

Cycle Model Studio

Version 9.1.0

Flop Behavior in Cycle Models: When Data and Clocks Change Simultaneously

Non-Confidential



Cycle Model Studio

Flop Behavior in Cycle Models: When Data and Clocks Change Simultaneously

Copyright © 2017 ARM Limited. All rights reserved.

Release Information

The following changes have been made to this document.

Change History			
Date	Issue	Confidentiality	Change
February 2017	A	Non-Confidential	Rebrand/Release with 9.1.0.

Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of ARM Limited (“ARM”). **No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.**

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, ARM makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version shall prevail.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to ARM’s customers is not intended to create or refer to any partnership relationship with any other company. ARM may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any signed written agreement specifically covering this document with ARM, then the signed written agreement prevails over and supersedes the conflicting provisions of these terms.

Words and logos marked with ® or ™ are registered trademarks or trademarks of ARM Limited or its affiliates in the EU and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. You must follow the ARM trademark usage guidelines <http://www.arm.com/about/trademarks/guidelines/index.php>.

Copyright © ARM Limited or its affiliates. All rights reserved.
ARM Limited. Company 02557590 registered in England.
110 Fulbourn Road, Cambridge, England CB1 9NJ.

In this document, where the term ARM is used to refer to the company it means “ARM or any of its subsidiaries as appropriate”.

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

<http://www.arm.com>

Flop Behavior in Cycle Models: When Data and Clocks Change Simultaneously

The values on Cycle Model inputs, outputs, and nets are updated whenever the test driver calls the `carbonSchedule` function. This document explains how the Cycle Model processes simultaneous transitions on clock and data inputs.

Asynchronous resets are handled in the same manner as clocks in Cycle Models. For the purposes of this discussion, the term “clock” refers to both clocks and asynchronous resets.

1.1 Simple Clocks

1.1.1 Data flows immediately through one flop

By default, clocks are assumed to be faster than the data input, so clocks win any race. Therefore, when the data and clock transition at the same time, the Cycle Model passes data input through the flop in the same call to `carbonSchedule` that the data changes.

The following example shows a primary input `pi1` and primary clock `pclk` that transition at the same time. (For simplicity in these examples, the data is assumed to pass unchanged through the combinational block; if 0x7 goes in, then 0x7 comes out.) Note that the output of the flop, `q1`, transitions at the same time as the input.

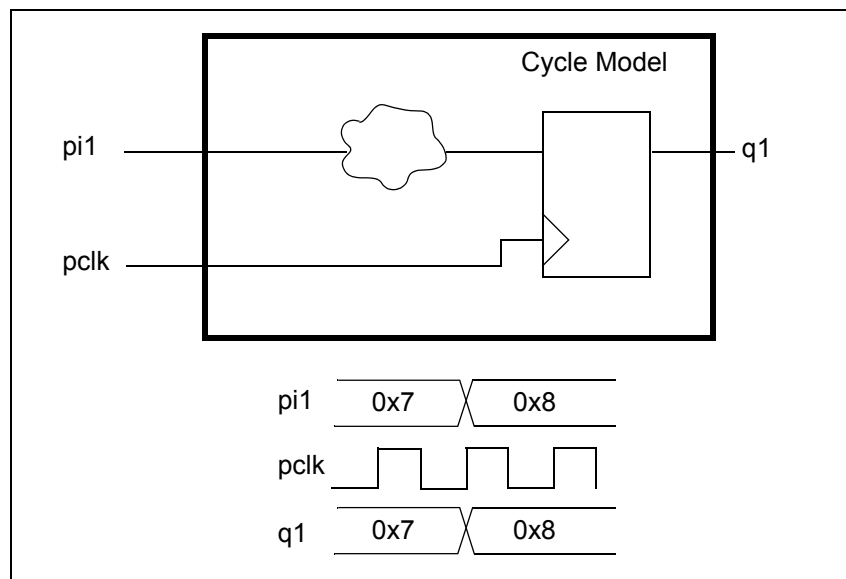


Figure 1.1 Data from a primary input flows immediately through a flop

1.1.2 Data does not flow immediately through two flops

The Cycle Model does not pass data through two flops in the same call to `carbonSchedule`. Consider the following example where a combinational block is fed both by a primary input `pi2` and a flop output `q1`. The result is stored in a combinational net called `c1`. As explained in the previous section, if `pi2` and `pclk` transition at the same time, `q2` reflects the new value of `pi2`. However, if `pi1` transitions at the same time as `pi2` and `pclk`, its new value is reflected in `q1` but not `q2` in the same `carbonSchedule` call. The new value is reflected in `q2` in the subsequent `carbonSchedule` call (that is, at the next active clock edge).

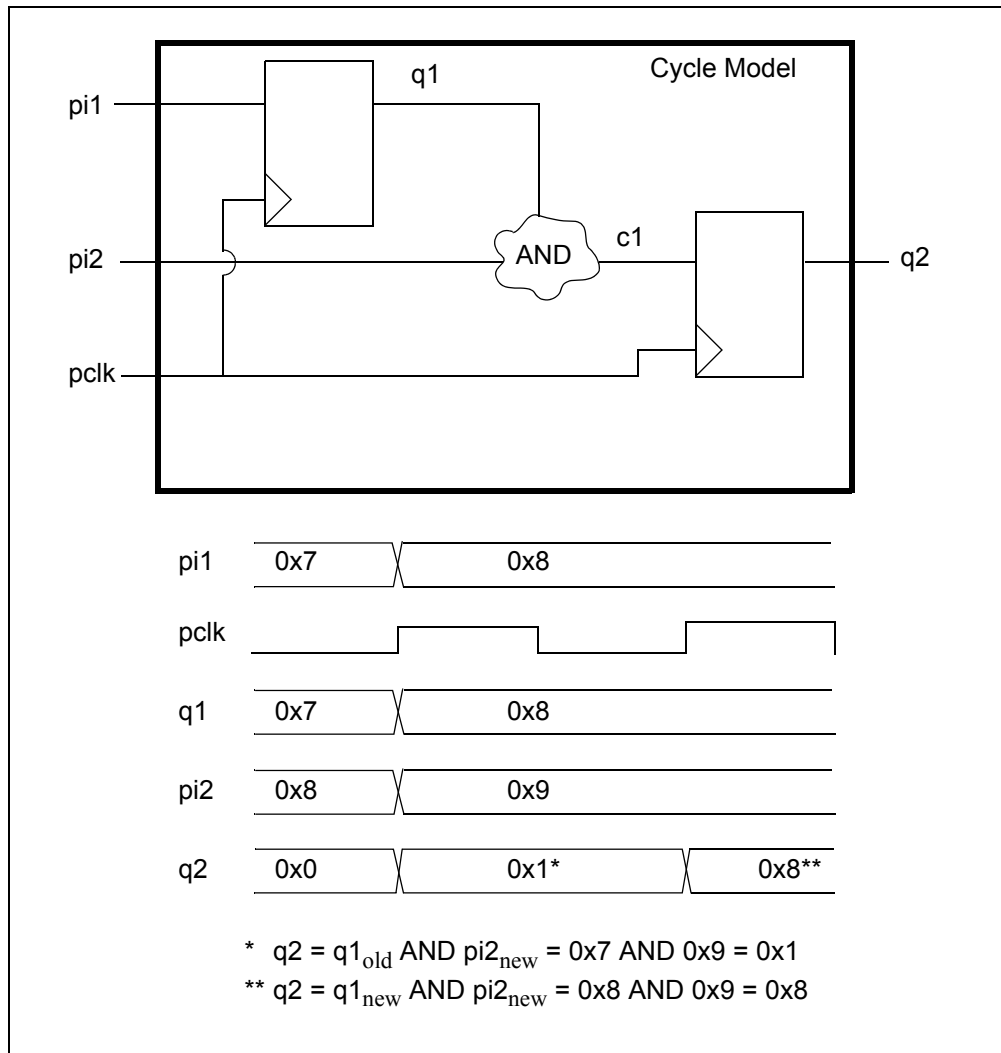


Figure 1.2 Data does not flow through two flops in the same call to `carbonSchedule`

1.2 Clocks as Data

When a signal acts both as a clock and as data, its data behavior is different than in the previous examples. If the source of the clock or clock tree is a primary input, that data is delayed so that it does not make it through the data pin of a flop in a single `carbonSchedule` call. In the following example, the primary input `pi1` flows immediately into `q1`. However, `q2` always shows the old value of `pclk1` on the rising edge of `pclk2`. This means that `q2` is the inverse of `pclk1`.

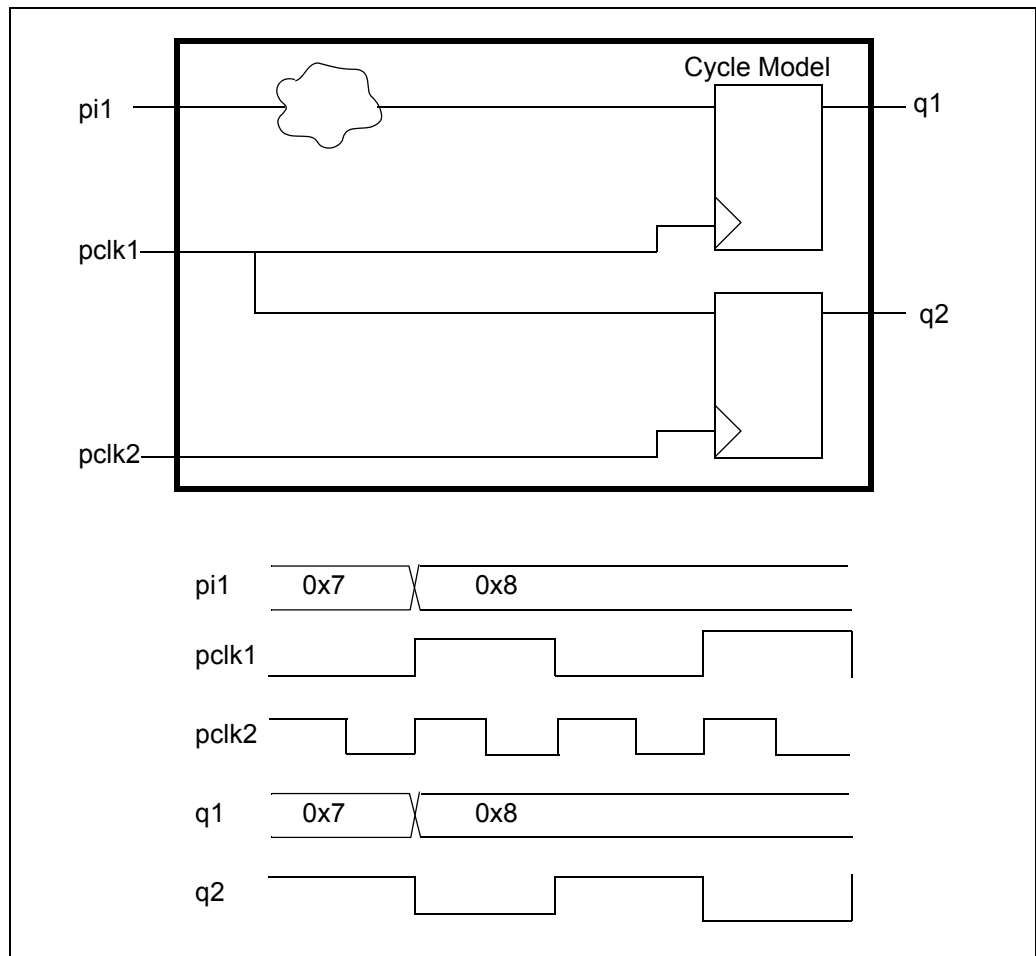


Figure 1.3 When a clock signal is also a data signal, the clock-as-data does not flow through a flop on the first call to `carbonSchedule`

1.3 Derived Clocks and Clock Trees

Derived clocks and clock trees behave in the same manner as primary clocks, which means that when they also supply a data line, their data is not passed through a flop on a single `carbonSchedule` call. In the following example, `q2` always sees the old value of `dclk` and therefore also sees the old value of the clock tree, `pen` and `pclk1`.

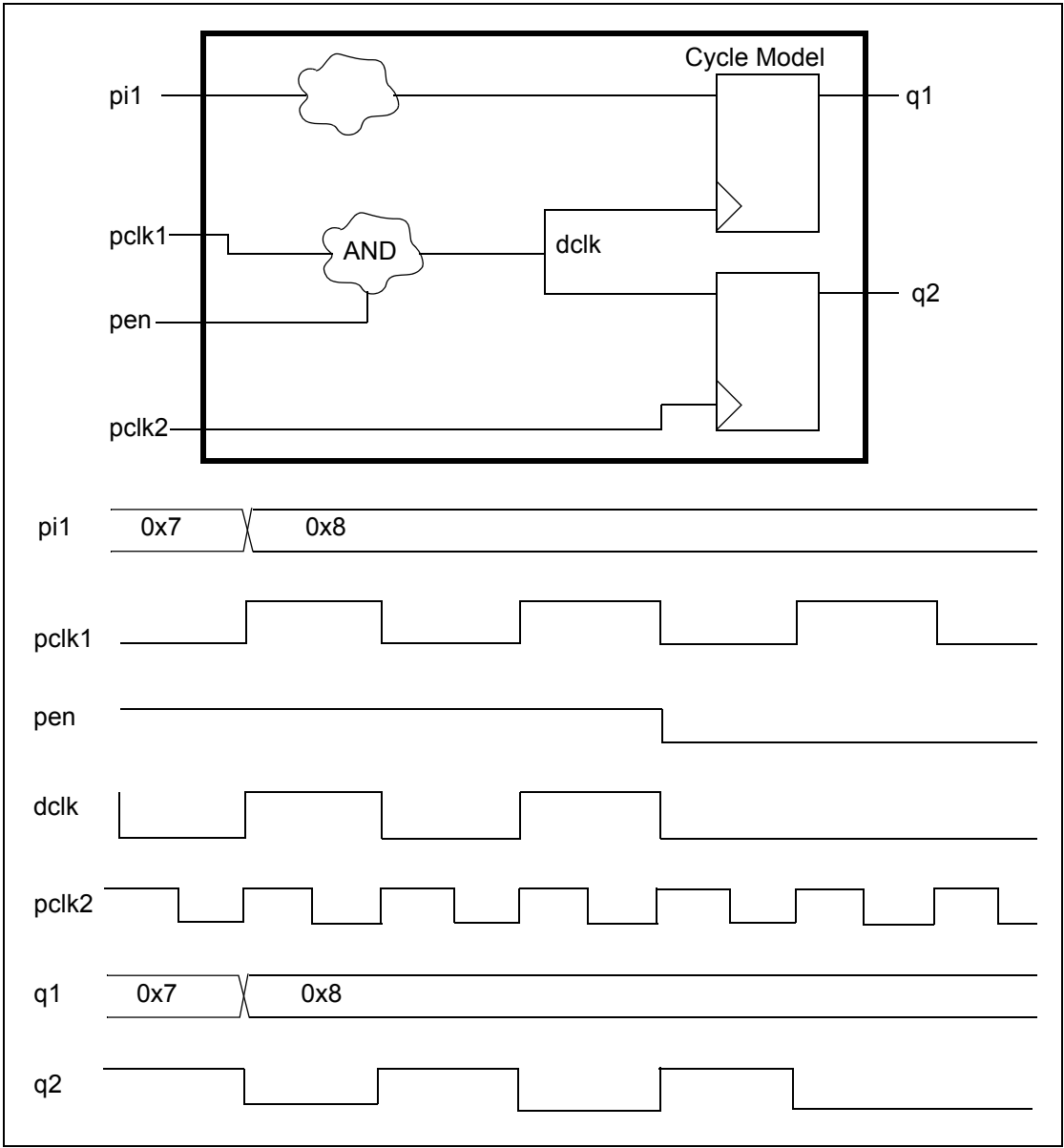


Figure 1.4 Derived clocks show the same behavior as clocks-as-data

In the following example, even though `pclk1` is not explicitly used as a clock, it is part of the clock tree for flop `q1`. Therefore its value is delayed by one `carbonSchedule` call (that is, until the next `pclk2` rising edge) before being passed through the `q2` flop.

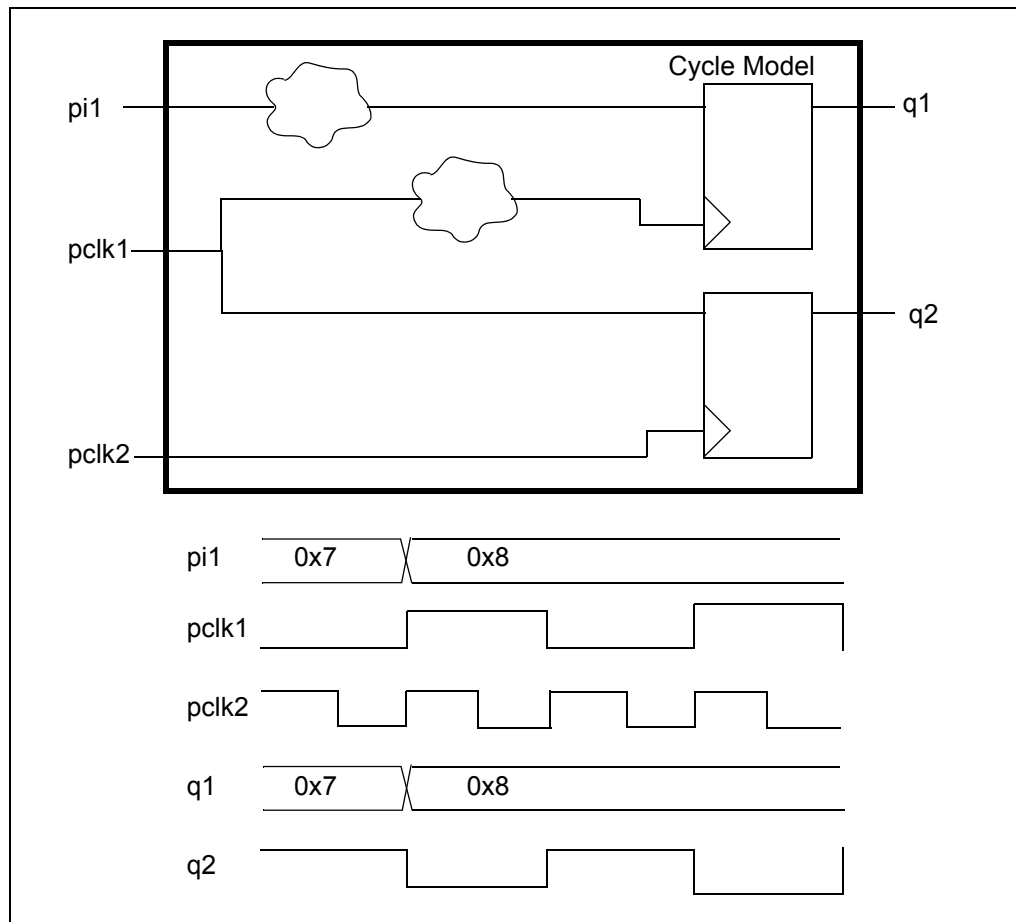


Figure 1.5 Data inputs that feed a clock tree show the same behavior as clocks-as-data

