



1
2
3
4
5
6
7
8
9

Clock Domain Crossing Standard Version 0.3 Draft for Public Review

July 15, 2024

1

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

16

17 **Keywords:** CDC, clock domain crossing, glitch, IP-XACT, RDC, reset domain crossing, functional
18 verification.

19

20

21

22

23

24

25

26

27

28

29

30

31 AMBA® is a registered trademark of Arm® Limited (or its subsidiaries) in the US and/or elsewhere

1

Notices

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

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

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

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

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

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

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

39

40

41

42

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

43

44

45

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

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

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

6 Authorization to photocopy portions of any individual standard for internal or personal use must be granted
7 by Accellera, provided that permission is obtained from and any required fee is paid to Accellera. To arrange
8 for authorization please contact Lynn Garibaldi, Accellera Systems Initiative, 8698 Elk Grove Blvd Suite 1,
9 #114, Elk Grove, CA 95624, phone (916) 670-1056, e-mail lynn@accellera.org. Permission to photocopy
10 portions of any individual standard for educational classroom use can also be obtained from Accellera.

11 Suggestions for improvements to the Clock Domain Crossing Standard 0.3 Draft for Public Review are wel-
12 come. They should be posted to the Clock Domain Crossing (CDC) community forum at:

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

14 The current Working Group (WG) web page is:

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

1 Participants

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

4

5 **Iredamola Olopade**, Intel Corporation, *Chair*

6 **Sean ODonohue**, Synopsys Inc, *Vice-Chair*

7 **Farhad Ahmed**, Siemens EDA, *Secretary*

8

9 **Agnisys, Inc:** Anupam Bakshi, Devender Khari, Yogita Koli, Pinku Kumar, Raushan Kumar, Sunil
10 Kumar, Mayank Nigam, Abhinandan Reddy, Vineet Sharma, Rajiv Singh

11 **Analog Devices:** Gopalakrishnan Krishnan

12 **AMD:** Apoorv Aggarwal, David Courtright, Vishwanath Sundararaman, Jia Zhu

13 **ARM, Ltd.:** Edwin Dankert, Sylvain Duvillard, Pat Overs, Stephen Hill, David Murray, Xiaodong
14 Zhuang

15 **Arteris, Inc.:** Said Derradji

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

17 **Cadence Design Systems, Inc.:** Pradeep B, Luis Humberto Rezende Barbosa, Aparna Dey, Larry
18 Lai, Shivaji Magadum, Greg Milano, Mamta Parab, Souradip Sarkar, Anshuman Seth, Konrad
19 Sikora

20 **Huawei Technologies Sweden AB:** Bochen Yang

21 **Infineon Technologies:** John Jebakumar, Ambuja Rashinkar, Joachim Voges, Joseph Yackzan

22 **Intel Corporation:** Jeremy Anderson, Boon Chong Ang, Mallikarjuna Badam, Lauren Carlson,
23 Saransh Choudhary, Sharvil Desai, Pawel Duc, Eldad Falik, Sachin Jain, Dinesh M, William Mok,
24 Iredamola Olopade, ATif Razak, Rohit Sinha, Chee Yoong Tan Lee, Jebin Vijai, Lee Fueng Yap

25 **Marvell International, Ltd.:** Sam Bueti, Gaurav Chhabra, Thien Le, Chetan Choppali Sudarshan

26 **Microchip Technology Inc.:** Don Mills

27 **Microsoft Corporation:** Shelly Henry, Serena Badran-Louca

28 **NVIDIA Corporation:** Sangeetha Sudha Nakerikanti, Ping Yeung,

29 **NXP Semiconductors:** Inayat Ali, Lei Deng, Vishal Jain, Shweta Pujar, Gaurav Saini

30 **Qualcomm Incorporated:** Suman Chalana, Prasad Nandipati

31 **Renesas Electronics Corp.:** Kaiwen Chin, Ciro Ceissler, Abhay Kejriwal, Kranthi Pamarthi, Esra
32 Sahin Basaran, Fengzhou Wang

33 **Robert Bosch GmbH:** Hayek, Jan

34 **Siemens EDA:** Farhad Ahmed, Manish Bhati, Abdul Moyeen, Andrew Seawright

35 **STMicroelectronics:** Jean-Christophe Brignone, Diana Kalel, Laurent Maillet-Contoz,
36 Julian Massicot, Ashish Soni

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

38 **Texas Instruments, Inc.:** Lakshmanan Balasubramanian, Abhinav Parashar

39

¹ At the time of standardization, the CDC WG had the following eligible voters:

Analog Devices	Microchip Technology Inc.
Agnisys, Inc.	Microsoft
ARM Limited	NVIDIA Corporation
Arteris, Inc.	Qualcomm Technologies, Inc.
Blue Pearl Software	Renesas Electronics Corp.
Cadence Design Systems, Inc.	Robert Bosch GmbH
Huawei Technologies Sweden AB	Siemens EDA
Infineon Technologies AG	STMicroelectronics N.V.
Intel Corporation	Synopsys, Inc.
Marvell Technology, Inc.	Texas Instruments Incorporated

1

2

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

3 Introduction

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

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

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

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

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

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

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

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

¹None of the approaches above is able to provide sufficient quality in reasonable time. However, defining a
²standard format to capture clock domain crossing (CDC), reset domain crossing (RDC), and glitch intent
³enables interoperability of CDC collateral generated by any CDC verification tool.

1 Contents

2	List of figures.....	11
	List of tables.....	12
4	1. Overview.....	13
5	1.1 Scope.....	13
6	1.2 Purpose.....	14
7	1.3 Word usage.....	14
8	1.4 Contents of this standard.....	14
9	2. References.....	16
10	3. Definitions, acronyms, and abbreviations.....	17
11	3.1 Definitions.....	17
12	3.2 Acronyms and abbreviations.....	17
13	4. CDC Attributes.....	19
14	4.1 Module attribute.....	24
15	4.2 Parameter attributes.....	24
16	4.3 Port attributes.....	26
17	4.3.1 Port attributes: name, direction, and type.....	26
18	4.3.2 Port attribute: associated_from_clocks.....	27
19	4.3.3 Port attributes: associated_to_clocks.....	28
20	4.3.4 Defining a virtual clock.....	29
21	4.3.5 Port attribute: associated_from_reset and associated_to_reset.....	30
22	4.3.6 Port attribute: ignore.....	31
23	4.3.7 Port attribute: cdc_static.....	32
24	4.3.8 Port attribute: constant.....	34
25	4.3.9 Port attribute: associated_outputs.....	36
26	4.3.10 Port attribute: logic.....	36
27	4.4 async_reset.....	38
28	4.5 Modeling abstracted blocks.....	45
29	4.6 Clock definitions.....	48
30	4.7 Clock relationships.....	51
31	4.8 Failure modes.....	56
32	5. Support for RDC verification.....	60
33	5.1 Requirements for IP with control outside the IP.....	60
34	5.1.1 Scenario 1.....	60
35	5.1.2 Scenario 2.....	61
36	5.1.3 Scenario 3.....	62
37	5.1.4 Scenario 4.....	63
38	5.2 Requirements for IP with control within the IP.....	64
39	5.2.1 Scenario 1.....	64
40	5.2.2 Scenario 2.....	66
41	5.2.3 Scenario 3.....	68
42	5.2.4 Scenario 4.....	69
43	5.2.5 Scenario 5.....	70

16.	CDC TCL format	72
2	6.1 cdc_set_module	72
3	6.1.1 Syntax example	72
4	6.2 cdc_set_port	72
5	6.2.1 Syntax example	73
6	6.3 cdc_set_clock_group	74
7	6.3.1 Syntax example	74
8	6.4 cdc_set_param	75
9	6.4.1 Syntax example	75
10	7. CDC IP-XACT format	76
11	7.1 Top-level elements	76
12	7.2 Top element—accellera:wire	76
13	7.3 Data port type—accellera-cdc:data	77
14	7.3.1 IP-XACT code for accellera-cdc:data	78
15	7.4 Reset port type—accellera-cdc:asyncReset	79
16	7.4.1 IP-XACT code for accellera-cdc:asyncReset	79
17	7.5 Control port type—accellera-cdc:cdcControl and accellera-cdc:rdcControl	80
18	7.5.1 IP-XACT code for accellera-cdc:cdcControl	80
19	7.6 Control port type—accellera-cdc:componentCDCDef	80
20	7.6.1 IP-XACT code for accellera-cdc:componentCDCDef	81
21	Annex A	
22	(informative)	
23	Bibliography	82

1 List of figures

2	Figure 1—Module attribute	24
3	Figure 2—Port attributes: name, direction, type.....	26
4	Figure 3—Port attribute: associated_from_clocks	27
5	Figure 4—Specifying a virtual clock.....	29
6	Figure 5—Port attribute: associated_from_reset and associated_to_reset	30
7	Figure 6—Ignore: hanging.....	31
8	Figure 7—Ignore: blocked.....	31
9	Figure 8—Example showing the need to describe a pseudo static behavior on the data crossing domains .	32
10	Figure 9—Example of the DFT structure usage in a digital design	34
11	Figure 10—Example of a digital design explaining usage of the “constant” attribute.....	35
12	Figure 11—Feedthrough logic.....	36
13	Figure 12—Internal synchronizer and internal combinatorial logic on paths through ports.....	37
14	Figure 13—Module mod0 with abstracted input ports	46
15	Figure 14—Module mod0 with abstracted output ports	47
16	Figure 15—Clock definition A	48
17	Figure 16—Clock definition B	49
18	Figure 17—Clock definition C	50
19	Figure 18—One clock domain.....	52
20	Figure 19—Two clock domains	53
21	Figure 20—Three clock domains	54
22	Figure 21—Two clock domains	55
23	Figure 22—Failure mode due to missing synchronizer.....	56
24	Figure 23—Failure mode due to missing qualifier.....	57
25	Figure 24—Failure mode due to missing reset synchronizer	58
26	Figure 25—Failure mode due to combo logic driving clock/reset	59
27	Figure 26—External rstb with internal synchronization logic.....	61
28	Figure 27—rstb2b of IP with associated reset	62
29	Figure 28—External rdcq on the data signal of the IP	63
30	Figure 29—External rdcq on the clock signal of the IP	64
31	Figure 30—rstb of IP driven by multiple resets	65
32	Figure 31—rstb of IP driven by multiple resets. Destination flop has no async reset pin	67
33	Figure 32—rstb of IP is controlling a flop that is driven by multiple resets	68
34	Figure 33—rdcq is gating the interface clk	70
35	Figure 34—Internal rdcq	70
36	Figure 35—Schema for top-level elements	76
37	Figure 36—Schema for accellera:wire	77
38	Figure 37—Schema for accellera-cdc:data.....	78
39	Figure 38—Schema for accellera-cdc:asyncReset.....	79
40	Figure 39—Schema for accellera-cdc:cdcControl and accellera-cdc:rdcControl	80
41	Figure 40—Schema for accellera-cdc:componentCDCDef.....	81

1 List of tables

2	Table 1—CDC Attributes	19
3	Table 2—Parameter attributes	24
4	Table 3—Parameter usage	25
5	Table 4—Associated clocks usage for input ports.....	28
6	Table 5—Associated clocks usage for output ports.....	29
7	Table 6—Supported async_reset attributes	38
8	Table 7—Example cases.....	40
9	Table 8—Failure modes.....	42
10	Table 9—CDC Port Attributes	73

1

2 Clock Domain Crossing Standard

3 Version 0.3 Draft for Public Review

4 1. Overview

5 This common CDC interface standard provides the following:

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

11 A limited feasibility study was conducted on a subsystem with multiple IPs connected by Arm® AMBA®
12 interfaces across three vendor tools with limited support from the vendors.¹ It showed that 99.5% of what
13 was identifiable in a flat run (using any of the three vendor tools) was also identifiable if the native
14 abstraction collateral was replaced with an XML representation and translated across the vendor tools.²

15 1.1 Scope

16 The scope of work of the Clock Domain Crossing Standard is limited to:

- 17 a) Support tool-independent output collateral for CDC, RDC, and glitch structural analysis
- 18 b) Provide human-readable and machine-parsable attributes
- 19 c) Support customizable extensions (for example, to support complex user conditions)
- 20 d) Support hierarchical analysis
- 21 e) Support power-aware designs
- 22 f) Support multi-modal IP/SOC analysis
- 23 g) Support multi-instance IPs
- 24 h) Support parameterized IPs
- 25 i) Support multiple interface protocols (for example, AMBA, I2C, PCIe, UCIe, and so forth)
- 26 j) Provide extensibility to cover input collateral cases (for example, constraints, waivers, and so forth)
27 to enable high-quality re-verification of an IP using alternate tools
- 28 k) Support assertions necessary to complement CDC, RDC, and glitch structural analyses

¹AMBA® is a registered trademark of Arm® Limited (or its subsidiaries) in the US and/or elsewhere

²This feasibility study was only for CDC and did not include reset domain crossing (RDC) or glitch analysis.

1 I) Support other design styles (for example, FPGA and Analog) that follow similar standards

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

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

9 1.2 Purpose

10 The purpose of the Clock Domain Crossing standard is to:

- 11 a) Enable EDA vendors to develop CDC, RDC, and/or glitch analysis tools that meet the specification
12 defined in this standard to generate tool-agnostic collateral
- 13 b) Enable IP companies to use their tool(s) of choice to perform CDC, RDC, and/or glitch analysis on
14 their IP and generate said collateral
- 15 c) Enable SOC companies to consume collateral generated by different IP vendors from their tool(s) of
16 choice, into the SOC company’s own tool(s) of choice

17 1.3 Word usage

18 The word *shall* indicates mandatory requirements strictly to be followed in order to conform to the standard
19 and from which no deviation is permitted (*shall equals is required to*).^{3,4}

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

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

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

27 1.4 Contents of this standard

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

- 29 — [Clause 2](#) provides references to other applicable standards that are assumed or required for this stan-
30 dard.
- 31 — [Clause 3](#) defines terms and acronyms used throughout the different specifications contained in this
32 standard.
- 33 — [Clause 4](#) lists CDC attributes along with their type, accepted values, whether they are mandatory,
34 and clarifying comments.

³ The use of the word *must* is deprecated and cannot be used when stating mandatory requirements; *must* is used only to describe unavoidable situations.

⁴ The use of *will* is deprecated and cannot be used when stating mandatory requirements; *will* is only used in statements of fact.

- 1 — [Clause 5](#) lists RDC attributes and provides scenarios to describe support for an RDC interface at the
- 2 top level without resynthesizing the RTL of the IP.
- 3 — [Clause 6](#) describes the CDC format that is based on the TCL language for CDC specification from
- 4 an output collateral perspective.
- 5 — [Clause 7](#) describes the CDC format that is based on the IP-XACT standard.
- 6 — [Annex A](#) provides an informative bibliography.

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

IEEE Std 1800TM-2017, IEEE Standard for SystemVerilog Unified Hardware Design, Specification and Verification Language.^{5, 6}

9

⁵The IEEE standards or products referred to in this clause are trademarks of the Institute of Electrical and Electronics Engineers, Inc.

⁶IEEE publications are available from the Institute of Electrical and Electronics Engineers, Inc., 445 Hoes Lane, Piscataway, NJ 08854, USA (<https://standards.ieee.org/>).

1 3. Definitions, acronyms, and abbreviations

2 For the purposes of this document, the following terms and definitions apply. *The Authoritative Dictionary*
 3 of *IEEE Standards Terms* [B1]⁷ should be referenced for terms not defined in this clause.

4 3.1 Definitions

5 **Accellera Vendor extensions:** Extensions to the IEEE Standard for IP-XACT that capture attributes related
 6 to clock and reset to support the CDC Standard.

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

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

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

13 **CDC control signal:** A signal that is either tool-inferred or user-specified as a clock domain crossing con-
 14 trol to a data path (scalar or vector) as it crosses clock domains.

15 3.2 Acronyms and abbreviations

16 CDC	clock domain crossing
17 Combo	combinational logic
18 DFT	design for test
19 EDA	electronic design automation
20 HDL	hardware description language
21 inout	bidirectional port
22 internal_sync	internal synchronizer
23 IP	intellectual property
24 IP-XACT	IEEE standard describing an XML metadata schema for documenting IP; used
25	in the development, implementation, and verification of electronic systems
26 LRM	language reference manual
27 MTBF	mean time before failure
28 PWG	proposed working group
29 RDC	reset domain crossing

⁷The numbers in brackets correspond to those of the bibliography in [Annex A](#).

1	QoR	quality of results
2	RTL	register transfer level
3	SDC	Synopsys Design Constraints
4	SOC	system on chip or products that consist of various IPs and glue-logic
5	SVA	SystemVerilog Assertions
6	TCL	Tool Command Language
7	VIP	validation IP or test environment used to test aspects of an IP
8	WG	working group
9	XML	eXtensible Markup Language
10		NOTE—CDC WG refers to CDC-, RDC-, and glitch-related (asynchronous data crossing) checks
11		necessary for correct functionality of clock, reset, and data glitch (associated with clock and reset
12		crossings).
13		

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 (see [Clause 6](#) and [Clause 7](#)). In this version of the LRM, we have included pseudo code for the Tcl to support EDA vendors with tool development.

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 CDC and RDC 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. (Also see [Clause 6](#) for additional information about RDC attributes.) 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 attribute](#)
- b) [Parameter attributes](#)
- c) [Port attributes](#)
- d) [async_reset](#)
- e) [Modeling abstracted blocks](#)
- f) [Clock definitions](#)
- g) [Clock relationships](#)
- h) [Failure modes](#)

Table 1—CDC Attributes

Domain	Attribute	Type	Values	Mandatory	Comments
module	name	string	{module name}	Yes	See also: 4.1
parameter	name	string	{parameter name}	Yes	See also: 4.2
parameter	value	range-list	{values}	Optional	See also: 4.2
parameter	type	defined set	{string, Boolean, number (hex, decimal, oct, binary)}	Optional	See also: 4.2
parameter	ignore	Boolean	{true or false}	Optional	See also: 4.2
port	name	string	{port name}	Yes	See also: 4.3.1
port	direction	defined set	{input, output, inout}	Yes	See also: 4.3.1

Table 1—CDC Attributes (*continued*)

Domain	Attribute	Type	Values	Mandatory	Comments
port	type	defined set	{data, clock, virtual_clock, async_reset, cdc_control, rdc_control, virtual_reset}	Yes	Mandatory “Yes” applies to async reset only. For async resets, type is async_reset; for sync resets, type is data. Type virtual_clock defines a virtual clock port that does not match an actual port in the module. See also: 4.3.1 , 4.3.4 , 4.4 , and Clause 5
port	logic	defined set	{combo, buffer, inverter, glitch_free_combo, internal_sync}	Optional	Without this attribute, the assumption is that the port directly reaches a sequential element. See also: 4.3.10
port	cdc_control_from_clock	; separated list	{clock names}	Optional	Required for cdc_control and data with cdc_control, async is implied when data does not have -associated_clock Only supports one clock See the definition in the next row. See also: Figure 14
<p>Definition: Associates a control to a clock domain where the data is coming from.</p> <p>Example:</p> <pre>port -name sync_valid -type cdc_control -associated_to_clocks rx_clock -cdc_control_from_clock VCLK_bus</pre>					
port	associated_from_clocks	; separated list	{clock-names}	Yes	See also: 4.3.2 and 4.5
port	associated_to_clocks	; separated list	{clock-names}	Optional	For default values, see 4.3.3 . See also 4.5 and 5.1.4

Table 1—CDC Attributes (*continued*)

Domain	Attribute	Type	Values	Mandatory	Comments
port	associated_inputs	; separated list	{ports}	Optional	See the definition in the next row. See also: 4.5
Definition: Used for feedthrough configuration where one or more inputs reach one or more outputs; applicable to outputs. Used also for the CDC control signal configuration to associate a cdc_control to a vector signal; applicable to inputs if considering a CDC control signal input.					
port	associated_outputs	; separated list	{ports}	Optional	See the definition in the next row. See also: 4.3.9
Definition: Used for a feedthrough configuration where one or more outputs trace from one or more inputs; applicable to inputs. The cdc_control signal configuration might be useful.					
port	cdc_control	; separated list	{associated-ports}	Optional	See also: 4.5
port	polarity	defined set	{high, low, low_high}	Yes	“Yes” only applicable to async reset
port	ignore	defined set	{blocked, hanging}	Optional	See also: 4.3.6
port	cdc_static	; separated list	{clock-names}	Optional	See also: 4.3.7
port	constant	; separated list	{binary, hex, and of any length}	Optional	See the definition in the next row. See also: 4.3.8
Definition: Sets a constant value on a net, pin, or port for the current analysis. From the CDC model perspective, it is applicable to a port only. It is relevant for all signal types (data, control, etc.).					
port	gray_coded	Boolean	{true, false:default}	Optional	See the definition in the next row. Deferred to LRM v0.5
Definition: Implies mutually exclusiveness signals that are part of a bus corresponding to Gray coding (only 1 bit changing sequentially). gray_coded ports are used to eliminate convergence and glitch violations inside an IP. If a user has specified these on input ports, then an assertion is generated to check for non-gray coded transitions. There isn't any validation other than assertions that can be done on this constraint.					
port	clock_period	string	{clock period}	Optional	Deferred to LRM v0.5
clock_period is used to identify relative clock periods for recognizing a fast to slow CDC and where an assertion could be created for detecting if data loss could occur in the data transfer. Validation could be done to compare the clock_periods between the top and an abstract model.					

Table 1—CDC Attributes (*continued*)

Domain	Attribute	Type	Values	Mandatory	Comments
port	associated_from_reset	; separated list	{reset-names}	Optional	Defines the driver reset of a port See also: 4.3.5
port	associated_to_reset	; separated list	{reset-names}	Optional	Defines the receiver reset of a port See also: 4.3.5
port	rdc_control_from_reset	; separated list	{reset-names}	Optional	Used with <code>rdc_control</code> . Defines the source reset of the RDC; i.e., the start point's reset. This is the source reset being blocked by <code>-rdc_control</code> . See also: Clause 5
port	rdc_control_to_reset	; separated list	{reset-names}	Optional	Used with <code>rdc_control</code> . Defines the destination reset of the RDC; i.e., the end point's reset. See also: Clause 5
port	rdc_control_to_clock	; separated list	{clock-names}	Optional	Used with <code>rdc_control</code> . Defines the destination clock of the RDC end point's clock. This is the destination clock being gated by <code>-rdc_control</code> . See also: Clause 5
port	rdc_clock_gate_location	defined set	{external or internal}	Optional	Used with <code>rdc_control</code> . Defines the location of a clock gate; i.e., internal versus external. See also: Clause 5

Table 1—CDC Attributes (continued)

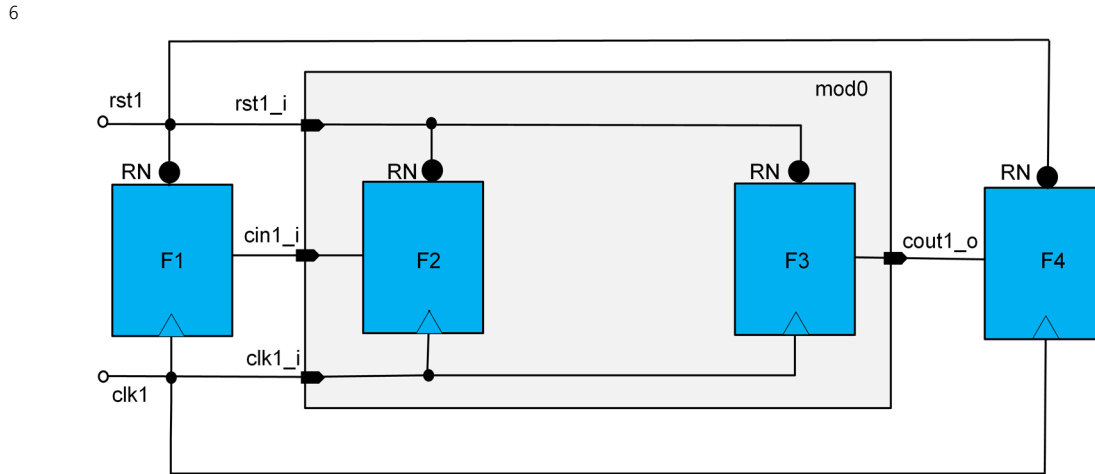
Domain	Attribute	Type	Values	Mandatory	Comments
tool	name	string	{tool name}	Yes	If no EDA tool name is captured, it implies hand crafted by user
tool	version	string	{tool version}	Yes	N/A if user generated
design	version	string	{design milestone}	Optional	
design	date	string	{collateral generation date}	Yes	
design	username	string	{user/tool that generated the collateral}	Optional	
design	description	string	{description}	Optional	
set_cdc_clock_group					See the definition in the next row. See also 4.7
<p>Definition: A set of clocks that are synchronous to each other in a design.</p> <p>Example:</p> <pre>set_cdc_clock_group -name small_domain -clocks {ck} set_cdc_clock_group -name large_domain -clocks {gck0;gck1}</pre>					
	clocks	; separated list	{clock-names}	Yes	
	name	string	{group-name}	Optional	
set_reset_group					See the definition in the next row. See also 5.2.3
<p>Defines the reset relation. There are no RDC violations between resets that appear together in the same reset group.</p> <p>Example:</p> <pre>set_reset_group -name reset_domain_1 -reset {virtual_reset_a; rstb}</pre>					

1

1 4.1 Module attribute

2 The attributes include one module attribute. Refer to the following pseudo code and [Figure 1](#), which depict
3 the mod0 module.

```
4
5 module -name mod0
```



7 **Figure 1—Module attribute**

8 4.2 Parameter attributes

9 The `parameter` command is used to define parameters within the scope of the block, e.g.,

```
10
11 parameter PARAM1 -type int -value 0;
```

12 [Table 2](#) lists the attributes that will be used with the `parameter` command.

13

Table 2—Parameter attributes

Attribute	Values	Definition
-name	String	Defines the name of the parameter. Is Mandatory.
-type	Integer/string/Boolean	Specifies type of the value stored by the parameter
-value	Range-list	Specifies the value of the parameter
-ignore	N/A	Tells that the parameter should be ignored. Any parameter that is ignored but still referred to in the model in following commands is a failure.

14 [Table 3](#) contains usage examples for the `parameter` command with the `port` command.

1

Table 3—Parameter usage

Number	Scenario	Example command
1	Sample integer type parameter with value 32	<code>parameter -name PARAM1 -type int -value 32</code>
2	Sample string type parameter with value DEFAULT_CASE	<code>parameter -name CASE_VAR -type string -value DEFAULT_CASE</code>
3	Sample Boolean type parameter with value “false”	<code>parameter -name SELECT -type boolean -value false</code> or <code>parameter -name SELECT -type boolean -value 0</code>
4	A bus named DATA 7 down to 0 defined using the parameter for MSB and LSB.	<code>parameter -name MSB -type int -value 7</code> <code>parameter -name LSB -type int -value 0</code> <code>port -name DATA[7:0] -type data</code> or <code>port -name DATA[MSB:0] -type data</code> or <code>port -name DATA[7:LSB] -type data</code> or <code>port -name DATA[MSB:LSB] -type data</code>
5	Individual bits and slice of a bus named DATA 7 down to 0 defined using the parameter for MSB and LSB. Also using expressions with parameters.	<code>parameter -name MSB -type int -value 7</code> <code>parameter -name LSB -type int -value 0</code> Only the 8th bit <code>port -name DATA[MSB] -type data</code> Only the first four MSB bits <code>port -name DATA[MSB:MSB-3] -type data</code> Only the last 2 LSB bits <code>port -name DATA[LSB+1:LSB] -type data</code> 3-bit slice from the middle of the bus <code>port -name DATA[MSB-2:LSB+3] -type data</code>
6	Example of usage of Boolean type parameter for a port P1 to be tied to constant value 1	<code>parameter -name SEL_VAL -type boolean -value true</code> <code>cdc_constant -port P1 -value SEL_VAL</code>

2

3

1 4.3 Port attributes

2 Several port attributes are detailed in this section.

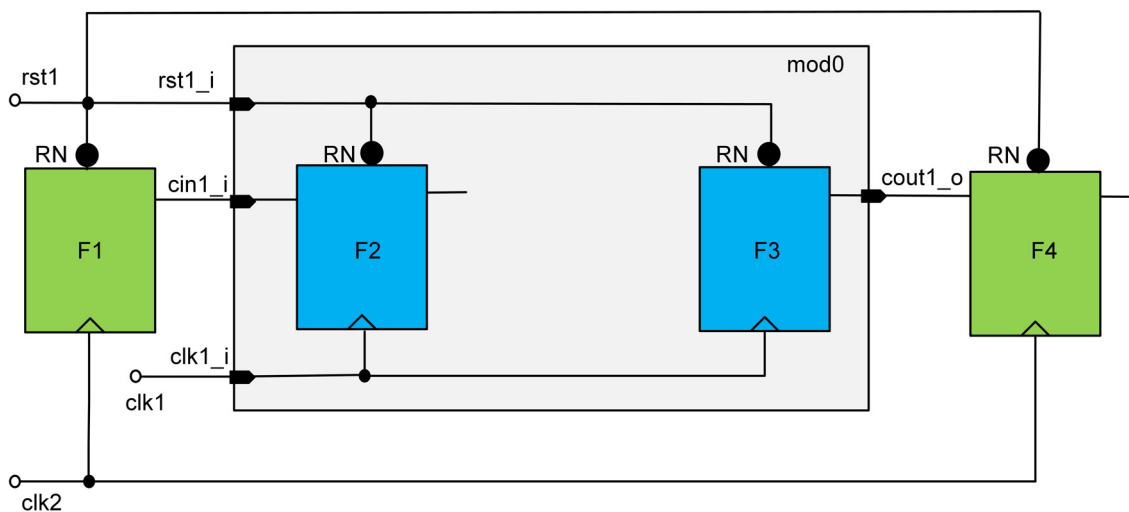
3 4.3.1 Port attributes: name, direction, and type

4 Refer to the following pseudo code and [Figure 2](#), which depict the name, direction, and type
 5 attributes. Also see [4.4](#) for additional information about the type attribute and the following sections about
 6 the clock attributes.

```

    7
    8     port -name cin1_i -direction input -type data -associated_from_clocks
    9         virtual_clk -associated_to_clocks clk1_i
    
```

10



11

Figure 2—Port attributes: name, direction, type

12 For clock attributes see [4.3.2](#), [4.3.3](#), [4.3.4](#).

14.3.2 Port attribute: associated_from_clocks

2 The following pseudo code and [Figure 3](#) depict the mandatory associated_from_clocks attribute.

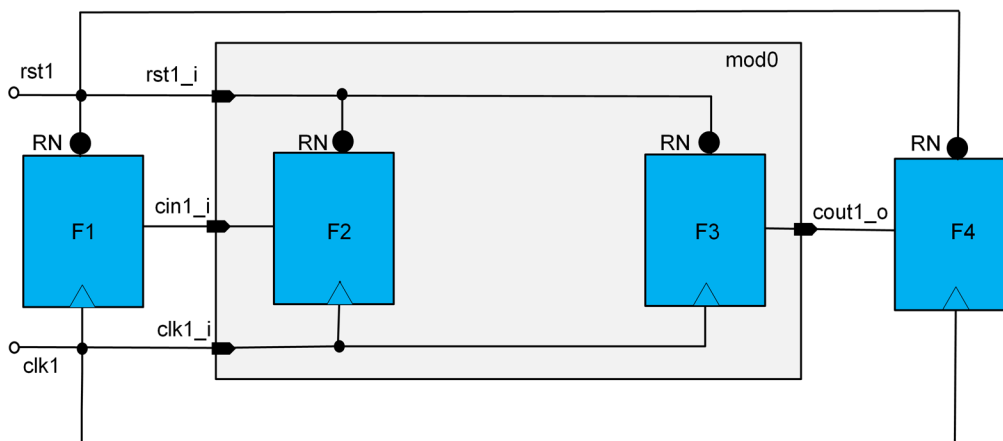
3 For an input port, the associated_from_clocks attribute is part of the external view describing the
4 driver of the input. For an output port, it is part of the internal view describing the driver of the output.

5

```
6 port -name cin1_i -direction input -type data -associated_from_clocks clk1_i
7 port -name cout1_i -direction output -type data -associated_from_clocks clk1_i
```

8 For ports being associated to a virtual clock refer to [4.3.4](#).

9



10 **Figure 3—Port attribute: associated_from_clocks**

11

1 4.3.3 Port attributes: associated_to_clocks

2 The attribute associated_to_clocks is optional.

3 If it is not used:

4 — The default assumption for input ports is: associated_from_clocks and
 5 associated_to_clocks are synchronous to each other; i.e., there is no CDC on the path
 6 through the port.

7 — There is no default assumption for output ports.

8 [Table 4](#) provides a description of the usage of associated_to_clocks and
 9 associated_from_clocks for input ports.

10

Table 4—Associated clocks usage for input ports

For input port	-associated_to_clocks defined e.g., -associated_to_clocks clk2	-associated_to_clocks Not defined
associated_from_clocks defined e.g., -associated_from_clocks clk1	Driver should come from clock clk1 and only be used in clk2; otherwise, it is a failure. If clk1 and clk2 are asynchro- nous to each other, an appropri- ate synchronization scheme is expected in the receiving circuit to avoid a CDC violation.	Driver should come from clock clk1 and only be used by a synchronous receiver; otherwise, it is a CDC viola- tion .

1 [Table 5](#) provides a description of the usage of `associated_to_clocks` and
 2 `associated_from_clocks` for output ports.

3

4

Table 5—Associated clocks usage for output ports

For output port	-associated_to_clocks defined e.g., -associated_to_clocks clk2	-associated_to_clocks Not defined
associated_from_clocks defined e.g., -associated_from_clocks clk1	If clk1 and clk2 are asynchronous to each other, an appropriate synchronization scheme is expected in the receiving circuit to avoid a CDC violation.	The receiving clock is found in the actual encompassing design. If it is asynchronous to clk1, an appropriate synchronization scheme is expected in the receiving circuit to avoid a CDC violation.

5 4.3.4 Defining a virtual clock

6 In the CDC LRM 0.3, virtual clocks must be explicitly declared by defining a port with the type
 7 `virtual_clock`. This is a virtual port that does not match an actual port in the module. For example,
 8 `vclk` used in [Figure 4](#) to define the `-associated_from_clocks` of port `in1_i` must be declared
 9 with type `virtual_clock` as shown below.

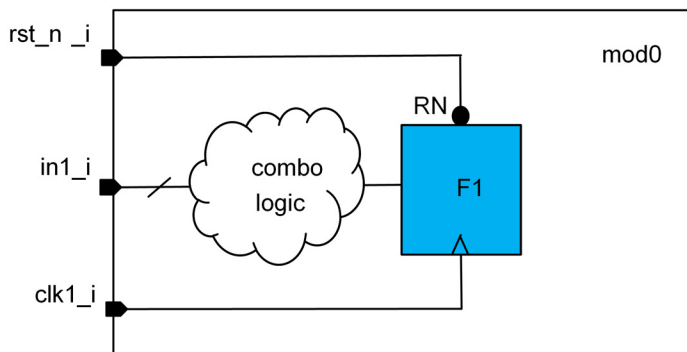
10

```

11 port -name clk1_i -type clock -direction input
12 port -name vclk -type virtual_clock -direction input
13 port -name in1_i -type data -direction input -associated_from_clocks vclk
14 port -name in1_i -type data -direction input -associated_to_clocks clk1_i
15     -logic combo
16 port -name rst_n_i -type async_reset direction input -associated_from_clocks
17     clk1_i
    
```

18

19



20

Figure 4—Specifying a virtual clock

1 4.3.5 Port attribute: `associated_from_reset` and `associated_to_reset`

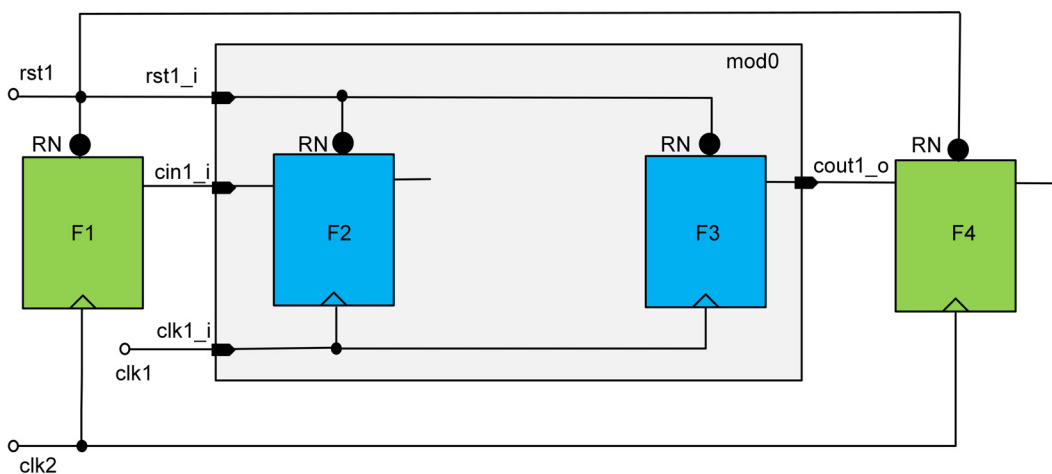
2 The following pseudo code and [Figure 5](#) depict the optional `associated_from_reset` and
 3 `associated_to_reset` attributes.

```

    4
    5 port -name cin1_i -direction input -type data -associated_from_clocks vclk
    6     -associated_to_reset rst1_i
    7 port -name cout1_o -direction output -type data -associated_from_clocks clk1_i
    8     -associated_from_reset rst1_i
    9
    
```

10 The list of `associated_from_clocks` contains a virtual clock `vclk` (see [4.3.4](#)) because this clock is
 11 not visible at the interface of the abstracted block.

12



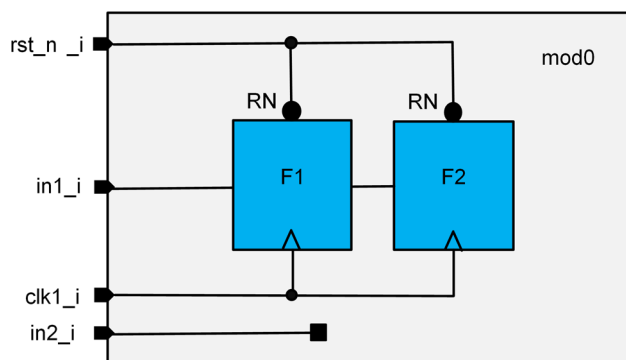
13

Figure 5—Port attribute: `associated_from_reset` and `associated_to_reset`

14.3.6 Port attribute: ignore

2 The `-ignore` attribute is used to indicate an interface port is not being analyzed for a clock domain
 3 crossing that is hanging or blocked. For example, the hanging value is used when an input port is
 4 dangling as shown in [Figure 6](#).

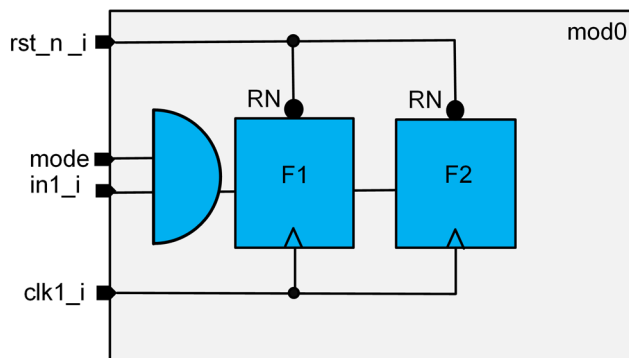
```
5
6 port -name in2_i -ignore hanging
```



8 **Figure 6—Ignore: hanging**

9 The `blocked` value is used when the propagation of an interface port is blocked as shown in [Figure 7](#).
 10 `in1_i` in [Figure 7](#) is blocked due to a constant constraint at port `-name mode`. RTL tie-off can also
 11 cause an interface port to be blocked, and therefore, ignored. The `-ignore` attribute is generated by the
 12 EDA tool during CDC analysis. Both the RTL and input constraints are used by the EDA tool to determine
 13 when an interface port is not being considered for CDC analysis.

```
14
15 port -name mode -constant 0
16 port -name in1_i -ignore blocked
```



18 **Figure 7—Ignore: blocked**

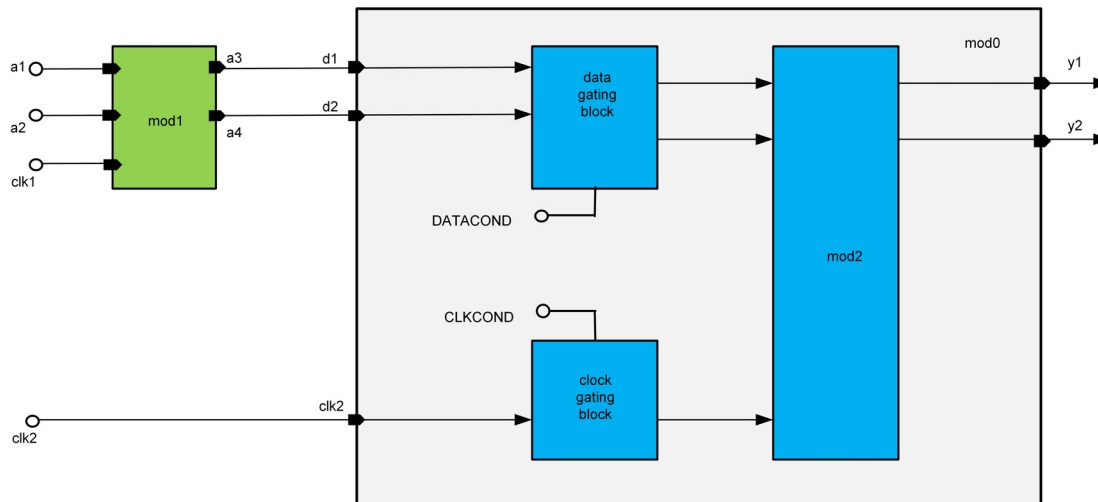
1 4.3.7 Port attribute: `cdc_static`

2 This section discusses the need and usage for the `cdc_static` attribute.

3 4.3.7.1 Need for the “`cdc_static`” attribute

4 In some cases, when the data is crossing the domain boundaries, we may not always want to add synchronization
 5 schemes in between to ensure less area overhead. This requires some pseudo static behavior on the data that is crossing
 6 the domain boundary to avoid any metastable behavior. For example, in [Figure 8](#), module `mod1` only works on `clk1`
 7 and `mod2` only works on `clk2`. `mod0` has a clock gating block on the clock path driving `mod2` and a data gating block
 8 on the data path for the incoming port `d1`, `d2`. If it is guaranteed by the design that `mod0`'s `clk2` shall always be gated
 9 before the driver for port `d1`, and port `d2` changes (i.e., port `a3` and port `a4` of `mod1`), the `cdc_static` attribute can
 10 be applied to port `d1` and port `d2` during `mod0`'s CDC analysis. Hence, CDC analysis shall not be done on these
 11 primary interface ports of `mod0`. It is important to have all `cdc_static` attributes validated in the assertion flow to
 12 ensure the functionality of the design. `mod0`'s internal nets `DATACOND` and `CLKCOND` illustrated in [Figure 8](#) are not
 13 currently supported in LRM0.3 because the current LRM focuses on capturing the interface information. However,
 14 `mod0`'s internal nets `DATACOND` and `CLKCOND` might be used in a future LRM, where a regular expression would be
 15 added to describe the `cdc_static`'s condition observed on port `d1` and port `d2`.

16



17 **Figure 8—Example showing the need to describe a pseudo static behavior on the data**
 18 **crossing domains**

19

20 4.3.7.2 Usage for the “`cdc_static`” attribute

21 The `cdc_static` attribute associates a data port with a clock or multiple clocks to indicate when the data is
 22 changing. It is expected that the clock or clocks will be gated; hence, no CDC verification is required.

23

24 Generic Usage:

25

```
26 port -name <name of the port> -direction <input|output>
```

```
27     -cdc_static <clock or list of clocks that will be gated>
```

28 Usage based on `mod0` of [Figure 8](#):

29


```
1 port -name d1 -direction input -cdc_static clk2  
2 port -name d2 -direction input -cdc_static clk2
```

3 Note: In contrast to a constant signal (see [4.3.8](#)) the value of a static signal is not known.

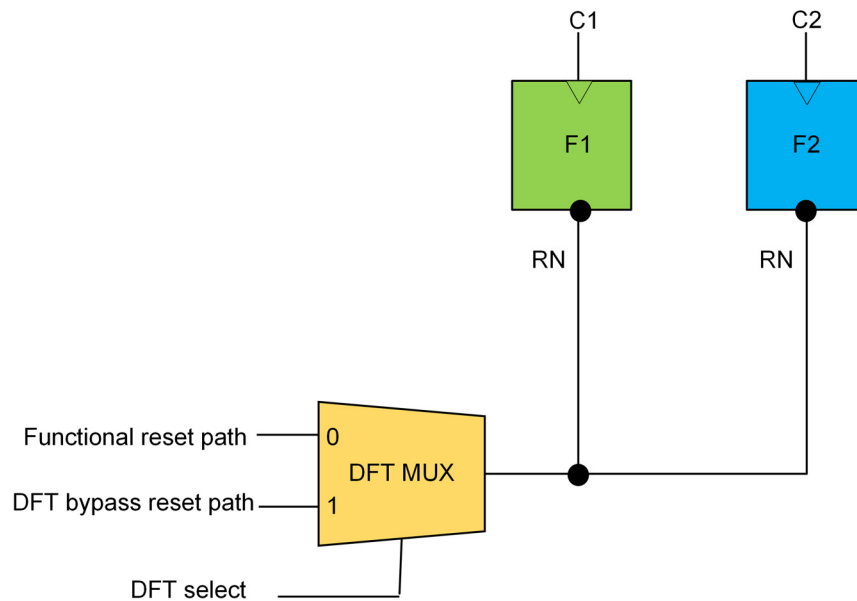
1 4.3.8 Port attribute: constant

2 This section discusses the need and usage for the `constant` attribute.

3 4.3.8.1 Need for the “constant” attribute

4 [Figure 9](#) below shows two different flops in two asynchronous clock domains. When the design for test
 5 (DFT) select is 0, then the functional reset path is activated, which clearly indicates a domain crossing
 6 violation on one of the reset paths because the same reset signal is consumed in two asynchronous clock
 7 domains. While in the case of a DFT select set to 1, the DFT flow shall ensure that functionally, there are no
 8 domain crossings on clock or reset paths by making sure that the clock and reset paths are separately
 9 controllable. Based on this, we can safely mark the DFT bypass reset path and the DFT select path as
 10 constant so that they are ignored in the CDC analysis.

11



12

Figure 9—Example of the DFT structure usage in a digital design

13

14 4.3.8.2 Usage for the “constant” attribute

15 [Figure 10](#) shows the design structure with associated ports. `clk1`, `clk2`, and `rstn` are functional ports
 16 while `dft_bypass_rstn` and `dft_bypass_rstn_select` are the DFT-specific ports. For the
 17 functional CDC analysis, these ports (`dft_bypass_rstn` and `dft_bypass_rstn_select`) can be
 18 declared as constant using the `constant` attribute.

19 Generic usage:

20

```
21 port -name <name of the port> -direction <input/output>
22     -constant <constant value>
```

23 Example:

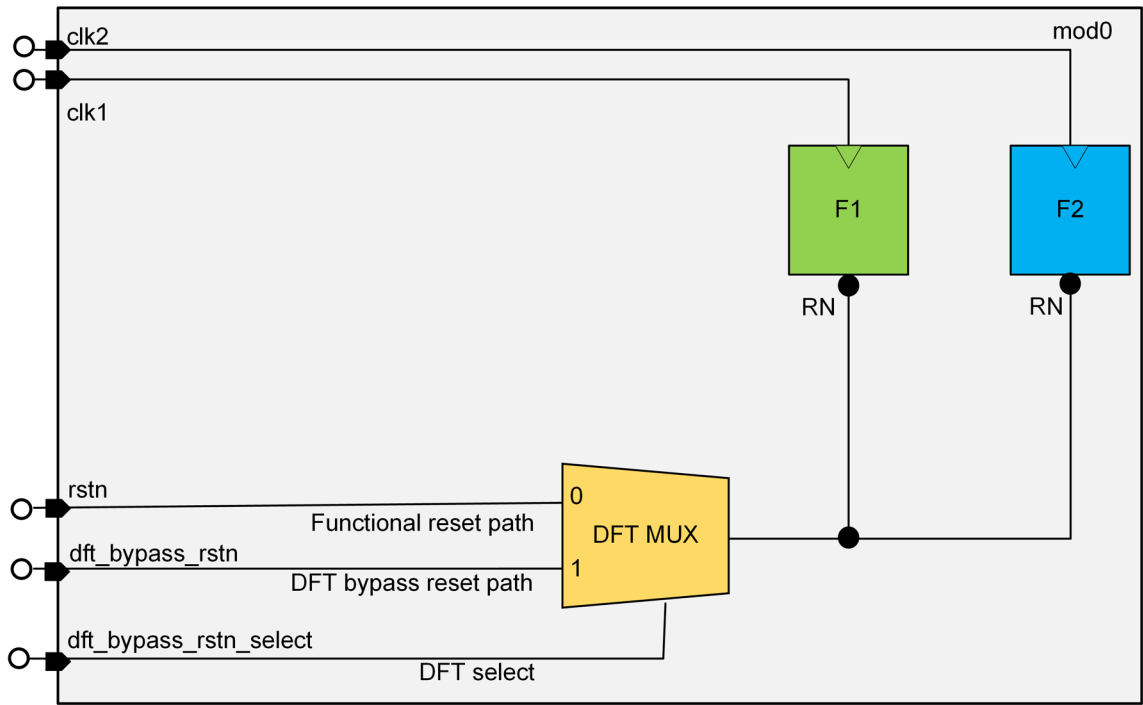
24

```

1 port -name dft_bypass_rstn -direction input -constant 0
2 port -name dft_bypass_rstn_select -direction input -constant 0

```

3
4



5 **Figure 10—Example of a digital design explaining usage of the “constant” attribute**

6 While using the constant attribute, only the name and direction of the ports are mandatory attributes. Other
7 attributes like type, associated_from_clocks, etc. are not required to be defined.

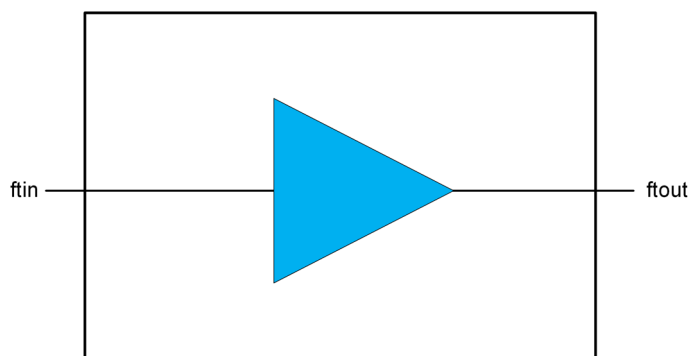
8

1 4.3.9 Port attribute: associated_outputs

2 [Figure 11](#) depicts feedthrough logic. The associated pseudo code includes the optional attribute
3 associated_outputs for the feedthrough output.

```
4
5 port -name ftin -direction input -type data -associated_from_clocks vclk
6     -associated_outputs ftout
```

7



8

Figure 11—Feedthrough logic

9

10 4.3.10 Port attribute: logic

11 The port attribute logic describes the internal elements in the fan-out of an input port or in the fan-in of an
12 output port.

13 The attribute value internal_sync indicates a multi-flop synchronizer on the path through the port.

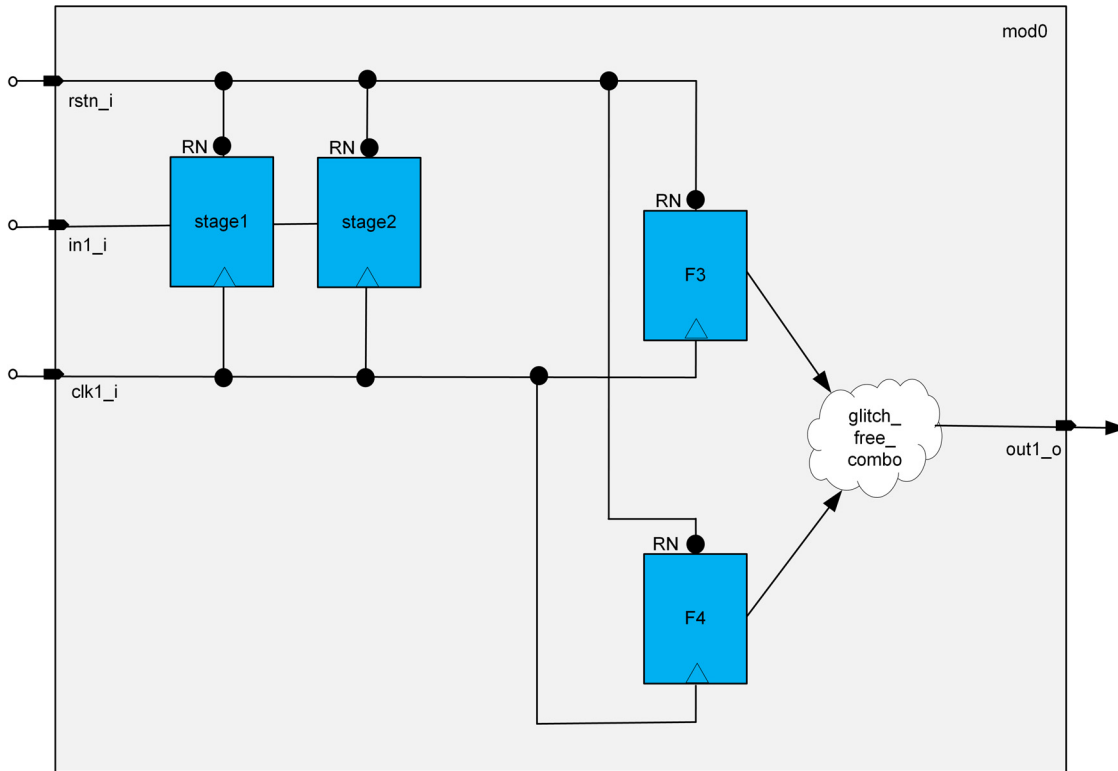
14 The attribute value glitch_free_combo indicates that the path through the port contains combinatorial
15 logic that does not generate glitches in any of the valid operation modes or at reset assertion.

16

```
17 port -name in1_i -direction input -type data -associated_from_clocks vclk
18     -associated_to_reset rstn_i -logic internal_sync
19 port -name out1_o -direction output -type data -associated_from_clocks clk1_i
20     -associated_from_reset rstn_i -logic glitch_free_combo
```

21

1



2 **Figure 12—Internal synchronizer and internal combinational logic on paths through ports**

3

1 4.4 async_reset

2 To define an asynchronous reset, port should include the `-type` attribute with the value `async_reset`.

3 All synchronous resets will be considered data ports.

4 If the port command's `-type` attribute has the value `async_reset`, the attributes in [Table 6](#) will be
5 applicable and the rest of the attributes of the port command can be ignored.

6 Sample command:

```
7 port -name <port_name> -type async_reset -polarity low -direction input
8     -associated_from_clocks TXCLK -associated_to_clocks RX_CLK -logic inverter
9     -ignore
```

10

Table 6—Supported `async_reset` attributes

Attribute	Values	Definition
<code>-name</code>	<code><port name></code>	Defines the name of a physical port when used with <code>-type async_reset</code> . This attribute is mandatory for <code>-type async_reset</code> .
<code>-polarity</code>	<code>low/high/low_high</code>	<code>-polarity low</code> signifies <code>active_low</code> asynchronous reset <code>-polarity high</code> signifies <code>active_high</code> asynchronous reset <code>-polarity low_high</code> signifies the reset is used by some flops as <code>active_low</code> and by some flops as <code>active_high</code> This attribute is mandatory for <code>async_reset</code> .
<code>-direction</code>	<code>input/output/inout</code>	<code>input</code> signifies that the port is an input port. <code>output</code> signifies that the port is an output port. <code>inout</code> signifies that the port is an inout port. This attribute is mandatory.
<code>-associated_from_clocks</code>	<code><clock_name></code>	Specifies the driver clock of a reset signal. This attribute is mandatory. If <code>associated_to_clocks</code> is not specified, reset will be considered deasserting with regard to the specified from clock. Without logic <code>internal_sync</code> in the port attribute, if both <code>associated_from_clocks</code> and <code>associated_to_clocks</code> are specified, the clock domain for both clocks should match.

Table 6—Supported `async_reset` attributes

Attribute	Values	Definition
<code>-associated_to_clocks</code>		
	<code><clock_name></code>	<p>Specifies the receiver clock.</p> <p>This attribute is optional.</p> <p>If <code>associated_to_clocks</code> is specified, reset will be considered deasserting with regard to the specified clock.</p> <p>Without <code>logic internal_sync</code> in the port attribute, if both <code>associated_from_clocks</code> and <code>associated_to_clocks</code> are specified, the clock domain for both clocks should match.</p>
<code>-logic</code>	<code>{combo, buffer, inverter, glitch_free_combo, internal_sync}</code>	<p><code>combo</code>: Signifies that there is combo logic just inside the port.</p> <p><code>inverter</code>: Signifies the signal is inverted, and accordingly, polarity will be considered inverted.</p> <p><code>glitch_free_combo</code>: Is irrelevant for <code>async_reset</code> ports.</p> <p><code>internal_sync</code>: For an <code>async_reset</code> type of input port, this signifies that the port is feeding a reset synchronizer.</p> <p>For an <code>async_reset</code> type of output port, <code>internal_sync</code> signifies that the signal is generated by a reset synchronizer.</p> <p>This attribute is optional. The attribute shall reflect what is present in the design:</p> <ul style="list-style-type: none"> — <code>combo</code>: the default is <i>combo not present</i> — <code>inverter</code>: the default is <i>inverter not present</i> — <code>glitch_free_combo</code>: irrelevant for <code>async_reset</code> ports — <code>internal_sync</code>: the default is <i>internal sync not present</i>
<code>-ignore</code>	N/A	<p>No checks are required for this port</p> <p>This attribute is optional.</p> <p>For high quality of CDC/RDC checks, <code>-type async_reset</code> port should not have <code>-ignore</code> set.</p>

1

2 [Table 7](#) provides example cases.

3

4

Table 7—Example cases

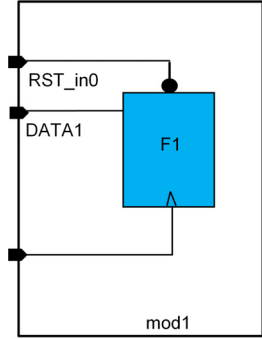
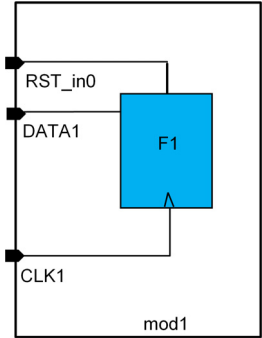
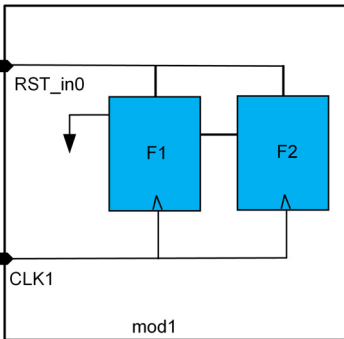
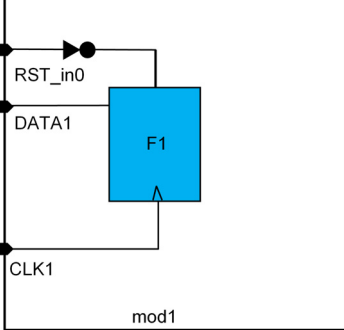
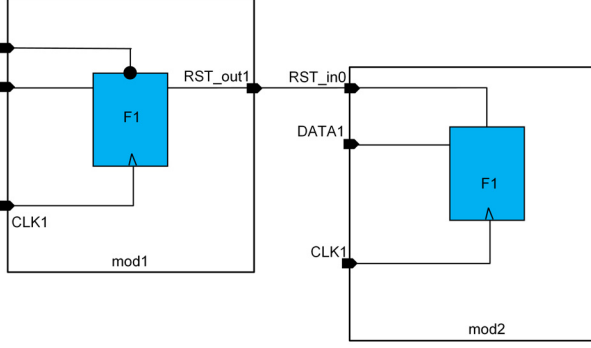
	Example	Diagram
1	<p>active_low reset on an input port with a virtual associated_from_clocks. This means that the signal driving RST_in0 is asynchronous to CLK1. With this modeling, the IP will have an internal reset deassertion violation. If the signal driving RST_in0 is from CLK1, the associated_from_clocks should be CLK1.</p> <p>Sample command:</p> <pre> module -name mod1 port -name RST_in0 -type async_reset -direction input -polarity low -associated_from_clocks VCLK1 (mandatory) port -name RST_in0 -associated_to_clocks CLK1 (optional) </pre>	
2	<p>active_high reset on an input port with a virtual associated_from_clocks. This means that the signal driving RST_in0 is asynchronous to CLK1. With this modeling, the IP will have an internal reset deassertion violation. If the signal driving RST_in0 is from CLK1, the associated_from_clocks should be CLK1.</p> <p>Sample command:</p> <pre> module -name mod1 port -name RST_in0 -type async_reset -direction input -polarity high -associated_from_clocks VCLK1 (mandatory) port -name RST_in0 -associated_to_clocks CLK1 (optional) </pre>	

Table 7—Example cases

	Example	Diagram
3	<p>active_high reset on an input port feeding a reset synchronizer</p> <p>Sample command:</p> <pre> module -name mod1 port -name RST_in0 -type async_reset -direction input -polarity high -associated_from_clocks VCLK1 -logic internal_sync (mandatory) port -name RST_in0 -associated_to_clocks CLK1 (optional) </pre>	
4	<p>active_high on an input port and feeding an inverter</p> <p>Sample command:</p> <pre> module -name mod1 port -name RST_in0 -type async_reset -direction input -polarity high -associated_from_clocks CLK1 -logic inverter (mandatory) port -name RST_in0 -associated_to_clocks CLK1 </pre>	
5	<p>active_high on an input port with associated_to_clocks and known associated_from_clocks</p> <p>Sample command:</p> <pre> module -name mod2 port -name RST_in0 -type async_reset -direction input -polarity high -associated_from_clocks CLK1 (mandatory) port -name RST_in0 -associated_to_clocks CLK1 (optional) </pre>	

1

2 [Table 8](#) illustrates failure modes.

3

Table 8—Failure modes

	Description	Diagram
1	<p>Reset driven by wrong clock domain</p> <p>Example: Clock mismatch. Assuming CLK1 and CLK2 to be async</p> <pre> module -name mod1 port -name RST_out1 -type async_reset -direction out -polarity low -associated_from_clocks CLK1 (mandatory) module -name mod2 port -name RST_in1 -type async_reset -direction in -polarity low -associated_from_clocks CLK2 (mandatory) </pre> <p>Expectation: An unsynchronized crossing on port RST_in1 should be reported during top CDC analysis for clock domain mismatch.</p>	
2	<p>Reset driven by correct clock domain but incorrect polarity</p> <p>Example: Simple incorrect polarity</p> <pre> module -name mod1 port -name RST_out1 -type async_reset -direction out -polarity low -associated_from_clocks CLK1 (mandatory) module -name mod2 port -name RST_in1 -type async_reset -direction in -polarity high -associated_from_clocks CLK1 (mandatory) port -name RST_in1 -associated_to_clocks CLK1 (optional) </pre> <p>Expectation: A conflict for polarity should be reported on port RST_in1 during top CDC analysis.</p>	

Table 8—Failure modes

	Description	Diagram
3	<p>Multiple ports</p> <p>Example 1: Multiple input ports in the same clock domain but with different polarity driven by the same reset</p> <pre> module -name mod1 port -name RST_out1 -type async_reset -direction out -polarity low -associated_from_clocks CLK1 (mandatory) module -name mod2 port -name RST_in1 -type async_reset -direction in -polarity low -associated_from_clocks CLK1 (mandatory) port -name RST_in1 -associated_to_clocks CLK1 (optional) port -name RST_in2 -type async_reset -direction in -polarity high -associated_from_clocks CLK1 (mandatory) port -name RST_in2 -associated_to_clocks CLK1 (optional) </pre> <p>Expectation: A conflict for polarity should be reported for the second port RST_in2 and not for RST_in1.</p>	<p>The diagram illustrates two modules, mod1 and mod2, connected. Module mod1 contains a block F1 with two inputs: DATA1 and CLK1, and one output: RST_out1. Module mod2 contains two blocks, F2 and F3, both with an input: CLK1. The output RST_out1 from mod1 is connected to the RST_in1 input of block F2 and the RST_in2 input of block F3. The RST_in1 input of F2 is marked with a low polarity symbol (an inverted triangle), while the RST_in2 input of F3 is marked with a high polarity symbol (a triangle). This setup creates a polarity conflict for the RST_in2 port in mod2, which is not reported because it is not associated to the clock CLK1.</p>

Table 8—Failure modes

	Description	Diagram
	<p>Example 2: Multiple input ports in the same clock domain with the same polarity, but one is inverted and driven by the same reset</p> <pre> module -name mod1 port -name RST_out1 -type async_reset -direction out -polarity low -associated_from_clocks CLK1 (mandatory) module -name mod2 port -name RST_in1 -type async_reset -direction in -polarity low -associated_from_clocks CLK1 (mandatory) port -name RST_in1 -associated_to_clocks CLK1 (optional) port -name RST_in2 -type async_reset -direction input -polarity high -associated_from_clocks CLK1 -logic inverter (mandatory) port -name RST_in2 -associated_to_clocks CLK1 </pre> <p>Expectation: A conflict for polarity should be reported for the second port RST_in2 and not for RST_in1.</p>	
4	<p>Non-reset signal feeding into reset signal</p> <p>Example:</p> <pre> module -name mod1 port -name DATA_out1 -type data -direction out -associated_from_clocks CLK1 (mandatory) module -name mod2 port -name RST_in1 -type async_reset -direction input -polarity low -associated_from_clocks CLK1 (mandatory) port -name RST_in1 -associated_to_clocks CLK1 </pre> <p>Expectation: A conflict for the signal type should be reported for port RST_in1. This might be expected, but it must be reviewed during top CDC analysis.</p>	

1 4.5 Modeling abstracted blocks

2 The model of the input interface of an abstracted block can be described

3 — in terms of requirements to the driving circuit in the fan-in of the block (external model) or

4 — in terms of the relevant structures inside the block (internal model).

5 The external model is a mandatory part of the abstraction. The additional attributes of the internal model are
6 an optional part. The following examples describe the circuit in [Figure 13](#). For clarity, both flavors of the
7 model appear in separate lines. More than one line per port is allowed.

8 External:

```
9     port -name d1_i -type data -direction input -associated_from_clocks clk2_i
```

10 Internal:

```
11     port -name d1_i -associated_to_clocks clk1_i -logic combo -cdc_control q1_i
```

12 External:

```
13     port -name q1_i -type cdc_control -direction input -cdc_control_from_clock  
14         clk2_i -associated_to_clocks clk1_i -associated_inputs d1_i
```

15 Internal:

```
16     port -name q1_i -logic combo
```

17 External:

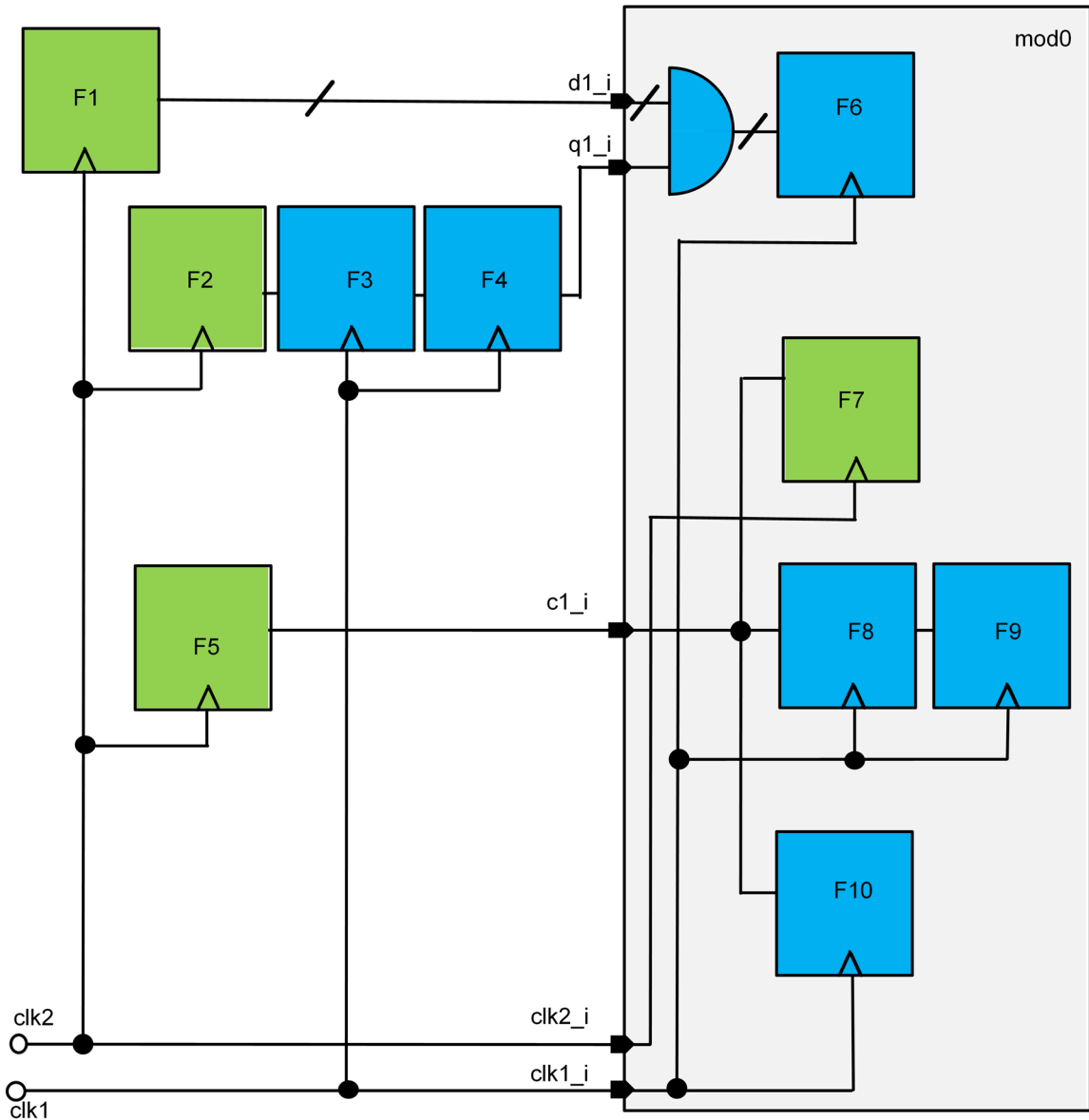
```
18     port -name c1_i -type data -direction input -associated_from_clocks clk2_i
```

19 Internal:

```
20     port -name c1_i -associated_to_clocks clk2_i  
21     port -name c1_i -associated_to_clocks clk1_i -logic internal_sync  
22     port -name c1_i -associated_to_clocks clk1_i
```

23 In an environment with asynchronous clocks being connected to `clk1_i` and `clk2_i`, the structure in the
24 last line should be reported as violated because of missing synchronization.

1



2

Figure 13—Module mod0 with abstracted input ports

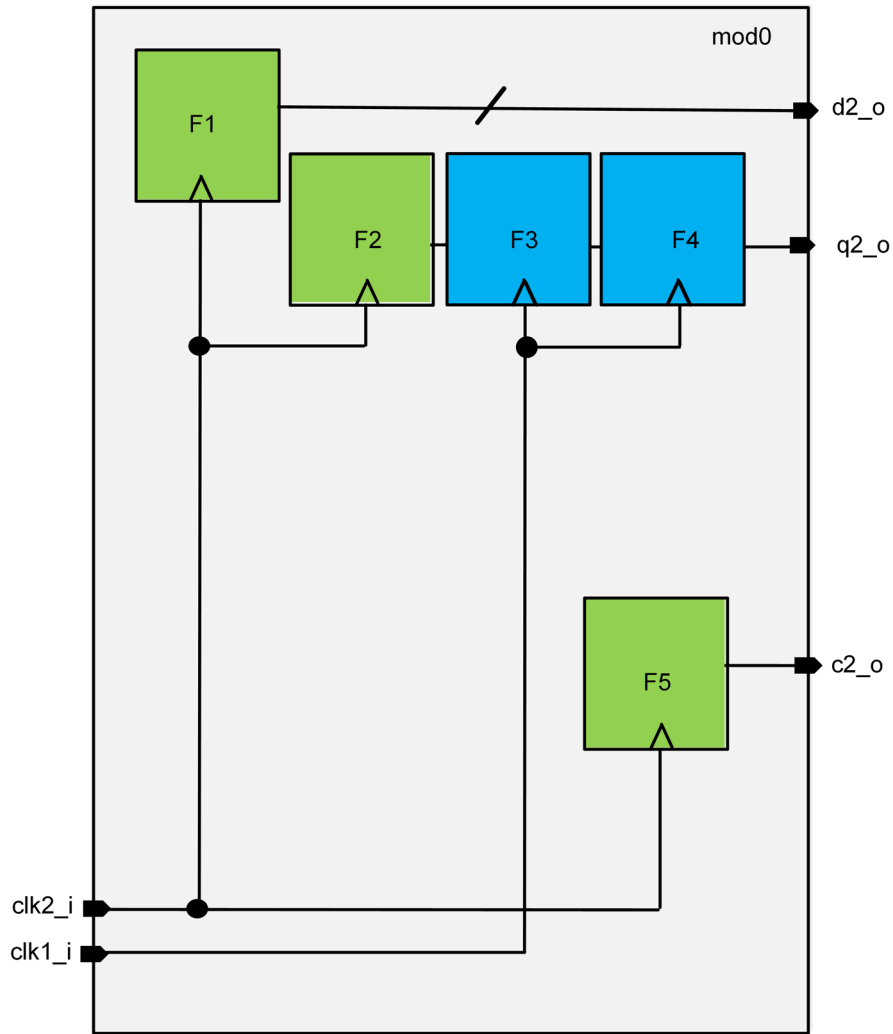
3 The model of the output interface of an abstracted block is described in terms of the relevant structures
 4 inside the block (internal model). Hence, the mandatory models for both inputs and outputs describe the
 5 structure of their drivers. The following examples describe the circuit in [Figure 14](#).

6

```

7 port -name d2_o -type data -direction output -associated_from_clocks clk2_i
8     -cdc_control q2_o
9 port -name q2_o -type cdc_control -direction output -cdc_control_from_clock
10     clk2_i -associated_from_clocks clk1_i -associated_outputs d2_o
11 port -name c2_o -type data -direction output -associated_from_clocks clk2_i
    
```

1



2

Figure 14—Module mod0 with abstracted output ports

3

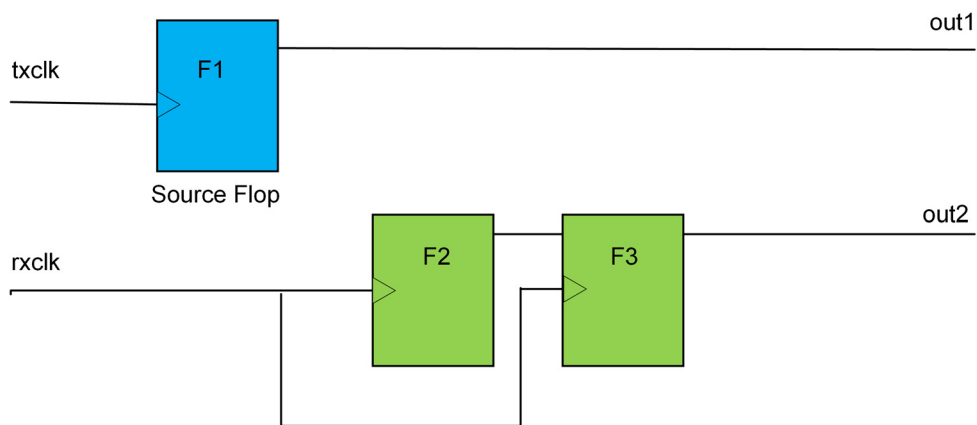
1 4.6 Clock definitions

2 This section shows three examples of clock definitions. [Figure 15](#) depicts two asynchronous clocks tx and
 3 rx.

```

    4
    5 port -name txclk -direction input -type clock
    6 set_cdc_clock_group -name tx -clocks {txclk}
    7
    8 port -name rxclk -direction input -type clock
    9 set_cdc_clock_group -name rx -clocks {rxclk}
    10
    
```

11



12

Figure 15—Clock definition A

13

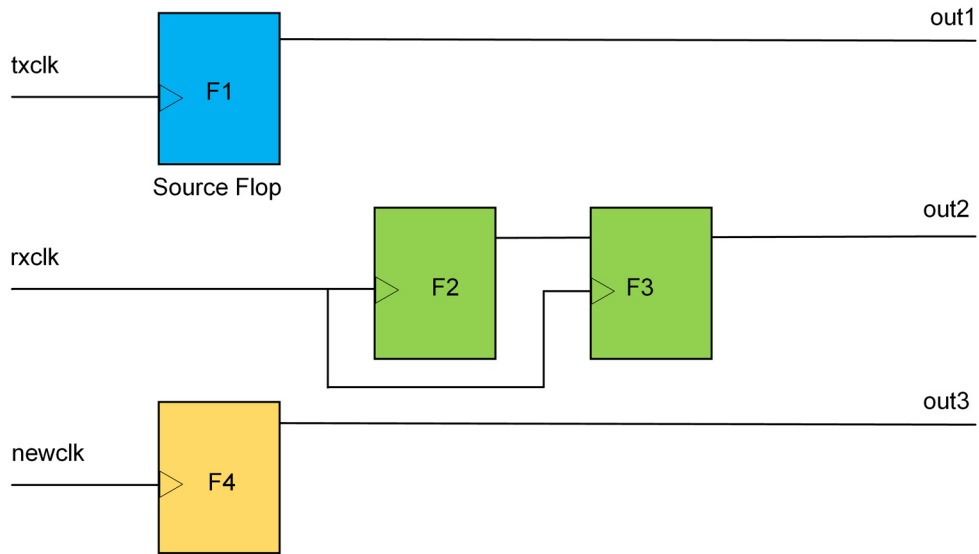
1 [Figure 16](#) depicts three asynchronous clocks txclk, rxclk, and newclk.

```

2
3 port -name txclk -direction input -type clock
4 set_cdc_clock_group -name tx -clocks {txclk}
5
6 port -name rxclk -direction input -type clock
7 set_cdc_clock_group -name rx -clocks {rxclk}
8
9 port -name newclk -direction input -type clock
10 set_cdc_clock_group -name new -clocks {newclk}
11

```

12



13

Figure 16—Clock definition B

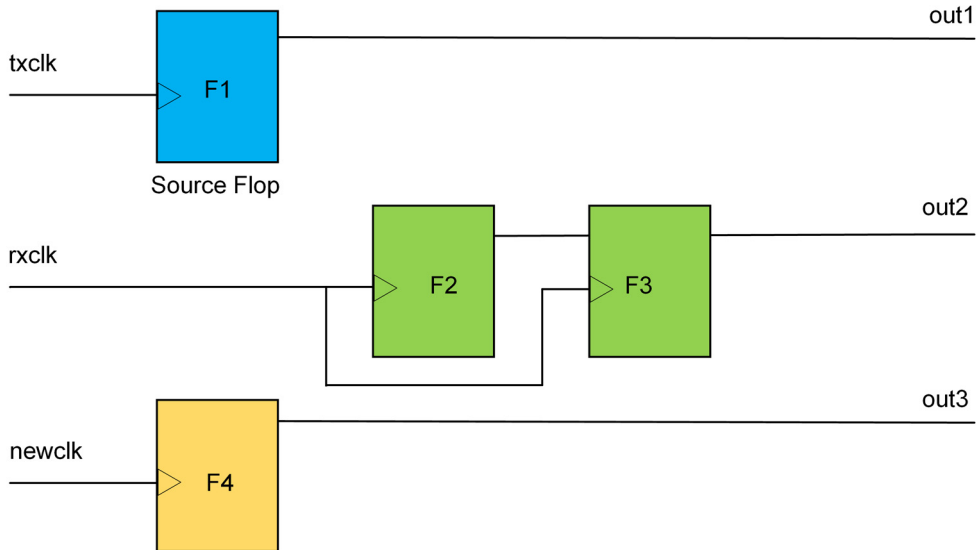
14

1 [Figure 17](#) depicts three clocks txclk, rxclk, and newclk. In this example, tx and new are asynchronous
 2 to rx.

```

    3
    4 port -name txclk -direction input -type clock
    5 set_cdc_clock_group -name tx -clocks {txclk}
    6
    7 port -name rxclk -direction input -type clock
    8 set_cdc_clock_group -name rx -clocks {rxclk}
    9
    10 port -name newclk -direction input -type clock
    11 set_cdc_clock_group -name new -clocks {newclk}
    12
    
```

13



14 **Figure 17—Clock definition C**

15

1 4.7 Clock relationships

2 The command `set_cdc_clock_group` is used to clearly define clock relationships in terms of
3 synchronicity. The default assumption for CDC is that clocks are asynchronous to each other. Any two
4 clocks that are members of a common CDC clock group are considered synchronous.

5 The command `set_cdc_clock_group` supports an optional attribute `-name`. If names are assigned to
6 CDC clock groups, the names must be unique. Even if a clock group is a transitive continuation of others, it
7 shall have a unique name. This approach avoids synchronicity relations between groups with the same name
8 that may appear distributed over the abstracted model. Instead of introducing several groups with equal
9 names, merge them into one group with a unique name. The `-name`'s space is local to an IP.

10 In the following example, the same synchronicity relation is described by different grouping and naming of
11 the CDC clock groups. When expanding the groups to the set of all pairs of clocks that you can build from
12 the group members, you receive the same set in both cases. In other words, both the example of one group
13 and the example of three groups listed below are equivalent. Despite having three different group names, all
14 `{clk1;clk2;clk3;clk4}` are synchronous to each other.

15 In one group:

16

```
17 set_cdc_clock_group -clocks {clk1;clk2;clk3;clk4} -name sys_clk_domain
```

18 In three groups:

19

```
20 set_cdc_clock_group -clocks {clk1;clk2;clk3} -name center_and_left
21 set_cdc_clock_group -clocks {clk2;clk3;clk4} -name center_and_right
22 set_cdc_clock_group -clocks {clk1;clk4} -name balanced_branches
```

23 The usage of the groups in the context of CDC checks does not depend on preprocessing for merging or
24 expansion of the groups but can also keep them as given by the abstracted model. An EDA tool could
25 designate how to merge or expand these groups for top-level consumption.

26 The synchronicity relation of clocks `{A, B, C}` is reflexive (A sync to A) and symmetrically (A sync to B
27 means B sync to A). In most applications, it is also transitive:

28 A sync to B and B sync to C means A sync to C.

29 In this case, the largest possible sets of synchronous clocks are disjoint; i.e., after merging the CDC clock
30 groups as far as the synchronicity allows, each clock appears in exactly one of the CDC clock groups.

31

32 However, there are also applications with a non-transitive synchronicity of clocks, e.g. a root clock driving
33 two generated clocks with “odd” division factors (large value for least common multiple) or two generated
34 clocks that propagate to branches of the clock tree with a large physical distance. In both cases, it may be
35 very expensive to implement synchronous timing arcs between the generated clocks, so instead, they may be
36 considered as asynchronous to each other.

37 In cases of non-transitive synchronicity, the largest possible sets of synchronous clocks are maximum sets of
38 compatibility that may have common members. However, it is not required to find these largest possible
39 sets. Subsets and even pairs can describe the same synchronicity relation. The above-mentioned groups
40 `center_and_left` and `center_and_right` without the last group `balanced_branches` are an
41 example for a non-transitive synchronicity relation. `clk1` and `clk4` can be understood as non-balanced
42 branches of the clock tree. Refer to [Figure 21](#) for an example of a non-transitive use model.

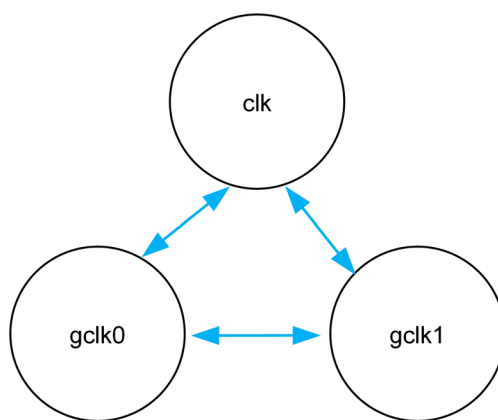
1 The following examples show different cases of synchronicity. Blue solid arrows denote synchronous clock
 2 relationships. Red dashed arrows denote asynchronous clock relationships.

3 [Figure 18](#) depicts three clocks. All are pairwise synchronous to each other so that they build a common clock
 4 group.

5

```
6 set_cdc_clock_group -name common_domain -clocks {clk;gclk0;gclk1}
```

7



8

Figure 18—One clock domain

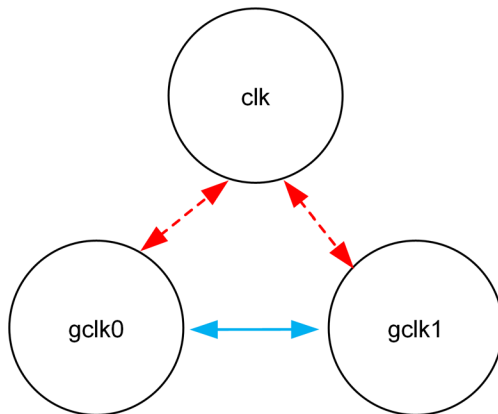
9

1 [Figure 19](#) depicts two clock groups. The clock `clk` builds a group on its own because it is asynchronous to
 2 the other two clocks.

```

    3
    4 set_cdc_clock_group -name small_domain -clocks {clk}
    5 set_cdc_clock_group -name large_domain -clocks {gclk0,gclk1}
    6
    
```

7



8

Figure 19—Two clock domains

9

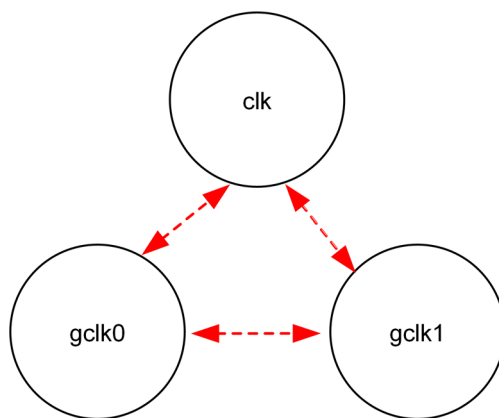
1 [Figure 20](#) depicts three clocks that are all pairwise asynchronous to each other so that every clock builds an
2 individual clock group.

3

```
4 set_cdc_clock_group -name domain_c -clocks {clk}  
5 set_cdc_clock_group -name domain_0 -clocks {gclk0}  
6 set_cdc_clock_group -name domain_1 -clocks {gclk1}
```

7

8



9

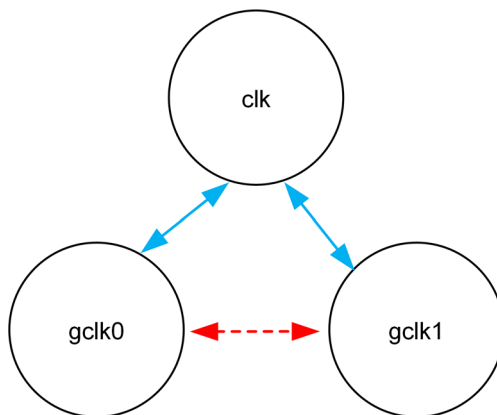
Figure 20—Three clock domains

10

1 [Figure 21](#) depicts a non-transitive synchronicity between three clocks. The root clock `clk` is synchronous to
2 both generated clocks `gclk0` and `gclk1`, but these are asynchronous to each other. The root clock `clk`
3 appears as a member in both clock groups, whereas `gclk0` and `gclk1` do not appear in a common group.
4 Both clock groups are described as maximum compatibility sets of clocks.

```
5  
6 set_cdc_clock_group -name clk_branch0 -clocks {clk,gclk0}  
7 set_cdc_clock_group -name clk_branch1 -clocks {clk,gclk1}  
8
```

9



10

Figure 21—Two clock domains

11

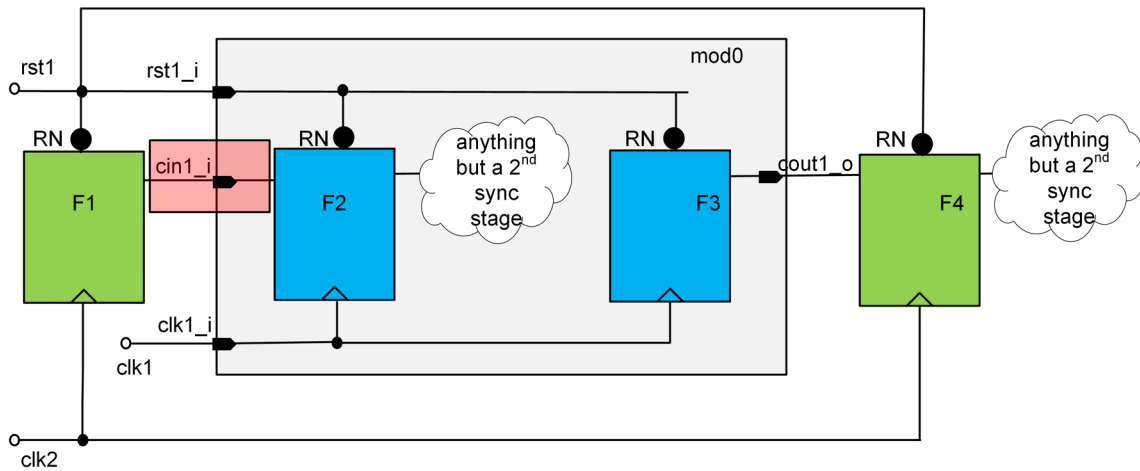
1 4.8 Failure modes

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

4 [Figure 22](#) depicts the propagation of metastability with a mean time before failure (MTBF) that is too low.
 5 The failure is due to a missing synchronizer. This scenario assumes that `cin1_i` has an asynchronous
 6 driver. This failure mode can occur at ports of type `data` or of type `cdc_control`, which are described in
 7 the following models.

```

    8
    9     port -name cin1_i -direction input -type data -associated_from_clocks clk1_i
    10
    11     port -name cin1_i -direction input -type cdc_control -cdc_control_from_clock
    12         clk2 -associated_to_clocks clk1_i
    13
    14
    
```

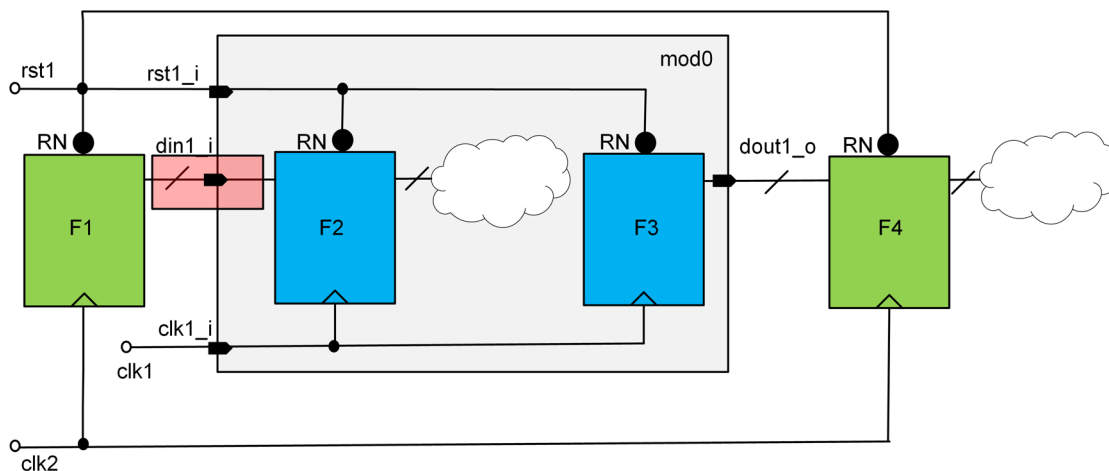


15 **Figure 22—Failure mode due to missing synchronizer**

16

1 [Figure 23](#) depicts the propagation of an intermediate (random or metastable) data value. The failure is due to
 2 a missing cdc_control.

3



4

Figure 23—Failure mode due to missing qualifier

5

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

8

```

9 module -name mod0
10 port -name din1_i -type data -associated_to_clocks clk1_i
11 -cdc_control qual
12
13 port -name qual -type cdc_control -cdc_control_from_clock clk2
14 -associated_to_clocks clk1_i
15 -associated_inputs din1_i

```

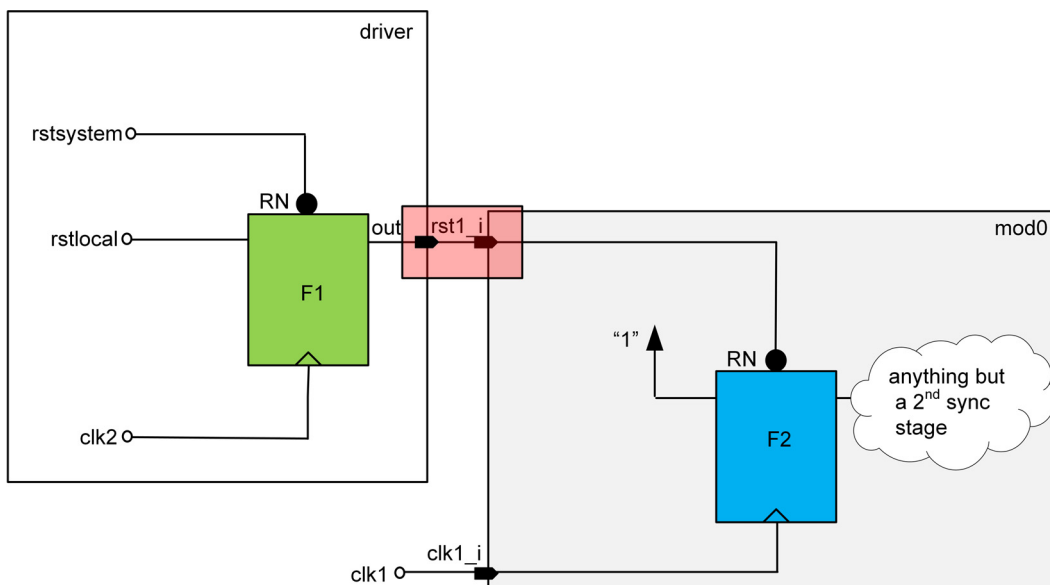
16

1 [Figure 24](#) depicts an asynchronous deassertion of an asynchronous reset (CDC on reset tree). The failure is
 2 due to a missing reset synchronizer.

```

3
4 module -name mod0
5 port -name rst1_i -direction input -type async_reset -associated_to_clocks
6     clk1_i
7 -polarity low
8
9 module -name driver
10 port -name out -direction output -type async_reset -associated_from_clocks
11     clk2 -polarity low
    
```

12



13

Figure 24—Failure mode due to missing reset synchronizer

14

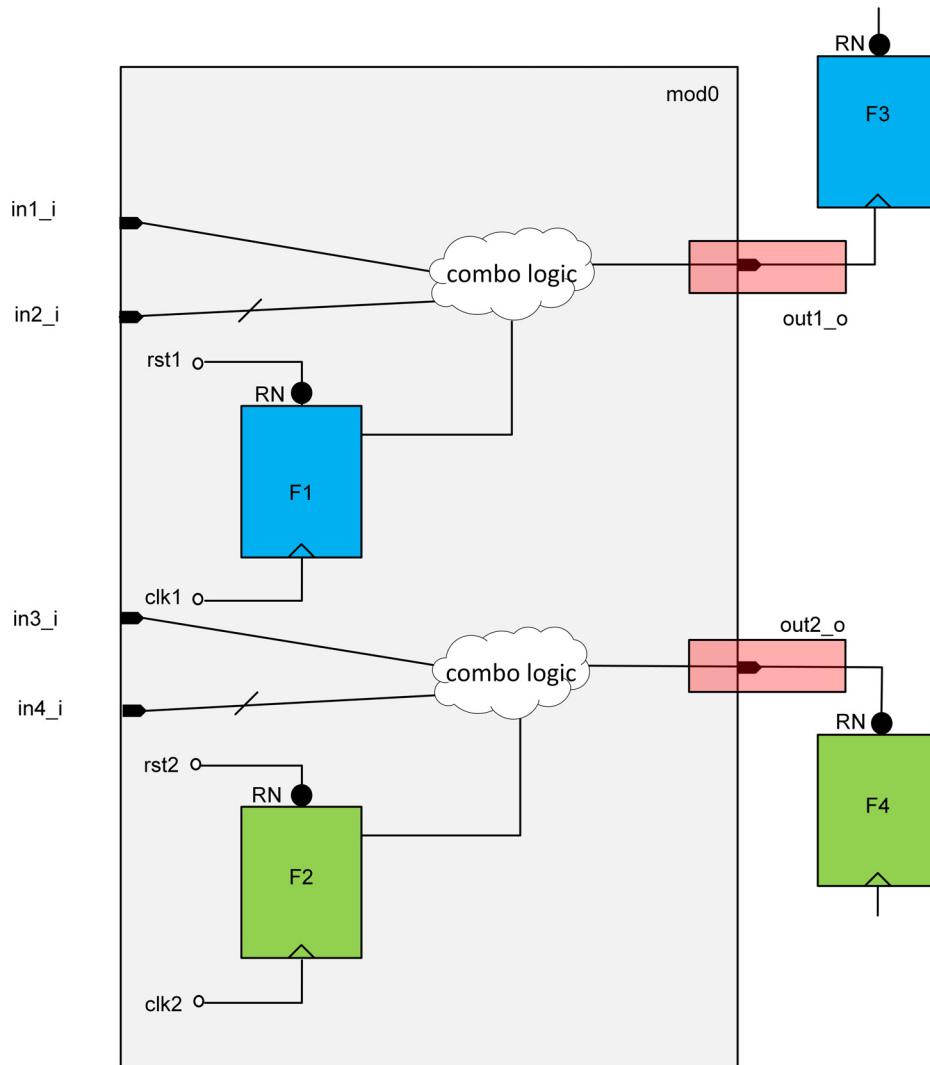
1 We cannot have combo logic that can glitch, for instance, on a clock or a reset tree. [Figure 25](#) shows that if
 2 the combo logic can glitch on the clock or reset network, there will be a violation.

```

3
4 module -name mod0
5   port -name out1_o -direction output -type data -logic combo
6     -associated_from_clocks clk1 -associated_inputs {in1_i; in2_i}
7
8   module -name mod0
9     port -name out2_o -direction output -type data -logic combo
10       -associated_from_clocks clk2 -associated_inputs {in3_i; in4_i}
    
```

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

13



14

Figure 25—Failure mode due to combo logic driving clock/reset

15

1 5. Support for RDC verification

2 This clause details output requirements for IP with multiple resets from the top level and RDC control within
3 and outside the IP. See [Table 1](#) for the supported RDC attributes along with their domain, type, accepted
4 values, whether they are mandatory, and clarifying comments.

5 To support RDC interface verification, the CDC attribute table ([Table 1](#)) has been updated with the
6 following content beginning with the CDC LRM version 0.3.

7 a) New Domain

8 1) `set_reset_group`: Defines the reset relation. There are no RDC violations between resets
9 that appear together in the same reset group.

10 b) New attributes

11 1) `-rdc_control_from_rst`: Defines the source reset of RDC, i.e., start point of the sequen-
12 tial's reset. This attribute shall be used together with `-rdc_control`. This is the source reset
13 being blocked by `-rdc_control`.

14 2) `-rdc_control_to_rst`: Defines the destination reset of RDC, i.e., the end point of the
15 sequential's reset. This attribute shall be used together with `-rdc_control`.

16 3) `-rdc_control_to_clock`: Defines the destination clock of the RDC end point of the
17 sequential's clock. This attribute shall be used together with `-rdc_control`. This is the des-
18 tination clock being gated by `-rdc_control`.

19 4) `-rdc_clock_gate_location`: Defines the location of the clock gate, i.e., whether it is
20 internal or external. This attribute shall be used together with `-rdc_control_to_clock`.

21 5) `-associated_from_reset`: Defines the source reset of a port.

22 6) `-associated_to_rst`: Defines the destination reset of a port.

23 c) New Values

24 1) `rdc_control` for the `-type` attribute: Defines the RDC control. The usage for this attribute
25 is the same as `cdc_control`.

26 2) `virtual_reset` for the `-type` attribute: Defines a virtual reset. The usage for this attribute
27 is the same as `virtual_clock`.

28 The usage of the new additions will be described with scenarios in the following sections.

29 Note—The images in this new clause are placeholders. They will be redrawn to match the style of existing images.

30 5.1 Requirements for IP with control outside the IP

31 This section describes the necessary interface information for an IP design that does not have an RDC
32 qualifier within the IP. It is assumed that when the output collateral for the IP is generated, the IP does not
33 have any knowledge of the number of resets driving the IP's interface reset ports.

34 5.1.1 Scenario 1

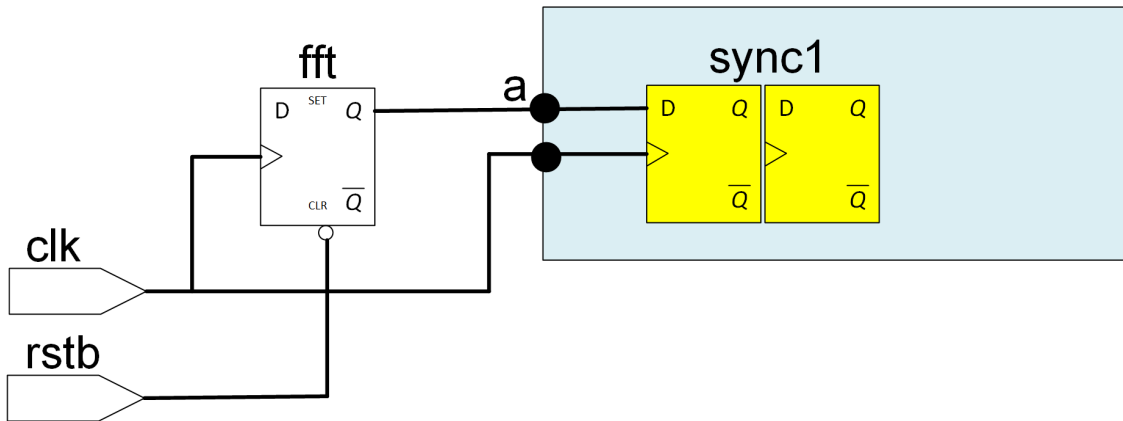
35 This section describes support for an RDC interface at the top level without resynthesizing the RTL of the IP
36 as shown in [Figure 26](#). The IP shall capture the following information in its output collateral for the top-level
37 interface RDC verification.

38

```
39 port -name a -type data -direction input -logic internal_sync
40     -associated_from_clocks clk
41 port -name a -associated_to_clocks clk
```

42

1



2

Figure 26—External rstb with internal synchronization logic

3 When the IP is integrated by the top-level logic as shown in [Figure 26](#), the connectivity, clock domain, and
 4 reset domain of the IP at the top shall match the IP’s modeling. As such, the following shall be checked by
 5 the tool:

6 When a receiving port has a synchronizer property, any RDC can be ignored because the synchronizer is
 7 present within the IP. In [Figure 26](#), port a of the IP is a receiving port connected to the input of the
 8 synchronizer.

9 If the IP and top have mismatches, the tool can report them to the user for investigation during the top-level
 10 verification.

11 5.1.2 Scenario 2

12 This section describes support for an RDC interface at the top level without resynthesizing the RTL of the IP
 13 as shown in [Figure 27](#). The IP shall capture the following information in its output collateral for the top-level
 14 interface RDC verification.

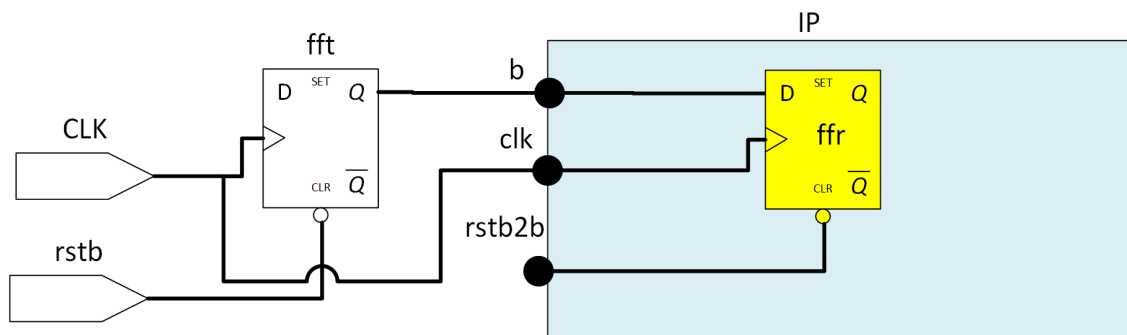
15

```

16 port -name b -type data -direction input -associated_from_clocks clk
17     -associated_from_rst -virtual_reset_b -associated_to_rst rstb2b
18 port -name b -associated_to_clocks clk
19 port -name rstb2b -type async_reset -polarity low -direction input
20     -associated_from_clocks clk
21 port -name rstb2b -associated_to_clocks clk
22 port -name virtual_reset_b -type virtual_reset
  
```

23

1



2

Figure 27—rstb2b of IP with associated reset

3 When the IP is integrated by the top-level logic as shown in [Figure 27](#), the connectivity, clock domain, and
 4 reset domain of the IP at the top shall match the IP’s modeling. As such, the following shall be checked by
 5 the tool:

- 6 a) The top level’s reset and clock signals for port `b` of IP shall align with the IP’s assumptions to
 7 ensure there are no CDC or RDC issues. Specifically, the tool shall compare the reset domain of
 8 `rstb` with that of the receiver reset `rstb2b` of IP. It is essential to ensure that `rstb2b` occurs
 9 before `rstb` to maintain the correct reset sequence, which is critical for the stable operation of
 10 sequential elements within the design.
- 11 b) If the design constraints guarantee that `rstb2b` precedes `rstb`, or if `rstb2b` of IP is properly con-
 12 nected at the interface level to `rstb`, there will be no RDC violations. Thus, if these conditions are
 13 met, no synchronization is necessary.

14 If the IP and top have mismatches, the tool can report them to the user for investigation during the top-level
 15 verification.

16 5.1.3 Scenario 3

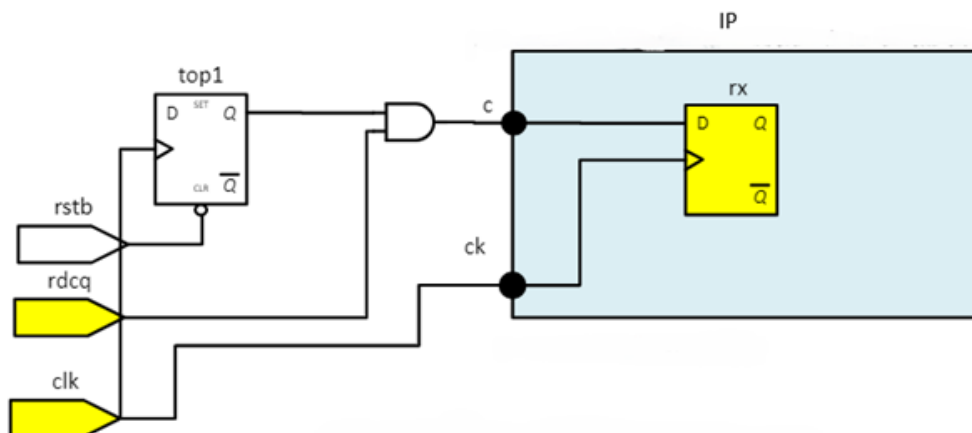
17 [Figure 28](#) captures how the RDC interface is managed when the control is handled outside the IP with port `C`
 18 being driven by logic from reset qualifier `rdcq`. This section describes support for an RDC interface at the
 19 top level without resynthesizing the RTL of the IP. The IP shall capture the following information in its output
 20 collateral for the top-level interface RDC verification.

21

```
22 port -name c -type data -direction input -associated_from_clocks ck
23 port -name c -associated_to_clocks ck
24 port -name ck -type clock -direction input
```

25

1



2

Figure 28—External rdcq on the data signal of the IP

3 When the IP is integrated by the top-level logic as shown in [Figure 28](#), the connectivity, clock domain, and
 4 reset domain of the IP at the top shall match the IP’s modeling. As such, the following shall be checked by
 5 the tool:

- 6 a) The external RDC qualifier `rdcq` shall ensure there is no RDC from the receiving port `c` of IP to
 7 the `rx` flop of IP at the top level due to a reset qualifier at the data pin. When `rstb` is asserted at
 8 the top level, it is blocked by `rdcq`, eliminating the need for an additional RDC qualifier at the IP
 9 level.
- 10 b) Upon integrating this IP at the top level, it is crucial to have an RDC qualifier to manage the RDC
 11 from `rstb` to the receiving port `c` of IP. The `rdcq` signal effectively blocks the RDC, and this
 12 mechanism is modeled as an RDC qualifier/control at the top level.

13 If the IP and top have mismatches, the tool can report them to the user for investigation during the top-level
 14 verification.

15 5.1.4 Scenario 4

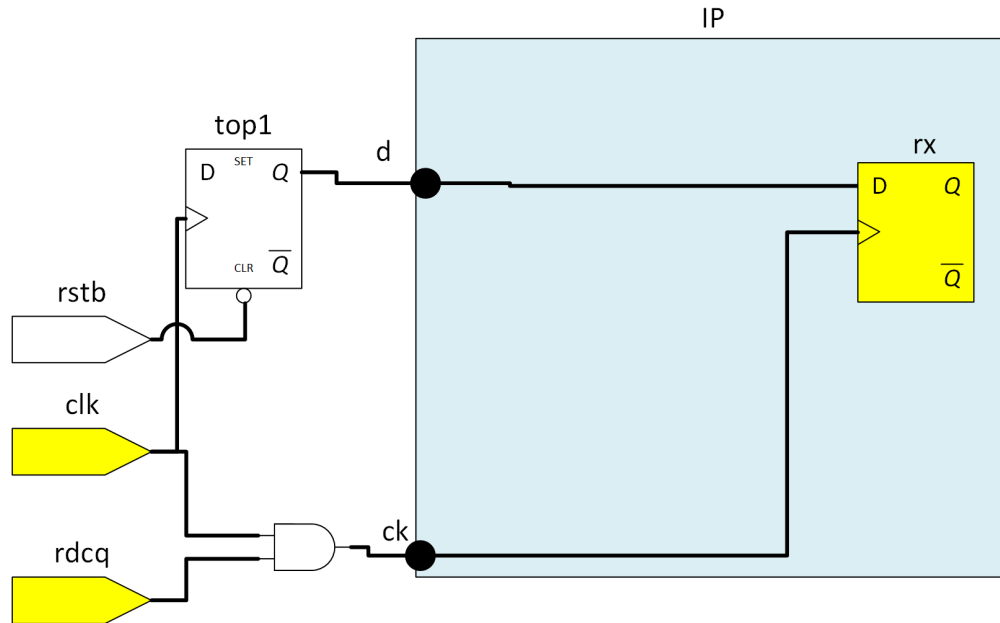
16 [Figure 29](#) captures using a reset qualifier on a clock signal. It is crucial to ensure that the data port receives
 17 the clock signal that has been properly qualified by the reset. This section describes support for an RDC
 18 interface at the top level without resynthesizing the RTL of the IP. The IP shall capture the following information
 19 in its output collateral for the top-level interface RDC verification.

20

```
21 port -name d -type data -direction input -associated_from_clocks ck
22 port -name d -associated_to_clocks ck
23 port -name ck -type clock -direction input
```

24

1



2

Figure 29—External rdcq on the clock signal of the IP

3 When the IP is integrated by the top-level logic as shown in [Figure 29](#), the connectivity, clock domain, and
 4 reset domain of the IP at the top shall match the IP’s modeling. As such, the following shall be checked by
 5 the tool:

- 6 a) In the CDC LRM version 0.1, `-associated_to_clocks` is listed as optional; however, in this
 7 scenario, it becomes necessary to ensure correct mapping and to prevent assuming that there is no
 8 CDC violation. Thus, the `-associated_to_clocks` attribute shall be included to ensure accu-
 9 rate verification.
- 10 b) The tool shall verify that the data port is correctly receiving the reset-qualified clock signal. Since
 11 the clock `ck` is present in the IP and is the driver of receiving port `d` of IP, it is essential to use this
 12 information for top-level analysis. Knowing that `ck` of IP is gated before `rstb` is asserted ensures
 13 no internal RDC violations occur.

14 If the IP and top have mismatches, the tool can report them to the user for investigation during the top-level
 15 verification.

16 5.2 Requirements for IP with control within the IP

17 This section describes the necessary interface information for an IP design that has RDC control within the
 18 IP and multiple resets driven from the top level. It is assumed that when the output collateral for the IP is
 19 generated, the IP has no knowledge of the number of resets that drive the IP’s interface reset ports. Five
 20 scenarios will be presented in this section with usage for the attributes introduced in the CDC LRM version
 21 0.3.

22 5.2.1 Scenario 1

23 This section describes support for an RDC interface at the top level without resynthesizing the RTL of the IP
 24 as shown in [Figure 30](#). It is recommended for the IP to capture the following information in its output
 25 collateral for the top-level RDC interface verification.


```

1
2 port -name rstb -type async_reset -polarity low -direction input
3   -associated_from_clocks clkx
4 port -name rstb -associated_to_clocks clkx (optional)
5 port -name rstx -type async_reset -polarity low -direction input
6   -associated_from_clocks clkx
7 port -name rstx -associated_to_clocks clkx (optional)
8 port -name rdcq -type rdc_control -direction input -rdc_control_from_rst rstb
9   -rdc_control_to_rst rstx -associated_from_clocks clkx
10 port -name clkx -type clock -direction input
11
12

```

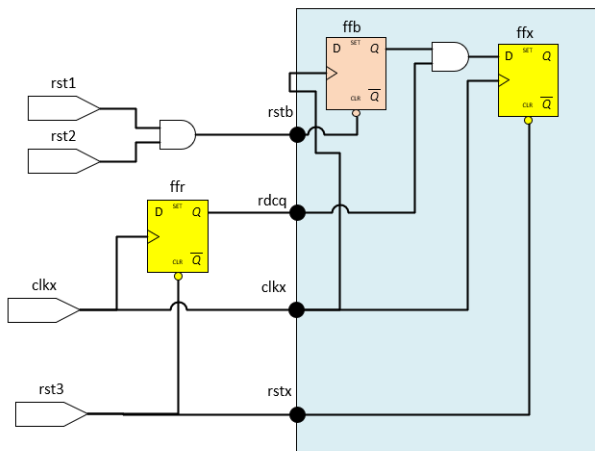


Figure 30—rstb of IP driven by multiple resets

15 -associated_from_clocks for port -name rstb and port -name rstx conveys
16 information used by the IP to ensure there is no reset deassertion issue within the IP. Although this
17 information is not needed for RDC interface modeling, it is needed for CDC interface verification. It is
18 illustrated in this example because the output collateral covers interface information needed for both CDC
19 and RDC verification.

20 As described in [Table 1](#), -associated_to_clocks for port -name rstb and port -name
21 rstx is optional information captured at the interface. With -associated_from_clocks and
22 -associated_to_clocks information for port -name rstb and port -name rstx respectively,
23 this indicates that the signal driving the async reset pin is synchronous to the receiver flop’s clock. EDA
24 tools may determine how much information will be generated for the IP’s output collateral.

25 -type rdc_control was added to the CDC LRM version 0.3 to support RDC modeling. The usage is
26 similar to that of cdc_control, which was introduced in the CDC LRM version 0.1. The only difference is
27 that rdc_control is used for blocking RDC instead of CDC.

28 Along with the introduction of the -type rdc_control attribute, the following four new attributes are
29 also added: -rdc_control_from_rst, -rdc_control_to_rst, -rdc_control_to_clock,
30 and -rdc_clock_gate_location. These attributes are used with -type rdc_control to
31 describe the blocking conditions. For example,

32

```

1  port -name rdcq -type rdc_control -direction input -rdc_control_from_rst rstb
2      -rdc_control_to_rst rstx -associated_from_clocks clkx

```

3 The command above specifies that `rdcq` is an RDC control signal that is used to block any RDC from
4 `rstb` to `rstx`. The signal `rdcq` is driven from `clkx`. From the above example, it also infers that `rdcq` is
5 driven from the same reset domain as `rstx`.

6 The usage of `-rdc_control_to_clock` and `-rdc_clock_gate_location` is shown in Scenario
7 2 (see [5.2.2](#)).

8 When the IP is integrated by the top-level logic as shown in [Figure 30](#), the connectivity, clock domain, and
9 reset domain of the IP at the top shall match the IP's modeling. As such, the following shall be checked by
10 the tool.

- 11 a) The top level's reset for `rdcq` is aligned with the IP's assumption such that `rdcq` is driven by a
12 reset that is in the same reset domain as the IP. `rstx`.
- 13 b) The top level's clock for `rdcq` is aligned with the IP's assumption such that `rdcq` is driven by a
14 clock that is in the same clock domain as IP. `clkx`.
- 15 c) Based on the reset relationship at the top level, the following assumption shall be true at the top
16 level:

```

17  port -name rdcq -type rdc_control -direction input -rdc_control_from_rst
18      top.rst1 top.rst2 -rdc_control_to_rst top.rst3 -associated_from_clocks
19      top.clkx

```

20 If the IP and top have mismatches, the tool can report them to the user for investigation during the top-level
21 verification.

22 5.2.2 Scenario 2

23 This section describes support for an RDC interface at the top level without resynthesizing the RTL of the IP
24 as shown in [Figure 31](#). It is recommended for the IP to capture the following information in its output
25 collateral for the top-level RDC interface verification.

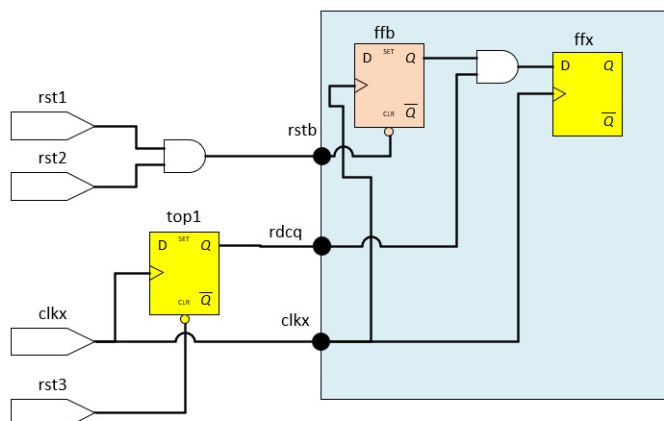
```

26
27  port -name rstb -type async_reset -polarity low -direction input
28      -associated_from_clocks clkx
29  port -name rstb -associated_to_clocks clkx (optional)
30  port -name rdcq -type rdc_control -direction input -rdc_control_from_rst rstb
31      -rdc_control_to_clock clkx -rdc_clock_gate_location external
32      -associated_from_clocks clkx -associated_from_reset virtual_reset_rdcq
33  port -name clkx -type clock -direction input
34  port -name virtual_reset_rdcq -type virtual_reset -direction input

```

35

1



2 **Figure 31—rstb of IP driven by multiple resets. Destination flop has no async reset pin**

3 When `-rdc_control_from_rst` and `-rdc_control_to_clock` are used together with `-type`
 4 `rdc_control`, they specify that when the source reset (the reset specified with
 5 `-rdc_control_from_rst`) goes into reset, it is expected that the RDC endpoint's flop's clock (the
 6 clock specified with `-rdc_control_to_clock`) shall be gated to prevent metastability.
 7 `-rdc_clock_gate_location` defines the location of the clock gate cell. In this Scenario 2 example,
 8 the IP expects `clkx` to be gated by the SoC. The attribute `-associated_from_reset` is intended for
 9 use by the top level to ensure there is no RDC from the reset specified by `-associated_from_reset`
 10 to the RDC endpoint's flop in the IP after integration.

11 When the IP is integrated by the top-level logic as shown in [Figure 31](#), the connectivity, clock domain, reset
 12 domain, clock, and reset relationship of the IP at the top level shall match the IP's modeling. As such, the
 13 following shall be checked by the tool.

- 14 a) The top level's reset and clock for `rdcq` is aligned with the IP's assumption such that when the reset
 15 from `rdcq` is asserted, the clock for the IP, i.e., `IP.clkx` shall be gated.

16
 17 The following interface information from the IP indicates that `IP.rdcq` is expecting `IP.clkx` to
 18 gate when the reset that drives `IP.rdcq` is asserted.

19
 20 `port -name rdcq -type rdc_control -direction input -rdc_control_from_rst`
 21 `rstb -rdc_control_to_clock clkx -rdc_clock_gate_location external`
 22 `-associated_from_clock clkx -associated_from_reset virtual_reset_rdcq`

- 23 b) Based on the reset relationship at the top level, the following assumption shall be true at the top
 24 level.

25
 26 `port -name rdcq -type rdc_control -direction input -rdc_control_from_rst`
 27 `top.rst1 top.rst2 -rdc_control_to_clock top.clkx`
 28 `-rdc_clock_gate_location external -associated_from_clocks top.clkx`
 29 `-associated_from_reset top.rst3`

30
 31 `port -name rdcq` shall block the RDC before `top.rst1` and `top.rst2` assert.

32 If the IP and top have mismatches, the tool can report them to the user for investigation during the top-level
 33 verification.

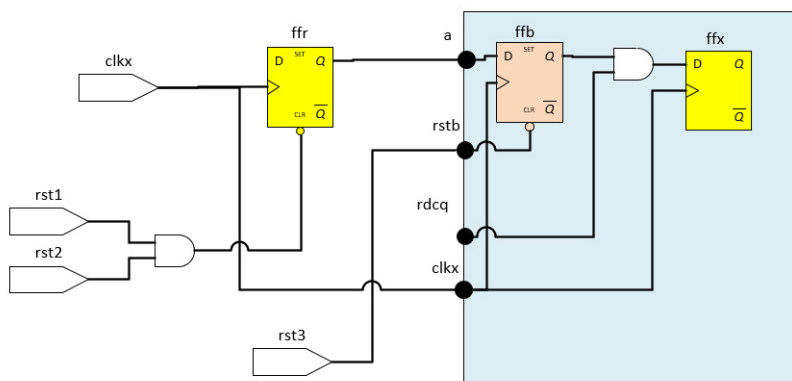
1 5.2.3 Scenario 3

2 This section describes support for an RDC interface at the top level without resynthesizing the RTL of the IP
 3 shown in [Figure 32](#). It is recommended for the IP to capture the following information in its output collateral
 4 for the top-level RDC interface verification.

```

5
6 port -name a -type data -direction input -associated_from_clocks clkx
7     -associated_from_reset virtual_reset_a -associated_to_rst rstb
8 port -name a -associated_to_clocks clkx (optional)
9 port -name rstb -type async_reset -polarity low -direction input
10    -associated_from_clocks clkx
11 port -name rstb -associated_to_clocks clkx (optional)
12 port -name rdcq -type rdc_control -direction input -rdc_control_from_rst rstb
13    -rdc_control_to_clock clkx -rdc_clock_gate_location external
14    -associated_from_clock clkx -associated_from_reset virtual_reset_rdcq
15 port -name clkx -type clock -direction input
16 port -name virtual_reset_rdcq -type virtual_reset -direction input
17 port -name virtual_reset_a -type virtual_reset -direction input
18 set_reset_group -name reset_domain_1 -reset {virtual_reset_a; rstb}
19
20

```



21 **Figure 32—rstb of IP is controlling a flop that is driven by multiple resets**

22 Scenario 3 shows the modeling of a data path in IP to support RDC interface verification. Within the IP,
 23 port -name a is a pure data path. To enable the RDC interface, both the driver and the receiver’s reset
 24 information shall be modeled at the interface; hence, -associated_from_reset and
 25 -associated_to_rst are used. For example,

```

26
27 port -name a -type data -direction input -associated_from_clocks clkx
28     -associated_from_reset virtual_reset_a -associated_to_rst rstb
29 set_reset_group -name reset_domain_1 -reset {virtual_reset_a; rstb}

```

30 The set_reset_group attribute above indicates that RDC from virtual_reset_a to rstb is safe
 31 from metastability issues. When the IP is integrated with the top-level logic as shown in [Figure 32](#), the
 32 connectivity, clock domain, reset domain, clock, and reset relationship of the IP at the top level shall match
 33 the IP’s modeling. As such, the following shall be checked by the tool.

- 1 a) The top level's reset and clock for IP . a aligns with the IP's assumption, so there will be no CDC or
 2 RDC. There shall be no RDC violation from top . rst1 and top . rst2 to top . rst3 such that
 3 the following assumption shall be true at the top level.

```
4
5 port -name top.ffr.Q -type data -associated_from_clocks clkx
6 -associated_from_reset top.rst1 top.rst2 -associated_to_rst
7 top.rst3
```

8
 9 top . rst3 shall belong to the same reset domain as top . rst1 and top . rst2.

- 10 b) When mapped to the top level, when the driver of virtual_reset_rdcq asserts, it is expected
 11 that top . clkx will be gated to fulfill the requirement of the IP's assumption.

```
12
13 port -name rdcq -type rdc_control -direction input
14 -rdc_control_from_rst rstb -rdc_control_to_clock clkx
15 -rdc_clock_gate_location external -associated_from_clock clkx
16 -associated_from_reset virtual_reset_rdcq
```

17 If the IP and top have mismatches, the tool can report them to the user for investigation during the top-level
 18 verification.

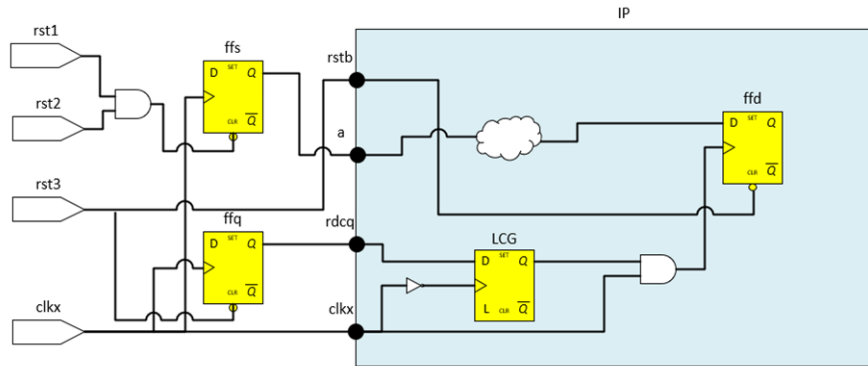
19 5.2.4 Scenario 4

20 This section describes support for an RDC interface at the top level without resynthesizing the RTL of the IP
 21 shown in [Figure 33](#). It is recommended for the IP to capture the following information in its output collateral
 22 for the top-level RDC interface verification.

```
23
24 port -name a -type data -direction input -associated_from_clocks clkx
25 -associated_from_reset virtual_reset_a -associated_to_rst rstb
26 port -name a -associated_to_clocks clkx -logic combo (optional)
27 port -name rstb -type async_reset -polarity low -direction input
28 -associated_from_clocks clkx
29 port -name rstb -associated_to_clocks clkx (optional)
30 port -name rdcq -type rdc_control -direction input -rdc_control_from_rst
31 virtual_reset_a -rdc_control_to_reset rstb -rdc_control_to_clock clkx
32 -rdc_clock_gate_location internal -associated_from_clocks clkx
33 -associated_from_reset virtual_reset_rdcq -associated_input a
34 port -name clkx -type clock -direction input
35 port -name virtual_reset_a -type virtual_reset
36 port -name virtual_reset_rd -type virtual_reset
```

37

1



2

Figure 33—rdcq is gating the interface clk

3 Without the `rdcq` modeling shown in [Figure 33](#), RDC from `virtual_reset_a` to `rstb` would have
 4 been a violation within the IP because there is no definition of a relationship between these two resets. The
 5 `rdcq` modeling indicates that `rdcq` will be used to block RDC for port `-name a` if the source reset is
 6 from `virtual_reset_a` and the destination reset is `rstb` where the RDC's endpoint clock will be gated
 7 with internal clock gating present in the IP.

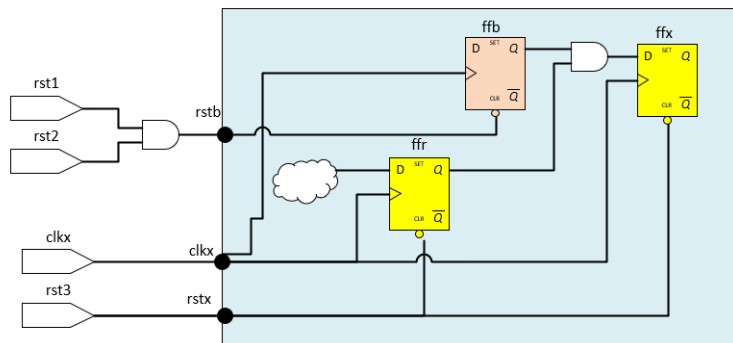
8 If the same RDC control has been used by multiple ports, `-associated_input` can have a list of ports.
 9 [Figure 33](#) shows an example where only port `a` has used the `rdc_control`.

10 When the IP is integrated by the top-level logic as shown in [Figure 33](#), the connectivity, and clock domain of
 11 the IP at the top level shall match the IP's modeling. As such, the followings shall be checked by the tool.

- 12 a) The clock domain and reset domain for `IP.rstb` and `IP.rdcq` shall retain the same domains at
- 13 the top level.
- 14 b) `IP.clkx` shall be gated by `rdcq` before the assertion of the top-level `rst1` or `rst2`.

15 5.2.5 Scenario 5

16



17

Figure 34—Internal rdcq

18 This scenario is not supported in the CDC LRM version 0.3. Because `ffr`, which is the RDC control from
 19 `IP.rstb` to `IP.rstx` resides within the IP, there is no interface port available for the RDC interface for

1 the top-level RDC verification. To verify this kind of design, the RDC control information shall be made
2 available at the interface.

3 This will be a future focus for the CDC WG.

1 6. CDC TCL format

2 This clause describes the CDC format that is based on TCL language for CDC specification from an output
3 collateral perspective. It is assumed that users have one file per block/module in which users can specify the
4 CDC attributes for ports of that module and users can also specify clock groups. We expect to add further
5 commands for input collateral going forward. This clause describes the syntax and semantics for each CDC
6 command.

7 It is expected that EDA tools should process the CDC specification in TCL format and generate
8 corresponding IP-XACT collateral. Similarly, there could be a requirement for generating TCL specification
9 of CDC from IP-XACT.

10 6.1 cdc_set_module

11

Purpose	Indicates the module/block for which CDC specification is provided in further commands. It is specified in the beginning of a file.	
Syntax	<code>cdc_set_module module-name</code>	
Arguments	<code>module-name</code>	The name of the module for which CDC specification is being created.
Return value	Returns an empty string if successful or raises a TCL_ERROR if not.	

12 This command allows users to indicate the module/block name for which CDC specification is being
13 provided in a file. The subsequent commands are applicable to the module that is specified as *module-name*
14 in the `cdc_set_module` command.

15 There will be an error if *module-name* is not specified or does not exist in the design.

16 6.1.1 Syntax example

```
17 # Set a module for CDC specification
18 cdc_set_module ALU
19 # other CDC commands
20 ...
```

21 6.2 cdc_set_port

22

Purpose	Sets the attributes of a port of a module/block.
Syntax	<code>cdc_set_port port-name</code> <code>-type port-type</code> <code>[-direction port-direction]</code> <code>[-polarity polarity]</code> ...

Arguments	<i>port-name</i>	The name of the port for which CDC is specified.
	-type <i>port-type</i>	The type of a port, which can be data, clock, <code>async_reset</code> , or <code>cdc_control</code> .
	-direction <i>port-direction</i>	The direction of a port, which can be input, output, or inout. RTL already has direction, so it can be optional here, but it will be useful when the containing module is a black box or in the case of an inout port.
	-polarity <i>polarity</i>	Polarity is applicable for the <code>async_reset</code> port type only.
Return value	Returns an empty string if successful or raises a <code>TCL_ERROR</code> if not.	

1 This command allows users to set attributes of a port on a single line. The CDC attributes for a port are listed
2 in the [Table 9](#) below.

3 There will be an error if:

- 4 a) *port-name* is not specified or does not exist in the module.
- 5 b) a module has not been set using the `cdc_set_module` command.
- 6 c) an attribute is specified for a particular type of port that is not applicable to that type of port.

7 6.2.1 Syntax example

```

8 # Set a module
9 cdc_set_module ALU
10 # Set attributes of a port
11 cdc_set_port CLK -type clock -virtual_port p2 -frequency value ...
12
13 cdc_set_port RegA -type data ...
14
15 cdc_set_port RESET -type reset -polarity high ...
16 ...
17
18
```

Table 9—CDC Port Attributes

Attribute	Type	Values	Mandatory
Name	string	{port name}	Yes
Direction	defined set	{input, output, inout}	Yes
Type	defined set	{data, clock, <code>async_reset</code> , <code>cdc_control</code> }	Yes
Logic	defined set	{combo, buffer, inverter, glitch_free_combo, internal_sync}	Optional
cdc_control_from_clock	string	{clock name}	Optional
associated_from_clocks	; separated list	{clock-names}	Yes

Table 9—CDC Port Attributes (*continued*)

Attribute	Type	Values	Mandatory
<code>associated_to_clocks</code>	; separated list	{clock-names}	Optional
<code>associated_from_resets</code>	; separated list	{reset-names}	Optional
<code>associated_to_resets</code>	; separated list	{reset-names}	Optional
<code>associated_inputs</code>	; separated list	{ports}	Optional
<code>associated_outputs</code>	; separated list	{ports}	Optional
<code>cdc_control</code>	; separated list	{associated-ports}	Optional
Polarity	defined set	{high, low, low_high}	Yes
Ignore	defined set	{blocked, hanging}	Optional
<code>cdc_static</code>	; separated list	{can be any, <clocks to which it is cdc_static>}	Optional
Constant	; separated list	{binary, hex, and of any length}	Optional
<code>gray_coded</code>	Boolean	{true, false:default}	Optional
<code>clock_period</code>	string	{clock period}	Optional

1 6.3 cdc_set_clock_group

2

Purpose	Creates a clock group for a module/block.	
Syntax	<code>cdc_set_clock_group</code> [-name <i>clock-group-name</i>] -clocks { <i>clock-name</i> +}	
Arguments	<code>-name <i>clock_group_name</i></code>	The name of the clock group. It is optional.
	<code>-clocks <i>clock-name</i>+</code>	TCL list consisting of one or more clock names that are synchronous to each other.
Return value	Returns an empty string if successful or raises a <code>TCL_ERROR</code> if not.	

3 This command allows users to set clock groups of synchronous clocks. A clock group can be specified as a
4 TCL list having one or more clock names that are synchronous.

5 There will be an error if one or more specified clock names do not exist in the design.

6 6.3.1 Syntax example

```

7   # Set a module
8   cdc_set_module ALU
9   # Set attributes of a port
10  cdc_set_port CLK -type clock -virtual_port p2 -frequency value ...
11
12  cdc_set_port RegA -type data ...
13
```

```

1  cdc_set_port RESET -type reset -polarity high ...
2  ...
3  # set cdc clock group
4  cdc_set_clock_group -clocks {clk1 clk2}

```

6.4 cdc_set_param

6

Purpose	Defines a parameter within the scope of a module.	
Syntax	<pre> cdc_set_param -name parameter-name -type type [-value value] [-ignore ignore] </pre>	
Arguments	-name <i>parameter-name</i>	The name of the parameter. It is mandatory.
	-type <i>type</i>	Can be any of int/string/Boolean. It is optional. Default: int
	-value <i>value</i>	Value can be a string, number, or Boolean based on the type of the parameter.
	-ignore <i>ignore</i>	If true, the parameter is ignored. It flags an error if it is used in subsequent commands. By default, it is true if no value is specified.
Return value	Returns an empty string if successful or raises a TCL_ERROR if not.	

7 This command allows users to define parameters within the scope of a module.

8 There will be an error if:

- 9 a) *parameter-name* is not specified.
- 10 b) If parameter is ignored and still referred to in subsequent commands.

6.4.1 Syntax example

```

12  # Set a module
13  cdc_set_module ALU
14  # Set a parameter for the module
15  cdc_set_param -name MSB -type int -value 15
16  cdc_set_param -name LSB -type int -value 0
17
18  cdc_set_port RegA[MSB:LSB] -type data ...

```

7. CDC IP-XACT format

This section describes the IP-XACT format for CDC specification. It is an extension to the IP-XACT standard. The IP-XACT standard allows users to capture some attributes related to clock and reset. Other attributes that are currently not part of the IP-XACT standard are defined as Vendor Extensions. IP-XACT allows extending the standard using these vendor extensions. Vendor extensions for CDC are defined as Accellera Vendor extensions at different elements in the design hierarchy. They are described in detail in the following sections.

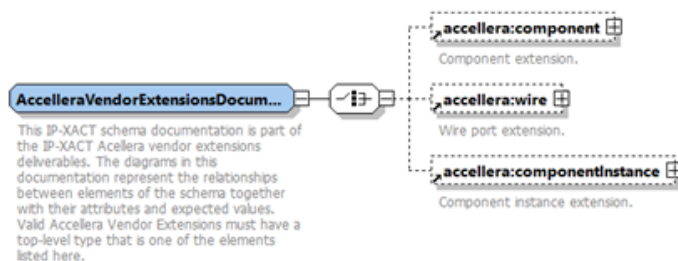
7.1 Top-level elements

The Accellera vendor extensions for CDC must have documents with the following top-level types:

- `accellera:component`
- `accellera:wire`
- `accellera:componentInstance`

This is demonstrated in the schema diagram in [Figure 35](#).

14



15

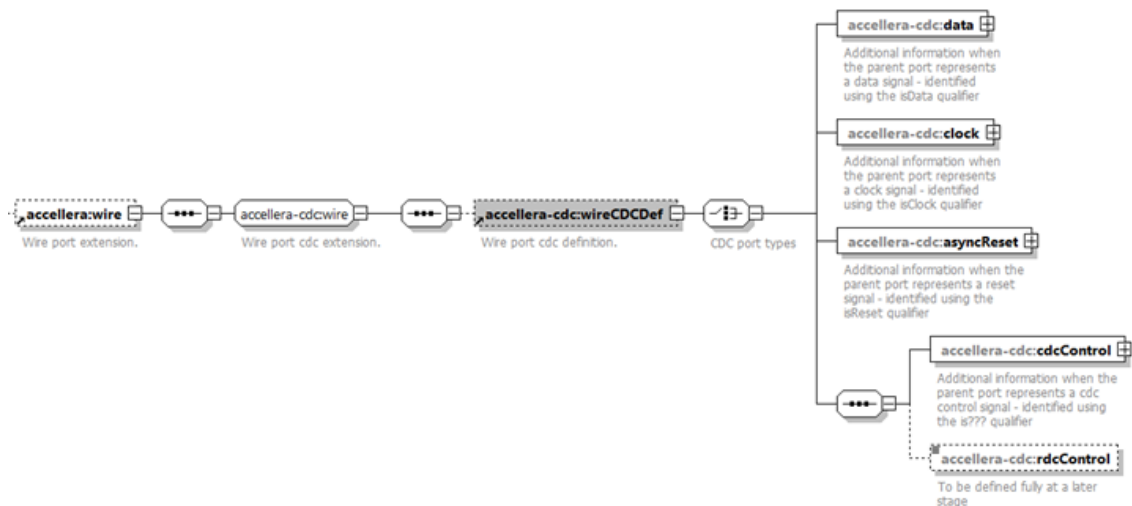
Figure 35—Schema for top-level elements

7.2 Top element—`accellera:wire`

The top-level element `accellera:wire` is used to define an Accellera-specific wire port extension, which contains the wire port CDC definition, `accellera-cdc:wireCDCDef`. The `wireCDCDef` element allows defining one of the CDC port types, that is, data, clock, reset, or control as shown in the schema diagram in [Figure 36](#).

20

1



2

Figure 36—Schema for accellera:wire

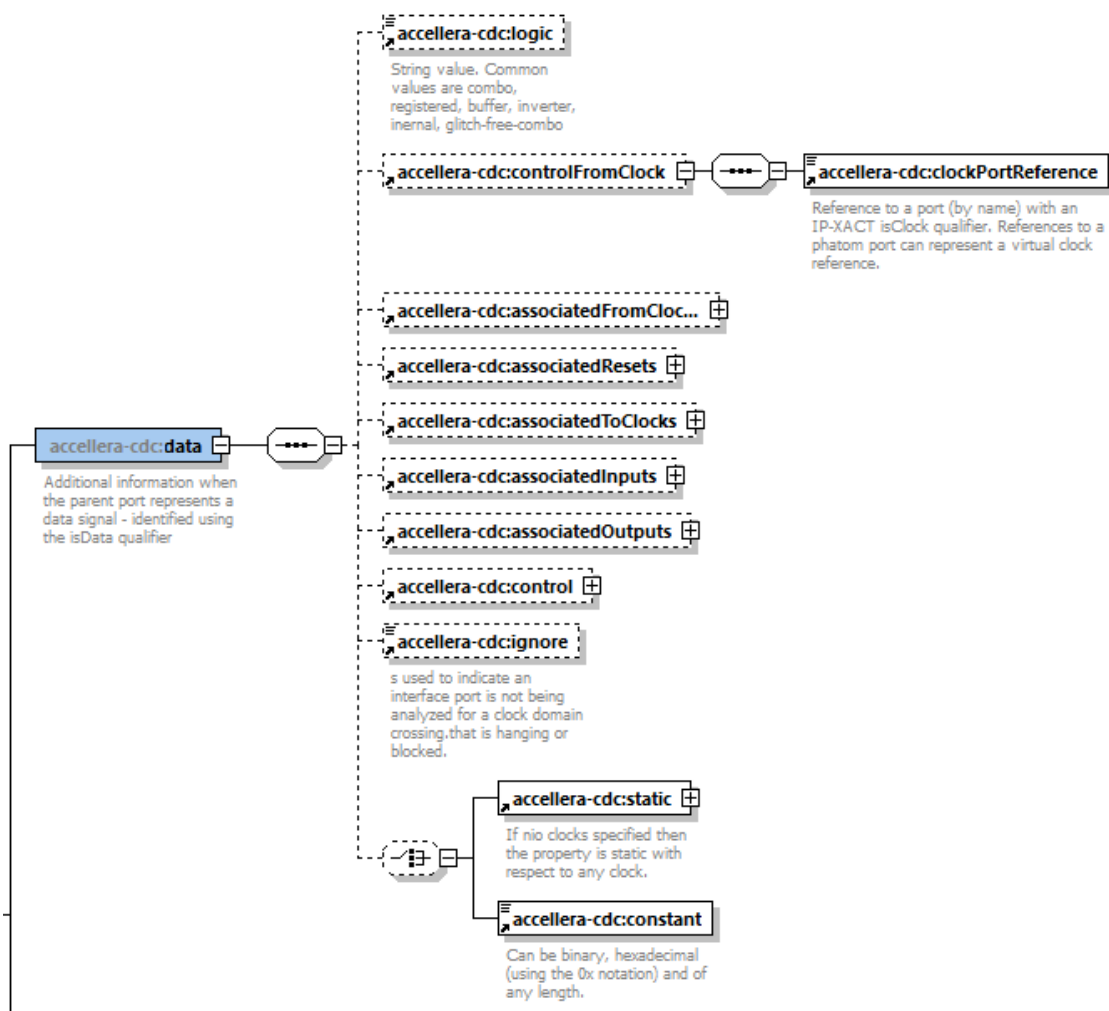
3 Each element of a wireCDCDef is defined as follows:

- 4 a) accellera-cdc:data describes the attributes of a data port type
- 5 b) accellera-cdc:clock describes the attributes of a clock port type
- 6 c) accellera-cdc:asyncReset describes the attributes of a reset port type
- 7 d) accellera-cdc:cdcControl describes the attributes of a CDC control port type
- 8 e) accellera-cdc:rdcControl describes the attributes of a CDC reset port type

9 **7.3 Data port type—accellera-cdc:data**

10 When a port represents a data signal, all attributes required for CDC are captured in the extension,
 11 accellera-cdc:data, as shown in the schema diagram in [Figure 37](#). Each element of the
 12 accellera-cdc:data extension map to an attribute for the data port as defined in [Clause 4, Table 1](#).

1



2

Figure 37—Schema for accellera-cdc:data

3 7.3.1 IP-XACT code for accellera-cdc:data

```

4 <ipxact:port>
5   <ipxact:name>i_data</ipxact:name>
6   <ipxact:wire>
7     <ipxact:direction>in</ipxact:direction>
8     <ipxact:vector>
9       <ipxact:left>7</ipxact:left>
10      <ipxact:right>0</ipxact:right>
11    </ipxact:vector>
12  </ipxact:wire>
13  <ipxact:vendorExtensions>
14    <accellera-cdc:wire>
15      <accellera-cdc:wireCDCDef>
16        <accellera-cdc:data>
17          <accellera-cdc:associatedFromClocks>
18            <accellera-cdc:clockPortReference>i_clk</accellera-
19 cdc:clockPortReference>
20          </accellera-cdc:associatedFromClocks>

```

```

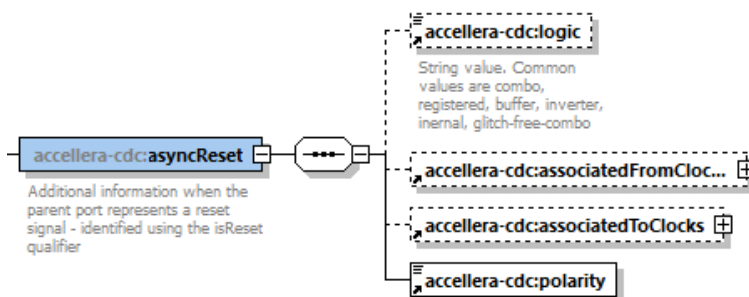
1         </accellera-cdc:data>
2         </accellera-cdc:wireCDCDef>
3     </accellera-cdc:wire>
4 </ipxact:vendorExtensions>
5 </ipxact:port>

```

6 7.4 Reset port type—accellera-cdc:asyncReset

7 When a port represents a reset signal, the reset attributes for CDC are captured in the extension,
8 `accellera-cdc:asyncReset`, as shown in the schema diagram in [Figure 38](#). In addition to the
9 `accellera-cdc:logic` attribute in the `accellera-cdc:asyncReset` extension, there are
10 additional available attributes; e.g., `polarity`, `associatedFromClocks`, and
11 `associatedToClocks` as defined in [Clause 4, Table 1](#).

12



13

Figure 38—Schema for accellera-cdc:asyncReset

14 7.4.1 IP-XACT code for accellera-cdc:asyncReset

```

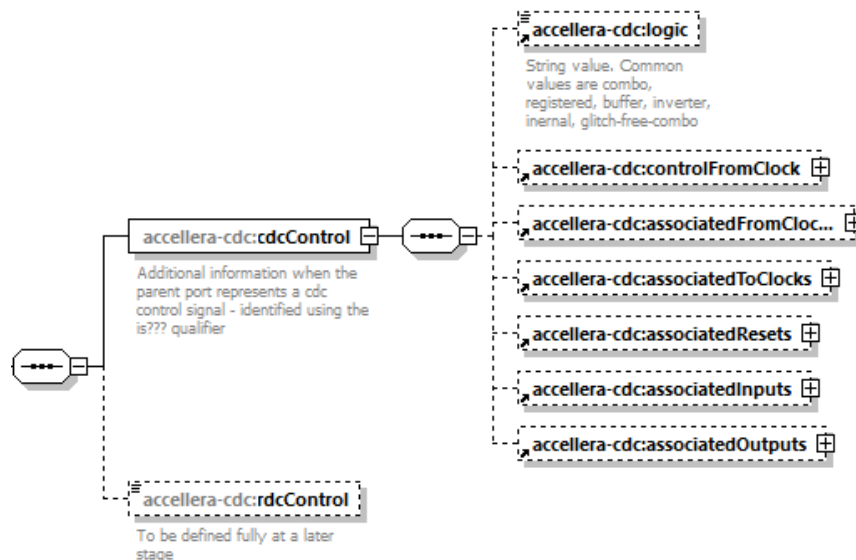
15 <ipxact:port>
16     <ipxact:name>i_rst</ipxact:name>
17     <ipxact:wire>
18         <ipxact:direction>in</ipxact:direction>
19         <ipxact:qualifier>
20             <ipxact:isReset>true</ipxact:isReset>
21         </ipxact:qualifier>
22     </ipxact:wire>
23     <ipxact:vendorExtensions>
24         <accellera-cdc:wire>
25             <accellera-cdc:wireCDCDef>
26                 <accellera-cdc:asyncReset>
27                     <accellera-cdc:associatedFromClocks>
28                         <accellera-cdc:clockPortReference>i_clk
29                         </accellera-cdc:clockPortReference>
30                     </accellera-cdc:associatedFromClocks>
31                     <accellera-cdc:asyncReset>
32                         <accellera-cdc:wireCDCDef>
33                     </accellera-cdc:wire>
34                 </ipxact:vendorExtensions>
35     </ipxact:port>

```

17.5 Control port type—`acellera-cdc:cdcControl` and `acellera-cdc:rdcControl`

2 When a port represents a CDC or RDC control signal, the corresponding attributes are captured in the
 3 extension, `acellera-cdc:cdcControl` or `acellera-cdc:rdcControl`, as shown in the
 4 schema diagram in [Figure 39](#). While attributes for CDC control signals are currently defined, the RDC
 5 control attributes will be defined in a future version of the CDC LRM.

6



7 **Figure 39—Schema for `acellera-cdc:cdcControl` and `acellera-cdc:rdcControl`**

8 7.5.1 IP-XACT code for `acellera-cdc:cdcControl`

```

9 <ipxact:port>
10 <ipxact:name>r_valid</ipxact:name>
11 <ipxact:wire>
12 <ipxact:direction>in</ipxact:direction>
13 </ipxact:wire>
14 <ipxact:vendorExtensions>
15 <acellera-cdc:wire>
16 <acellera-cdc:wireCDCDef>
17 <acellera-cdc:cdcControl>
18 <acellera-cdc:controlFromClock>
19 <acellera-cdc:clockPortReference>i_clk
20 </acellera-cdc:clockPortReference>
21 </acellera-cdc:controlFromClock >
22 <acellera-cdc:cdcControl>
23 </acellera-cdc:wireCDCDef>
24 </acellera-cdc:wire>
25 </ipxact:vendorExtensions>
26 </ipxact:port>

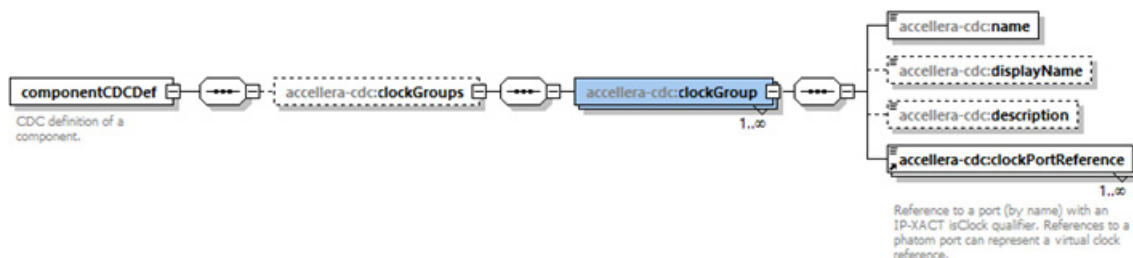
```

27 7.6 Control port type—`acellera-cdc:componentCDCDef`

28 A component is a top element in IP-XACT encompassing bus interfaces, memory maps, models (views and
 29 ports), power domains, parameters, etc. In CDC, there are clock groups that are associated with components.

1 The extension `accellera-cdc:componentCDCDef` is defined to capture the clock groups of a
 2 component using the `accellera-cdc:clockGroup` extensions shown in the schema diagram in
 3 [Figure 40](#). It contains references to clock ports or phantom ports (virtual clock) in addition to name,
 4 `displayName`, and `description`.

5



6 **Figure 40—Schema for accellera-cdc:componentCDCDef**

7 **7.6.1 IP-XACT code for accellera-cdc:componentCDCDef**

```

8 <ipxact:component>
9 <!-- Specify VLNV -->
10 ...
11 ...
12 <ipxact:vendorExtensions>
13 <accellera-cdc:clockGroups>
14 <accellera-cdc:clockGroup>
15 <accellera-cdc:name>grp_clk</accellera-cdc:name>
16 <accellera-cdc:clockPortReference>i_clk</accellera-
17 cdc:clockPortReference>
18 <accellera-cdc:clockPortReference>j_clk</accellera-
19 cdc:clockPortReference>
20 <accellera-cdc:clockPortReference>k_clk</accellera-
21 cdc:clockPortReference>
22 </accellera-cdc:clockGroup>
23 </accellera-cdc:clockGroups>
24 </ipxact:vendorExtensions>
25 </ipxact:component>
    
```

¹ **Annex A**

² (informative)

³ **Bibliography**

⁴ [B1] IEEE 100, *The Authoritative Dictionary of IEEE Standards Terms*, Seventh Edition. New York: Insti-
⁵ tute of Electrical and Electronics Engineers, Inc.

⁶