Specification-Driven Automated Conformance Checking for Virtual Prototype and Post-Silicon Designs

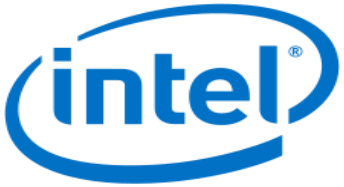
Haifeng Gu[1], Mingsong Chen[1], Tongquan Wei[1], Li Lei[2], and Fei Xie[3]

[1]Shanghai Key Lab of Trustworthy Computing, East China Normal University, China

[2]Intel Labs, United States

[3]Deptpartment of Computer Science, Portland State University, United States

華東師範大學
EAST CHINA NORMAL UNIVERSITY



Portland State
UNIVERSITY

DAC 2018, San Francisco, United States

Outline

- Motivation
- Formal Device Model Generation
  - Syntax Extensions for SystemRDL
  - Automated Generation of Formal Device Models
- Specification-Driven Conformance Checking
  - Device Trace Collection
  - Conformance Checking with FDMs
- Performance Evaluation Results
- Conclusion

Virtual Prototypes Are Increasingly Used

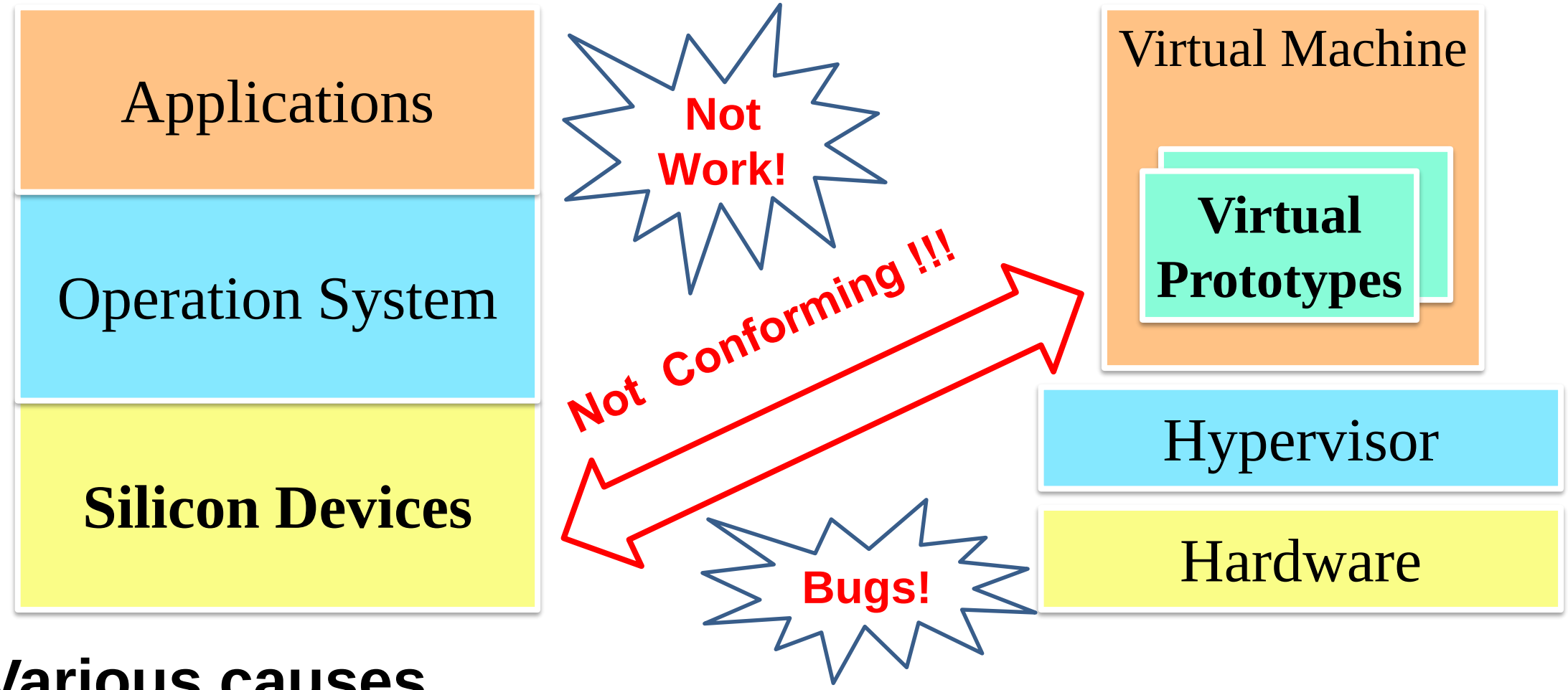
- **Virtualization prototyping can**
  - **Maximize device utilization, e.g., could computing**
  - **Improve production efficiency, e.g., ESL design**

Silicon Prototype

```
static void
set_phy_ctrl(E1000State *s, int index, uint16_t val)
{
    if ((val & MII_CR_AUTO_NEG_EN) && (val & MII_CR_RESTART_AUTO_NEG)) {
        e1000_link_down(s);
        s->phy_reg[PHY_STATUS] &= ~MII_SR_AUTONEG_COMPLETE;
        DBGOUT(PHY, "Start link auto negotiation\n");
        qemu_mod_timer(s->autoneg_timer, qemu_get_clock_ms(vm_clock) + 500);
    }
}

static void
e1000_autoneg_timer(void *opaque)
{
    E1000State *s = opaque;
    if (!qemu_get_queue(s->nic)->link_down) {
        e1000_link_up(s);
    }
    s->phy_reg[PHY_STATUS] |= MII_SR_AUTONEG_COMPLETE;
    DBGOUT(PHY, "Auto negotiation is completed\n");
}

static void (*phyreg_writeops[])(E1000State *, int, uint16_t) = {
    [PHY_CTRL] = set_phy_ctrl,
};

enum { NPHYWRITEOPS = ARRAY_SIZE(phyreg_writeops) };
```

Virtual Prototype

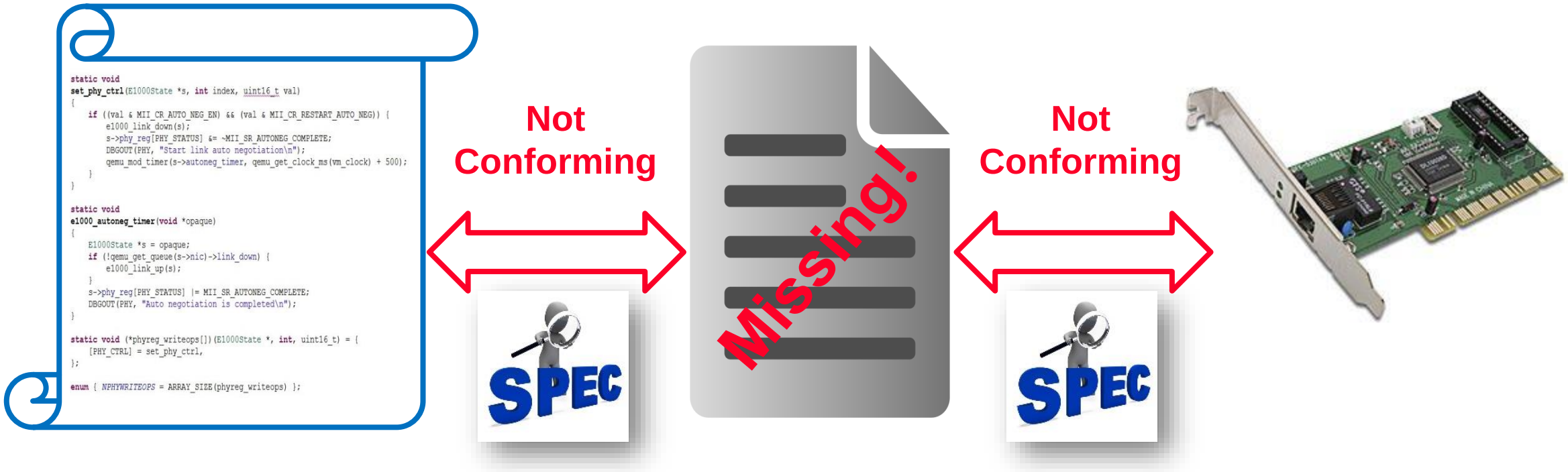
Challenges to be Addressed



- **Various causes**
  - Software bugs hidden on virtual prototypes, silicon hardware bugs
  - Observability is limited in silicon troubleshooting

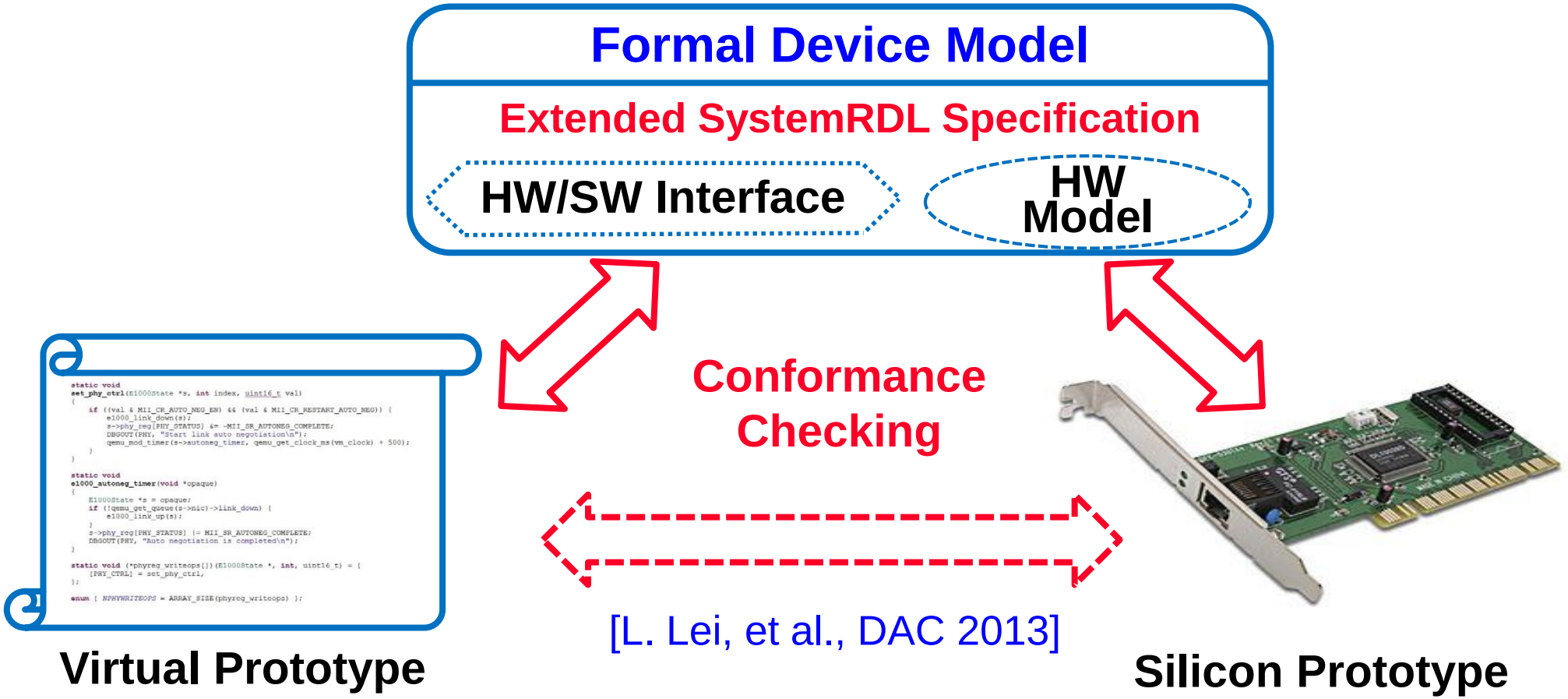
Things Are Even Worse …

- **Correctness of virtual/silicon devices cannot be guaranteed**
  - **Due to lack of golden reference models!**



Specification-Driven Conformance Checking

- **Formal device model generation using SystemRDL**
  - **Pros: Consistent register definitions across abstraction levels**
  - **Cons: Lack of register behavior modeling mechanisms**



L. Lei et al. Post-silicon Conformance Checking with Virtual Prototypes, *DAC 2013.*

Syntax Extensions for SystemRDL

- **Define macro and function components for registers' logics**
  - Macro facilitates the programming of function components
  - Function supports behavior modeling of interface registers

component_def ::= new_comp_body | component_type

component_name {{[component_def;] [property;][. . . ;]} * };

component_type ::= field | reg | regfile | addrmap | signal | enum | macro | function

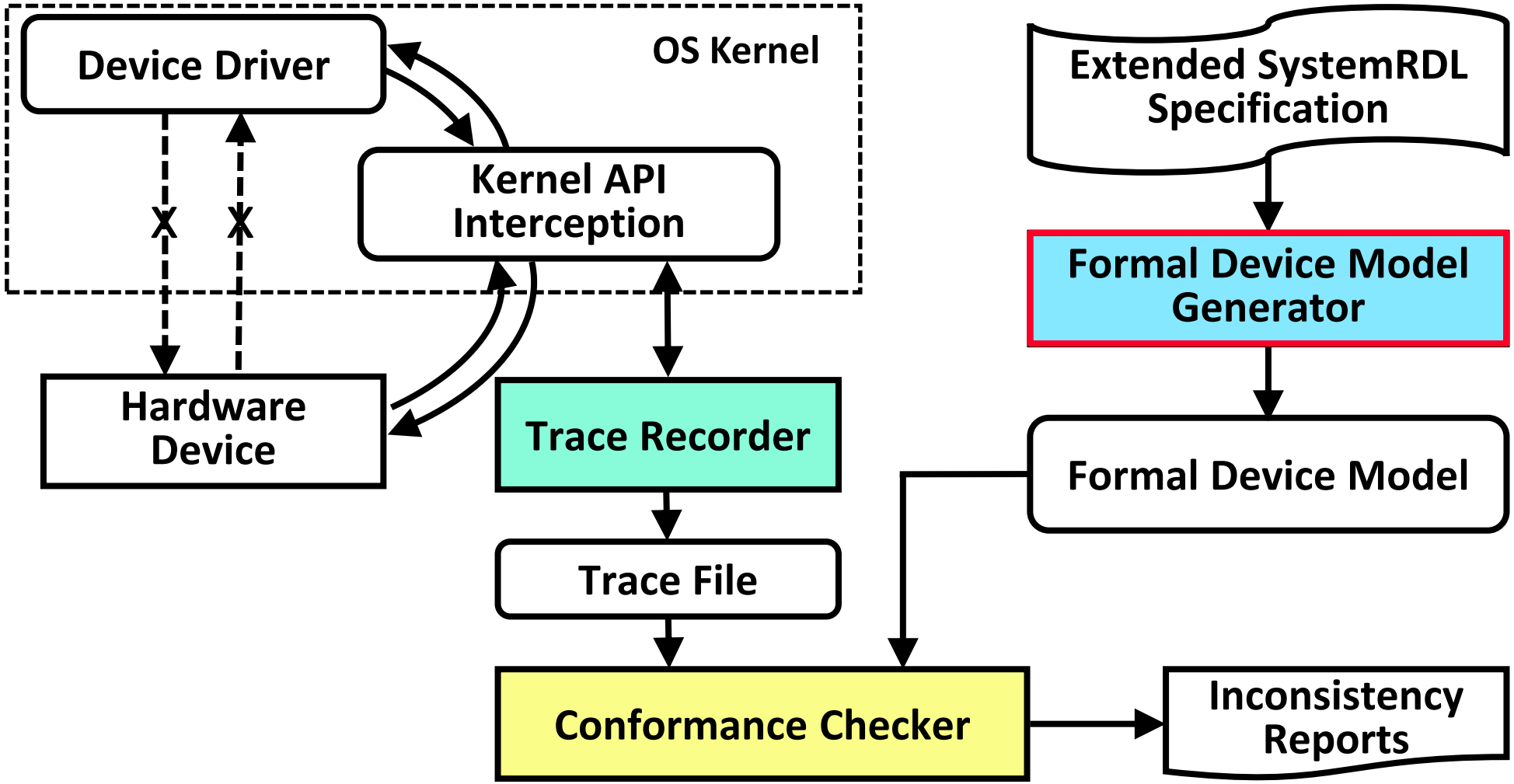
property ::= property_name = value;

new_comp_body ::= comp_body | macro_body | function_body

macro_body ::= {macro_name = value;}*

function_body ::= fun_type fun_name (argument_list) { statements }

Our Framework



Automated Generation of FDMs

Extended SystemRDL Specification

```
macro general_macro{
   E1000_TCTL_EN=0x00000002;
   …   };
reg Reg_TCTL_type {
   default sw=rw;
   regwidth = 32;
   accesswidth = 32;
   field {  desc = "…";  sw=rw;  swwe=true; } Rsv[0:0] = 0;
   …
   function syn_tctl_func {
      void write_tctl(DevState* ds, uint32_t val, uint64_t offset) {
            ds->this.value = val;
            ...        }
    };
   function asyn_tctl_func {
     void run_tctl(DeviceState* ds) {
       ds->reg_TDT_inst.value &= 0xffff;
       if(!(ds->this.value & E1000_TCTL_EN))   return;
       if(fdm_choice()) { … }
       …       }
  };
};
…
Reg_TCTL_type  inst1  @0x00400;
```

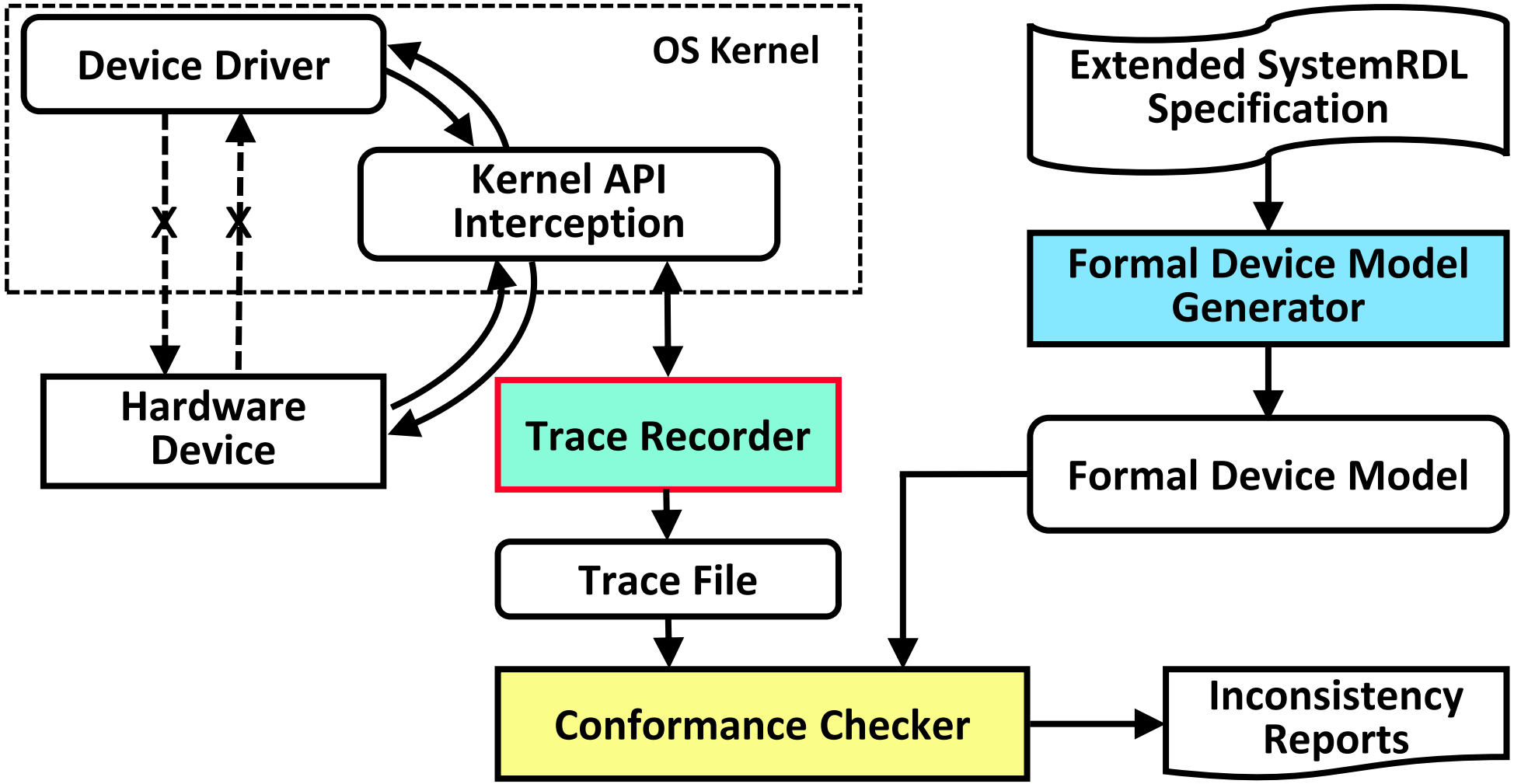
Transformation Rules

Register Behavior Modeling

```
void run_tctl_inst1(DevState* ds) {
      ds->reg_TDT_inst.value &= 0xffff;
      if(!(ds->inst1.value & E1000_TCTL_EN))
            return;
      if(fdm_choice()) { … }
      ...
}
...
void runDeviceTransactions(DevState* ds){
      ...
      switch (offset) {
          case 0x00400 : //TCTL
                run_tctl_inst1(ds); break;
                ...
      }
}
```

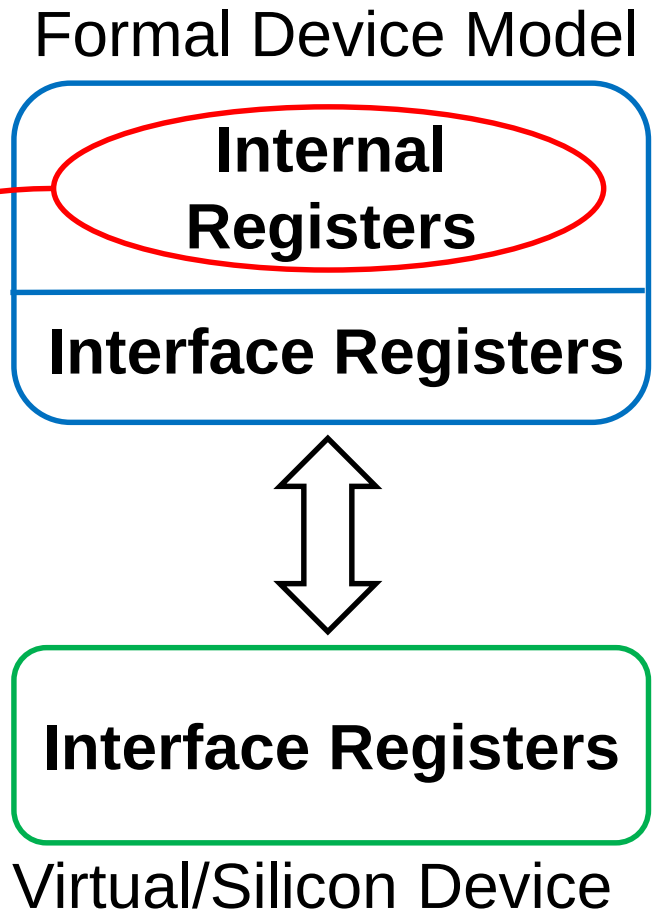
- A set of 10 rules is proposed for the transformation to FDMs

Our Framework



FDM State & Virtual/Silicon Device State

- **Formal Device Model state**
  - $F = <F_i, F_n>$
  - F_i: interface register state
  - F_n: internal register state

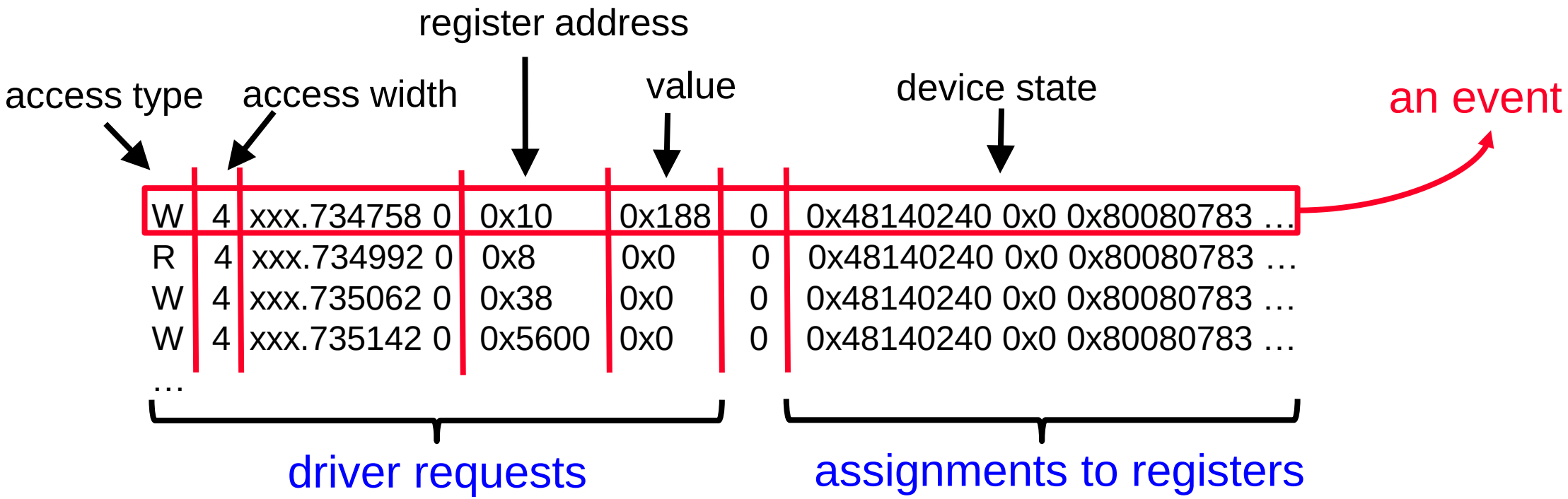


- **Virtual/Silicon device state**
  - S_i: interface register state

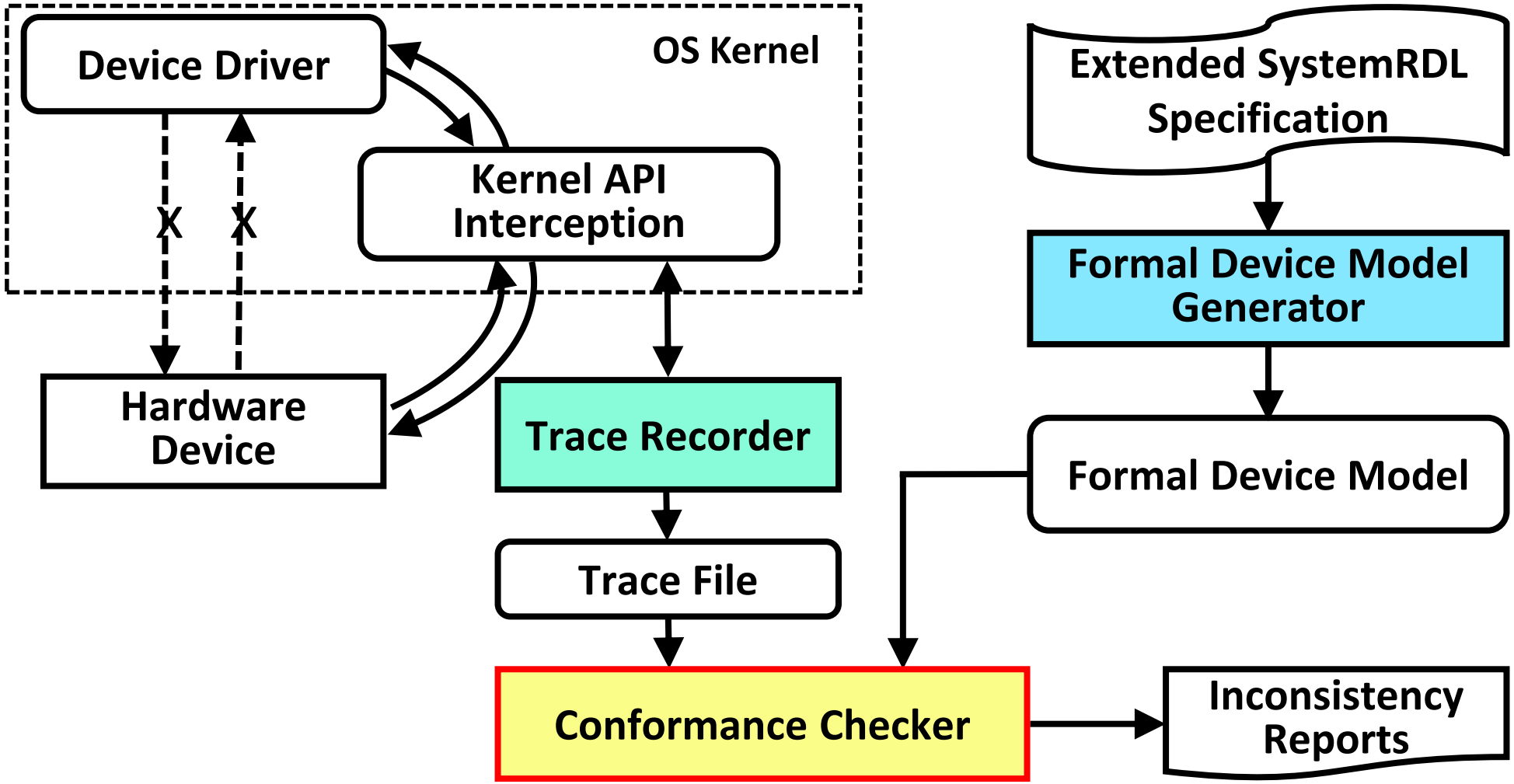
- **Definition of Conformance**
  - An FDM state F and a virtual/silicon device state S_i conform to each other if $S_i \models F$.

Device Trace Collection

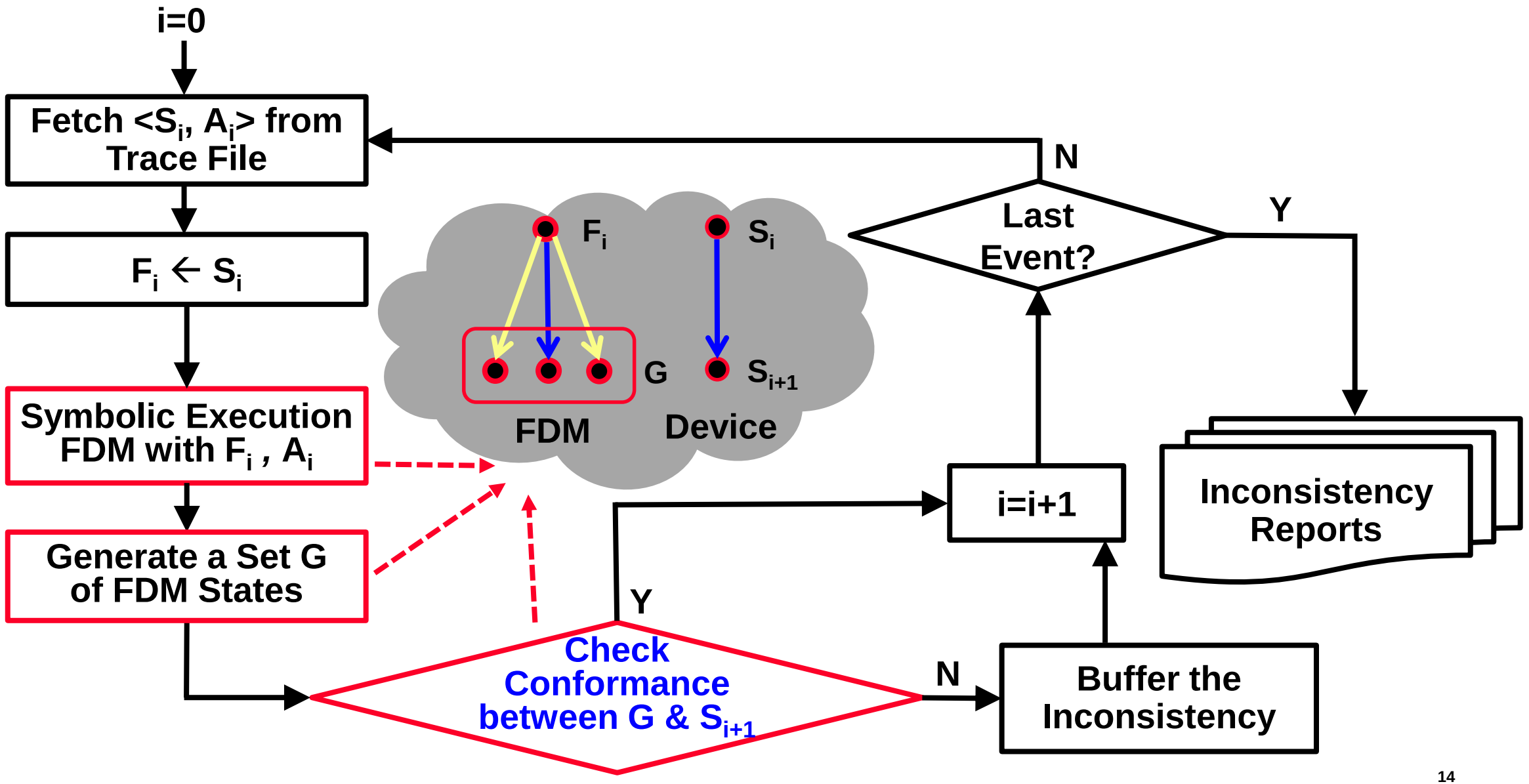
- **A device trace is a sequence of <S_i, A_i> pairs**
  - S_i : current assignments to all the registers
  - A_i : forthcoming driver request



Our Framework



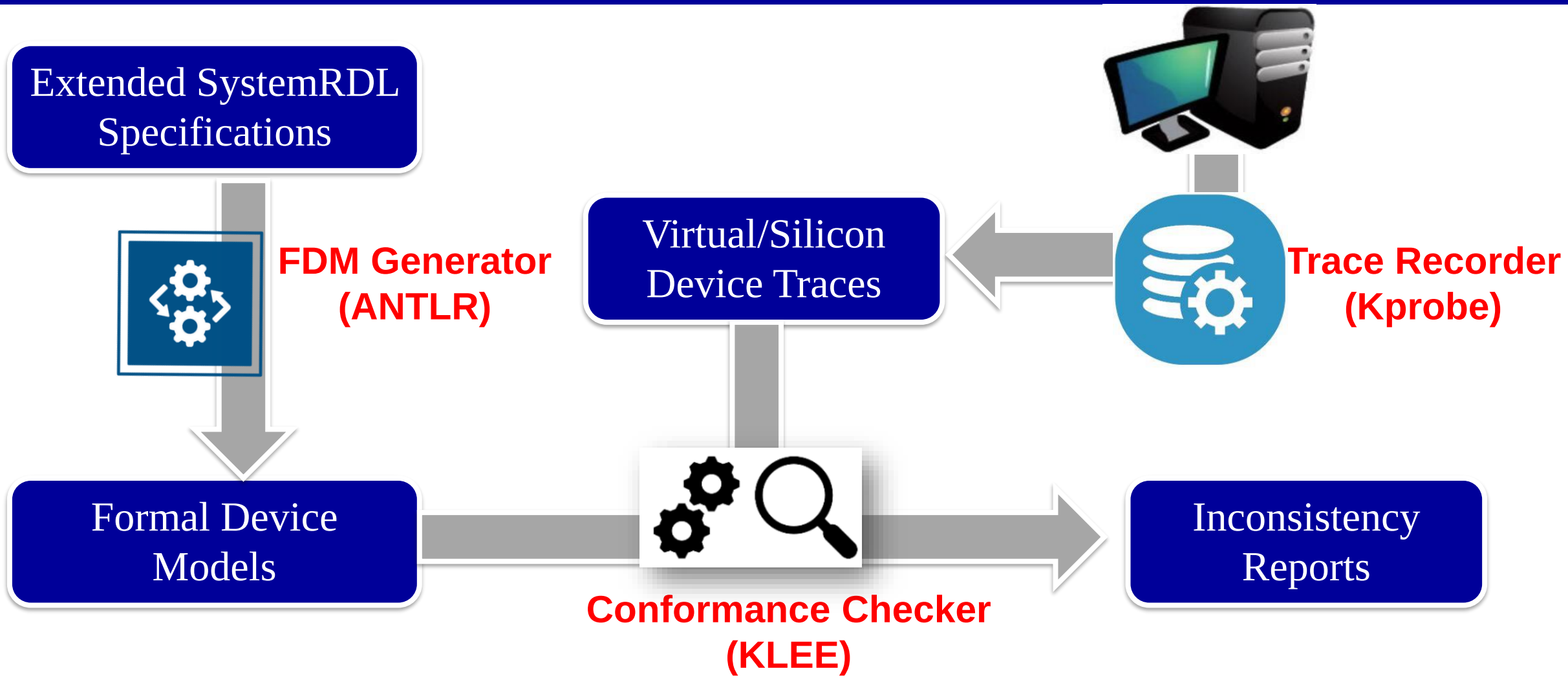
Conformance Checking Procedure



Outline

- Motivation
- Formal Device Model Generation
  - Syntax Extensions for SystemRDL
  - Automated Generation of Formal Device Models
- Specification-Driven Conformance Checking
  - Device Trace Collection
  - Conformance Checking with FDMs
- **Performance Evaluation Results**
- Conclusion

Tool Chain for Experiment



- All the experiments were obtained on an Ubuntu desktop with 3.2GHz AMD CPU and 16GB RAM

Experimental Settings

Virtual Devices:

- QEMU 0.15.1
- Contains e1000 and eepro100 virtual devices

Silicon Devices:

- Intel E1000 Gigabit NIC (e1000)
- Intel EEPro100 Megabit NIC (eepro100)

Table 1: Experimental Settings for Network Adapters

Devices	Spec. (LoC)	FDM (LoC)	VP (LoC)	Select. Captured Size (Bytes)
Intel e1000 Gigabit NIC	546	1805	2099	1224
Intel eepro100 Megabit NIC	587	903	2178	74

Experimental Results

- **Validated designs using 4 types of network commands**
  - **e.g., ifconfig, ping, scp, ifup, hping3, ...**
- **Detected 12 bugs from virtual/silicon devices**

Table 2: Bugs Identified from Virtual and Silicon Devices

Indices	Bug Types	Num.	Bug Sources
E1	Update the bits of reserved SD register	3	SD
E2	Generate unnecessary interrupts	1	VP
E3	Fail to update register when necessary	2	VP
E4	Write incorrect values to registers	3	VP
E5	Update the bits of reserved VP register	1	VP
E6	Driver issues a write to reserved registers	2	Driver
* VP and SD stand for Virtual Prototype and Silicon Device, respectively			

Driver issues invalid requests!

Experimental Results

- **Better false negative ratio due to more bugs detected**
  - **E5: Update the bits of reserved VP register**
  - **E6: Driver issues a write to reserved registers**

Table 3: Comparison of Different Methods

Bug Source	Bug Type	FDM-VP	FDM-SD	VP-SD [7]
Silicon Devices	E1	-	**3**	2
Virtual Devices	E2	1	-	1
	E3	2	-	2
	E4	3	-	3
	E5	**1**	-	-
Driver	E6	**1**	**1**	-

[7] L. Lei et al. Post-silicon Conformance Checking with Virtual Prototypes, *DAC 2013.*

Experimental Results

- **Better conformance checking time: Up to 67X improvement**
- **Better resource utilization: Up to 2X less memory used**

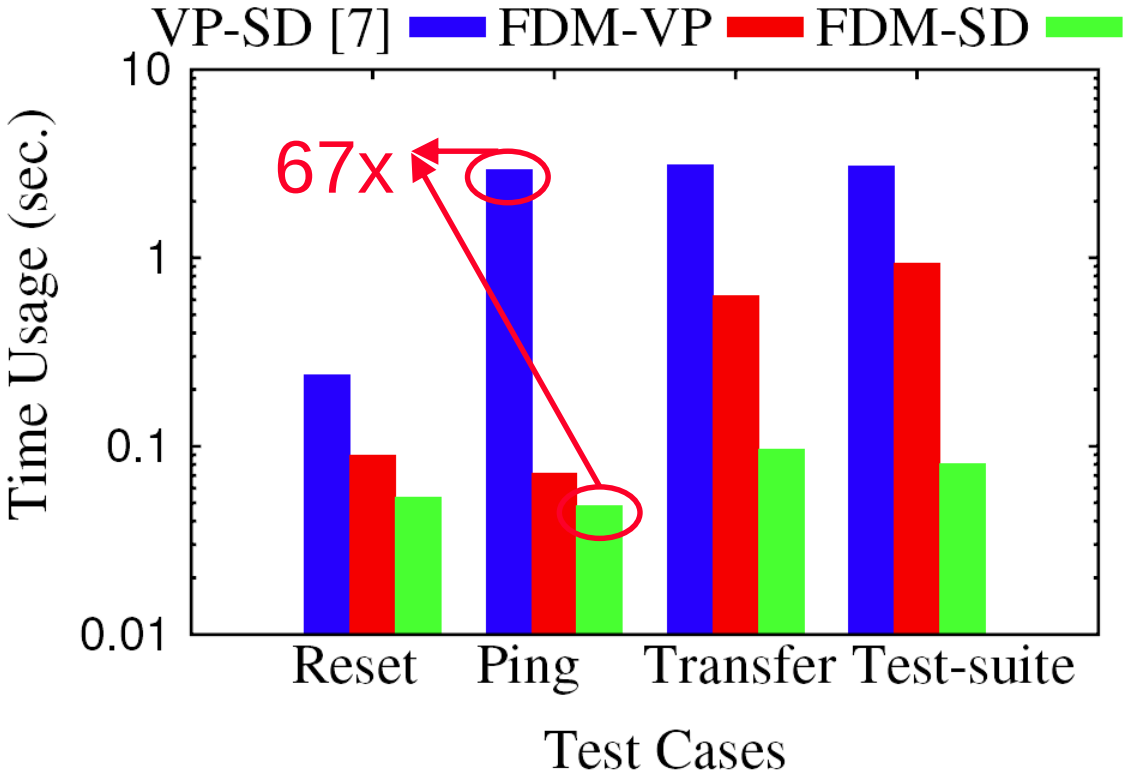


Fig.1 Time Usage for *e1000 NIC*

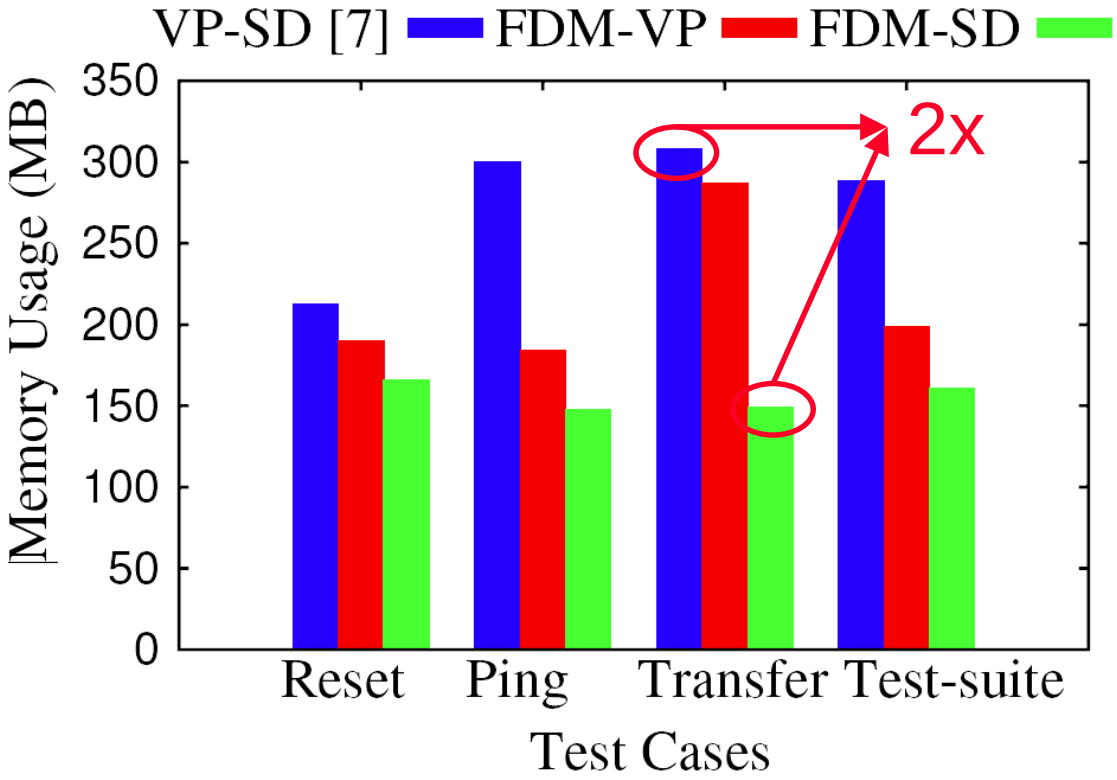


Fig.2 Memory Usage for *e1000 NIC*

Conclusion

- **Contributions:**
  - SystemRDL extension for register access behavior modeling
  - Transformation rules from extended SystemRDL to FDMs
  - Symbolic execution-based conformance checking framework
- **Experimental results on industrial network adapters**
  - New bugs found in virtual/silicon devices
  - Better performance than state-of-the-art methods
- **Future work**
  - Directed test generation for virtual/silicon prototypes
  - Runtime validation

Thank you !