RAM Control Block Code Review

Jake Humphrey

March 10, 2014

Code Block RAM Control Block

Reviewee Mike Li ⟨ml1811⟩

Reviewer Jake Humphrey ⟨jbh111⟩

Group Number 36

1 Synthesis Errors

```
@W:CL117 : rcb.vhd(303) | Latch generated from process for signal pxcache_pixopin(1 downto 0); possible
     missing assignment in an if or case statement.
@W:CL117 : rcb.vhd(303) | Latch generated from process for signal pxcache_wen_all; possible missing assignment
      in an if or case statement.
@W:CL111 : rcb.vhd(303) | All reachable assignments to pxcache_pw assign '1'; register removed by optimization
@W:CL117 : rcb.vhd(303) | Latch generated from process for signal vram_waddr(7 downto 0); possible missing
     assignment in an if or case statement.
@W:CL117 : rcb.vhd(303) | Latch generated from process for signal pxcache_pixnum(3 downto 0); possible missing
      assignment in an if or case statement.
@W:CL111 : rcb.vhd(303) | All reachable assignments to change_curr_word assign '1'; register removed by
     optimization
@W:CL117 : rcb.vhd(398) | Latch generated from process for signal ram_addr(7 downto 0); possible missing
     assignment in an if or case statement.
@W:CL111 : rcb.vhd(398) | All reachable assignments to ram_start assign '1'; register removed by optimization
@W:CL111 : rcb.vhd(94) | All reachable assignments to one_vector(0) assign '1'; register removed by
     optimization
@W:CL111 : rcb.vhd(94) | All reachable assignments to one_vector(1) assign '0'; register removed by
     optimization
@W:CL111 : rcb.vhd(94) | All reachable assignments to one_vector(2) assign '0'; register removed by
     optimization
@W:CL111 : rcb.vhd(94) | All reachable assignments to one_vector(3) assign '0'; register removed by
     optimization
@W:CL111 : rcb.vhd(94) | All reachable assignments to one_vector(4) assign '0'; register removed by
     optimization
@W:CL111 : rcb.vhd(94) | All reachable assignments to one_vector(5) assign '0'; register removed by
     optimization
@W:CL111 : rcb.vhd(94) | All reachable assignments to one_vector(6) assign '0'; register removed by
     optimization
@W:CL111 : rcb.vhd(94) | All reachable assignments to one_vector(7) assign '0'; register removed by
     optimization
@W:CL159 : ram_fsm.vhd(15) | Input cache_d is unused
@W:CL238 : rcb.vhd(122) | Latch state_transition.assert_sig0 enable evaluates to constant 0, optimized
```

rcb.vhd(303): The signals `pxcache_pixopin`, `pxcache_wen_all`, `vram_waddr`, `pxcache_pixnum` are all only assigned a value when this `if` condition is true, which means latches are generated to hold the previous value when the condition is false. Values must be assigned to these signals under all conditions in this process.

Also, there are some `std_logic` signals which are asserted to show a certain condition, but not deasserted when the condition does not hold, which is probably the intention. These signals are `pxcache_pw` and `change_curr_word`, and may also include some of the signals from the previous paragraph.

rcb.vhd(398): Similar to before, the signals `ram_addr` and `ram_start` must be assigned a value under all conditions.

rcb.vhd(94): `one_vector` should not be a signal. Since it is only used in one line (283) to increment a counter, and its value is never changed, it would make more sense to simply hard-code the value, or, even better, to use a constant.

ram_fsm.vhd(15): This warning that `cache_d` is unused stems from the fact that it is used in process `MERGE_PROC` to drive `data_merged`, but this signal is not read anywhere, and the whole process that drives only this signal will likely be optimised out. The intent was likely to output `data_merged` from this block in place of `data`.

rcb.vhd(122): This warning indicates that a latch was optimised out, so it is not necessary to fix it. The `assert` on line 140 will never execute, so this line is unnecessary.

2 FSMs

Not sure the `rangecheck` state is necessary. It seems to only decide which state to go to next, based solely on the current command. This decision-making could be done in the idle state. Furthermore, on moving out of the idle state, you can no longer be sure that the `dbb_bus` is unchanged. The command should be registered upon moving out of the idle state.

3 Reset

Reset input resets sub-entities and makes FSM transition back to the idle state. However, the `idle_counter_proc` and `current_word_register` processes are not reset.

4 Code Correctness

Many latches are instantiated, see section 1 for details. However, code is otherwise synthesisable (No times, `wait for`s, real numbers etc.).

The `for` loop in `ram_fsm.vhd(31)` loops over the size of the type `store_t`, which is defined in `pix_cache_pak.vhd`. Thus, the size of this loop is known (constant) at compile-time.

No complicated implementations such as gated/multiple clocks, signal-clocked flip-flops, or asynchronous sets or resets are used.

Code compiles properly, so nothing before the `wait for`s in clocked processes, or other non-VHDL 1993 code.

5 Design Correctness

Synplify gives no errors relating to signals not being singly-driven. Helper functions appear to return the right values given the input co-ordinates, according to the spec. Design appears sound.

6 Code Style

At `rcb.vhd(220)`, it is not possible for `vram_done` to take a value other than 0 or 1, so the final part of the `if` statement is unnecessary.

In `draw_px`, many `if elsif` constructs, where `case` would be more readable.