Administrivia

- Week-1 workshop (“Lab 0”) completed
- Week-2 lab (“Lab 1”) ongoing
  - Extra lab hall access difficult (other courses)
    - May install ModelSim on private computer (see intro email)
  - Show results to TAs during lab to get checked off!
  - Can’t accept emailed solutions!

Administrivia, cont.

- Post-workshop survey
  - Positive feedback in general, esp. for lab TAs 😀
- Lab-feedback timeslot: move lecture forward 1h but may encroach on lunch break?

(V)HDL design reviews

DAT093
lars.svensson@chalmers.se

Background

- Recent addition to DAT093: peer review of VHDL code.
  - Submit Lab-2 code by 3pm, Sep 20
    - Receive someone else's code for review
    - Submit review by noon, Sep 28
- Earn up to 0.4 grade points
  - Review quality will be graded, not code!

Why code review in DAT093?

- Learn more about VHDL and hardware design by considering received feedback
- Learn by formulating feedback
- Learn about review process
- Encourage certain good habits
- Bypass review bottleneck through parallel processing

Outline

- Why do code reviews?
- How does a VHDL review relate to SW and HW reviews?
- A VHDL review example
- Best practices (discussion)

Purpose of code / design reviews

1. Identify and eliminate bugs?

- How measure degree of success?
  - Number of bugs found!?

2. Verify correct functionality?

- How can it be better than simulation?

- Caveat: need to consider also “incorrect” inputs + states

  - Yes, they should never occur, but…

3. Human-understandable design / implementation?

- This looks better!
  - Something <u>beyond</u> bench testing
- How help humans understand code?

HW and SW reviews

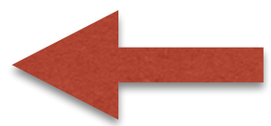
Traditional HW design review

- “Formal” occasion scheduled in time plan
  - Designer walks through design + docs
  - Supervisor, experienced co-workers review, clarify, ask questions
- Often associated with project “toll gate”: required before investment increase
  - E.g. before prototype build, chip mfg, etc
  - Goal: signoff by project owner

Traditional SW code review

- Often less “formal”
  - …except for medical systems, etc
- Less clearly connected with toll gate, if used
- Smaller chunks to review
  - May be associated with version control system code commit
  - Ad-hoc or distributed rather than planned meetings

“Pair programming”

- Agile practice of 2-person-team code development
  - Shared workstation and keyboard
  - “Driver” and “navigator” roles
    - “Writer” and “reviewer” 
    - Swap roles frequently

Why not try it in 2nd half of lab series?

HW review vs SW review

- SW review practices follow “agile” trend
  - Less emphasis on pre-planning
  - Less emphasis on documentation
  - More emphasis on learning
- So is HW review practices (becoming) old-fashioned? useless? waste of time?

[Cue vigorous student discussion]

VHDL design review

Hardware guy on VHDL reviews:

What VHDL Adds to the Review Process

- Probably, an awful lot more work!!
- VHDL introduces serious problems:
  - It hides design details
  - It is not WYSIWYG: What you see (as your design concept in VHDL) may not be what you get (as an output of the synthesizer)
  - Coupled with FPGAs, it encourages bad design practices
- Understanding design by reading code extremely difficult

 [Rich Katz: VHDL Design Review And Presentation; 2004 MAPLD]

Ouch :-(

There’s more.

VHDL Hides Design Details

- Connectivity hard to follow in VHDL files
- Behavior of sequential circuits can be hard to follow through processes
- Interactions between modules can be difficult to understand
- Spelling errors → undetected circuit errors

There's more..

Following Connectivity

Simple Input and Output Example

```
MA1: COUNT23    port map
                (RST_N => SIM_CLR_N, SYNC_CLR => EQUALS_internal, CLOCK => C5MHZ,
                 Q => TIME_NOW_internal
                );
MA3: MUX_23_4   port map
                (A => R1HZ, B => R6HZ, C => R8HZ, D => R10HZ, T => THZ,
                 T_SEL => TEST_SEQ, S1 => RATE_SEL(1), S0 => RATE_SEL(0),
                 Y => Y
                );
MA2: COMP23     port map
                (DATAA => TIME_NOW_internal, DATAB=> Y,
                 AEB   => EQUALS_internal
                );
MA7: GATE_RANGE port map
                (RST_N => RST_N,
                 CLOCK => C5MHZ,
                 OPEN_VALUE => OPEN_VALUE,CLOSE_VALUE => CLOSE_VALUE,
                 TIME_NOW   => TIME_NOW_internal,
                 GATE => GATE
                );
MA8: BIN2GRAY23 port map
                (A => TIME_NOW_internal,
                 Y => GRAYTIME
                );

 EQUALS    <= EQUALS_internal;
 TIME_NOW <= TIME_NOW_internal;
end RTL_ARCH;
```

Note: Had to print out the entity just to make this slide.

Inputs
Outputs

There's more…

Following Connectivity

Signals In a Simple Module

```
MA1: COUNT23     port map
                 (RST_N => SIM_CLR_N, SYNC_CLR => EQUALS_internal, CLOCK => C5MHZ,
                  Q => TIME_NOW_internal
                 );
MA3: MUX_23_4    port map
                 (A => R1HZ, B => R6HZ, C => R8HZ, D => R10HZ, T => THZ,
                  T_SEL => TEST_SEQ, S1 => RATE_SEL(1), S0 => RATE_SEL(0),
                  Y => Y
                 );
MA2: COMP23      port map
                 (DATAA => TIME_NOW_internal, DATAB=> T,
                  AEB   => EQUALS_internal
                 );
 MA7: GATE_RANGE port map
                 (RST_N => RST_N,
                  CLOCK => C5MHZ,
                  OPEN_VALUE => OPEN_VALUE,CLOSE_VALUE => CLOSE_VALUE,
                  TIME_NOW   => TIME_NOW_internal,
                  GATE => GATE
                 );
 MA8: BIN2GRAY23 port map
                 (A => TIME_NOW_internal,
                  Y => GRAYTIME
                 );

 EQUALS   <= EQUALS_internal;
 TIME_NOW <= TIME_NOW_internal;
end RTL_ARCH;
```

Making this chart was a lot of work and was error prone.

Was this fair?

What do you think?

There's more….

VHDL and Bad Design Practices

- VHDL and FPGAs combine to allow designers to treat design as software
  - Especially for FPGAs for which there is no reprogramming penalty, e.g., Xilinx
- Rather than designing by analysis, designers simply "try" design concepts

There's more…..

E.g., part of a 16 page process

```
 -- V1.02 & V2.2
    -- DATA WILL STOP TANSFERING IFF BOTH HOLD AND OUTPUT ENABEL
   ARE
    -- ACTIVE FOR THE SAME PORT

--     HOLD2 <= ((((HLD2TX_N_Q AND O_EN_Q(2)) OR
--     (HLDTX_N_Q  AND O_EN_Q(1)) OR
--     (ROFRDY_N_Q AND O_EN_Q(0))) AND
--      NOT(BYPASS_EN_Q AND  (HLDTX_N_Q AND O_EN_Q(1)))));

    HOLD1_I <= ((HLDTX_N_Q  AND O_EN_Q(1)) OR (ROFRDY_N_Q AND
  O_EN_Q(0)));--  V2.2
```

```
  HOLD2 <= ((((HLD2TX_N_Q AND O_EN_Q(2)) OR
  (HLDTX_N_Q  AND O_EN_Q(1)) OR
  (ROFRDY_N_Q AND O_EN_Q(0))) AND
   NOT(BYPASS_EN_Q AND  (HLDTX_N_Q AND O_EN_Q(1)))) OR
       (((HLD2TX_N_Q AND O_EN_Q(2)) OR (HLDTX_N_Q  AND
O_EN_Q(1)))
      AND  (BYPASS_EN_Q AND HLDTX_N_Q AND O_EN_Q(1)));
```

There's more……

Simplifying

Let:

a=HDL2TX_N_Q and O_EN_Q(2)
b=HLDTX_N_Q and O_EN_Q(1)
c=ROFRDY_N_Q and O_EN_Q(0)
d=BYPASS_EN_Q

Then

HOLD2=(a+b+c)·(d·b)' + (a+b)·(d·b) = a+b+c.

What happened to **d=BYPASS_EN_Q**??

There's more……

Lessons

- Don't just try things, think about what you're doing
  - Either **`BYPASS_EN_Q`** is needed or it's not – what's the requirement of the system?
- Make modules small enough to test via VHDL simulation, and test them fully.
  - If this logic was tested by itself, the error would have been found.
- It's on orbit, now

How about this?
Was this fair?

What do you think?

There's more…….

Worst Case Result

- A design that works in simulation for expected conditions, but with flaws that show up in unusual conditions
- Passed on with little documentation by engineers who become unavailable

⇒**A total programmatic disaster**!!

An common occurrence!

Likely a Grumpy Old Man.

- Probably wears a beard and thick glasses, and a pocket protector.

- This is Dennis Ritchie, btw. Inventor of the C language and of Unix.

- Bearded old guys may have a point.

DMR, 1941–2011

What's that guy really saying?

- Does he say that the VHDL language is bad?
- Does he say that any HDL is bad?
- Does he say that VHDL encourages bad practice?
- Does he moan about bad software practices entering hardware design?

Best practices for good + understandable VHDL code?

[buzz groups]

My $.02:

- Refer to the style sheet (downloadable)
- Shorter is often better (not always)
  - Beware of “unknown idioms”
  - Beware of your own clever ideas
- Consider the synthesis results
  - Code may be “correct” but yield inefficient hardware anyway!

What goes into a quality VHDL review?

[buzz groups]

My $.02:

- Describe your considerations, even when no change is suggested
- When you suggest a change, explain <u>why</u> you believe it would be beneficial
- Be considerate even when anonymous

How should the code be organized to enable good review?

[buzz groups]

My $.02

- Large and monolithic is rarely easy to understand
- Maintain a uniform style
  - Follow project style guide if available
  - Example style guide in PingPong

Summary (of points)

- I'll summarize all your points in a summary on PingPong

Summary (of lecture)

- Giving and receiving constructive critique helps us learn!

Difficult!

  - Avoid recurring mistakes
  - Should be part of normal workflow
    - Agile practices turn down the drama
  - Guidelines help (but will not cover everything)
    - Need to be adapted per project