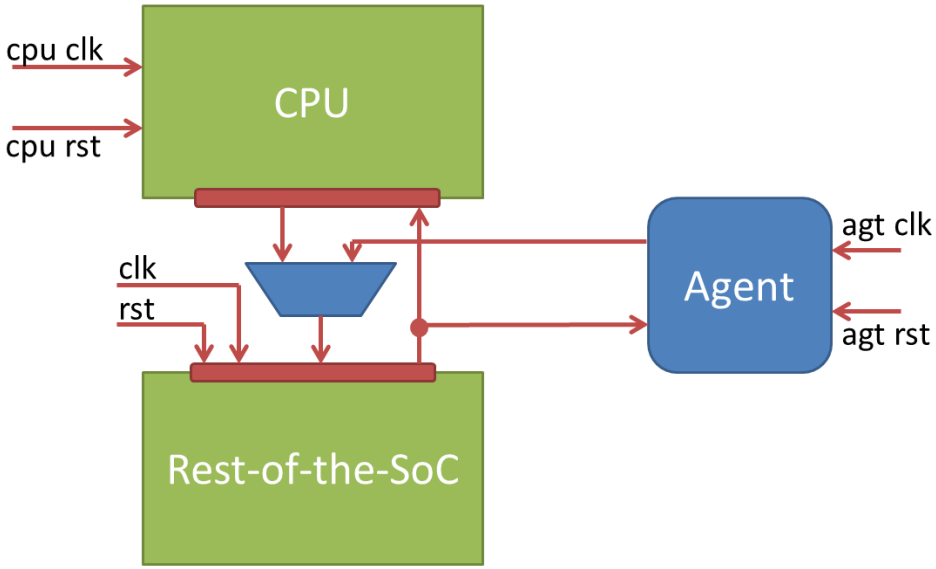
- Code the switch module
- Instantiate it, along with the agent
- Compile
- Run - use plusargs to select a mode

Applications

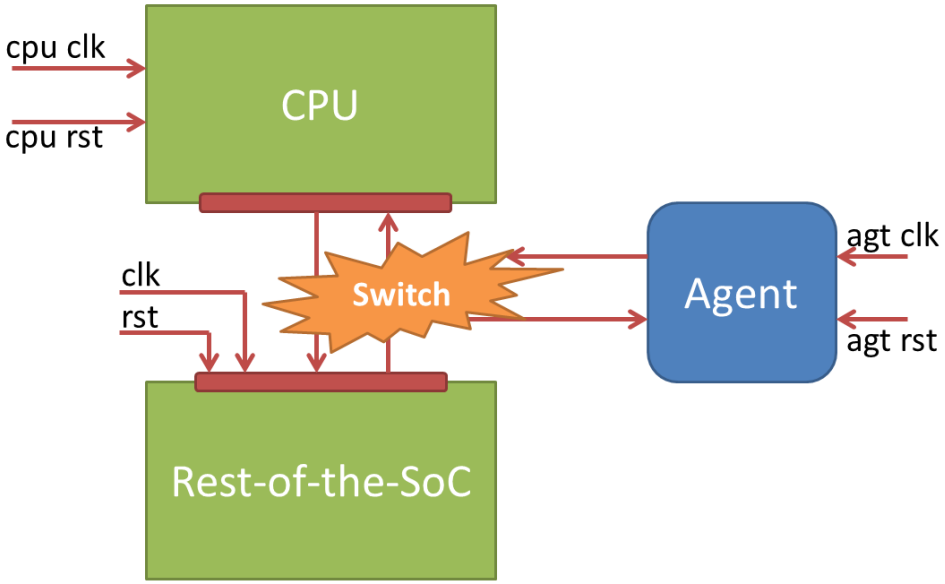
Where can I use it?

Injecting traffic into SOCs

Using a mux to drive an SOC with an agent

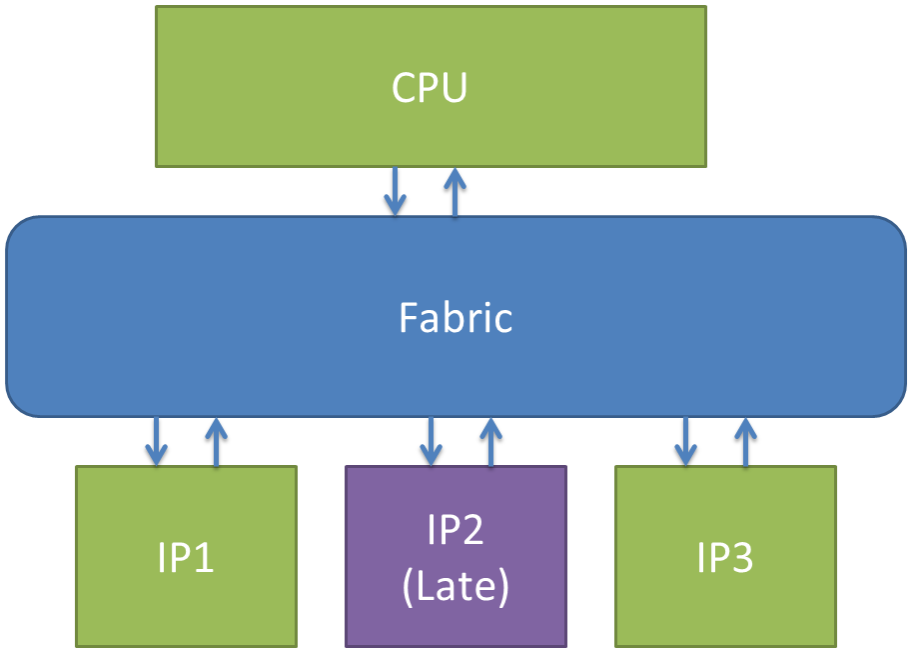


Using the switch to drive an SOC with an agent

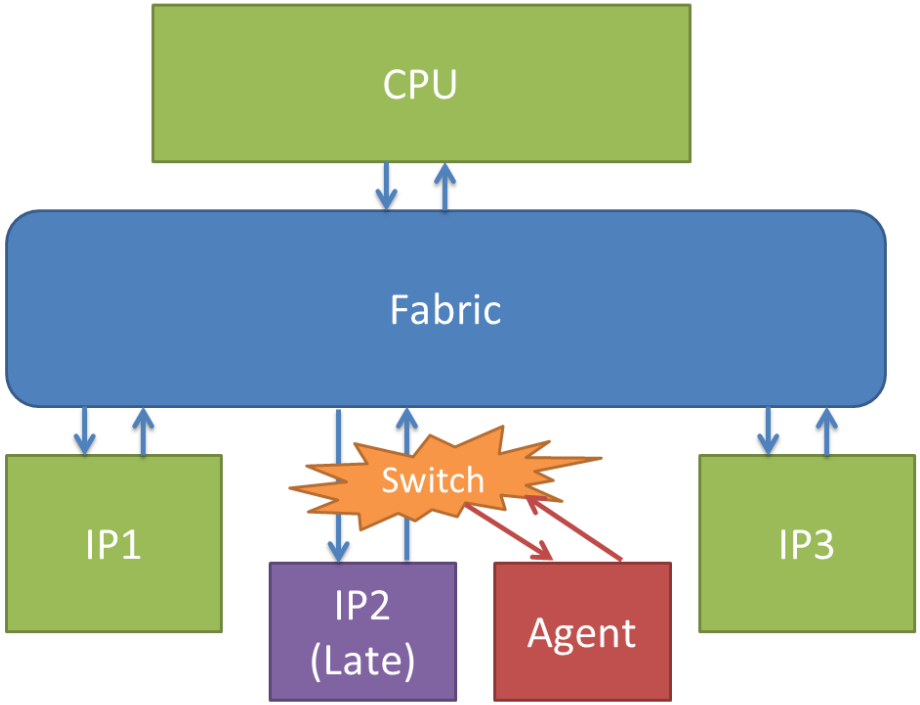


RTL components not ready

IP2 delivery late, delaying SOC verif schedule



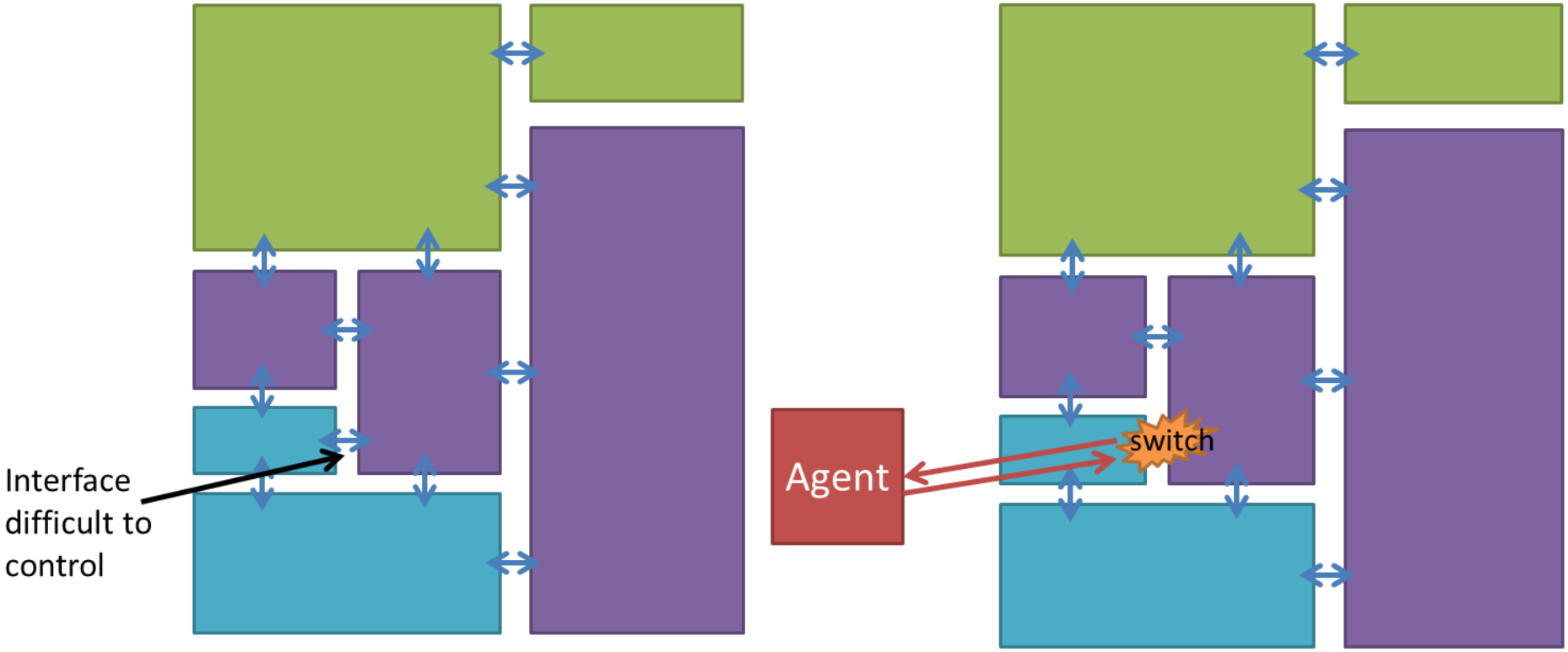
IP2 replaced with an agent using the switch



Ease of traffic generation

Difficult to control interface between blocks

Use the switch to control the interface with an agent



Discussion

Is there any supporting data? What can't it do?

Discussion

Results

- Stimulation of Tegra SOC
  - Used an AXI master agent injecting traffic through the switch
  - Simulation time greatly improved (about 14 hours to 1.5 hours)
  - Build time didn’t have any noticeable difference

- RAM wrapper delivered late
  - Replaced with an AXI slave agent using the switch
  - 2-3 days of extra effort to set it up. Save ~1 month of SOC verif schedule
  - Test development unhindered
  - Change command line when RTL available

Discussion

Results

- GPU's interface with the rest of the DUT
  - Difficult to control to get coverage
  - Replaced with a GPU model (agent)
  - Coverage closure became easier
  - Sanity tests run with GPU RTL in place

Discussion

Challenges

- Forcing a wire forces everything connected to it

```
ip1 u_ip1(.port1(wire));
ip2 u_ip2(.port2(wire));
```

- Width of aliased signals should match

```
wire dummy = {6'b0, sig3};
alias switch.rtl_input = { sig1, sig2, dummy, sig4 };
```

Discussion

Limitations and Drawbacks

- Only works for port-accurate models

- Slight degradation in build time and simulation memory footprint because both agent and RTL are present in the same simv

Discussion

Enhancements to the original idea

- Multiple instances of the switch

```
parameter unique_string = "";

string agt_only_mode = $psprintf("%s_agt_only_mode", unique_string);

if ($test$plusargs(agt_only_mode)) begin
end
```

Parameter on the module

Now you can say:

```
simv +abc_agt_only_mode +def_rtl_only_mode +other_args
```

Can control different switches separately

Discussion

Enhancements to the original idea

- Dynamic switching from agent to RTL

```
if ($test$plusargs(mix_mode)) begin
        //code for agent only mode
        force a = b;
        force c = d;
        //wait for event
        @(posedge change_mode);
        //code for rtl only mode
        release a;
        release c;
end
```

When “change_mode” goes high, mode switches from agent mode to RTL mode

Example: Boot using agent, Test using RTL

Discussion

Future Work

- Replacing multiple interfaces of a block
- System level simulations with all-but-one block in RTL
- Try with other type of models, eg. SystemC models

Thank You