DVCon 2010

Design Verification Conference and Exhibition

February 22-25, 2010

Experiencing Checkers for a Cache Controller Design

Ben Cohen	Srinivasan Venkataramanan and Ajeetha Kumari	Lisa Piper
VhdlCohen Publishing 	CVC Pvt.Ltd., Bangalore, India	IC Verification Consultant
ben@SystemVerilog.us	srini@cvcblr.com akumari@cvcblr.com	lisa@SystemVerilog.us

Purpose of Study

- **Are checkers useful?**
  - Easy to use?
    - Create, instantiate, Intuitive results, Easy debug
- **What was our experience?**
  - Process of using checkers
  - Application issues
  - Tool issues
  - Conclusions

Checkers are COOL!

Agenda

- Classification of assertions
- Classification of checker instantiations
- The cache controller model
- Example of checker declaration and instantiation
- Design experience
  - Without checkers
  - With checkers
- Impressions and conclusions

Classification of Checkers Assertions and Instances

DVCon 2010
Design Verification Conference and Exhibition

VhdlCohen Publishing

```
checker chk(...);
 ...
 ap_static:  assert property(...);

 always @ (clk) begin : c_a1
   ap_proc1: assert property(...);
   ap_immd1: assert(...);
 end : c_a1

 final begin : c_final
   ap_immd2: assert(...);
 end : c_final

 endchecker  : chk
```

Static concurrent assertion

Procedural concurrent assertion

Immediate assertion

Checker Declaration

```
module m( ...);
 ...
 chk chk1(...);

 always @ (posedge clk) begin : m_a1
   ...
   if(enb) begin  : m_if1
     chk chk2(a, ...);
   end   : m_if1
 end : m_a1

 endmodule : m
```

Checker Instantiation

Static checker instance

Procedural checker instance

Checker assertion type has significance when instantiated

Assertion
- Static
- procedural

Checker instance
- Static
- Procedural

Emulation of Static Checker Instance (*chk1*) in a Module

VhdlCohen Publishing

```
checker chk(...);
 ...
 ap_static:  assert property(...);

 always @ (clk) begin : c_a1
   ap_proc1: assert property(...);
   ap_immd1: assert(...);
 end : c_a1

  final begin : c_final
    ap_immd2: assert(...);
  end : c_final

endchecker  : chk
```

Static concurrent assertion

Procedural concurrent assertion

Immediate assertion

```
module m(..);
  chk chk1(...;

endmodule : m
```

Checker Instantiation

EMULATION

```
module m(...);
 ...
 // chk chk1(...);
 ...
 ap_static:  assert property(...);

 always @ (clk) begin : c_a1
   ap_proc1: assert property(...);
   ap_immd1: assert(...);
 end : c_a1

  final begin : c_final
    ap_immd2: assert(...);
  end : c_final

endmodule : m
```

Emulation of static checker instance

Static concurrent assertion

Procedural concurrent assertion

Final procedure

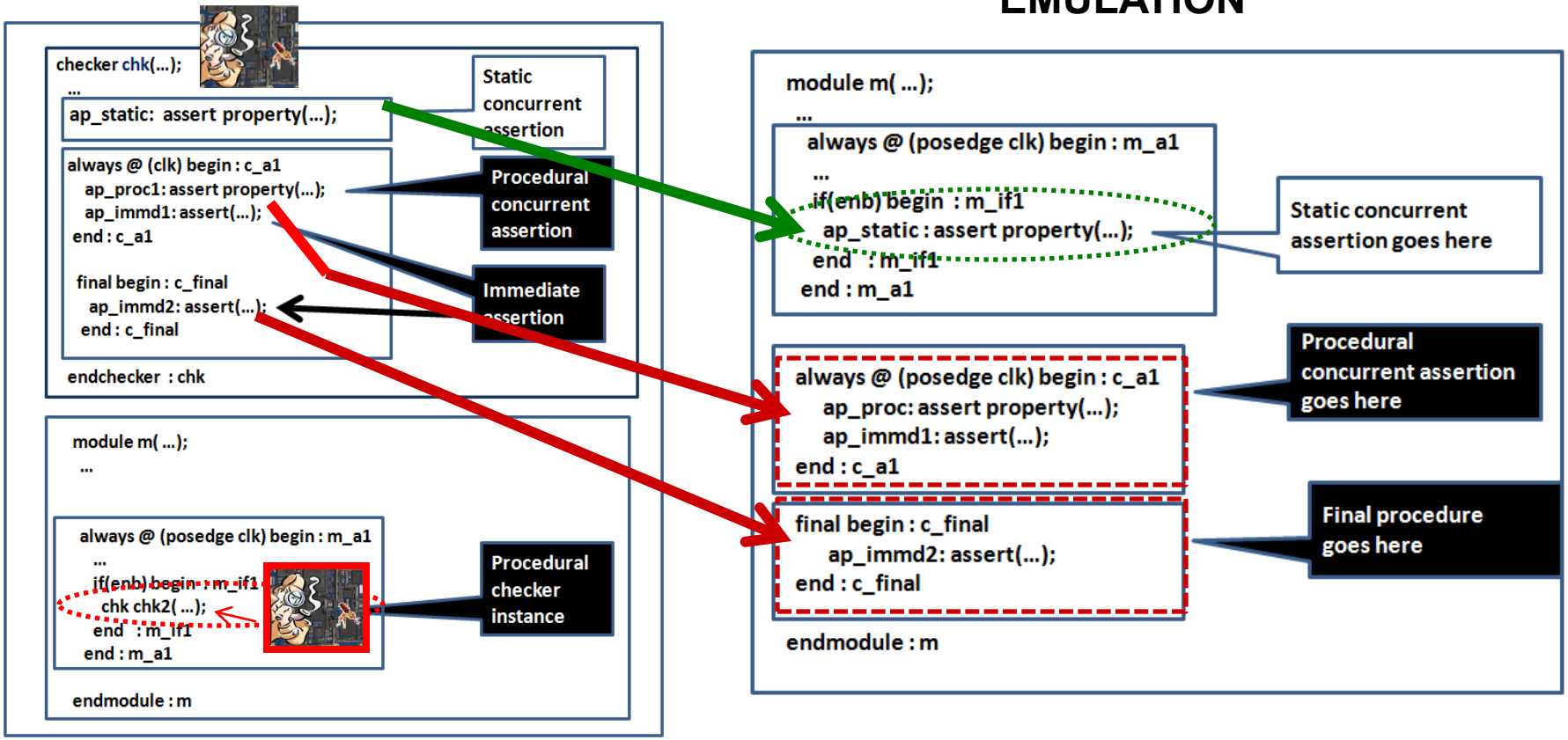
Static instantiation is identical to module instantiation

When a checker is instantiated as a static checker instance, all of its code behaves as if it were instantiated directly in the module after the proper argument associations are made.

Emulation of Procedural Checker Instance in a Module

DVCon 2010
Design Verification Conference and Exhibition

VhdlCohen Publishing



- Procedural concurrent assertions of checker instanced procedurally behave in-block.
- Static assertions of that checker behave inline to where instanced

The Model – Cache Controller

VhdlCohen Publishing

User interface

data_vld
busy
rd
wr
addr
rd_data
wr_data
clk
rst_n

Cache Controller

Checker

Control Logic

Cache Memory

cache_exists
cache_data

miss= !cache_exists

Main Memory Interface

mem_rd
mem_wr
mem_addr
mem_rd_data
mem_wr_data

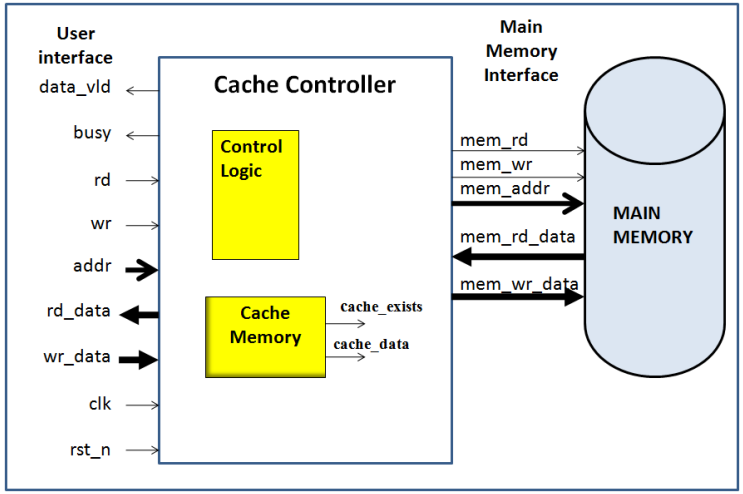
MAIN MEMORY

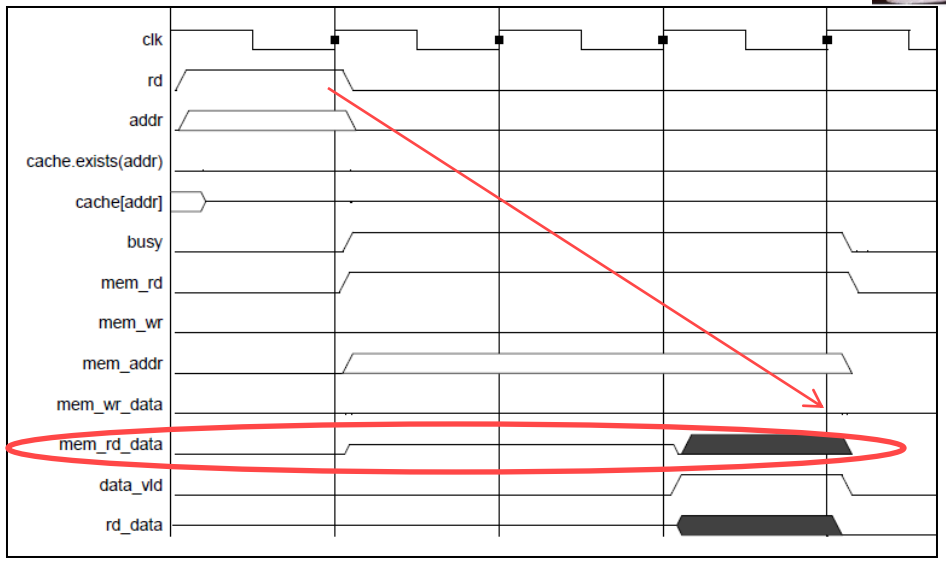
Many requirements!

Cache Controller Timing

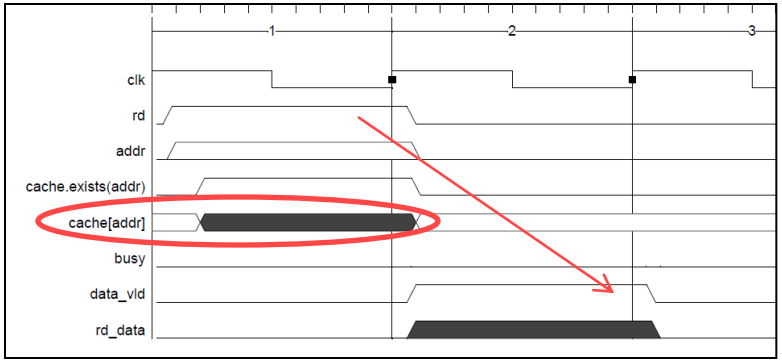
read and write-through cache

VhdlCohen Publishing

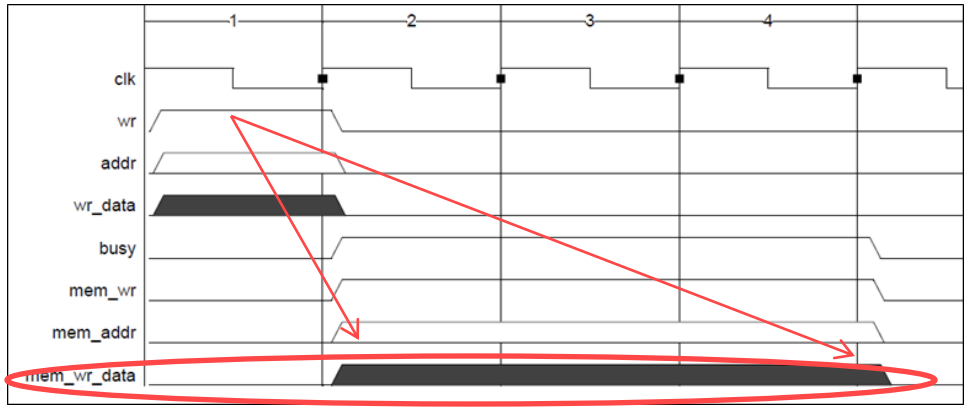




Main Memory Read with Cache Miss



Read with data in Cache

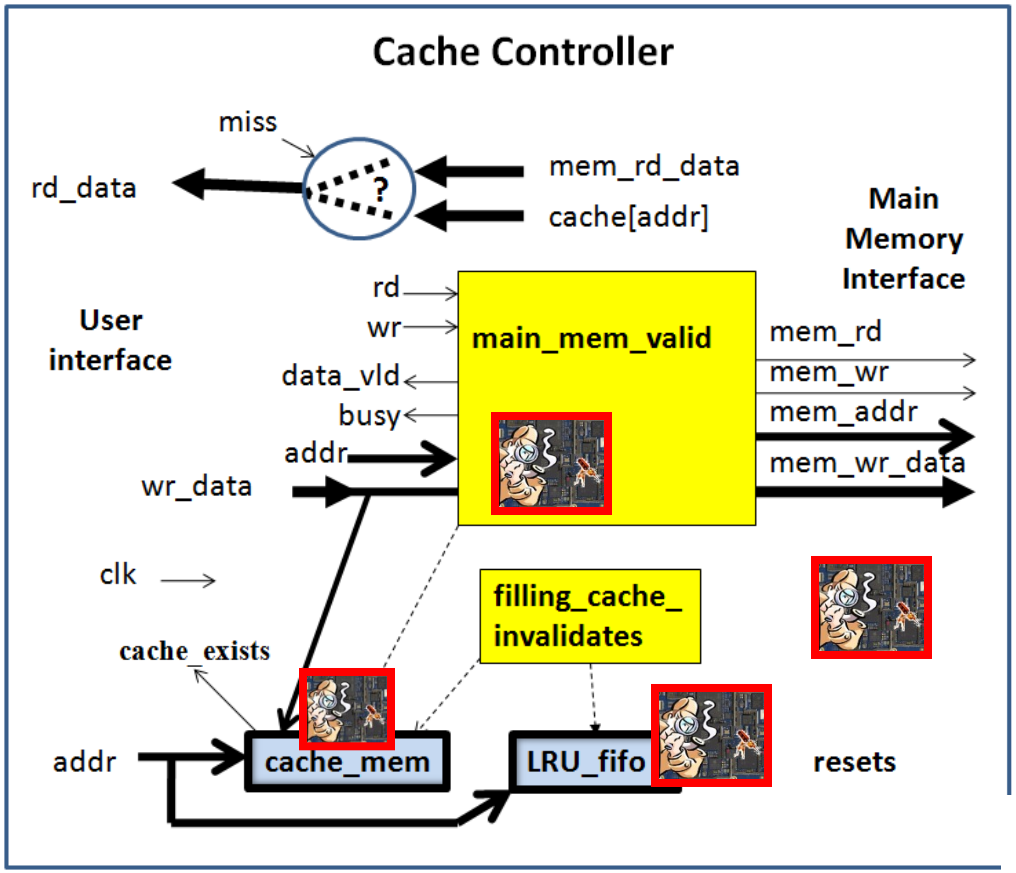


User Interface Write

Memory rd/wr access parameterized

Cache Controller Architecture

VhdlCohen Publishing

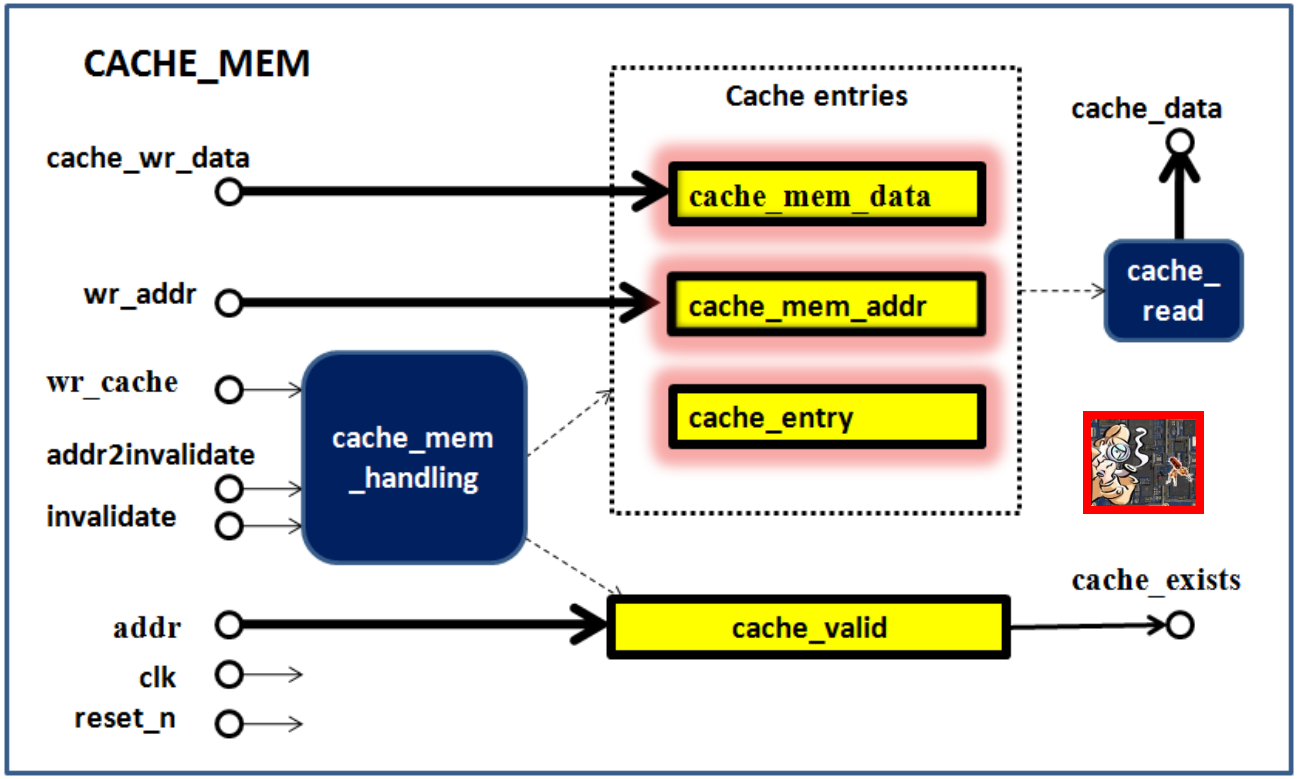


Checker	Function
chk_immd.	Inline assertions, *instantiated procedurally*
chk_rd_cntrl.	Read miss/hit, *instantiated statically*
chk_rd_wr_cntlr	RD/WR miscellaneous, *instantiated statically*
chk_wr_cntrl.	Write through and cache, *instantiated statically*
chk_reset.	Reset, *instantiated statically*
chk_fifo	LRU Fifo assertions, *instantiated statically*

Lots to check and cover!

VhdlCohen Publishing

Cache Memory Architecture



Lots to check and cover!

Checker	Function
chk_invalidate.	Cache invalidates, *instantiated procedurally*
chk_immd.	Inline assertions, *instantiated procedurally*

Example of a Checker Declaration

```
import cache_ctl_pkg::*;
checker chk_invalidate(
  logic [0: 2**MAIN_MEM_ADDRESS_WIDTH-1] cache_valid,
  logic [MAIN_MEM_ADDRESS_WIDTH-1 : 0]
          [0 : (CACHE_SIZE-1)] cache_mem_addr,
  logic [CACHE_SIZE-1: 0]  cache_entry,
  logic [MAIN_MEM_ADDRESS_WIDTH-1:0] addr2invalidate,
  logic found_existing_entry,
  logic clk, rst_n);
 timeunit 1ns;   timeprecision 100ps;
 if (CACHE_SIZE > 1024) $error("cache size is too large");
 default clocking default_clk @ (posedge clk); endclocking
 default disable iff !rst_n;
```

Formal arguments

Can also be property , sequence , event

Elaboration time check

Defaults

Example of a Checker Declaration

VhdlCohen Publishing

```
function logic check_cache_entry4MT();
   automatic logic success   =0; // If==1 then it was not invalidated
   automatic int   i;
    for (i=0; i <= CACHE_SIZE -1; i++) begin : for1
      if(cache_entry[i]==1 && cache_mem_addr[i] == addr2invalidate) begin : if_1
        success  = 1; // found cache line
        break;
      end : if_1
    end : for1
   if (success) check_cache_entry4MT  =0; // was not invalidated
   else check_cache_entry4MT   =1; // was invalidated
  endfunction : check_cache_entry4MT
 // static concurrent assertion
 ap_invalidate : assert property(  // Static assertion
      ##1 cache_valid[addr2invalidate] ==0 && check_cache_entry4MT());
 // static concurrent assertion
 ap_nothing2invalidate: assert property (found_existing_entry)
    else $error("nothing to invalidate");
endchecker : chk_invalidate
```

Function -- Must return a single value

Function used here

To be instantiated procedurally

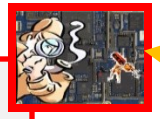
Example of a Checker Instantiation *(cache_mem.sv)*

DVCon 2010 Design Verification Conference and Exhibition

VhdlCohen Publishing

```
always @ (posedge clk) begin : cache_mem_handling
   automatic logic success, found_existing_entry, found_empty_line;
   automatic int   i, j, found_index;
   success =0;  found_existing_entry =0;
   found_index =0;  found_empty_line =0;
   if (invalidate) begin : if1
     cache_valid[addr2invalidate]<= 1'b0;
     for (int j=0; j<= CACHE_SIZE-1; j=j+1) begin : for1
        if(cache_entry[j]==1 && cache_mem_addr[j] == addr2invalidate) begin : if2
        found_existing_entry= 1'b1;
        found_index= j;
        success=1;
        if (found_existing_entry) cache_entry[j] <=1'b0;  //  empty line
        break;
      end : if2
     end : for1
     chk_invalidate chk_invalidate_1(.*);    // checks for invalidates
  end : if1
```

```
checker : chk_invalidate
  ap_invalidate : assert property( …);
 ap_nothing2invalidate: assert property (..);
endchecker : chk_invalidate
```



Example of a checker application

DVCon 2010
Design Verification Conference and Exhibition

VhdlCohen Publishing

```
checker chk_immd(logic the_what,
                 string msg,
                 logic clk);
  // static concurrent assertion
  ap_test_now: assert property(@ (posedge clk) the_what) else
        $error("msg, the_what=%b at %t", the_what, $time);
endchecker : chk_immd
```

Need a static concurrent assertion to enable inline behavior. Immediate assertion is a concurrent assertion

```
// cache_controller
always @ (posedge clk) begin : main_mem_accesses
    …
    if (wr) begin : wr_2main // Start of a write
        chk_immd chk_immd_wr_cycle_timer_not_0(
            .the_what(wr_cycle_timer==0),
            .msg("wr signal when memory counter !=0"),
            .clk(clk));
```

Potential Use Model of Checkers

VhdlCohen Publishing

- Marks the need of checks during RTL design definition
- Not yet know what is in the checker
  - but feel that there should be something
  - The what's in the checker can be defined later

- Example :

```
module dut(..);
  ...
  always @ (posedge clk) begin
   if (rd && mode==FAST) begin
      some_code;
      chk_rd_mode_fast  chk_rd_mode_fast_1 (.*);
  ...
```

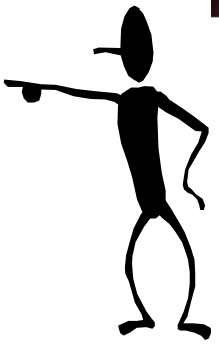
Design Experience with Checkers

- **Started with a <u>single checker declared within the module</u>**
  - Because requirements were too loose
  - Single checker became too complex
- **<u>Reconsidered the use model</u>**
  - Smaller targeted checkers
  - As design matured:
    - More checkers created
    - More static concurrent assertions within a checker
    - More elaboration time checks
    - Less clutter in RTL
  - Assertions challenge the design and requirements !
    - Clarify requirements and ease RTL design
      - Even if assertions are correctly or incorrectly written
    - Review of assertions helped the RTL design and debug

Incorrect Assertions yield False Sense of Security

DVCon 2010
Design Verification Conference and Exhibition

VhdlCohen Publishing



Assertions collocated in checkers help in the review process

```
sequence q_wr_wr;
  int v_addr;
  @ (posedge clk) ($rose(wr), v_addr=addr) ##[1:$] wr && addr==v_addr;
 endsequence : q_wr_wr

 property p_wr_rd;
  int v_addr, v_data;
   disable iff(q_wr_wr.triggered)
 ($rose(wr), v_addr=addr, v_data=wr_data) |->
    ##[1:$] rd && addr==v_addr ##[1:3] rd_data==v_data;
 endproperty : p_wr_rd
 ap_wr_rd : assert property(@ (posedge clk) p_wr_rd);
```

INCORRECT ASSERTION

Data read must be same as what was written

Correct assertion

```
property p_wr_rd;
  int v_addr, v_data;
    first_match(($rose(wr), v_addr=addr, v_data=wr_data) ##1
    !(wr && addr==v_addr)[*1:$] ##1 rd && addr==v_addr) |->
                              ##[1:MEM_CYCLES] rd_data==v_data;
 endproperty : p_wr_rd
 always @ (posedge clk)
   ap_wr_rd : assert property(p_wr_rd);
```

VhdlCohen Publishing

Checker / Code Ratio

- **cache_controller** ~ 200 lines with some comments
  - Includes checker instantiations ~ 18 lines
- **cache_mem** ~140 lines with some comments
  - Includes checker instantiations ~ 9 lines
- **fifo_mem** ~100 lines
  - Includes ~100 lines
- **Checkers** 
  - chk_rd_cntrl ~60 lines
  - chk_invalidate ~38 lines
  - chk_immd ~ 7 lines
  - chk_wr_cntrl ~90 lines
  - chk_rd_wr_cntlr ~30 lines
  - chk_fifo ~90 lines
  - chk_reset ~ 7 lines

Checkers	
LOC	330
Assertions	34
Cover property	6
Declarations	7
Instantiations	10
DUT	
LOC	440
LOC: checker/DUT	**~ 75%**
Checker instances/DUT	**< 3%**

Checker Impressions -1

Clear demarcation between RTL and verification code

DVCon 2010
Design Verification Conference and Exhibition

VhdlCohen Publishing

- **Verification and supporting code outside RTL code**
  - Reduced clutter in RTL

    Supporting code (signal / functions)
  - No concern about interference or confusion with the RTL
    - Organized and structured solution
    - Amenable to building small to medium verification units

- **Instantiated statically or procedurally within RTL**
  - If instantiated procedurally, then must differentiate
    - static assertions => behave inline
    - procedural assertions => in-block
  - Must be careful in using static concurrent vs procedural concurrent assertions

- **Collocate common assertions statements**
  - Otherwise multiple instantiations will cause duplicate assertions

Checker Impressions -2

- **Facilitates reviews**
  Verification code is collocated
- **Adopts a higher level view of verification**
  Each checker envelops a related aspect of verification
- **Configuration checks at elaboration time are useful**
  Elaboration checks can be included in checkers
- **Shorter code**
  CLK & reset inference
- **Ignored by synthesis tools**

Checker Impressions -3

- **Limitations – Not a big issue**
  - But … Must understand the rules
- **Limitations on what can be <u>declared</u>**
  - What's **<u>not</u>** in the body of a checker
    - parameter, localparam and specparam
    - Module, interface, program, class
    - task, void functions, blocking assignments,
    - Functions with side effects
    - **if, for, case** (in **always**, **initial** procedures)
    - hierarchical references

- **Limitations on where instantiated**
  - Wherever a concurrent assertion may appear
    - Statically or procedurally in **always, initial, final**
  - Illegal in ***fork…join, fork…join_any, or fork…join_none blocks***

Conclusions --1

- **Are checkers useful?**
  - Yes, group related code and assertions into single entities
  - Can be instantiated inline with RTL or statically
  - Significantly reduces LOC of RTL
  - Eases review process of verification code
- **What was our experience?**
  - Process of using checkers:
    - Easy & like it; checkers are cousins of modules
    - Procedural concurrent assertions: useful for in-block behavior when instantiated
      - Different style than normal
    - static concurrent assertions: useful for inline behavior when instantiated procedurally
  - Tool issues
    - Very few vendors support checkers, as of December 2009
  - Conclusions
    - Checkers represent a good addition to IEEE 1800-2009
    - Checkers should be used in design and verification

Questions?

VhdlCohen Publishing

And, yes, you can stuff many assertions into one checker !

Slides and code can be downloaded from
http://SystemVerilog.us/DvCon2010/

Backup slides

- To be used only if questions arise

Checkers Illegal constructs

- Those include:
  a) Parameter, localparam and specparam
  b) Module, interface, program, class
  c) Task, void functions, blocking assignments,
  d) Functions with side effects
  e) **if, for, while, case** statements (in **always,** and **initial** procedures)
  f) All hierarchical referencing, into or out of a checker, is disallowed

Checkers Illegal constructs

```
checker chk_rdhit(
    logic data_vld, busy, rd, wr, cache_exists, clk, reset_n);
  always @ (posedge clk) begin : a1

    if(rd) begin : rd1 // Illegal in a checker
      assert(wr  == 1'b0);
      if ($past(cache_exists))  // if is illegal in a checker
        assert (busy==1'b0);
    end : rd1
  end: a1
```

If, for, case are not allowed here

```
  function logic rd_cache(logic rd, wr, cache_exists, busy);
    if(rd && !wr && cache_exists & !busy)
      rd_cache  = 1'b1;
    else
      rd_cache  = 1'b0;
  endfunction : rd_cache
```

if, for, while, case allowed in functions

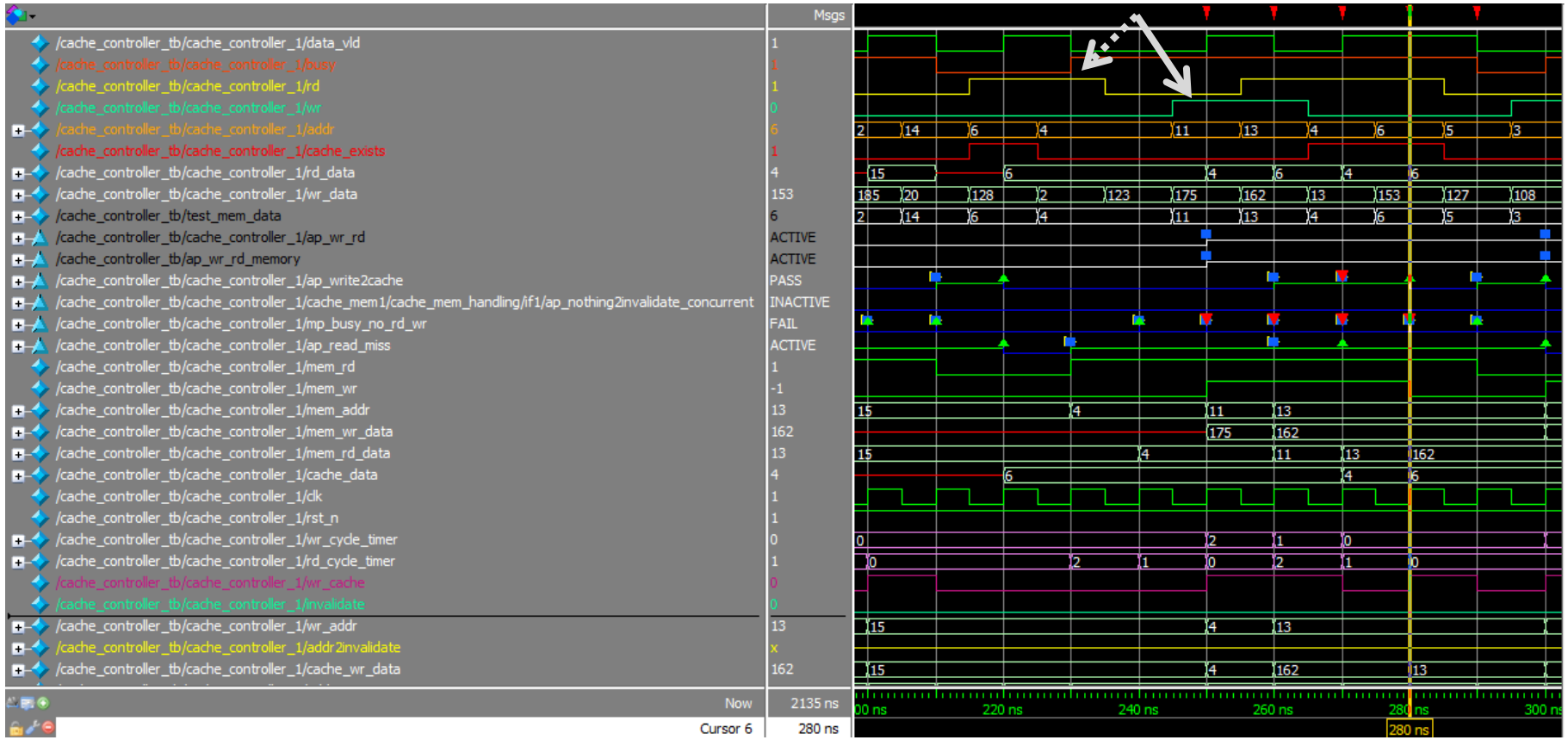
```
 ap_rd_hit: assert property(@ (posedge clk)
              rd_noWr(rd, wr, cache_exists, busy));
 ap_rd_hit2: assert property(@ (posedge clk)
        rd && cache_exists |-> !wr ##1 !busy);
endchecker : chk_rdhit
```

Checkers Illegal Instantiations

- It is illegal to instantiate checkers in
  - **fork…join**,
  - **fork…join_any**, or
  - **fork…join_none** blocks

mp_busy_no_rd_wr

wr when busy



Read hit and miss

VhdlCohen Publishing

	Msgs
/cache_controller_tb/cache_controller_1/data_vld	0
/cache_controller_tb/cache_controller_1/busy	1
/cache_controller_tb/cache_controller_1/rd	1
/cache_controller_tb/cache_controller_1/wr	0
/cache_controller_tb/cache_controller_1/addr	10
/cache_controller_tb/cache_controller_1/cache_exists	0
/cache_controller_tb/cache_controller_1/rd_data	4
/cache_controller_tb/cache_controller_1/wr_data	207
/cache_controller_tb/test_mem_data	10
/cache_controller_tb/cache_controller_1/ap_wr_rd	ACTIVE
/cache_controller_tb/ap_wr_rd_memory	ACTIVE
/cache_controller_tb/cache_controller_1/wr	0
/cache_controller_tb/cache_controller_1/cache_mem1/cache_mem_data[7]	9
/cache_controller_tb/cache_controller_1/cache_mem1/cache_mem_handling/i...	INACTIV
/cache_controller_tb/cache_controller_1/mp_busy_no_rd_wr	INACTIV
/cache_controller_tb/cache_controller_1/ap_read_miss	ACTIVE
/cache_controller_tb/cache_controller_1/mem_rd	1
/cache_controller_tb/cache_controller_1/mem_wr	0
/cache_controller_tb/cache_controller_1/mem_addr	10
/cache_controller_tb/cache_controller_1/mem_wr_data	214
/cache_controller_tb/cache_controller_1/mem_rd_data	9
/cache_controller_tb/cache_controller_1/cache_data	4
/cache_controller_tb/cache_controller_1/clk	1
/cache_controller_tb/cache_controller_1/rst_n	1
/cache_controller_tb/cache_controller_1/wr_cycle_timer	0
/cache_controller_tb/cache_controller_1/rd_cycle_timer	2
/cache_controller_tb/cache_controller_1/wr_cache	0
/cache_controller_tb/cache_controller_1/invalidate	1
/cache_controller_tb/cache_controller_1/wr_addr	9
/cache_controller_tb/cache_controller_1/addr2invalidate	7

Now 635 ns
Cursor 6 2.6 ns

1980 ns 2000 ns 2020 ns 2040 ns 2060 ns 2080 ns 2100

Good invalidate

VhdlCohen Publishing

	Msgs
/cache_controller_tb/cache_controller_1/rd_data	4
/cache_controller_tb/cache_controller_1/wr_data	64
/cache_controller_tb/cache_controller_1/cache_mem1/invalidate	0
/cache_controller_tb/cache_controller_1/cache_mem1/wr_cache	1
/cache_controller_tb/cache_controller_1/cache_mem1/wr_addr	4
/cache_controller_tb/cache_controller_1/cache_mem1/addr2invalidate	x
/cache_controller_tb/cache_controller_1/cache_mem1/cache_wr_data	64
/cache_controller_tb/cache_controller_1/cache_mem1/cache_data	4
/cache_controller_tb/cache_controller_1/cache_mem1/cache_exists	1
/cache_controller_tb/cache_controller_1/cache_mem1/clk	1
/cache_controller_tb/cache_controller_1/cache_mem1/reset_n	1
/cache_controller_tb/cache_controller_1/cache_mem1/cache_mem_data	17 10 9 11 2
[7]	17
[6]	10
[5]	9
[4]	11
[3]	214
[2]	4
[1]	15
[0]	6
/cache_controller_tb/cache_controller_1/cache_mem1/cache_mem_addr	14 10 9 11 5
[7]	14
[6]	10
[5]	9
[4]	11
[3]	5
[2]	4
[1]	15
[0]	6
/cache_controller_tb/cache_controller_1/cache_mem1/cache_entry	11111111
[7]	1

Assertion Report with forced failures

VhdlCohen Publishing

Assertions

File Edit View Add Window

Assertions

Name	Assertion Type	Lan	Er	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Pe
/cache_controller_tb/ap_wr_rd_memory	Concurrent	SVAon		0	5	5	960B	1.3K	
/cache_controller_tb/cache_controller_1/ap_fifo_pop_on_miss_full	Concurrent	SVAon		0	4	0	0B	48B	
/cache_controller_tb/cache_controller_1/ap_fifo_push_on_miss	Concurrent	SVAon		0	9	0	0B	48B	
/cache_controller_tb/cache_controller_1/ap_invalidate	Concurrent	SVAon		0	11	0	0B	64B	
/cache_controller_tb/cache_controller_1/ap_miss_then_data_2output	Concurrent	SVAon		1	7	0	0B	144B	
/cache_controller_tb/cache_controller_1/ap_read_hit	Concurrent	SVAon		0	116	1	48B	96B	
/cache_controller_tb/cache_controller_1/ap_read_miss	Concurrent	SVAon		0	9	0	0B	128B	
/cache_controller_tb/cache_controller_1/ap_read_miss_busy_data_vld	Concurrent	SVAon		1	8	0	0B	192B	
/cache_controller_tb/cache_controller_1/ap_wr_rd	Concurrent	SVAon		0	5	5	960B	1.3K	
/cache_controller_tb/cache_controller_1/ap_write2cache	Concurrent	SVAon		1	20	0	0B	152B	
/cache_controller_tb/cache_controller_1/cache_mem1/cache_mem_handling/if1/ap_nothing2invalidate	Immediate	SVAon		1	10	0	0	0	
/cache_controller_tb/cache_controller_1/cache_mem1/cache_mem_handling/if1/ap_nothing2invalidate_concurrent	Concurrent	SVAon		1	10	0	0B	64B	
/cache_controller_tb/cache_controller_1/cache_mem1/cache_mem_handling/wr_cache_updates/else1_mt4wr/ap_no_cache_line_found_4write	Immediate	SVAon		0	12	0	0	0	
/cache_controller_tb/cache_controller_1/cache_mem1/cache_mem_handling/wr_cache_updates/else1_mt4wr/ap_no_cache_line_found_4write_concurrent	Concurrent	SVAon		0	12	0	0B	72B	
/cache_controller_tb/cache_controller_1/cache_mem1/cache_read/if_exists/immed__131	Immediate	SVAon		0	119	0	0	0	
/cache_controller_tb/cache_controller_1/cache_mem1/cache_read/if_exists/while1/ap_read_cache_line	Immediate	SVAon		0	885	0	0	0	
/cache_controller_tb/cache_controller_1/cache_mem1/cache_read/if_exists/while1/ap_read_cache_line_concurrent	Concurrent	SVAon		0	885	0	0B	480B	
/cache_controller_tb/cache_controller_1/LRU_fifo/a_illegal_fifo_cmd_1	Concurrent	SVAon		0	0	0	0B	0B	
/cache_controller_tb/cache_controller_1/LRU_fifo/ap_pop_error_1	Concurrent	SVAon		0	213	0	0B	48B	
/cache_controller_tb/cache_controller_1/LRU_fifo/ap_push_data	Concurrent	SVAon		0	19	0	0B	144B	
/cache_controller_tb/cache_controller_1/LRU_fifo/ap_push_error_1	Concurrent	SVAon		0	213	0	0B	48B	
/cache_controller_tb/cache_controller_1/main_mem_accesses/rd_from_main/ap_rd_cycle_timer_not_zero	Immediate	SVAon		0	9	0	0	0	
/cache_controller_tb/cache_controller_1/main_mem_accesses/wr_2main/ap_wr_cycle_timer_not_0	Immediate	SVAon		1	12	0	0	0	
/cache_controller_tb/cache_controller_1/main_mem_accesses/wr_2main/ap_wr_cycle_timer_not_0_concurrent	Concurrent	SVAon		1	12	0	0B	48B	
/cache_controller_tb/cache_controller_1/mp_busy_no_rd_wr	Concurrent	SVAon		4	56	0	0B	48B	
/cache_controller_tb/cache_controller_1/mp_no_simultaneous_rd_wr	Concurrent	SVAon		1	212	0	0B	48B	
/cache_controller_tb/test/r1/a1	Immediate	SVAon		0	100	0	0	0	

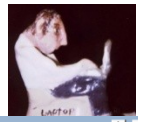
Cursor 6 603 ns 603 ns

0 ps to 2241800 ps

Documents emacs@BEN... SystemVerilo... 7 vish SpringSoft_Bi... SpringSoft_Bil... Desktop 11:20 AM

Assertion Report with no forced failures

VhdlCohen Publishing



Assertions

Name	Assertion Type	Lan	Er	Failure Count	Pass Count	Active Count	Memory	Peak Memory
/cache_controller_tb/ap_wr_rd_memory	Concurrent	SVAon		0	68	7	1.3K	1.7K
/cache_controller_tb/cache_controller_1/ap_fifo_pop_on_miss_full	Concurrent	SVAon		0	104	0	0B	48B
/cache_controller_tb/cache_controller_1/ap_fifo_push_on_miss	Concurrent	SVAon		0	109	0	0B	48B
/cache_controller_tb/cache_controller_1/ap_invalidate	Concurrent	SVAon		0	150	0	0B	64B
/cache_controller_tb/cache_controller_1/ap_miss_then_data_2output	Concurrent	SVAon		0	109	0	0B	72B
/cache_controller_tb/cache_controller_1/ap_read_hit	Concurrent	SVAon		0	102	0	0B	96B
/cache_controller_tb/cache_controller_1/ap_read_miss	Concurrent	SVAon		0	109	0	0B	128B
/cache_controller_tb/cache_controller_1/ap_read_miss_busy_data_vld	Concurrent	SVAon		0	109	0	0B	144B
/cache_controller_tb/cache_controller_1/ap_wr_rd	Concurrent	SVAon		0	68	7	1.3K	1.7K
/cache_controller_tb/cache_controller_1/cache_mem1/cache_mem_handling/if1/ap_nothing2invalidate	Immediate	SVAon		0	150	0	0	0
/cache_controller_tb/cache_controller_1/cache_mem1/cache_mem_handling/if1/ap_nothing2invalidate_concurrent	Concurrent	SVAon		0	150	0	0B	64B
/cache_controller_tb/cache_controller_1/cache_mem1/cache_mem_handling/wr_cache_updates/else1_mt4wr/ap_no_cache_line_found_4write	Immediate	SVAon		0	112	0	0	0
/cache_controller_tb/cache_controller_1/cache_mem1/cache_mem_handling/wr_cache_updates/else1_mt4wr/ap_no_cache_line_found_4write_concurrent	Concurrent	SVAon		0	112	0	0B	72B
/cache_controller_tb/cache_controller_1/cache_mem1/cache_read/if_exists/immed__131	Immediate	SVAon		0	159	0	0	0
/cache_controller_tb/cache_controller_1/cache_mem1/cache_read/if_exists/while1/ap_read_cache_line	Immediate	SVAon		0	722	0	0	0
/cache_controller_tb/cache_controller_1/cache_mem1/cache_read/if_exists/while1/ap_read_cache_line_concurrent	Concurrent	SVAon		0	722	0	0B	480B
/cache_controller_tb/cache_controller_1/LRU_fifo/a_illegal_fifo_cmd_1	Concurrent	SVAon		0	0	0	0B	0B
/cache_controller_tb/cache_controller_1/LRU_fifo/ap_pop_error_1	Concurrent	SVAon		0	1163	0	0B	48B
/cache_controller_tb/cache_controller_1/LRU_fifo/ap_push_data	Concurrent	SVAon		0	158	0	0B	72B
/cache_controller_tb/cache_controller_1/LRU_fifo/ap_push_error_1	Concurrent	SVAon		0	1163	0	0B	48B
/cache_controller_tb/cache_controller_1/main_mem_accesses/rd_from_main/ap_rd_cycle_timer_not_zero	Immediate	SVAon		0	109	0	0	0
/cache_controller_tb/cache_controller_1/main_mem_accesses/wr_2main/ap_wr_cycle_timer_not_0	Immediate	SVAon		0	106	0	0	0
/cache_controller_tb/cache_controller_1/main_mem_accesses/wr_2main/ap_wr_cycle_timer_not_0_concurrent	Concurrent	SVAon		0	106	0	0B	48B
/cache_controller_tb/cache_controller_1/mp_busy_no_rd_wr	Concurrent	SVAon		0	645	0	0B	48B
/cache_controller_tb/cache_controller_1/mp_no_simultaneous_rd_wr	Concurrent	SVAon		0	1163	0	0B	48B
/cache_controller_tb/test/begin_4wr/a0	Immediate	SVAon		0	4	0	0	0
/cache_controller_tb/test/r1/a1	Immediate	SVAon		0	1050	0	0	0