2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Detecting Harmful Race Conditions in SystemC Models Using Formal Techniques

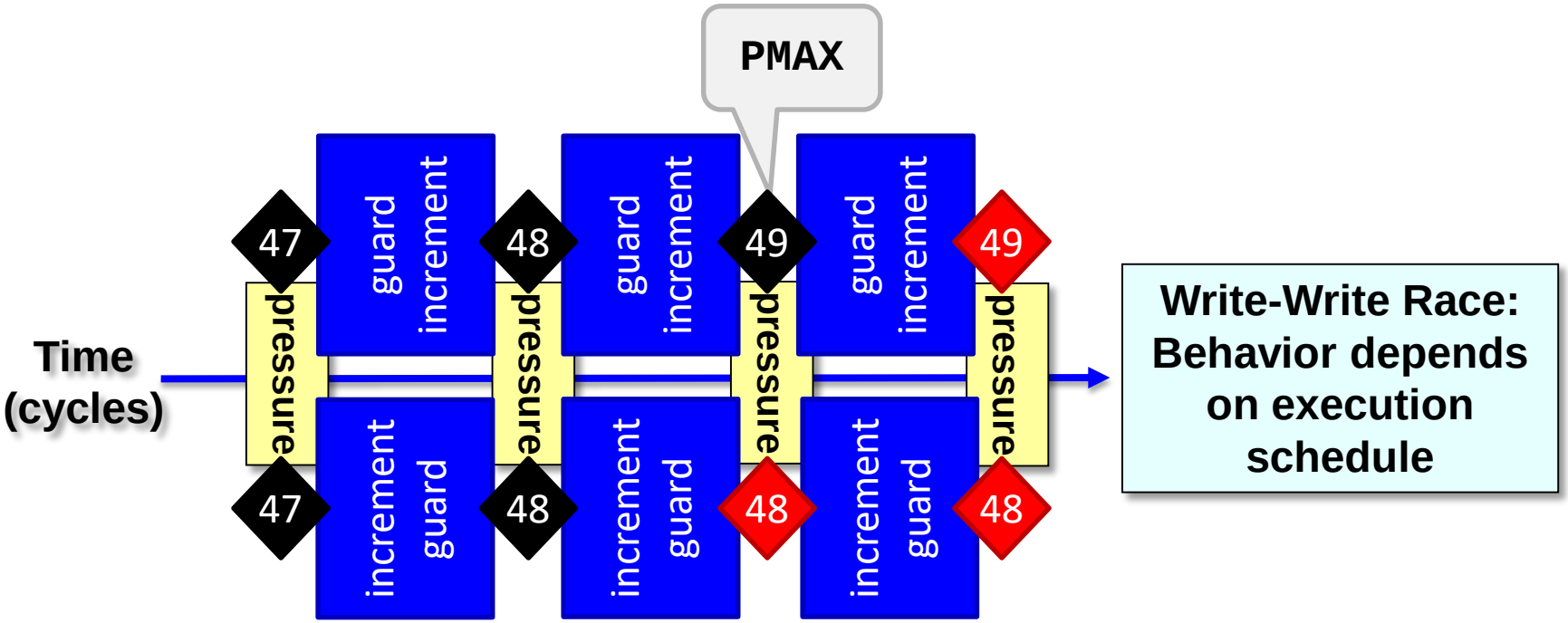
Sven Beyer, Dominik Strasser

OneSpin Solutions GmbH

Agenda

- Introduction
  - Race Conditions and SystemC
  - Related work
- Background
  - Formal analysis for race conditions in Verilog RTL
- SystemC-to-Verilog approach
  - Embedding SystemC to a formal property checker
  - Detecting race conditions in SystemC
  - Formal property checking on SystemC
- Conclusion

Race conditions & Execution schedules




```
void guard() {
  if (pressure == PMAX)
    pressure = PMAX - 1;
}
```

```
void increment() {
  pressure = pressure + 1;
}
```

Related work

- [1] D. Kroening, N. Blanc: Race Analysis for SystemC using Model Checking, ACM, 2010
  - Model checking on execution graphs to detect race conditions
- [2] H. Le, R. Drechsler: Toward Verifying Determinism of SystemC Designs, DATE, 2014
  - Input-output determinism checked with software model checking of C-code generated from SystemC

Hardware model checking

- Hardware languages == race conditions
- Exhaustive verification useful for race conditions
  - Standard verification flow (OneSpin 360DV): race condition reported with simulation trace from reset showing failure for easy debugging on RTL code
  - Assertion languages like SVA used for constraining
  - Tools scale to handling industrial size designs

Write-write race assertion

- For variable v with write location $i = 1, \ldots, n$
  - $write(v)_i$ is Boolean condition of write location i
  - $value(v)_i$ is write value of write location i
- $writewriterace(v) =$
  $\bigvee_{i,j}(write(v)_i \wedge write(v)_j \wedge (value(v)_i \neq value(v)_j))$
- Concurrent writes with same value not considered as write-write races
- Can be modelled bit-precise for hardware languages
- Same idea for read-write races

The SystemC-to-Verilog idea

- SystemC used to model hardware
  - Synthesis to standard RTL languages possible
- ➢Map it to similar hardware language like Verilog
  - ➢Methods and sensitivity to clocked processes
  - ➢SC_IN/SC_OUT to module ports
  - ➢…
- Build formal model out of SystemC
- Apply hardware model checking capabilities to this formal model

SystemC Example

```
const int PMAX = 49;
SC_MODULE(m) {
  sc_in<bool> clk, reset;
  sc_out<int> pressure;
  void guard() {
    if (pressure == PMAX)
      pressure = PMAX - 1;
  }
  void increment() {
    pressure = pressure + 1;
  }
  SC_CTOR(m) {
    pressure = 0;
    SC_METHOD(guard);     sensitive << clk.pos();
    SC_METHOD(increment); sensitive << clk.pos();
    async_reset_signal_is(reset,true);
  }};
```

pressure written in two threads – potential write-write-race

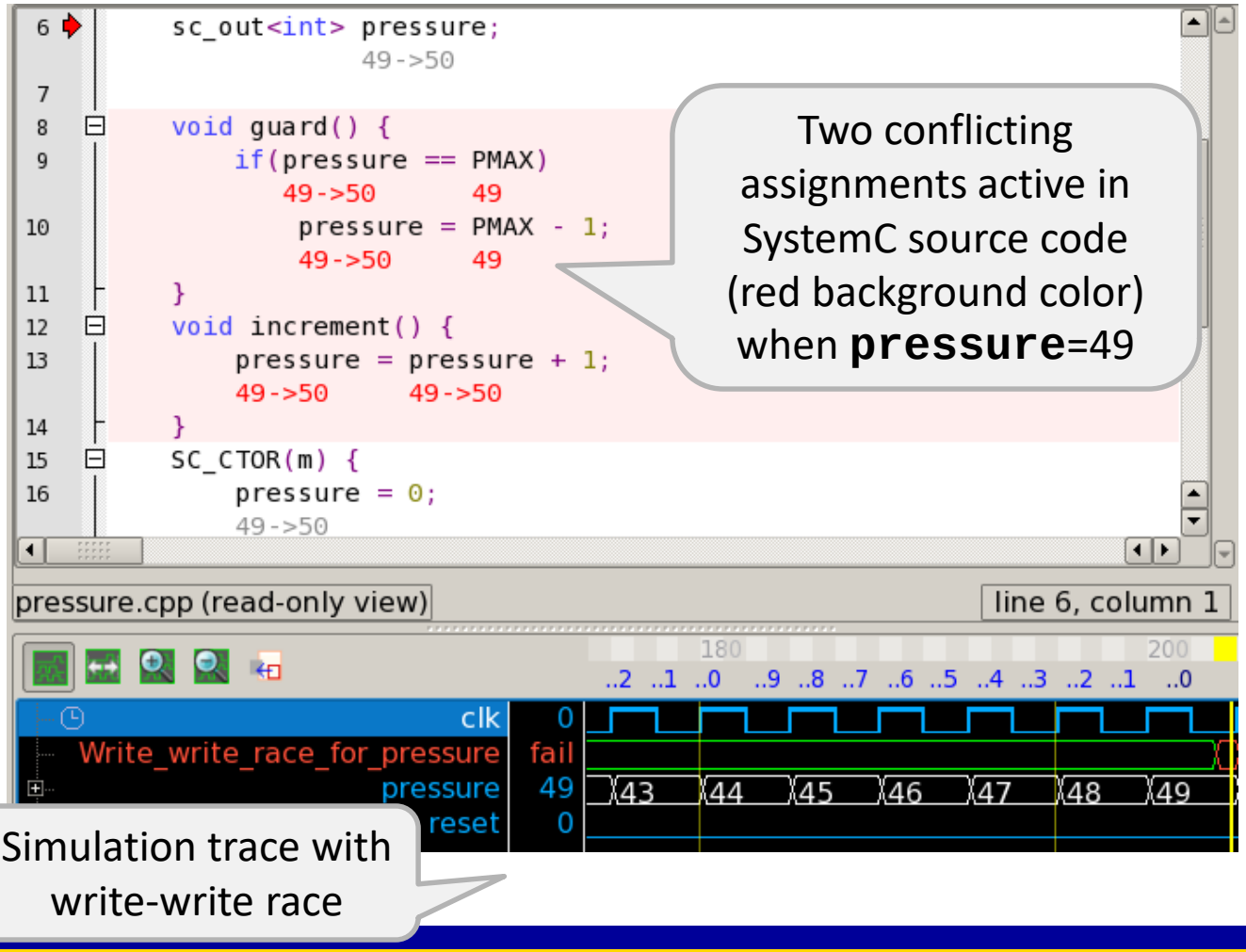
Verilog for SystemC example

```
localparam int PMAX=49;
module m(input clk, input reset,
         output int pressure);

always @(posedge clk or posedge reset)
begin: guard
  if (reset)
    pressure = 0;
  else if (pressure == PMAX)
    pressure = PMAX - 1;
end: guard
always @(posedge clk or posedge reset)
begin: increment
  if (!reset)
       pressure = pressure + 1;
end: increment
endmodule
```

Two clocked Verilog processes

Debug Write-Write Race



Adding guards & constraints

Adding guard (new input)

```
void increment() {
  if (guard_inc)
     pressure = pressure + 1;
}
```

Adding SVA constraint for guard

```
exclusive: assume property
  (@(posedge m.clk)
   m.pressure>=m.PMAX-1 |->
     ~m.guard_inc);
```

Absence of Write-Write Race can be proven on modified design with constraint

Conclusion

- Use well-established tool flow for detecting SystemC race conditions by mapping SystemC to Verilog
  - Typical tool flow for hardware development
  - Easy to understand representation of race conditions
  - Allows for standard SVA assertion usage for checking or constraining on SystemC, including timed SVA
- Future work
  - Enhance SystemC support based on customer needs
  - Add equivalence checking for HLS synthesis from SystemC