Automatic Register Cloning Avoidance

Arm® Partner Enablement Group

Confidential Proprietary notice

Arm Limited. Company 02557590 registered in England.
110 Fulbourn Road, Cambridge, England CB1 9NJ.
LES-PRE-20348

Web address

http://www.arm.com

Change history

Version	Revision	Date	Change
1.0	n/a	February 2015	Initial release
1.1	n/a	January 2018	Minor corrections

Contents

1 INTRODUCTION

Synthesis tools use register cloning to reduce the fan-out load on a register by making a mirror copy of the register and splitting the load between the two. A timing violation due to heavy load on a critical register can thus be avoided by insertion of an identical register that shares part of the original load.

Synthesis tools can pass the cloning information to formal logic equivalence checking (LEC) tools. LEC tools can use this additional information to identify a cloned register as identical to its master and not find a logic mismatch between RTL and the netlist with cloned registers.

When implementation tools automatically clone registers in order to meet timing requirements, identical copies of registers placed across the chip might initialize to different values on power-up. If all of the registers in the design are resettable, this will not be an issue because reset will initialize the circuit to a known start state. However, RTL designers commonly use non-reset registers[1] in data-path pipelines and other areas of the design to save power and area; in some cases it is not necessary and in other cases it is not feasible. If a register and its clone power up to different values and if reset cycling does not reach them, the underlying circuit will be in a state that was not originally specified and hence not validated by the RTL designer.

[1] This term refers to registers with no asynchronous set or reset.

2 DESCRIPTION OF REGISTER CLONING ISSUE

As shown in Figure 1 – Example of Register Cloning for Timing Closure below, synthesis tools can split a heavy load on the output of register R2 by creating a copy R2_clone which is then placed closer to the logic it drives and away from the original R2 location. A timing violation of -3ns thus becomes a timing path with a positive slack of 1ns.

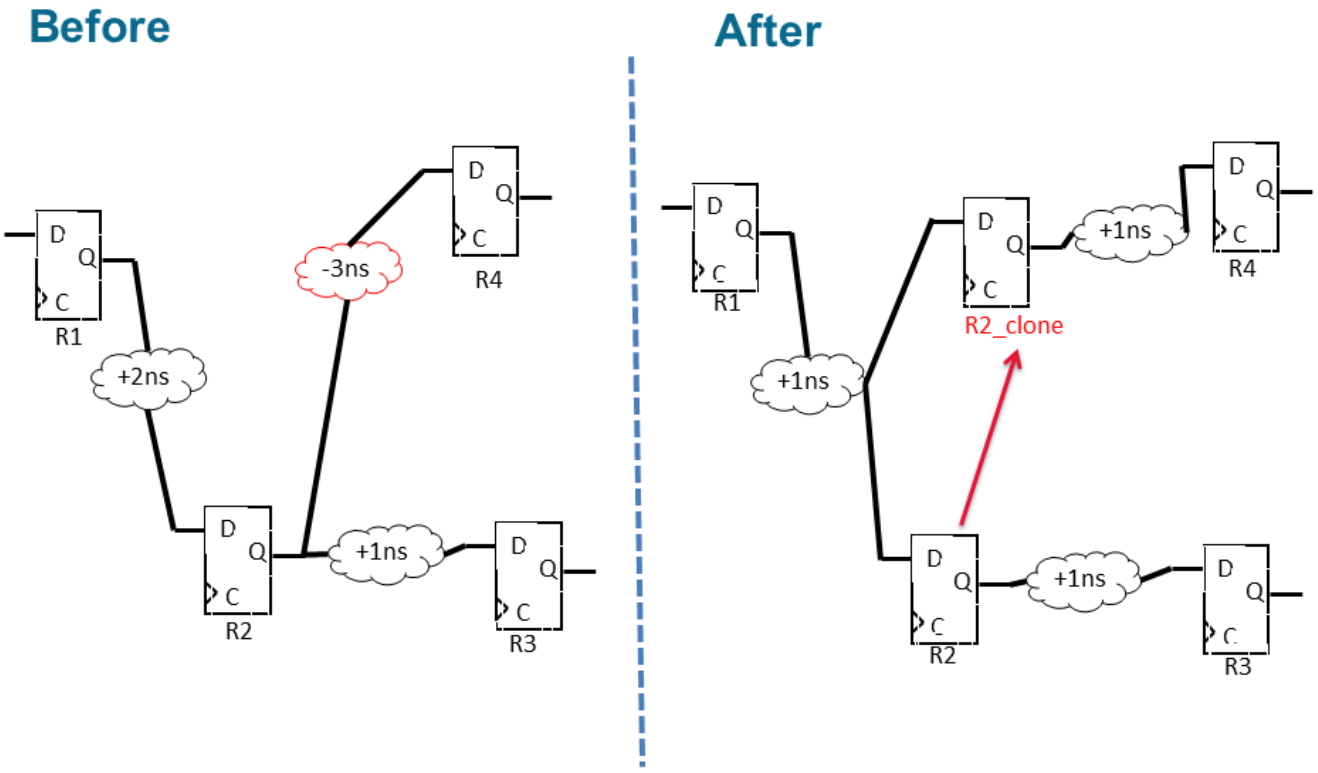

Figure 1 – Example of Register Cloning for Timing Closure

LEC can use the cloning information from synthesis to prove that the RTL logically matches the netlist with register clones. Timing closure becomes relatively straightforward for EDA tools with this technique compared to other more complex optimization algorithms, especially since this comes with an LEC matching guarantee.

A cloned register might not necessarily initialize to the same state as its master on power-up, and that would contradict the LEC match assertion made by synthesis. Additionally, there are other concerns related to register cloning that are not addressed by EDA tools:

- If cloning happens by default (this can change from release to release from the same vendor).
- If cloning is limited to registers with set or reset.
- If cloning is applied to clock enable registers driving clock gating logic.
- If cloning stops after scan chains are connected.
- If further optimizations treat copies of the same register individually by applying local optimizations (sizing, Vt, channel length, enable, etc.).

- If the tools are capable of identifying registers in a state machine compared to ones in pipelines for cloning.
- If registers from synchronizers are exempt from cloning[1].
- If registers with ECC protection are cloned[2].

Some of the above considerations are safe in the context of register cloning while others are not, and it would be highly desirable for the user to be able to control what is acceptable. Currently, such granularity in user control is not supported by EDA tools.

When non-reset registers are cloned, synthesis tools would ideally guarantee that these identical copies result in exactly the same functionality under all conditions – especially on power-up. Depending on how the tools transform local cones of logic independently with each copy, one branch of a clock tree might be turned off if a cloned register initializes with a 0 and was factored into the clock enable of an ICG (integrated clock gate) while the other copy of the same register might initialize with a 1, presenting a conflicting value to its downstream logic.

Many factors contribute to whether a pair of original/clone registers initialize differently on power-up: their location on the die, power grid layout, drive strength, Vt, process conditions, on-chip variation, clock tree, etc. A mismatched initialization injects a new state into the original FSM (finite state machine) that can potentially produce a dead lock, hang, or other unforeseen failures since verification did not account for this additional state.

[1] Synchronizer registers should not be cloned by EDA tools because cloning would split a single point of meta-stability.
[2] Registers with ECC should not be automatically cloned since ECC logic targets a specific flop to detect soft errors.

3 ARM'S RESPONSE TO THIS ISSUE

Automatic cloning of non-reset registers is potentially dangerous for any RTL (whether in Arm cores or in a partner's SoC logic) unless the synthesis or LEC tools guarantee that cloning registers results in precisely the same function behaviour under all conditions including power on reset.

When the default cloning of non-reset registers identified a hang in one of the Arm graphics cores[1] (where no such problem existed prior to synthesis tools changing their default to enable automatic cloning), Arm investigated and identified the problem as a subtle bug that only exhibited as a simulation vs. synthesis mismatch when the registers driving the logic were cloned and subsequently initialized to different states. Without register cloning the logic operated identically between RTL simulation and synthesized netlist.

Arm has re-examined internal review policies, design standards, formal and simulation based checking flows, RTL coding guidelines, etc. across the organization. The findings and current status (**as of the initial publication of this document in 2015**) are summarized below:

- Targeted simulations with bounds checking enabled were shown to detect the specific bug on the susceptible graphics cores (refer to the errata in the footnote). These simulations have been run on the most recent Arm IP (Cortex-A53, Cortex-A57, Cortex-A72, CCI-400/500, CCN-5XX, CoreSight SoC, GIC-500, etc.) and no further instances of the bug were detected.
- Arm is unaware of any failures in the field caused by register cloning on any Arm products in the past. This suggests that the older Arm products might not be affected by this problem.
- Testing of legacy products will carry practical limits on this effort (number of IP products/releases and time/effort involved), Arm will limit the testing to recent products only and not test older IP.
- Arm core and fabric reset standards require several cycles of clocked reset assertion to propagate the reset value across pipeline stages that are not explicitly reset.
- Arm RTL coding rules strongly discourage coding styles that lead to simulation vs. synthesis mismatches. This issue was only visible in combination with register cloning and had not previously been seen. It will now be covered explicitly by the Arm rules. Arm is also pursuing automated approaches to ensuring 100% compliance of future IP RTL releases to these enhanced RTL coding rules.

Unfortunately, even with the testing mentioned above, it is not possible to prove absolutely that Arm IP is immune to failures when cloned non-reset registers are introduced into the netlist. This is because netlist applicable test vectors cannot cover the entire design and there are too many combinations to test in RTL. Because the failure cited in the errata was unexpected, Arm does not know if there are other RTL coding styles that could provoke a failure in combination with the synthesis tools' cloning of non-reset registers.

For these reasons, Arm recommends that automatic non-reset register cloning feature is explicitly disabled in partner physical implementation work.

In addition, Arm recommends EDA vendors clarify the guarantees their synthesis tools offer around ensuring the same functional behaviour when non-reset registers are cloned and may power-up into different states.

[1] See implementation category B early errata defect IDs: 840169, 840220, 840219 notification from ARM dated 22nd Dec 2014.

4 GOING FORWARD

Arm is working closely with EDA vendors to discuss standard practices in register cloning, whether automatic register cloning should be enabled by default, if an explicit warning should be issued when non-reset registers are being cloned, etc.

Because the exact conditions assumed by the EDA tools in cloning a register are not documented and because a pair of cloned non-reset registers initializing differently on power-up can cause problems on silicon, **Arm strongly recommends that our partners disable this feature**.

The following switches can be used to accomplish that:

In Cadence[1],

set_attr iopt_sequential_duplication false /

In Synopsys[2],

set_app_var compile_register_replication false

set compile_register_replication_across_hierarchy false

If a partner enables this feature unknowingly (possibly because of a hidden tool default), Arm recommends that partner uses gate level simulations on the netlist with X-propagation targeting the pair of non-reset master/clone registers to rule out any unforeseen failures. A potential alternative is to perform a series of gate level simulations on the netlist with each group of cloned non-reset registers initialized to all possible permutations of power-up state.

No known issues exist with cloning of registers with set or reset[3].

In ongoing trials, Arm has not seen a performance drop-off when register cloning is explicitly disabled.

[1] As of RTL Compiler 14.10 release and Genus, register cloning is disabled by default.
[2] As of Design Compiler 2013.03 release, register cloning is enabled by default.
[3] As noted previously, synchronizers and registers with ECC should not be automatically cloned by EDA tools.