Clock Domain Crossing Standard
Version 0.1

October 2023

Abstract: In-house and externally purchased IPs are often combined in SOCs. It can prove challenging to verify these SOCs because a mixture of verification tools and methodologies that do not integrate can be used. Ensuring that a common clock domain crossing interface standard that every tool can translate their native format to and from is the intent of this standard. With this interface standard, every IP developer's verification tool of choice is run to verify and produce collateral, and the standard format is generated for SOCs that used a different tool. And with this standard, efficiently translating from provided collateral into a tool of choice is possible for every SOC. A way to specify all the information necessary to do accurate clock domain crossing, reset domain crossing, and glitch structural analysis is afforded. Attributes for a block that can be used to facilitate a correct clock domain crossing and reset domain crossing integration of that block in an encompassing design are identified. The limitations of the set of attributes are also addressed; that is, the clock domain crossing and reset domain crossing schemes for which the set of attributes is sufficient and the schemes the defined set of attributes is not guaranteed to support are identified.

Keywords: CDC, clock domain crossing, glitch, RDC, reset domain crossing, functional verification.

Notices

Accellera Systems Initiative (Accellera) Standards documents are developed within Accellera and the Technical Committee of Accellera. Accellera develops its standards through a consensus development process, approved by its members and board of directors, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are members of Accellera and serve without compensation. While Accellera administers the process and establishes rules to promote fairness in the consensus development process, Accellera does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

Use of an Accellera Standard is wholly voluntary. Accellera disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other Accellera Standard document.

Accellera does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or suitability for a specific purpose, or that the use of the material contained herein is free from patent infringement. Accellera Standards documents are supplied **"AS IS."**

The existence of an Accellera Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of an Accellera Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change due to developments in the state of the art and comments received from users of the standard. Every Accellera Standard is subjected to review periodically for revision and update. Users are cautioned to check to determine that they have the latest edition of any Accellera Standard.

In publishing and making this document available, Accellera is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is Accellera undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other Accellera Standards document, should rely upon the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of Accellera, Accellera will initiate action to prepare appropriate responses. Since Accellera Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, Accellera and the members of its Technical Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments for revision of Accellera Standards are welcome from any interested party, regardless of membership affiliation with Accellera. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Comments on standards and requests for interpretations should be addressed to:

Accellera Systems Initiative.
8698 Elk Grove Blvd Suite 1, #114
Elk Grove, CA 95624
USA

NOTE—Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. Accellera shall not

be responsible for identifying patents for which a license may be required by an Accellera standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Accellera is the sole entity that may authorize the use of Accellera-owned certification marks and/or trademarks to indicate compliance with the materials set forth herein.

Authorization to photocopy portions of any individual standard for internal or personal use must be granted by Accellera, provided that permission is obtained from and any required fee is paid to Accellera. To arrange for authorization please contact Lynn Garibaldi, Accellera Systems Initiative, 8698 Elk Grove Blvd Suite 1, #114, Elk Grove, CA 95624, phone (916) 670-1056, e-mail lynn@accellera.org. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained from Accellera.

Suggestions for improvements to the Clock Domain Crossing Standard 0.1 are welcome. They should be posted to the Clock Domain Crossing (CDC) community forum at:

https://forums.accellera.org/forum/56-cdc-draft-lrm-release-discussion/

The current Working Group (WG) web page is:

https://accellera.org/activities/working-groups/clock-domain-crossing

Participants

The CDC WG is entity-based. At the time this standard was developed, the CDC WG had the following active participants:

Iredamola Olopade, Intel Corporation, *Chair*
Sean ODonohue, Synopsys Inc, *Vice-Chair*
Farhad Ahmed, Siemens EDA, *Secretary*
Jeanne Foster, *Technical Editor*

Agnisys, Inc: Anupam Bakshi, Devender Khari, Pinku Kumar, Raushan Kumar, Abhinandan Reddy

AMD: Jia Zhu

ARM, Ltd.: Edwin Dankert, Stephen Hill, Xiaodong Zhuang

Arteris, Inc.: Said Derradji

Blue Pearl Software, Inc.: Bill Gascoyne, David Wallace

Cadence Design Systems, Inc.: Pradeep B, Aparna Dey, Greg Milano, Souradip Sarkar, Anshuman Seth, Konrad Sikora

Infineon Technologies: John Jebakumar, Ambuja Rashinkar, Joachim Voges, Joseph Yackzan
Intel Corporation: Jeremy Anderson, Boon Chong Ang, Mallikarjuna Badam, Lauren Carlson, Sharvil Desai, Pawel Duc, Eldad Falik, Sachin Jain, Dinesh M, William Mok, Iredamola Olopade, ATif Razak, Rohit Sinha, Chee Yoong Tan Lee, Jebin Vijai, Lee Fueng Yap

Marvell International, Ltd.: Chetan Choppali Sudarshan

Microsoft Corporation: Serena Badran-Louca

NVIDIA Corporation: Sangeetha Sudha Nakerikanti, Ping Yeung,

NXP Semiconductors: Inayat Ali, Shweta Pujar

Qualcomm Incorporated: Suman Chalana, Prasad Nandipati

Renesas Electronics Corp.: Kaiwen Chin, Abhay Kejriwal, Kranthi Pamarthi, Esra Sahin Basaran

Siemens EDA: Farhad Ahmed, Abdul Moyeen

STMicroelectronics: Jean-Christophe Brignone, Diana Kalel, Laurent Maillet-Contoz, Julian Massicot

Synopsys, Inc.: Jerome Avezou, Sudeep Mondal, Sean ODonohue

Texas Instruments, Inc.: Lakshmanan Balasubramanian, Abhinav Parashar

At the time of standardization, the CDC WG had the following eligible voters:

Agnisys, Inc.	**Microsoft**
ARM Limited	**NVIDIA Corporation**
Arteris, Inc.	**QualcommTechnologies, Inc.**
Blue Pearl Software	**Renesas Electronics Corp.**
Cadence Design Systems, Inc.	**Siemens EDA**

Infineon Technologies AG

Intel Corporation

Marvell Technology, Inc.

STMicroelectronics N.V.

Synopsys, Inc.

Texas Instruments Incorporated

This introduction is not part of IEEE P XXXX-20XX, IEEE Draft Standard for....

Introduction

The purpose of this standard is to provide the electronic design automation (EDA), semiconductor, and system design communities with a well-defined specification for unified handling of CDC by vendor tools used across IPs and SOCs.

Industry design style can be largely classified into two types (not exhaustive), namely:

Monolithic design: The entire product and its various hierarchies are all designed by one team. All aspects of the design are accessible (and editable) by that team. This type requires knowledge and expertise of all aspects of the design but provides control and autonomy over all aspects of the design (including tools and methodology). Depending on how extensive the product is, and how large (or small) the team is, this style can require a considerable time investment.

IP/SOC design: The product is composed of various IPs that can be designed in-house (by this team, or another team) or purchased externally. This means that the required knowledge and expertise of all aspects of the design are not available in the SOC team, and as a result, the SOC team has less autonomy and control over all aspects of the design. This style can significantly accelerate product development depending on the quality of the IPs and how easy it is to integrate into the product SOC.

NOTE—Most of the industry is shifting from monolithic design styles to IP/SOC design styles, and many IP companies provide their IP to multiple SOC product companies. In addition, there is no expectation that every IP and every SOC company is using the same tools and methodology for validating the design, including CDC analysis.

For most collateral types (for example, SystemVerilog, cluster test environment, low power, and so forth), there exist standards that govern its use; hence, integration for these types of collateral has been largely straight-forward. But for CDC collateral the industry has not had a clean way to handle tool and methodology differences across IPs and SOCs.

The following list describes various methods of handling the differences in the absence of this standard proposal:

a) Monolithic SOC design: Gives tool and methodology autonomy but delays time-to-market.
b) Black-boxing IP: Assumes the IP is clean and ignores checking for integration issues by black-boxing. This helps with initial time-to-market but introduces quality risks that can lead to silicon respin, which eventually impacts the product's time-to-market.
c) Re-verification of IP: Re-verifies IPs on CDC tools that might be different from those used by the IP provider. These teams lack the knowledge and expertise to do a thorough and efficient job, thus putting both time-to-market and quality at risk, even after employing considerable resources and effort.
d) Common tools between SOC and IPs: Requests all IP companies they purchase from provide collateral analyzed with their tool of choice. In this scenario, IP companies that provide IP to different SOC companies must run multiple tools to fit all SOC needs. Even if the IP company agrees, they push back on their delivery date to master new tools and collateral, which eventually impacts SOC time-to-market.

None of the approaches above is able to provide sufficient quality in reasonable time. However, defining a standard format to capture clock domain crossing (CDC), reset domain crossing (RDC), and glitch intent enables interoperability of CDC collateral generated by any CDC verification tool.

Contents

List of figures

List of tables

Clock Domain Crossing Standard Version 0.1

1. Overview

This common CDC interface standard provides the following:

a) Every vendor's tool can translate its native format to and from the standard to maintain its IP.
b) Every IP provider can run its tool of choice to verify and produce collateral and generate the standard format for SOCs that use a different tool.
c) Every SOC can quickly and safely integrate either native collateral or translate from the standard collateral into their tool of choice to ensure time-to-market goals and quality.

A limited feasibility study was conducted on a subsystem with multiple IPs connected by AMBA interfaces across three vendor tools with limited support from the vendors. It showed that 99.5% of what was identifiable in a flat run (using any of the three vendor tools) was also identifiable if the native abstraction collateral was replaced with an XML representation and translated across the vendor tools.

NOTE—This feasibility study was only for CDC and did not include reset domain crossing (RDC) or glitch analysis.

1.1 Scope

The following list details the Clock Domain Crossing Standard scope:

a) Support tool-independent output collateral for CDC, RDC, and glitch structural analysis
b) Provide human-readable and machine-parsable attributes
c) Support customizable extensions (for example, to support complex user conditions)
d) Support hierarchical analysis
e) Support power-aware designs
f) Support multi-modal IP/SOC analysis
g) Support multi-instance IPs
h) Support multi-parameters of IPs
i) Support multiple interface protocols (for example, AMBA, I2C, PCIe, UCIe, and so forth)
j) Provide extensibility to cover input collateral cases (for example, constraints, waivers, and so forth) to enable high-quality re-verification of an IP using alternate tools
k) Support assertions necessary to guarantee integration quality because CDC, RDC, and glitch analysis is only structural
l) Support other design styles (for example, FPGA and Analog) that follow similar standards

NOTE—The “interface protocols” item above is a way to ensure that multiple circuit styles (CDC crossing types) are supported to maintain good integration quality.

Using standard interfaces that can be verified independently (for example, with VIP from independent sources) for the integration of independently designed IPs limits the potential for bugs to be introduced. The above limitation does not prevent innovation between sub-blocks, but instead discourages any complications from these innovations from being spread across independent blocks that might not have been verified with the same tools. Using customizable extensions of the format can address exceptions, but these extensions are not guaranteed by the standard.

1.2 Purpose

This standard is intended to do the following:

a) Enable all EDA vendors to develop tools that meet this specification in generated collateral
b) Enable IP companies to generate collateral using various vendors and their tools
c) Enable SOC companies to consume generated collateral from different vendors’ tools into their tool of choice

1.3 Word usage

The word *shall* indicates mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (*shall* equals *is required to*).[1,2]

The word *should* indicates that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required (*should* equals *is recommended that*).

The word *may* is used to indicate a course of action permissible within the limits of the standard (*may* equals *is permitted to*).

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

1.4 Contents of this standard

The organization of the remainder of this standard is as follows:

— Clause 2 provides references to other applicable standards that are assumed or required for this standard.
— Clause 3 defines terms and acronyms used throughout the different specifications contained in this standard.
— Clause 4 lists CDC attributes along with their type, accepted values, whether they are mandatory, and clarifying comments.
— Annex A provides an informative bibliography.

[1] The use of the word *must* is deprecated and cannot be used when stating mandatory requirements; *must* is used only to describe unavoidable situations.

[2] The use of *will* is deprecated and cannot be used when stating mandatory requirements; *will* is only used in statements of fact.

2. References

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

IEEE Std 1685™-2022, IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows

IEEE Std 1800™-2017, IEEE Standard for SystemVerilog Unified Hardware Design, Specification and Verification Language.[3, 4]

[3]The IEEE standards or products referred to in this clause are trademarks of the Institute of Electrical and Electronics Engineers, Inc.

[4]IEEE publications are available from the Institute of Electrical and Electronics Engineers, Inc., 445 Hoes Lane, Piscataway, NJ 08854, USA (https://standards.ieee.org/).

3. Definitions, acronyms, and abbreviations

For the purposes of this document, the following terms and definitions apply. *The Authoritative Dictionary of IEEE Standards Terms* [B1][5] should be referenced for terms not defined in this clause.

3.1 Definitions

component: A physical and logical construction that relates inputs to outputs.

glitch: A transient pulse on a signal that is caused by a race between signals in its fan-in. This includes asynchronous data path (for example, combinational logic going into synchronizers or reconvergence of parallel synchronizers) as well as clock and reset trees.

port: A connection on the interface of a SystemVerilog module or very high speed integrated circuit (VHSIC) hardware description language (VHDL) entity.

qualifier: A signal that controls a crossing; e.g., the select of a mux in a mux-based synchronizer governing the crossing.

3.2 Acronyms and abbreviations

CDC	clock domain crossing
Combo	combinational logic
EDA	electronic design automation
HDL	hardware description language
inout	input/output port
internal_sync	internal synchronizer
IP	intellectual property blocks
LRM	language reference manual
MTBF	mean time before failure
PWG	proposed working group
RDC	reset domain crossing
QoR	quality of results
RTL	register transfer level
SDC	Synopsys Design Constraints
SOC	system on chip or products that consist of various IPs and glue-logic

[5]The numbers in brackets correspond to those of the bibliography in Annex A.

SVA	SystemVerilog Assertions
VIP	validation IP or test environment used to test aspects of an IP
WG	working group

NOTE—CDC WG refers to CDC-, RDC-, and glitch-related (asynchronous data crossing) checks necessary for correct functionality of clock, reset, and data glitch (associated with clock and reset crossings).

4. CDC Attributes

The CDC attributes are expressed in Tcl for the CDC tool-level format, but you have the option to translate this Tcl to IP-XACT for packaging the IP. In this version of the LRM, we have included pseudo code for the Tcl to support EDA vendors with tool development. In addition, we will include the IP-XACT schema for completeness in a future version of the LRM.

The Tcl captures data required from input, output, and verification collateral in a human-readable and machine-parsable format to provide accurate CDC, RDC, and glitch structural analysis by any EDA tool. The CDC attributes format supports customizable extensions for complex user conditions. In addition, these attributes are applicable to other design styles (e.g., FPGA and Analog) that follow similar standards.

The CDC attributes facilitate a correct CDC and RDC integration of blocks in an encompassing design and address a specific set of known industry standard interfaces. The CDC standard identifies both the CDC and RDC schemes the attributes support and those schemes the defined set of attributes is not guaranteed to support. Table 1 groups the supported attributes by domain (module, parameter, port, tool, and design) and lists them along with their type, accepted values, whether they are mandatory, and clarifying comments. Use the drawings and accompanying pseudocode examples that include the attributes in the remaining sections of this clause to understand clock relationships and various crossings and failure modes:

a) Module and parameter attributes
b) Port attributes
c) Clock definitions
d) Clock relationships
e) Failure modes

Table 1—CDC Attributes

Domain	Attribute	Type	Values	Mandatory	Comments
module	**name**	string	{module name}	Yes	
parameter	**name**	string		Yes	
parameter	**value**	range-list	{values}	Optional	
parameter	**type**	string	{int, string, boolean}	Optional	
parameter	**ignore**	string	{ignore}	Optional	
port	**name**	string	{name}	yes	
port	**direction**	defined set	{input, output, inout, internal}	yes	
port	**type**	defined set	{data, clock, async_reset, qualifier}	yes	Mandatory “yes” applies to async reset only. For async resets, type is async_reset; for sync resets, type is data.
port	**logic**	defined set	{combo, buffer, inverter, glitch_free_combo, internal_sync}	optional	

Table 1—CDC Attributes *(continued)*

Domain	Attribute	Type	Values	Mandatory	Comments
port	**qualifier_from_clock**	; separated list	{clock-names}	optional	required for qualifier and data with qualifier, async is implied when data does not have -associated_clock
port	**associated_clocks**	; separated list	{clock-names}	optional	the default for “to”
port	**associated_resets**	; separated list	{reset-names}	optional	
port	**associated_inputs**	; separated list	{ports}	optional	can be used for feed-through logic or qualifiers
port	**associated_outputs**	; separated list	{ports}	optional	
port	**clock_group**	string	{clock group}	yes	yes only applicable to clock type
port	**reset_group**	string	{reset group}	yes	yes only applicable to reset type
port	**qualifiers**	; separated list	{associated-ports}	optional	
port	**polarity**	defined set	{high, low, both}	yes	yes only applicable to async reset
port	**ignore**	string	{blocked, hanging}	optional	
port	**quasi_static**	; separated list	{can be any, <clocks to which it is quasi_static>}	optional	
port	**set_case_analysis**	; separated list	{binary, hex, and of any length}	optional	
port	**gray_coded**	boolean	{true, false:default}	optional	
port	**clock_period**	string	{clock period}	optional	
tool	**name**	string	{EDA name}	yes	if no EDA tool name is captured, it implies hand crafted by user
tool	**version**	string	{tool version}	yes	N/A if user generated
design	**version**	string	{design milestone}	optional	
design	**date**	string	{collateral generation date}	yes	
design	**username**	string	{user/tool that generated the collateral}	optional	
design	**description**	string		optional	

Table 1—CDC Attributes *(continued)*

Domain	Attribute	Type	Values	Mandatory	Comments
set_cdc_clock_group					
	name	string	{logical_ck1, logical_ck2...}	yes	

4.1 Module and parameter attributes

The attributes include one module attribute. Refer to the following pseudo code and Figure 1, which depict the `mod0` module.

```
module -name mod0
```

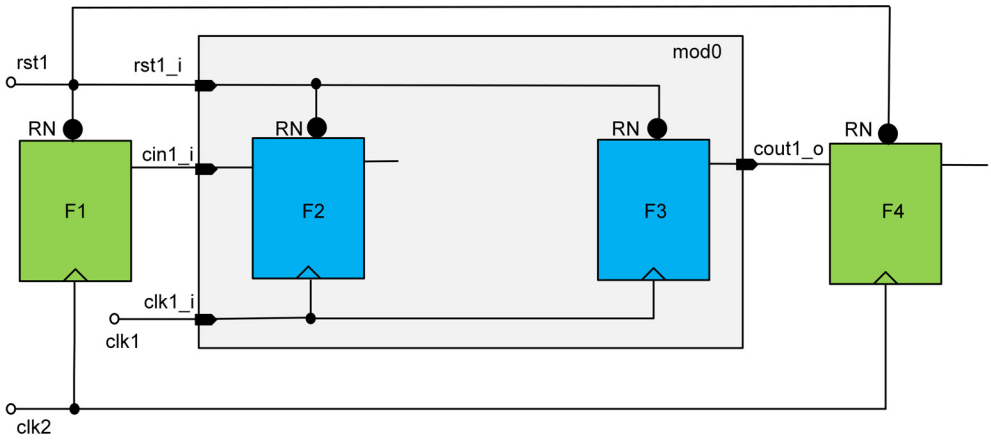

Figure 1—Module attribute

The attributes also include one mandatory and three optional parameters. The `name` parameter is mandatory. The `value`, `type`, and `ignore` attributes are optional. Refer to the following RTL code and pseudo code for a better understanding of parameters.

```
module mod0
#(
parameter       WIDTH = 32,
parameter       ID = FFF111)
(
...
);
```

```
parameter -name WIDTH -type decimal -value 32
parameter -name ID -ignore true
```

4.2 Port attributes

Several port attributes are detailed in this section. Refer to the following pseudo code and Figure 2, which depict the `name`, `direction`, and `type` attributes.

```
port -name cin1_i -direction input -type data
```

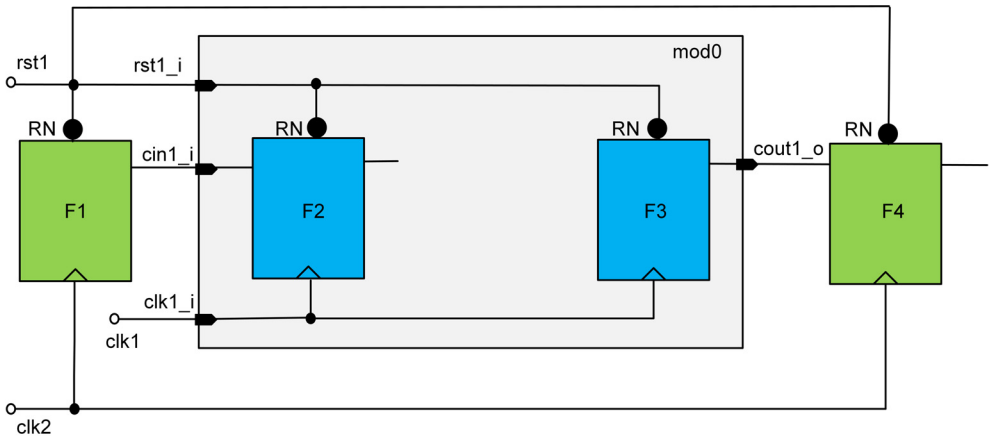

Figure 2—Port attributes: name, direction, and type

The following pseudo code and Figure 3 depict the `logic` attribute with the `combo` value.

```
port -name in1_i -direction input -type data -logic combo -associated_clocks
clk1_i
```

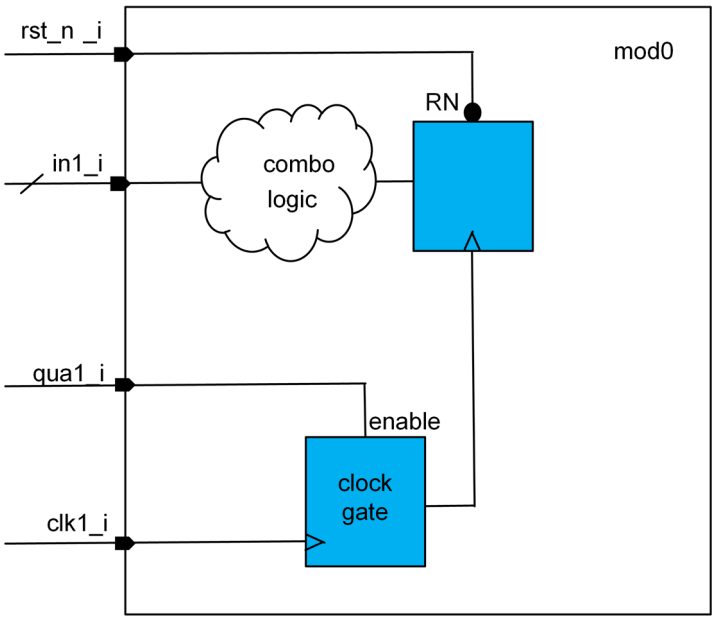

Figure 3—Port attribute: logic

The following pseudo code and Figure 4 depict the `qualifier_from_clock` attribute. This attribute is required for qualifiers and data with qualifiers.

```
port -name qua1_i -direction input -type qualifier -qualifier_from_clock vclk
-associated_clocks clk1_i -associated_inputs in1_i
```

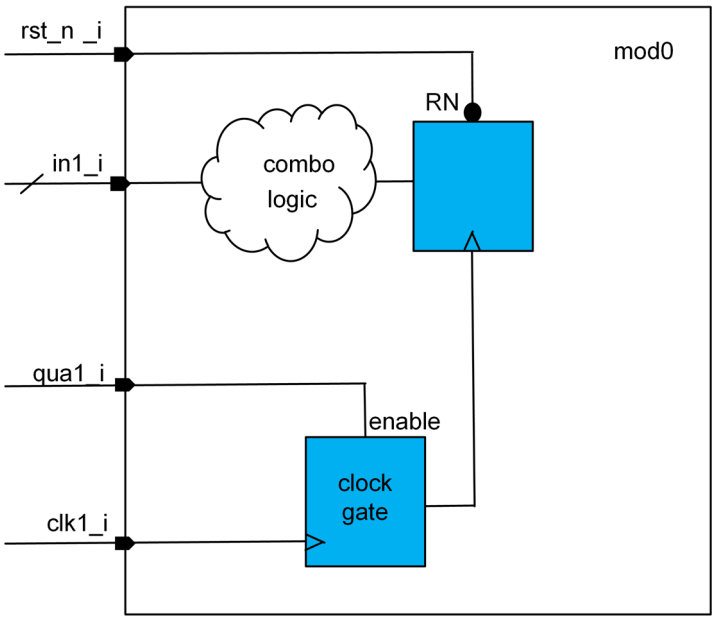

Figure 4—Port attribute: qualifier_from_clock

The following pseudo code and Figure 5 depict the optional `associated_clocks` attribute.

```
port -name cin1_i -direction input -type data -associated_clocks clk1_i
```

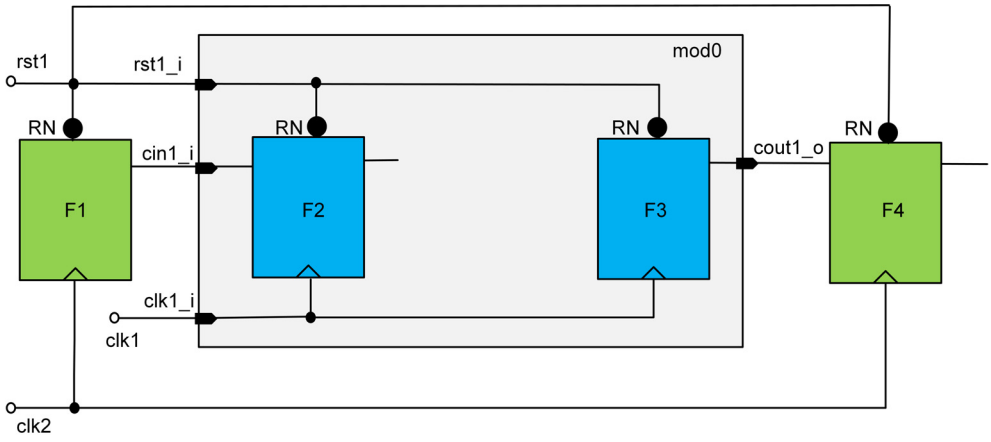

Figure 5—Port attribute: associated_clocks

The following pseudo code and Figure 6 depict the optional `associated_reset` attribute.

```
port -name cin1_i -direction input -type data -associated_reset rst1_i
```

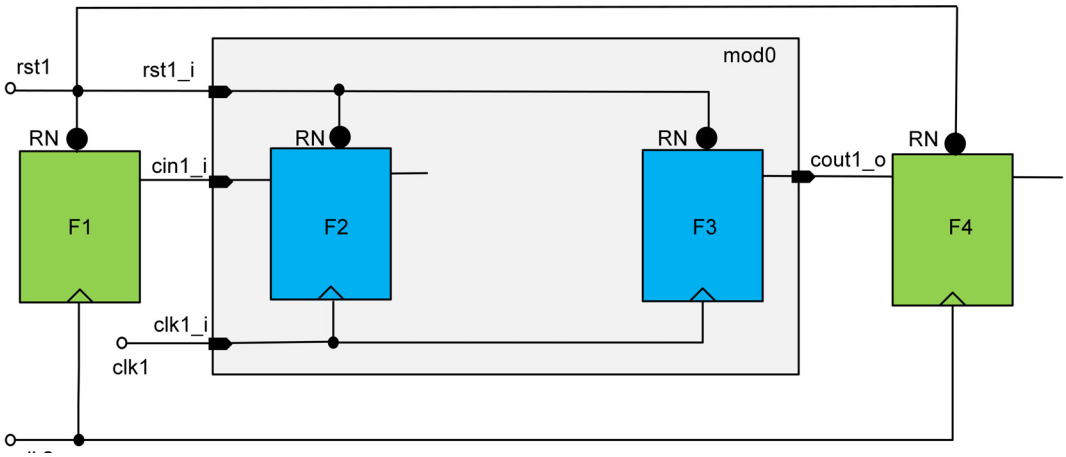

Figure 6—Port attribute: associated_reset

The following pseudo code and Figure 7 depict the optional `associated_inputs` attribute.

```
port -name qua1_i -direction input -type qualifier -qualifier_from_clock vclk
-associated_clocks clk1_i -associated_inputs in1_i
```

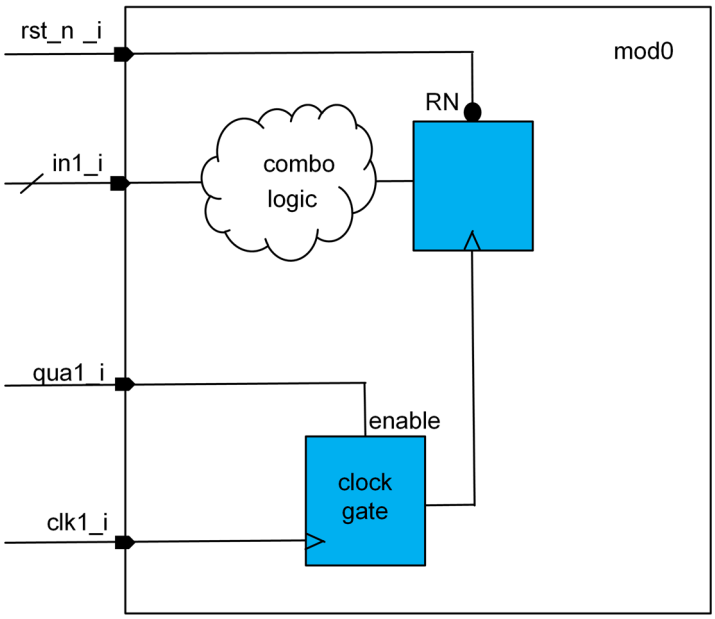

Figure 7—Port attribute: associated_inputs

Figure 8 depicts feedthrough logic. The associated pseudo code includes the optional attribute `-associated_outputs` for the feedthrough output.

```
port -name ftout -direction output -type data -associated_input ftin
```

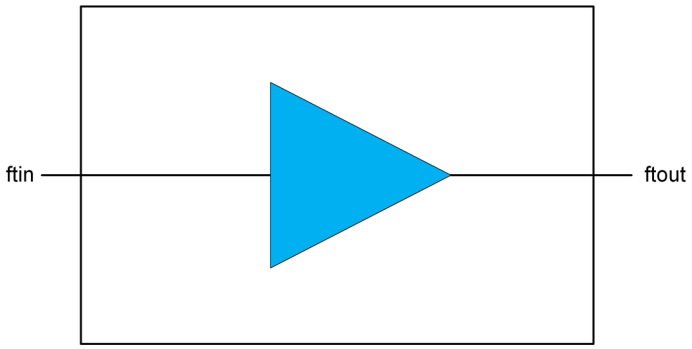

Figure 8—Feedthrough logic

4.3 Clock definitions

This section shows three examples of clock definitions. Figure 9 depicts two asynchronous clocks `tx` and `rx`.

```
port -name txclk -direction input -type clock -clock_group tx

port -name rxclk -direction input -type clock -clock_group rx
```

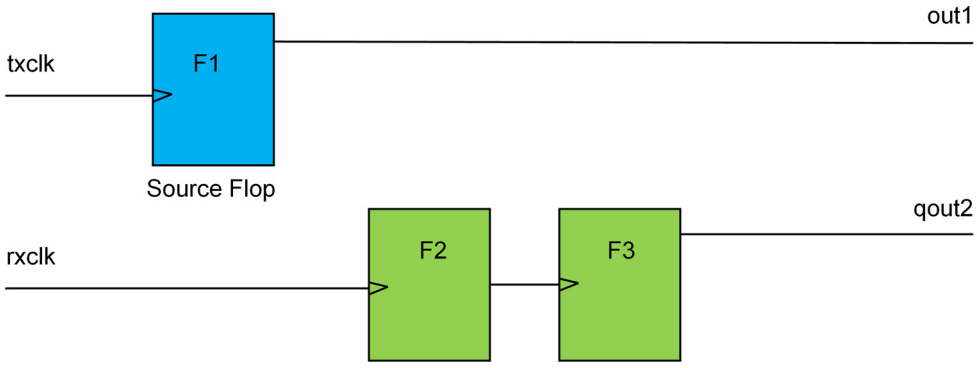

Figure 9—Clock definition A

Figure 10 depicts three asynchronous clocks `tx`, `rx`, and `new`.

```
port -name txclk -direction input -type clock -clock_group tx

port -name rxclk -direction input -type clock -clock_group rx

port -name newclk -direction input -type clock -clock_group new
```

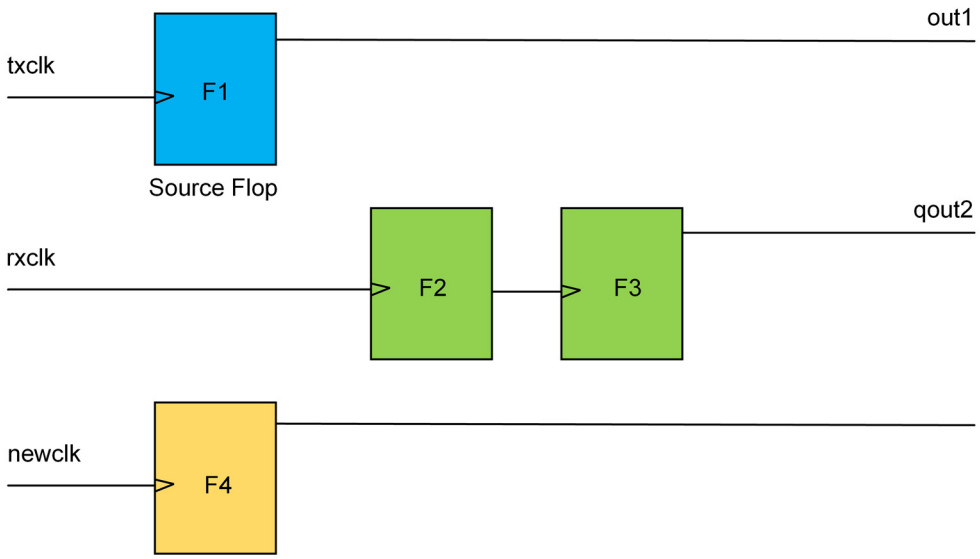

Figure 10—Clock definition B

Figure 11 depicts three clocks `tx`, `rx`, and `new`. In this example, `tx` and `new` are asynchronous to `rx`.

```
port -name txclk -direction input -type clock -clock_group txnew

port -name rxclk -direction input -type clock -clock_group rx

port -name newclk -direction input -type clock -clock_group txnew
```

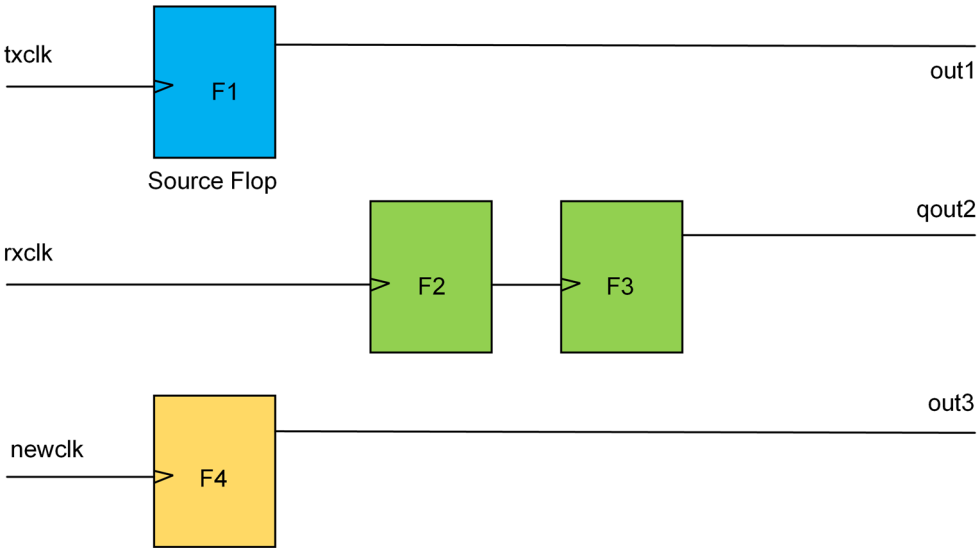

Figure 11—Clock definition C

4.4 Clock relationships

Relationships between clocks shall be explicitly defined regardless of the tool producing collateral. The `set_cdc_clock_group` command is used to clearly define clock relationships in terms of domains. The default assumption for CDC is that clocks are asynchronous unless specified as synchronous. A clock may be a member of more than one clock group. Two clocks are considered synchronous if a clock group contains both of them.

Each figure in this section includes three clock groups defined as follows:

```
port -name ck   -type clock -direction input -clock_group ck_logical
port -name gck0 -type clock -direction input -clock_group gck0_logical
port -name gck1 -type clock -direction input -clock_group gck1_logical
```

Blue solid arrows denote synchronous clock relationships. Red dashed arrows denote asynchronous clock relationships.

Figure 12 depicts one clock domain with synchronous clocks. The `set_cdc_clock_group` command defines the domain name and the synchronous clocks that belong to the domain.

```
set_cdc_clock_group -name ck_gck0_gck1_logical {ck gck0 gck1}
```

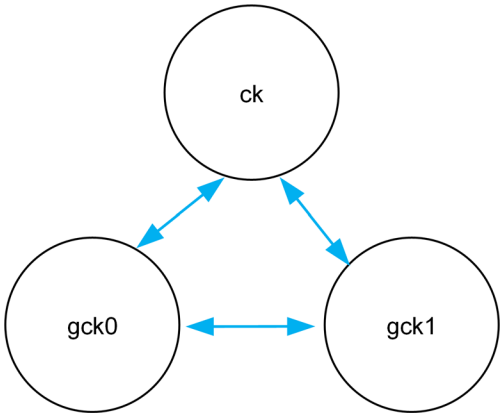

Figure 12—One clock domain

Figure 13 depicts two clock domains. The `ck_logical` domain contains `ck` and the `gck0_gck1_logical` domain includes clocks `gck0` and `gck1`.

```
set_cdc_clock_group -name ck_logical {ck}
set_cdc_clock_group -name gck0_gck1_logical {gck0 gck1}
```

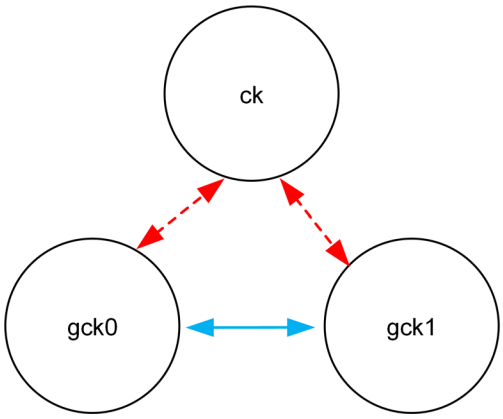

Figure 13—Two clock domains

Figure 14 depicts three clock domains with one clock in each. The `ck_logical` domain includes `ck`, the `gck0_logical` domain includes `gck0`, and the `gck1_logical` domain includes `gck1`.

```
set_cdc_clock_group -name ck_logical {ck}
set_cdc_clock_group -name gck0_logical {gck0}
set_cdc_clock_group -name gck1_logical {gck1}
```

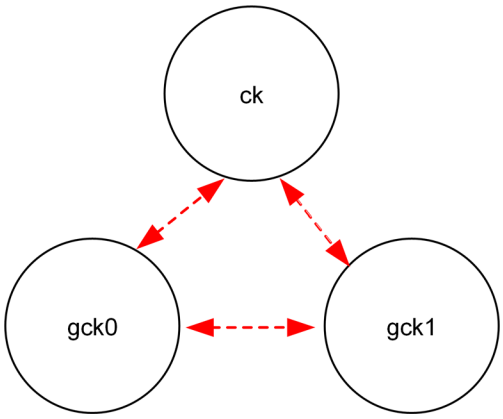

Figure 14—Three clock domains

Figure 15 depicts two clock domains. The `ck_gck0_logical` domain includes two clocks, `ck` and `gck0`. The `ck_gck1_logical` domain includes two clocks, `ck` and `gck1`.

```
set_cdc_clock_group -name ck_gck0_logical {ck gck0}
set_cdc_clock_group -name ck_gck1_logical {ck gck1}
```

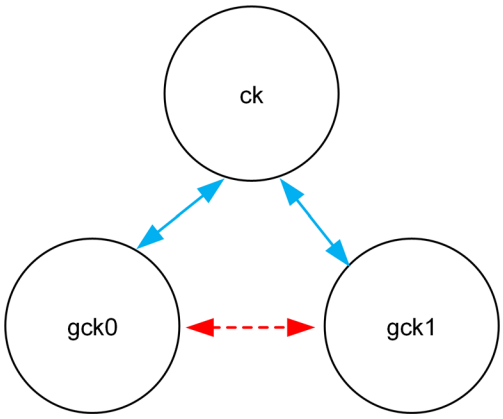

Figure 15—Two clock domains

4.5 Failure modes

This section shows attributes that may help a tool detect failure modes and their reasons in cases where an abstracted block is integrated into an encompassing design.

Figure 16 depicts the propagation of metastability with a mean time before failure (MTBF) that is too low. The failure is due to a missing synchronizer. This scenario assumes that `cin1_i` has an asynchronous driver. This failure mode can occur at inputs of type `data` or of type `qualifier`, which are described in the following models.

```
port -name cin1_i -direction input -type data -associated_clocks clk1_i

port -name cin1_i -direction input -type qualifier -qualifier_from_clock clk2
-associated_clocks clk1_i
```

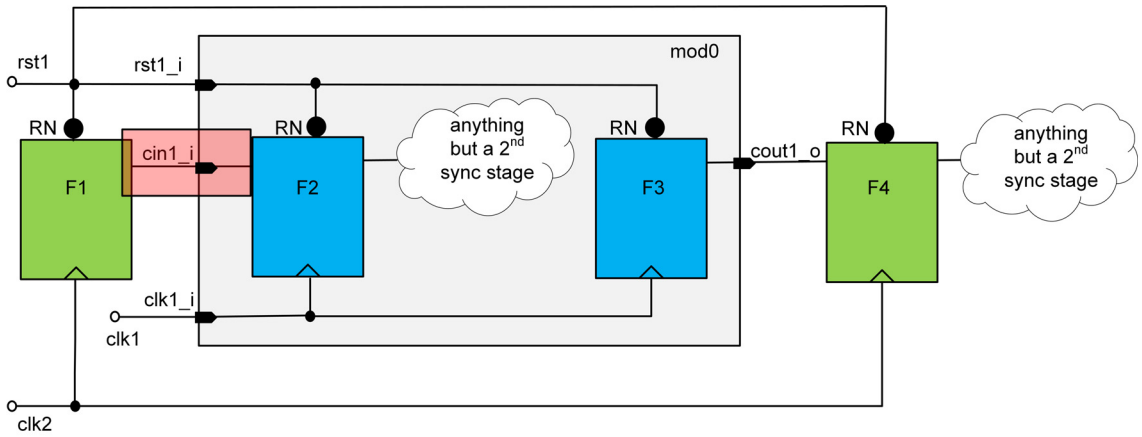

Figure 16—Failure mode due to missing synchronizer

Figure 17 depicts the propagation of an intermediate (random or metastable) data value. The failure is due to a missing qualifier.

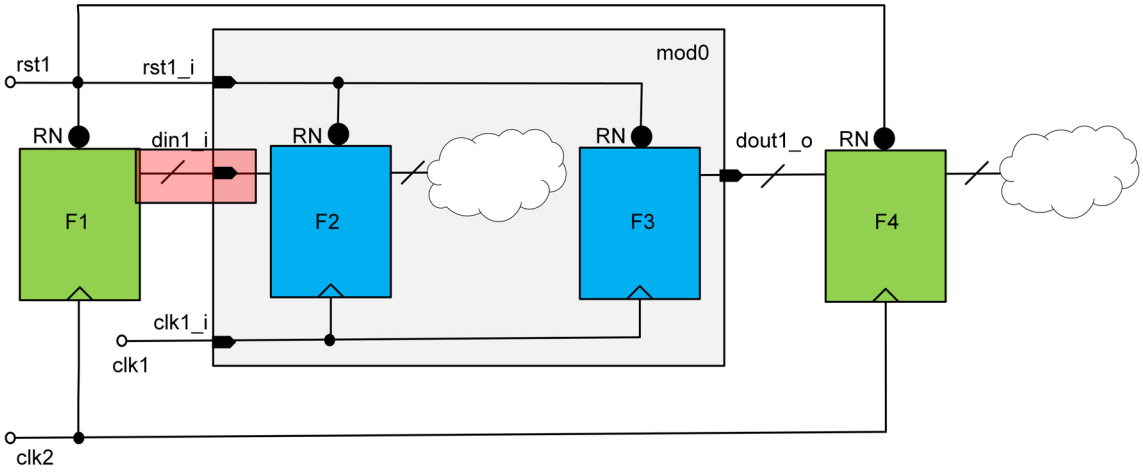

Figure 17—Failure mode due to missing qualifier

To fix the failure, add a synchronization scheme inside `mod0` that is controlled by a qualifier signal as shown in the following modeling.

```
module -name mod0
port -name din1_i -type data -associated_clocks clk1_i
-qualifier qual

port -name qual -type qualifier -qualifier_from_clock clk2
-associated_clocks clk1_i
-associated_inputs din1_i
```

Figure 18 depicts an asynchronous deassertion of an asynchronous reset (CDC on reset tree). The failure is due to a missing reset synchronizer.

```
module -name mod0
port -name rst1_i -direction input -type async_reset -associated_clocks clk1_i
-polarity low

module -name driver
port -name out -direction output -type async_reset -associated_clocks clk2
-polarity low
```

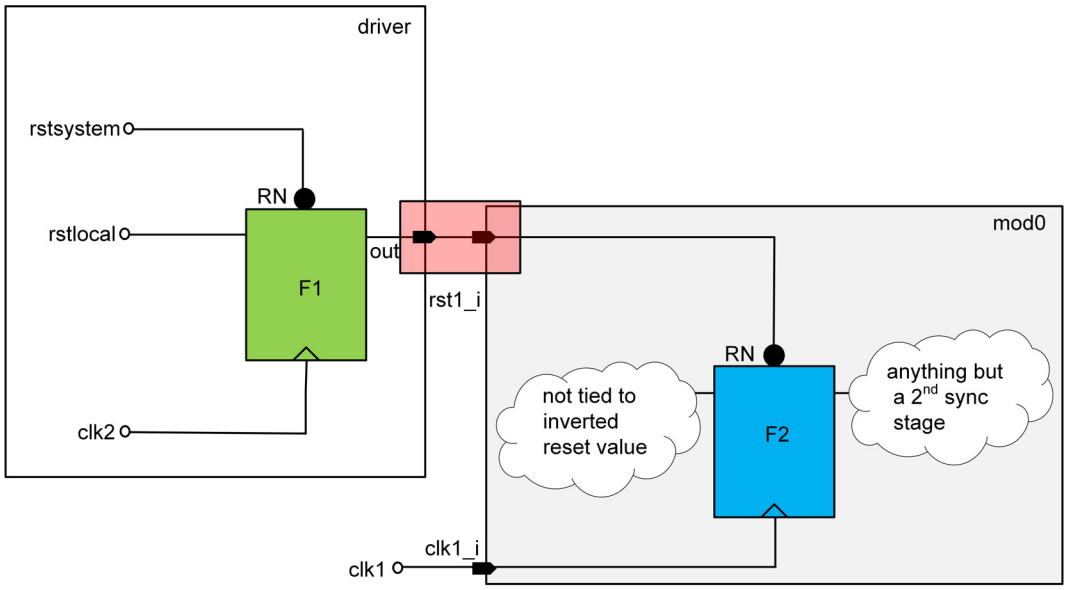

Figure 18—Failure mode due to missing reset synchronizer

We cannot have combo logic that can glitch on the CDC path and the reset path. Figure 19 shows that if the combo logic can glitch on a CDC path and reset network, there will be a violation.

```
module -name mod0
port -name out1_o -direction output -type data -logic combo
-qualifier_from_clock clk1 -associated_inputs {in1_i; in2_i}

module -name mod0
port -name out2_o -direction output -type data -logic combo
-qualifier_from_clock clk2 -associated_inputs {in3_i; in4_i}
```

NOTE—The attribute `-logic combo` describes a potentially glitching combinatorial logic. The attribute `-logic glitch_free_combo` describes the opposite.

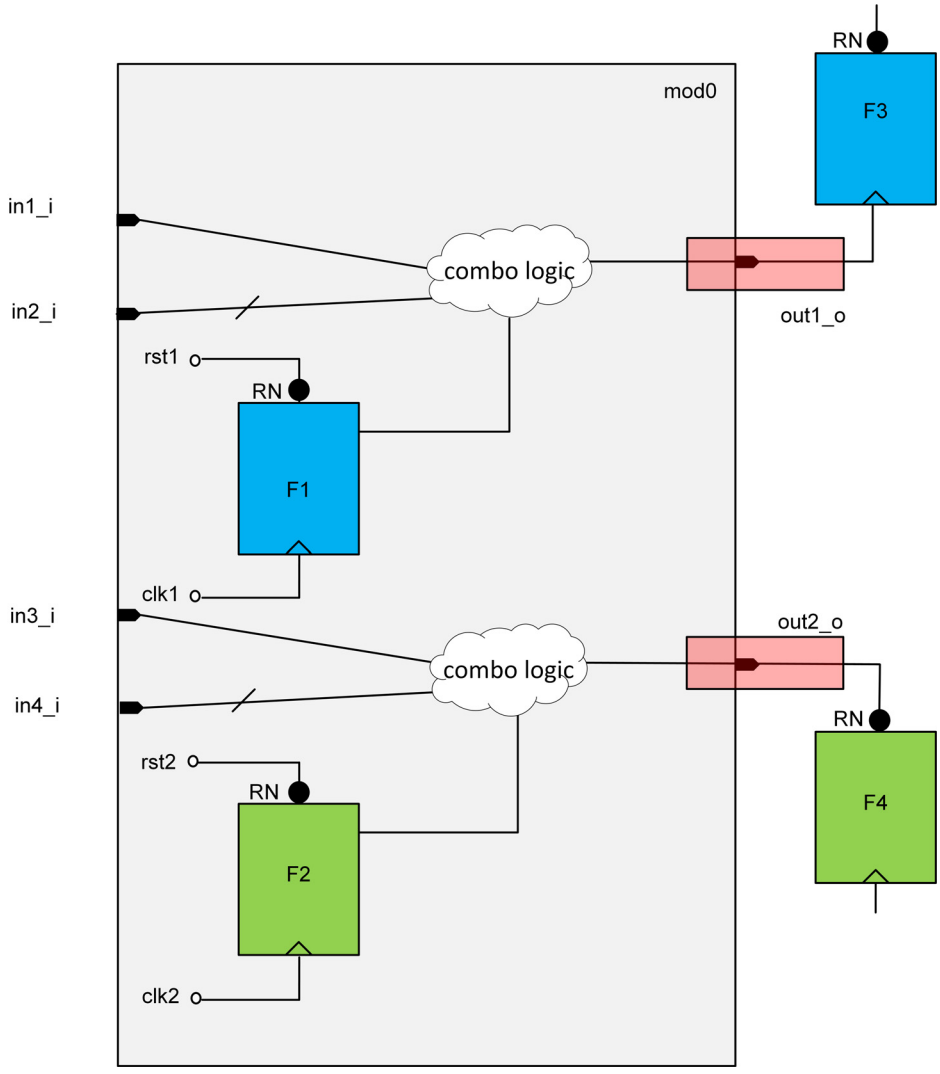

Figure 19—Failure mode due to combo logic driving clock/reset

Annex A

(informative)

Bibliography

[B1] IEEE 100, *The Authoritative Dictionary of IEEE Standards Terms,* Seventh Edition. New York: Institute of Electrical and Electronics Engineers, Inc.