2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SAN JOSE, CA, USA
MARCH 4-7, 2024

Hierarchical CDC and RDC closure with standard abstract models

Accellera CDC Working Group

Ping YEUNG — NVIDIA
Chetan CHOPPALI SUDARSHAN — MARVELL
Farhad AHMED — SIEMENS
Iredamola OLOPADE — intel
Sean O'DONOHUE — SYNOPSYS
Bill GASCOYNE — Blue Pearl Software
Kranthi PAMARTHI — RENESAS
Anupam BAKSHI — AGNISYS

Presenter Introduction

- Companies
  - Dammy Iredamola Olopade, Intel, leads Quality & Innovation and has been leading CDC/RDC at Intel for over 20 years.
  - Ping Yeung, Nvidia, works on CDC/RDC and Formal Verification; previously at 0-In, Mentor and Siemens.
  - Anupam Bakshi, Agnisys
- EDA Vendors
  - Bill Gascoyne, Blue Pearl Software, develops and delivers technical training; previously at Magma and LSI Logic.
  - Farhad Ahmed, Principal Product Engineer at Siemens EDA Software; previously at Synopsys, Cadence, and Ansys.
  - Sean O'Donohue, Synopsys

2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE 10 YEAR ANNIVERSARY

Making the impossible possible: CDC and RDC closure with abstracts from different tools

Accellera CDC Working Group

Diana KALEL
Jean-Christophe BRIGNONE
Farhad AHMED
Eldad FALIK
Jebin Mohandes
Jerome AVEZOU
Bill GASCOYNE
Kranthi PAMARTHI







Agenda

	Topic	Slide update/create	Presenter
#1	CDC-RDC (35min)		
	CDC-RDC Basic Knowledge	Bill Gascoyne, Blue Pearl	Bill Gascoyne, Blue Pearl
	Setup Constraints & Verification	Ping Yeung, Nvidia	Ping Yeung, Nvidia
	Structural CDC/RDC	Chetan Choppali Sudarshan, Marvell	TBD (Greg Milano, Cadence)
	CDC Assertion-Based Verification	Kranthi Pamarthi, Renesas Electronics	Anupam Bakshi, Agnisys
	CDC-RDC Hierarchical Flow	Farhad Ahmed, Siemens EDA	Farhad Ahmed, Siemens EDA
#2	Accellera CDC (55min)		
	Standard, 10min	Iredamola Olopade, Intel	Iredamola Olopade, Intel
	Format, 10min	Devender Khari, Agnisys	Anupam Bakshi, Agnisys
	Output, 10min	Sean O'Donohue, Synopsys	TBD (Sean O'Donohue, Synopsys)
	Assertion, 5min	Kranthi Pamarthi, Renesas Electronics	Ping Yeung, Nvidia
	Testing, 5min	Farhad Ahmed, Siemens EDA	Farhad Ahmed, Siemens EDA
	Training, 5min	JC Brignone, Diana Kalel, ST Micro	Bill Gascoyne, Blue Pearl

2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

CDC-RDC Basic Knowledge:

- Synchronous vs. asynchronous clocks
- Problems related to Clock Domain Crossing (CDC)
- CDC Synchronization
- Problems related to Reset Domain Crossing (RDC)
- RDC Synchronization

accellera SYSTEMS INITIATIVE

1. Synchronous vs. Asynchronous

- Synchronous clocks
  - Same source
  - Have an easily-established timing relationship
  - Static Timing Analysis works
- Asynchronous clocks
  - From different sources
  - Timing relationship unknown or difficult to establish
  - Static Timing Analysis doesn’t work
- Multi-clock designs, NOT clockless

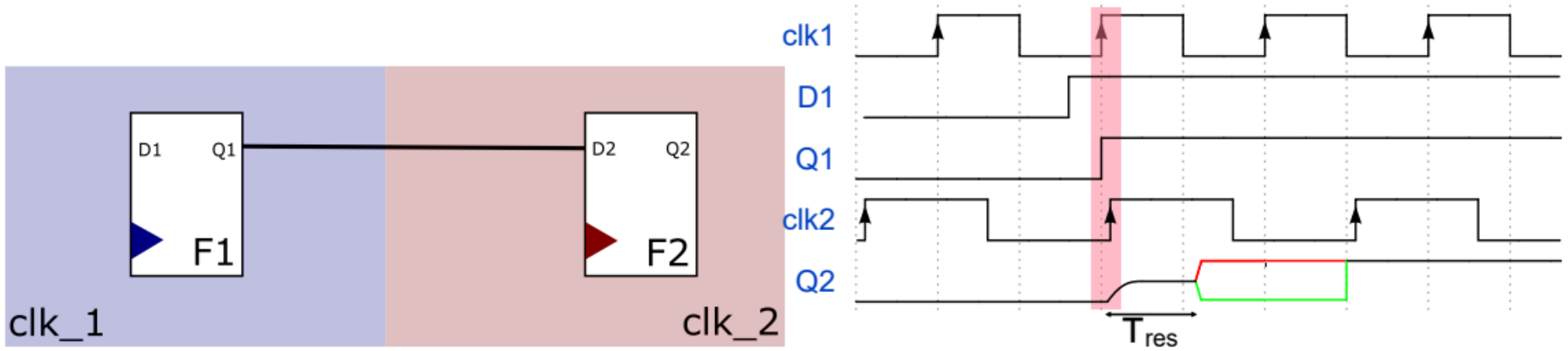
2. Coin Toss Analogy

- Think of a setup/hold violation result as the toss of a coin
  - Heads or Tails, but also very rarely it might just stay on its edge (metastability) before falling one way or the other
- Fixing metastability and fixing data coherency are independent
- For one bit, fixing metastability is enough
  - Coherency doesn't matter, since either heads or tails is fine
- For multiple bits, must fix metastability AND data coherency
  - Requires all heads or all tails from multiple coins
  - A losing bet!

3. Why do CDCs need fixing? (1/3)

- Metastability
  - Timing violations on registers resulting in an indeterminate state lasting more than one clock cycle

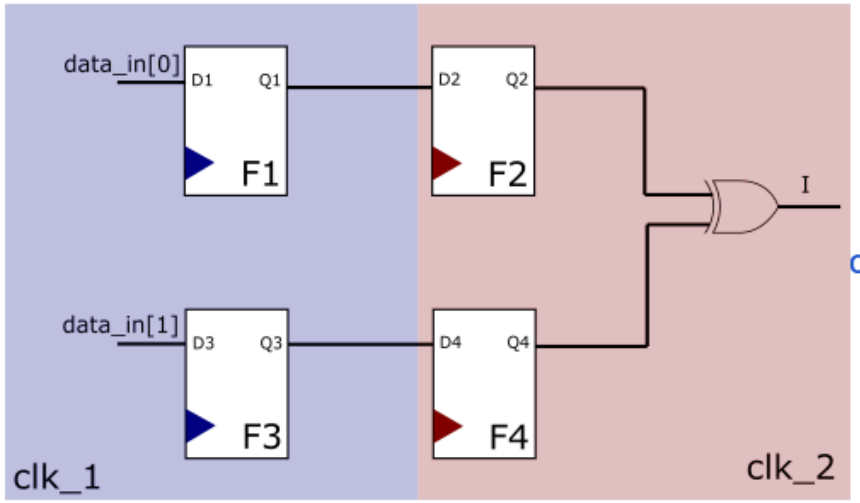
    The coin on its edge



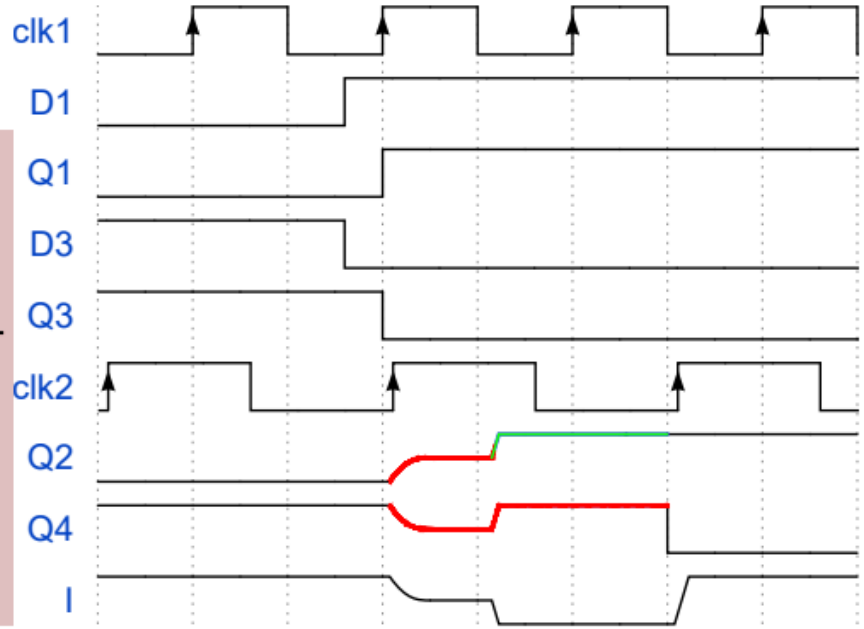
(a) Structure *(b) Waveform*

3. Why do CDCs need fixing? (2/3)

- Loss of Data Coherency
  - The indeterminate state settles to a random value



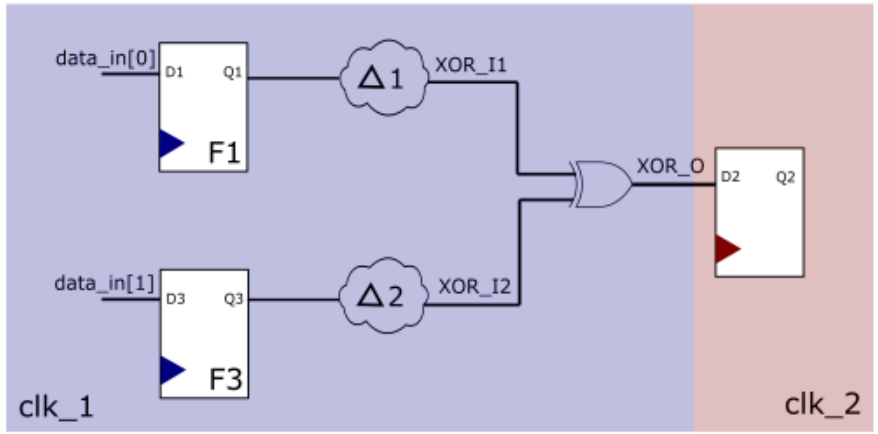
(a) Structure



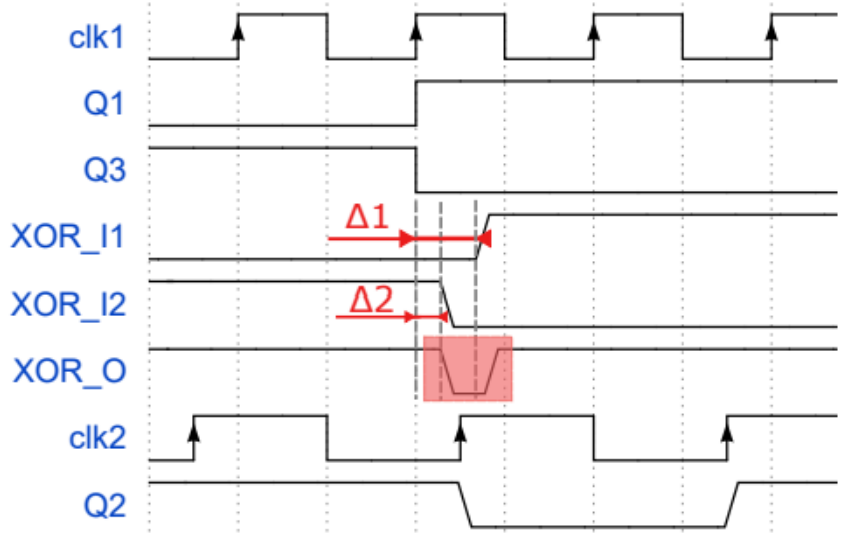
(b) Waveform

3. Why do CDCs need fixing? (3/3)

- Glitches
  - Multiple synchronized paths reconverge to cause unexpected momentary transitions



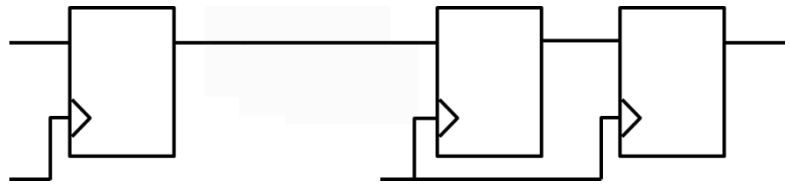
(a) Structure



(b) Waveform

4. Synchronization (1/3)

4.1 Multi-flop synchronizers



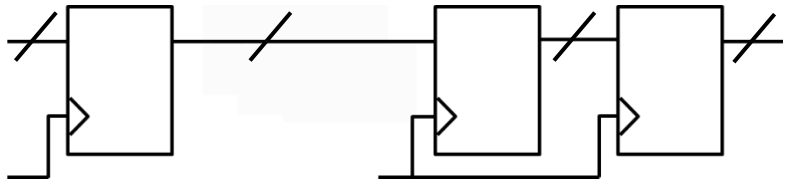
- Synchronizing one bit with two DFFs changes odds of metastability on the 2nd flop from ~1/p to ~1/p^2
  - The probability of a metastability event in a 2-flop metastability resolver

$$MTBF(t_r) = \frac{e^{\frac{t_r}{\tau}}}{T_0 * F_c * F_d} * \frac{e^{\frac{T_r}{\tau}}}{T_0 * F_c}$$

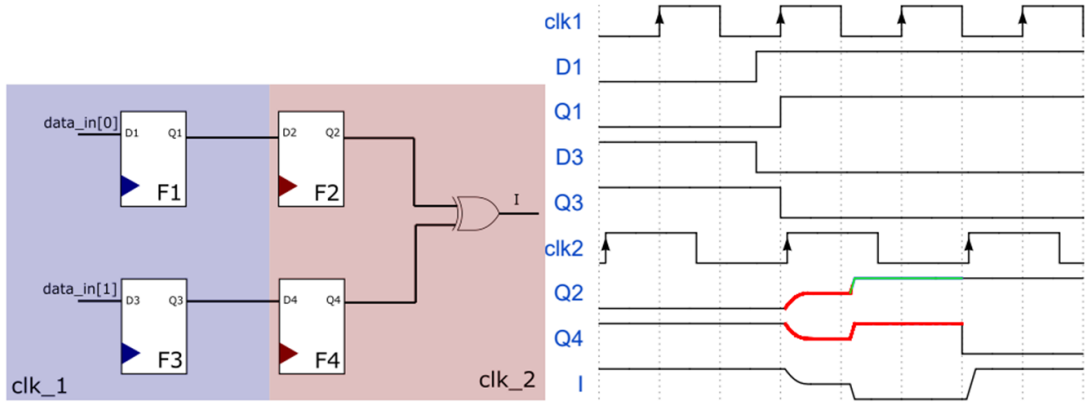
  - For a typical .25um ASIC technology, T_0=9.6nS,τ=0.31nS, and for T_r=2.3nS,F_c=100Mhz and F_d=1Mhz, the MTBF=20.1 days.
  - When using a 2-flop synchronizer, the MTBF at the output of the 2nd flop will be 9.57*10^10years.
  - Add a 3rd DFF for ~1/p^3
  - One bit matches cycle *n* or cycle *n+1* by coincidence

4. Synchronization (2/3)

4.1 Multi-flop synchronizers



- With multiple bits, metastability is still addressed but data coherency is a problem!
  - If multiple bits change on the same cycle, the result of each bit is random
  - This synchronization works only if the data is “gray” (only one bit changes)

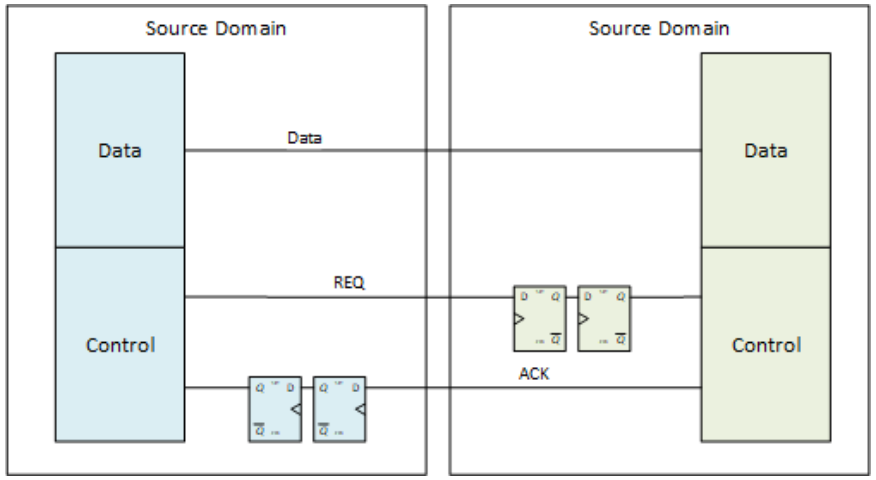


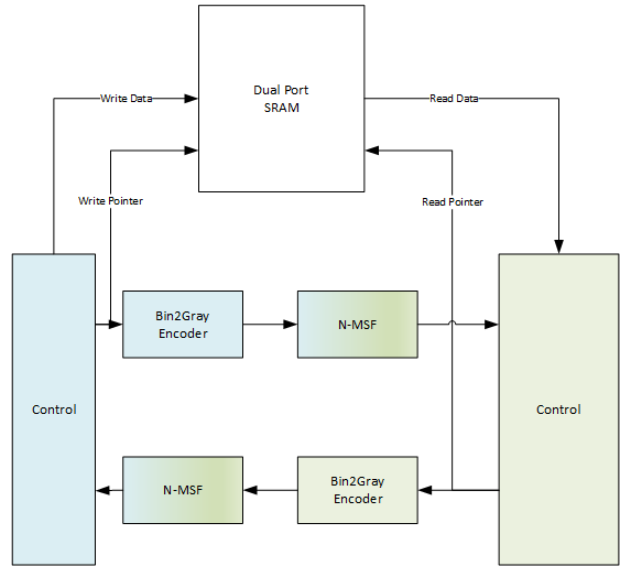
(a) Structure *(b) Waveform*

4. Synchronization (3/3)

4.2 Protocol-based synchronizers

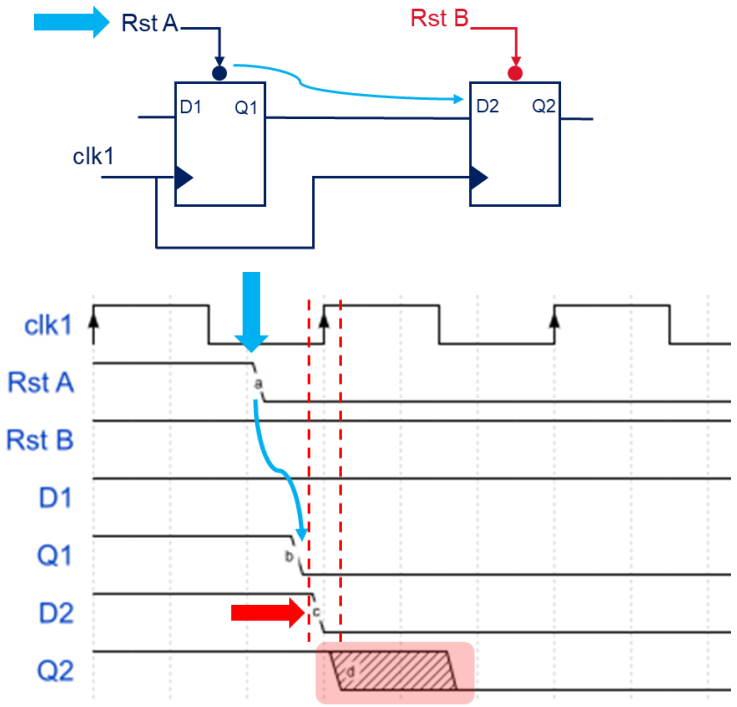
- Simple Qualifier
- Handshake protocol
- FIFO
  - Increased bandwidth
  - Throttling
  - Handles intermittent peaks of incoming data rate





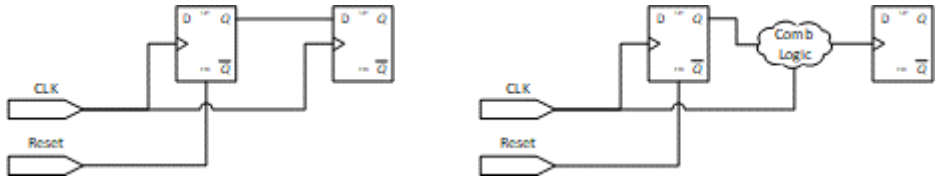
5. Asynchronous Reset Release

- In addition to setup and hold, DFF models also have **recovery time**
  - Time between asynchronous Set/Reset release and clock when data and output are different
  - Violating recovery time is no different than violating setup/hold
- Possible to synchronize asynchronous reset on release edge only
  - Static analysis is sufficient to make this determination

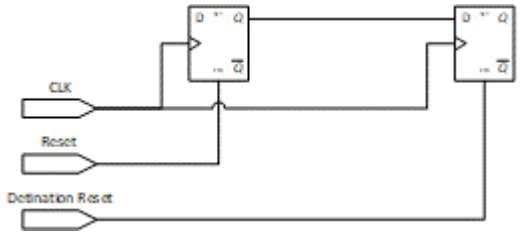


6. The Reset assertion RDC problem

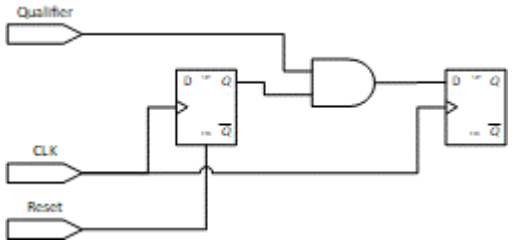
- Paths passing from CLR to Q are usually not timing closed



- Using reset ordering



- Using a qualifier
  - Qualifier must be synchronous with target domain



2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

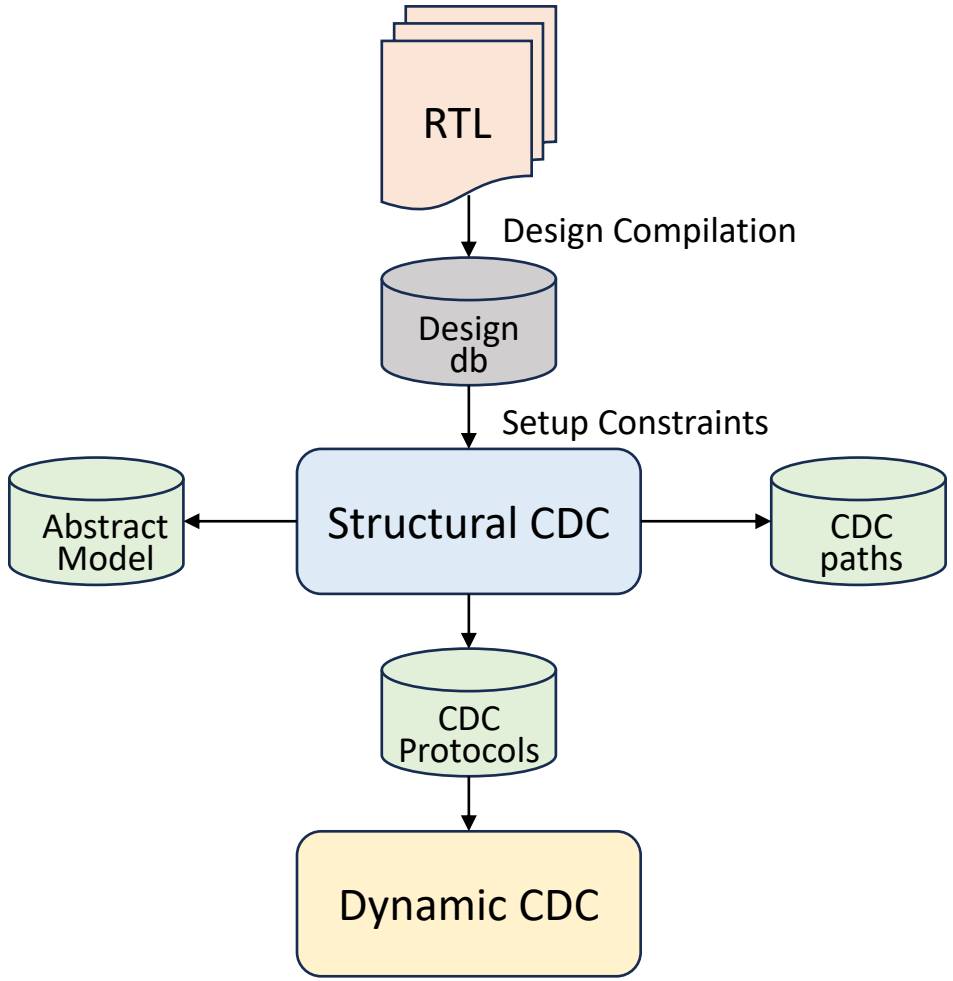
CDC verification on RTL

- Setup generation
- CDC setup check
- CDC structural checks
- CDC assertions-based verification
- Hierarchical CDC structural verification

accellera SYSTEMS INITIATIVE

1. CDC Verification flow

- Design Compilation
  - Parameters, defines
  - SV packages, SV configuration, SV interfaces
- Setup Constraints
  - Clock, reset, and IO signals
  - Configuration: stable, constant inputs
- Structural CDC Check
  - CDC schemes validation and debugging
- Abstract Model Generation
- Dynamic CDC Verification
  - CDC constraints and protocols



2. Setup Constraints

- The set of constraints used to guide CDC verification

- Clocks
- Resets
- Configuration signals
- Black boxes
- Primary inputs/outputs

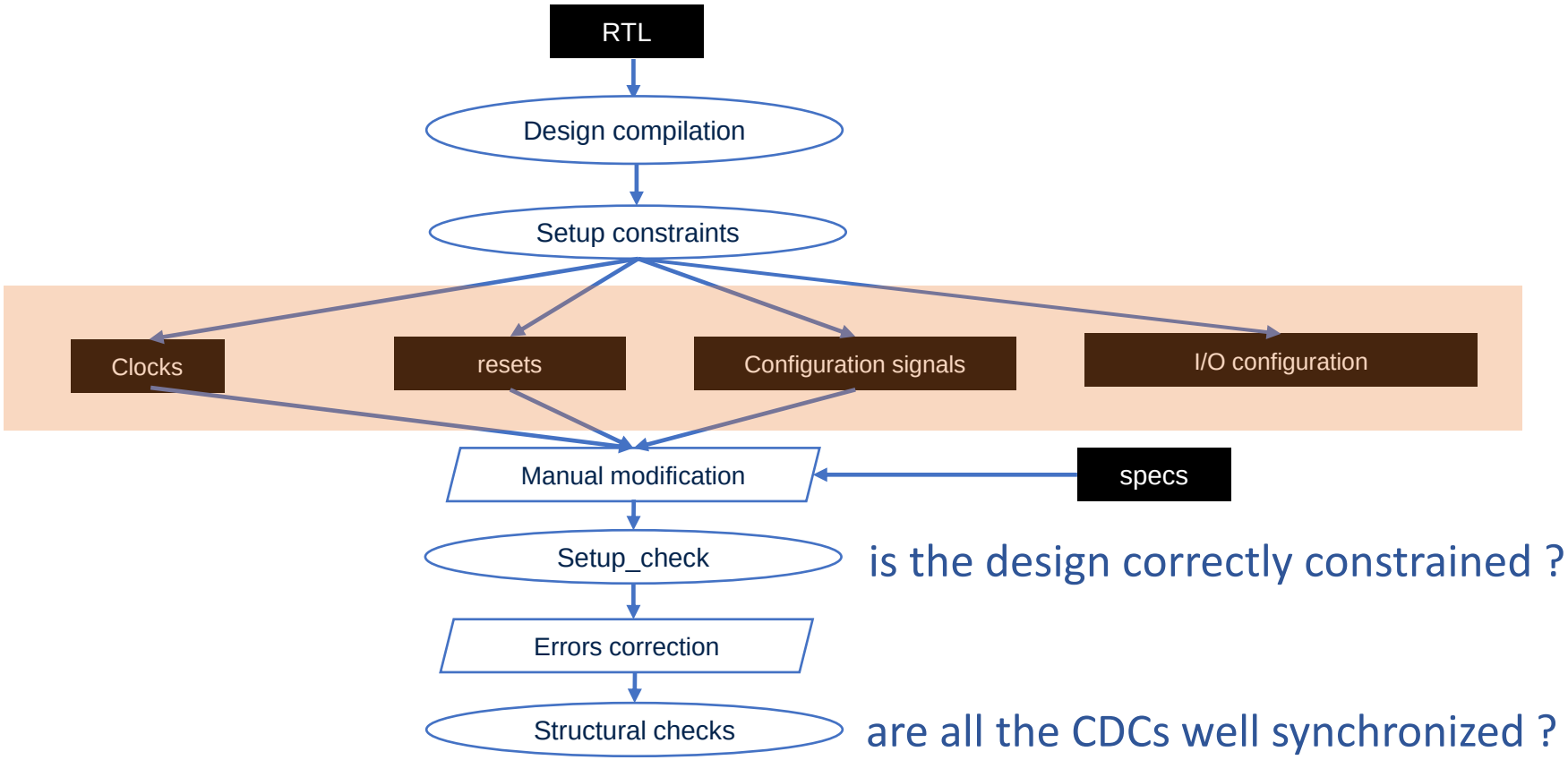
Don’t rely blindly on timing constraints

Reuse timing constraints is risky

- Pseudo-static signals
- Exclusive signals
- Gray coded buses
- Custom synchronizers
- False path

Clock groups for timing analysis =/= Clock groups for CDC analysis
Signal paths waived for time analysis =/= Signal paths waived for CDC analysis

1. Structural CDC Analysis



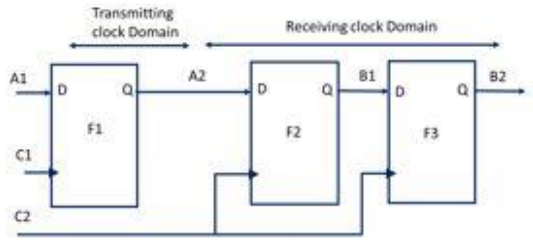
3. Dynamic CDC Verification

- Dynamic verification is to ensure
  - structural CDC check is done with the proper constraints and assumptions
  - the identified CDC paths follow the protcols defined by the CDC schemes

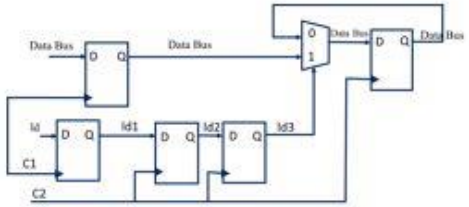
- CDC constraint properties
  - Assertions are generated based on the setup constraints
  - Ideally, should be done concurrently with structural CDC check
  - Violations can potentially invalidate the complete structural CDC
- CDC protocol properties
  - Assertions are generated based on the CDC paths
  - Violations can potentially invalidate the CDC paths

4. Structural CDC (1/9)

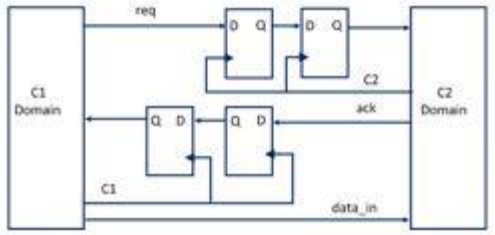
Commonly Used Synchronization Schemes



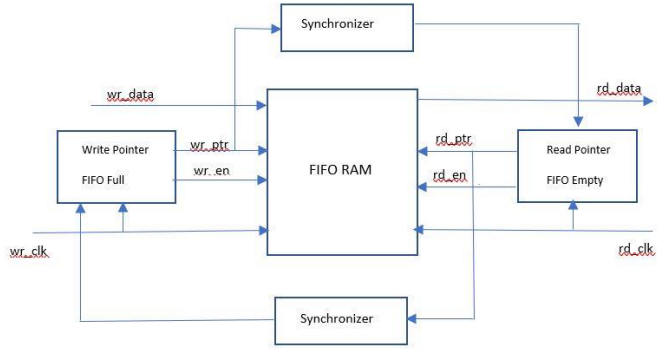
Double-FF synchronizer



MUX synchronizer



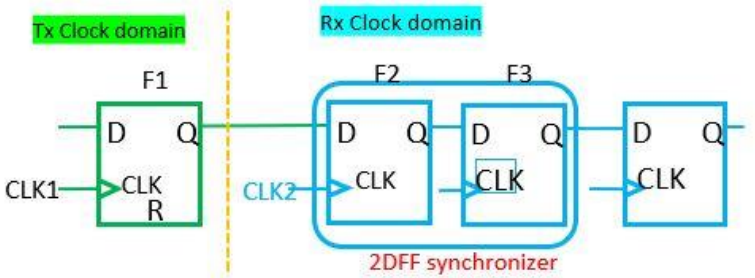
Handshake synchronizer



FIFO synchronizer

4. Structural CDC (2/9)

- Double FF Synchronizer
  - Most design houses prefer to use their own CDC components
    - Disable automatic detection of the specific synchronizer type that you don't want the tool to recognize automatically
    - Declare your own scheme as user-defined synchronizer (before scheme detection)
  - Example: Use my own 2DFF only
    - Disable auto-detection of 2DFF
    - Declare your own module as 2DFF



4. Structural CDC (3/9)

- Various Signal Configurations possible for structural CDC Analysis
  - Constant
  - Static
  - Mutually exclusive / Gray code
  - Externally synchronized
  - CDC False paths
    - *Not recommended (avoid using it to mask real CDCs)*
- Purpose
  - Define signal behavior that can help to reduce CDC analysis noise
    - Exclude certain paths which may not have any standard synchronizer but safe for CDC
    - Helps to speed up CDC analysis

4. Structural CDC (4/9)

- CDC Constraints – Constant Declaration
  - It can be applied on a port or on an internal signal
  - A constant signal does not change in a given mode and hence does not cause a CDC issue.

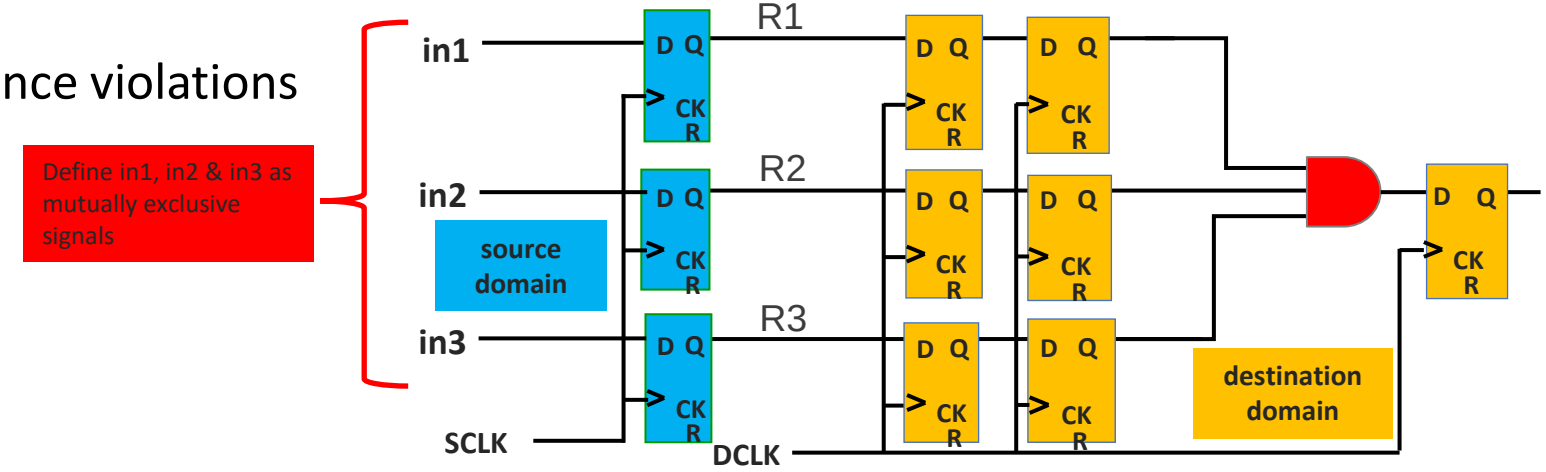
- Purpose
  - Define signal behavior that can help to reduce CDC analysis noise
    - Exclude certain paths which may not have any standard synchronizer but safe for CDC
    - Helps to speed up CDC analysis

4. Structural CDC (5/9)

- CDC Constraints – Static Declaration
  - Any signal that does not change while the destination is active
  - Same as quasi-static or pseudo-static
  - A static signal does not cause CDC issues because
    - The receiver clock is not active
    - The receiver is under reset

4. Structural CDC (6/9)

- CDC Constraints – Gray Coded Declaration
  - A bus can be specified as gray coded – Only one bit can toggle at a time
- CDC Constraints – Mutually Exclusive Toggle Declaration
  - A set of independent signals that can toggle only one at a time can be defined as mutually exclusive toggle
    - Helps in avoiding convergence violations

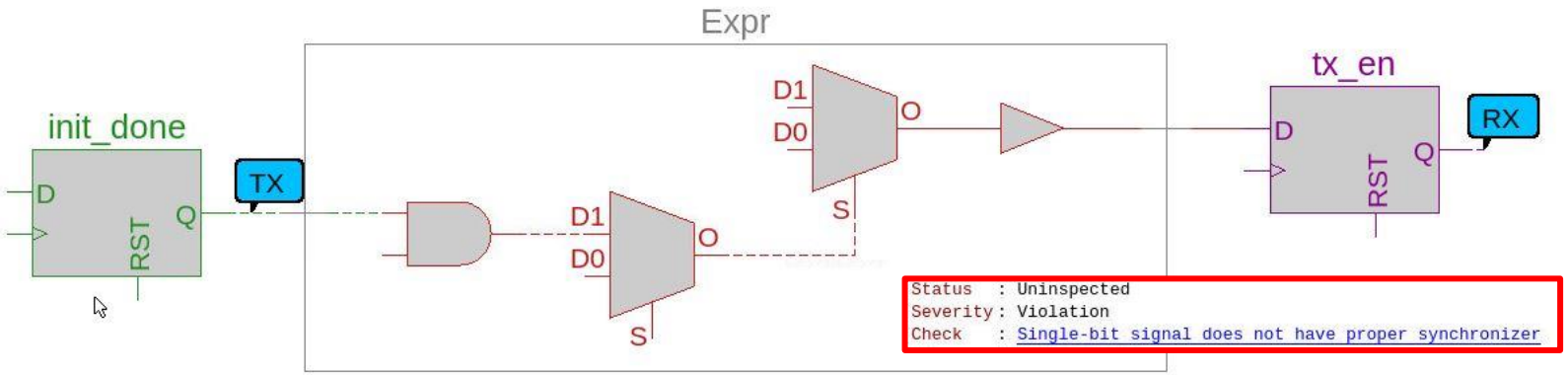


4. Structural CDC (7/9)

- CDC Constraints - Externally Synchronized
  - A block level input/output port can be declared as *externally synchronized*
    - Represents the output of a control synchronizer (2DFF/Edge/Pulse)
    - Can be used as the control path for complex synchronizers (MUX Synchronizer, Glitch Protector)
    - Helps in auto-detection of the above composite synchronization scheme types
- CDC Constraints - CDC False Path Declaration
  - CDC Checks can be disabled on certain paths by user-defined constraints
  - User can set a constraint to let the tool automatically identify a functionally false path and hence reports the path as a safe path

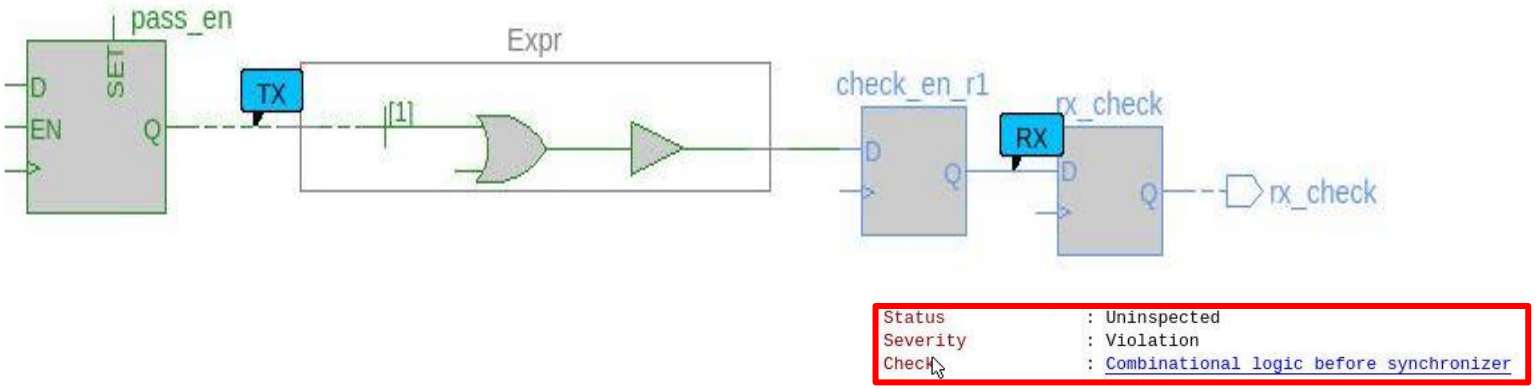
4. Structural CDC (8/9)

- Missing synchronizer on CDC path

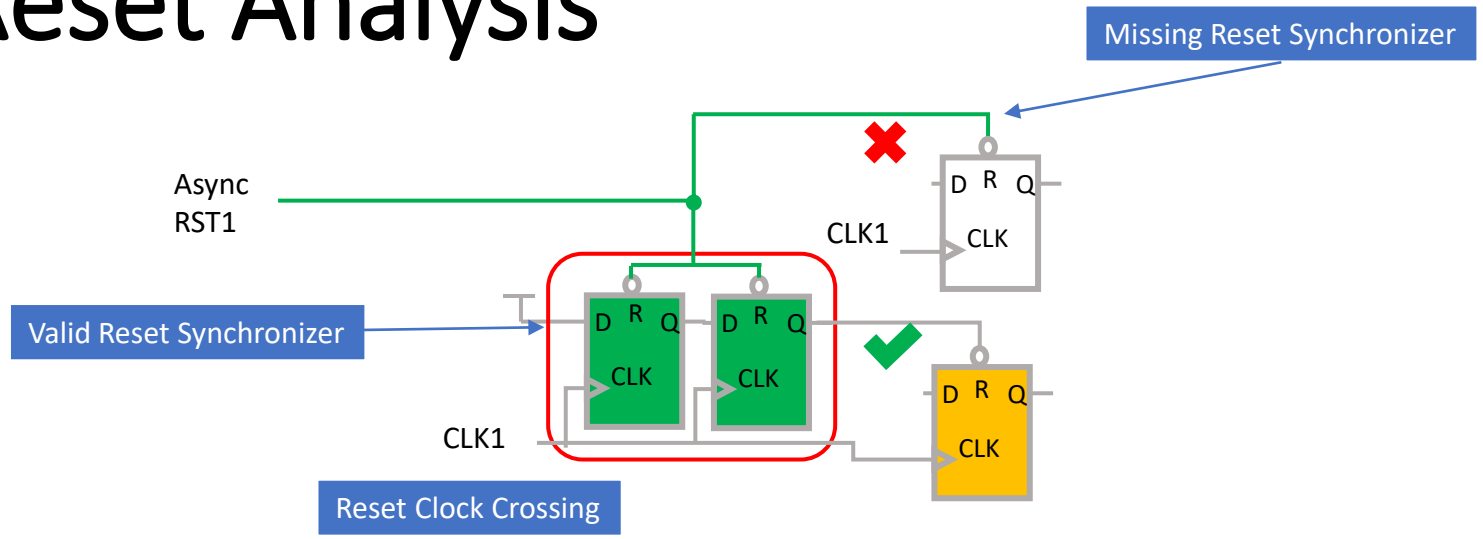


4. Structural CDC (9/9)

- Combo-logic before synchronizer on CDC path



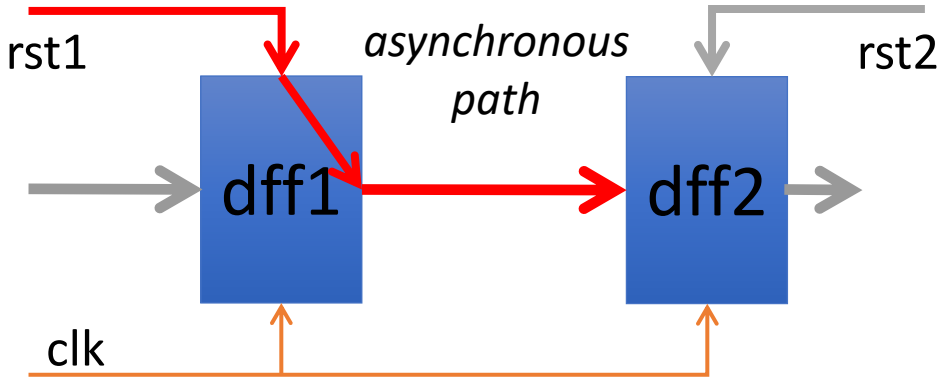
5. Reset Analysis



- Reset signal crossing from one clock domain to another
  - The asynchronous de-assertion of the reset signal at the destination flop can cause the signal to become metastable
  - Reset signals are required to be synchronized to destination domains for synchronous de-assertions

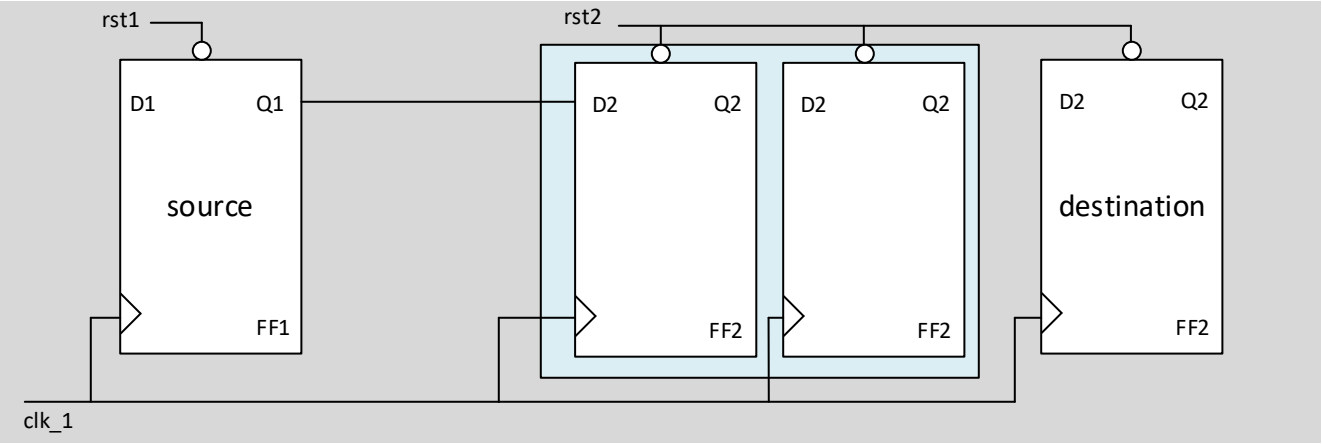
6. Reset-Domain Crossing (RDC)

- Asynchronous reset domains causes meta-stability
  - Contain registers whose resets are asserted asynchronously
  - Originate in one asynchronous reset domain
  - Sampled by register(s) in a different reset domain



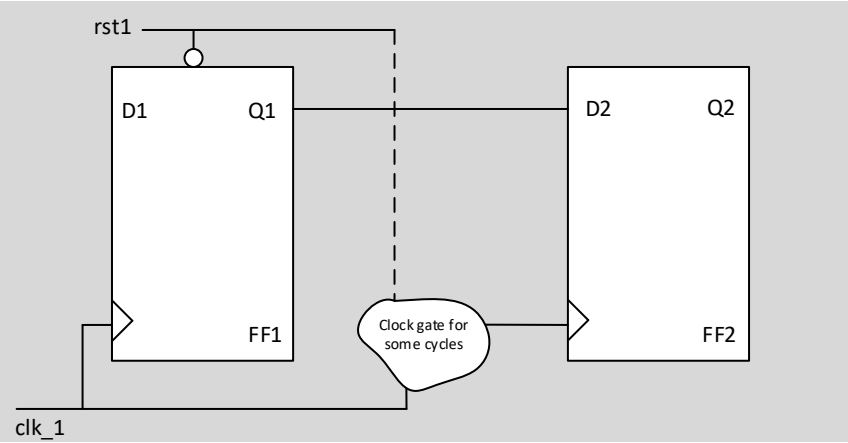
6. Reset-Domain Crossing

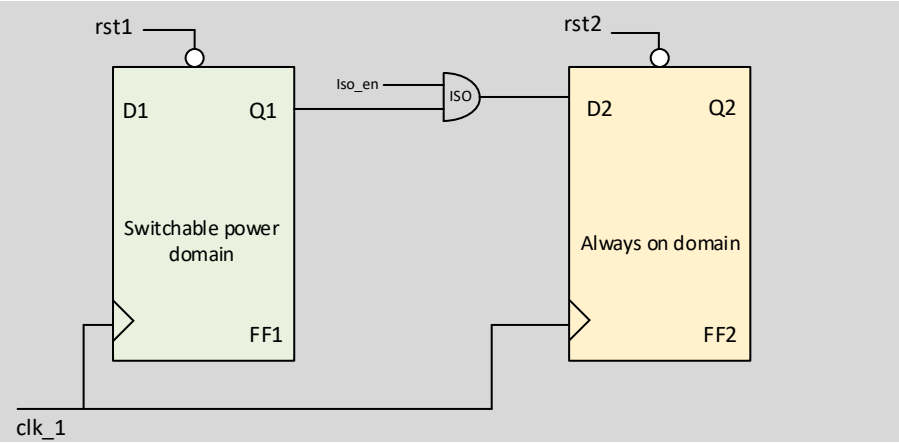
- Resolving RDC issues
  - Adding synchronizer in the destination domain
  - Reset ordering of different resets in the design



6. Reset-Domain Crossing

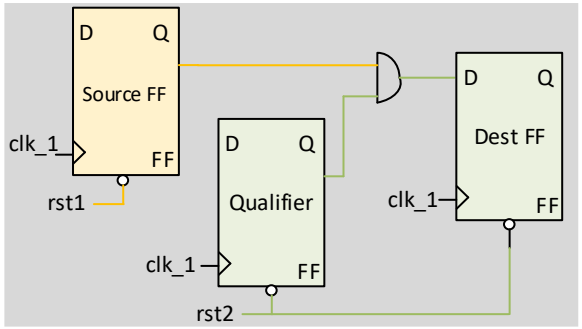
- Resolving RDC issues
  - Reset-less paths: Clock gate the downstream flops for some cycles when rst1 asserts to avoid glitch capture
  - Multipower domain designs : reset ordering, isolation en asserted



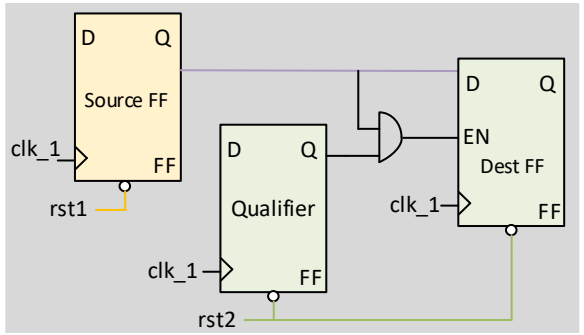


6. Reset-Domain Crossing

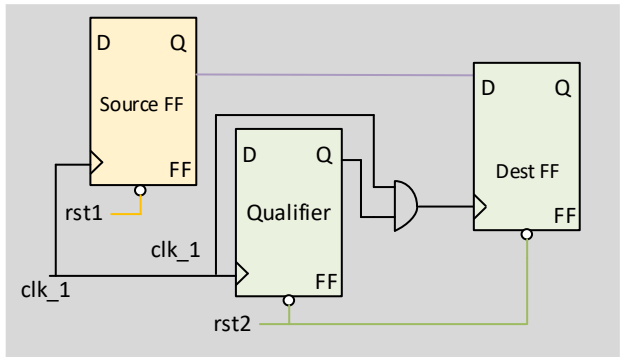
- Resolving RDC issues
  - Using Qualifiers
    - Data Qualifier
    - Enable Qualifier
    - Clock Qualifier



Data Qualifier



Enable Qualifier



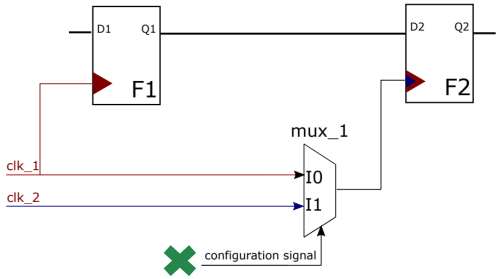
Clock Qualifier

Structural CDC/RDC - Limitations

1- Constraints based static checks

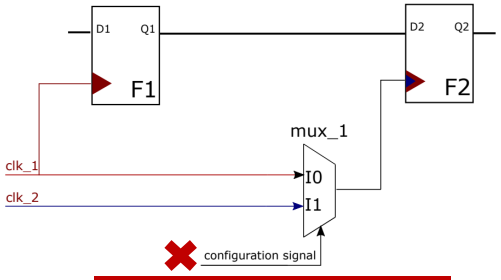
- Affect the results of the structural checks
- Are taken blindly for the structural verification
  - e.g., a CDC can be bypassed if the crossing signal is pseudo-static

✖ set_case_analysis 0 configuration_signal



Synchronous

✖ set_case_analysis 1 configuration_signal



Asynchronous

2- Rules based static checks

- Not possible to have rules for all architectures
- False positives / negatives
- Cannot verify correctness of design

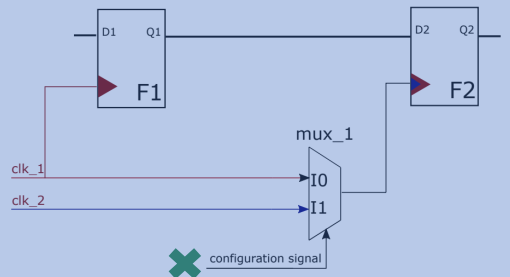
Overcoming Limitations – Assertions Based Verification

1- Constraints based static checks

- Affect the results of the structural checks
- Are taken blindly for the structural verification
  - e.g., a CDC can be bypassed if the crossing signal is pseudo-static

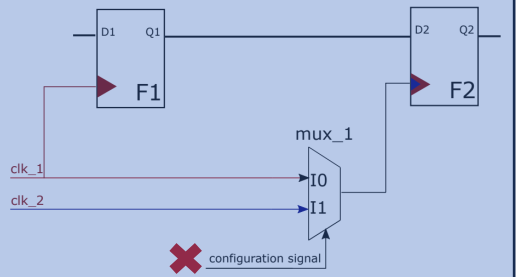
Assumptions

✖ set_case_analysis 0 configuration_signal



Synchronous

✖ set_case_analysis 1 configuration_signal



Asynchronous

2- Rules based static checks

- Not possible to have rules for all architectures
- False positives / negatives
- Cannot verify correctness of design

Assertions

Overcoming Limitations – Assertions Based Verification

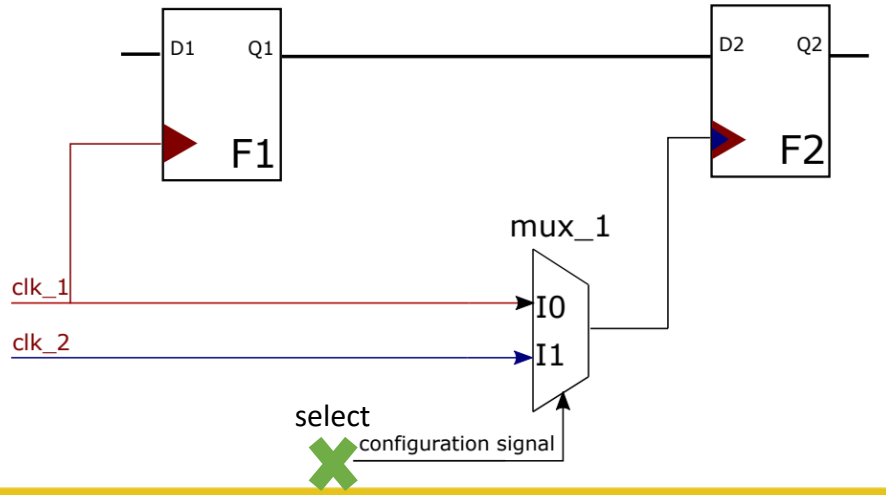
1- Constraints based static checks

- Constraints to be double checked with the functional verification

- Configuration signals

define_constant –value [0/1] –signal [signal name]

```
always@*
 begin
  assert_cdc_constant_prop : assert (select === value)
```



Overcoming Limitations – Assertions Based Verification

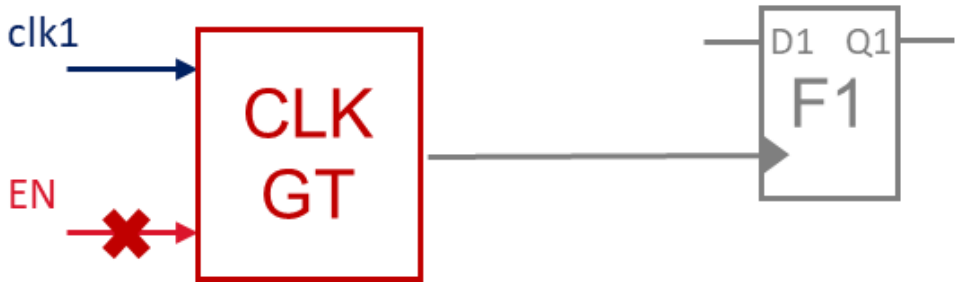
1- Constraints based static checks

- Constraints to be double checked with the functional verification

- Pseudo-static signals

```
define_pseudostatic –name [signal name] –stopped_clock [yes/no] –under_reset [yes/no]
```

```
property pseudo_static (EN);
   @(posedge clk) disable iff(reset)
     nexttime $stable(EN);
 endproperty
```



Overcoming Limitations – Assertions Based Verification

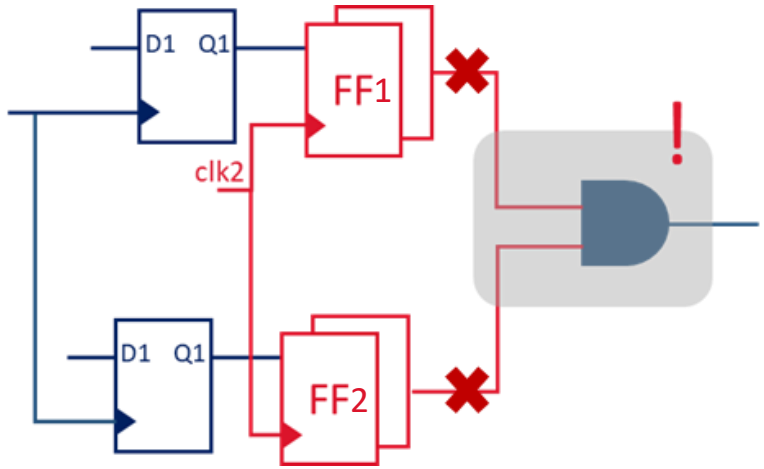
1- Constraints based static checks

- Constraints to be double checked with the functional verification

- Mutually exclusive

define_exclusive –signals [set of signals names]

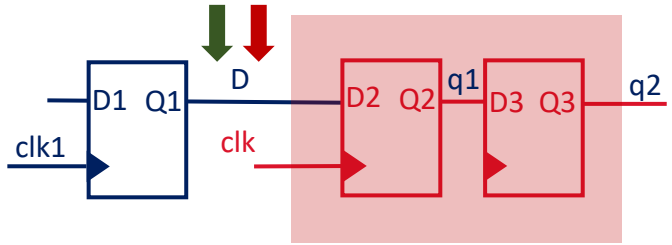
```
property mutex (data, clk);
   @(posedge clk)
    $onehot0(data ^ $past(data));
 endproperty
```



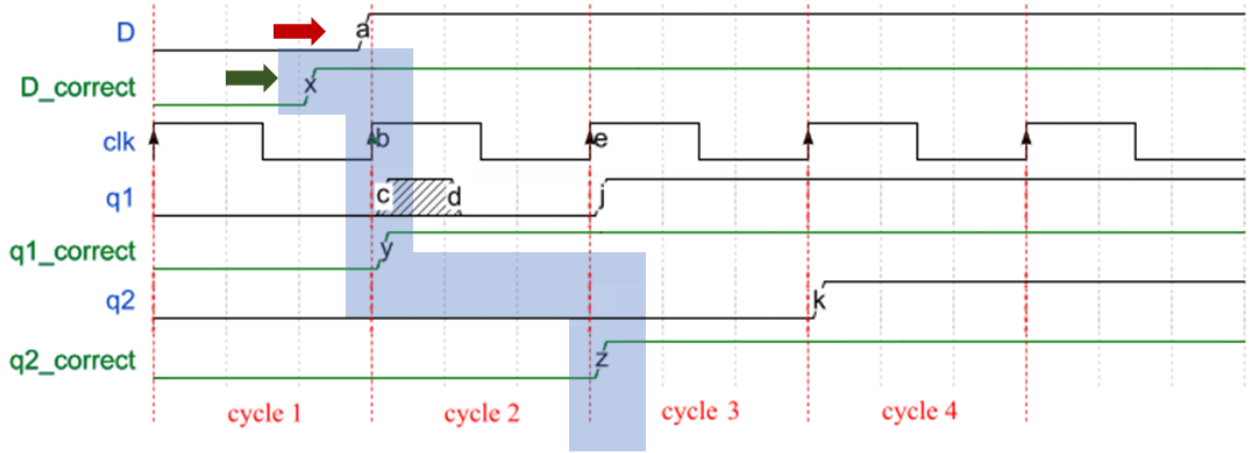
Overcoming Limitations – Assertions Based Verification

2- Rules based static checks

- Fundamentally target to verify **design intent**
- CDC paths are **not covered by STA** :
  - MFS : Make sure source data is stable while crossing.




```
property cdc_data_stable (D, clk2, reset, areset,
NUM_CYCLES);
    @(posedge clock) disable iff(areset)
    ##1 !reset && $changed(D) |=>
$stable(Q1)[*(NUM_CYCLES-1)];
 endproperty
```



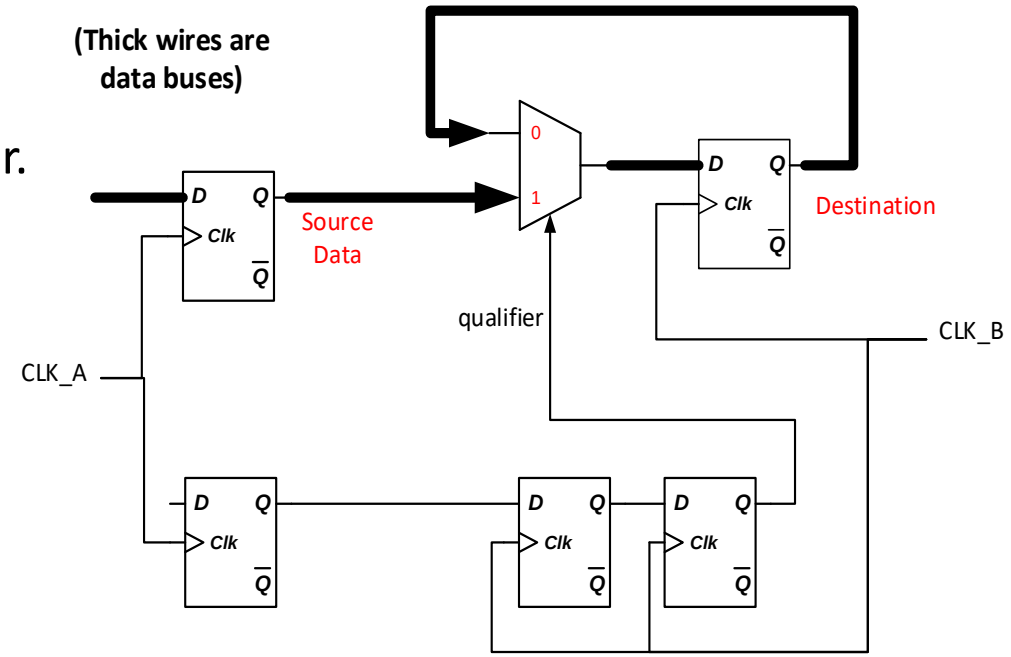
Overcoming Limitations – Assertions Based Verification

2- Rules based static checks

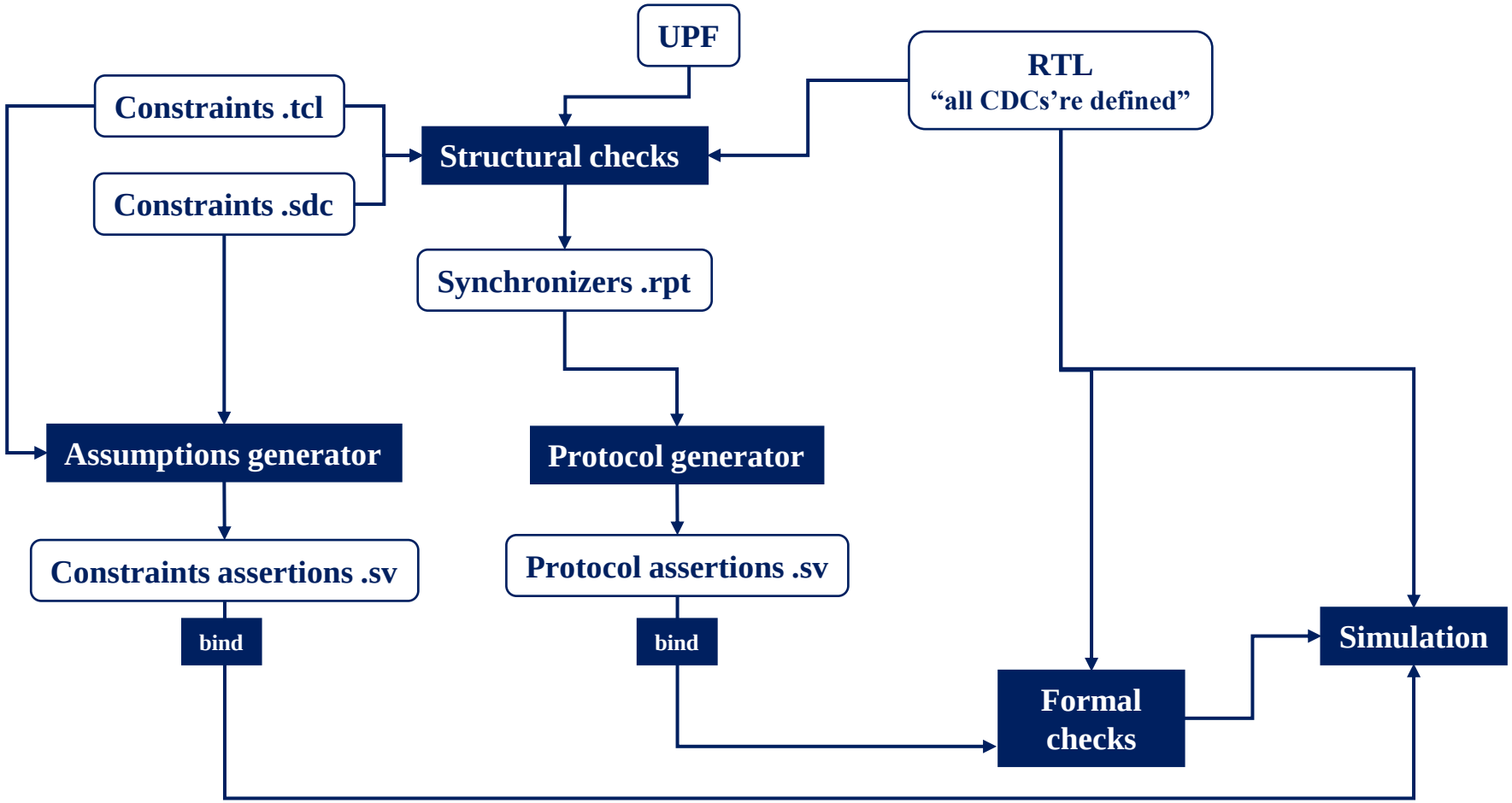
- Fundamentally target to verify **design intent**
- CDC paths are **not covered by STA** :
  - MFS : Make sure source data is stable for several cycles.
  - Enabler : Make sure source data is stable wrt to its enabler.

Source data must be static when qualifier is enabling.
Source data can toggle when qualifier is disabling.
Qualifier must be a known value

```
property qual_data_stable (clock, reset, areset, data,data_select);
  @(posedge clock) disable iff(areset)
   !reset && qualifier |=> $stable(data);
endproperty
```



ABV - Assertions Based Verification



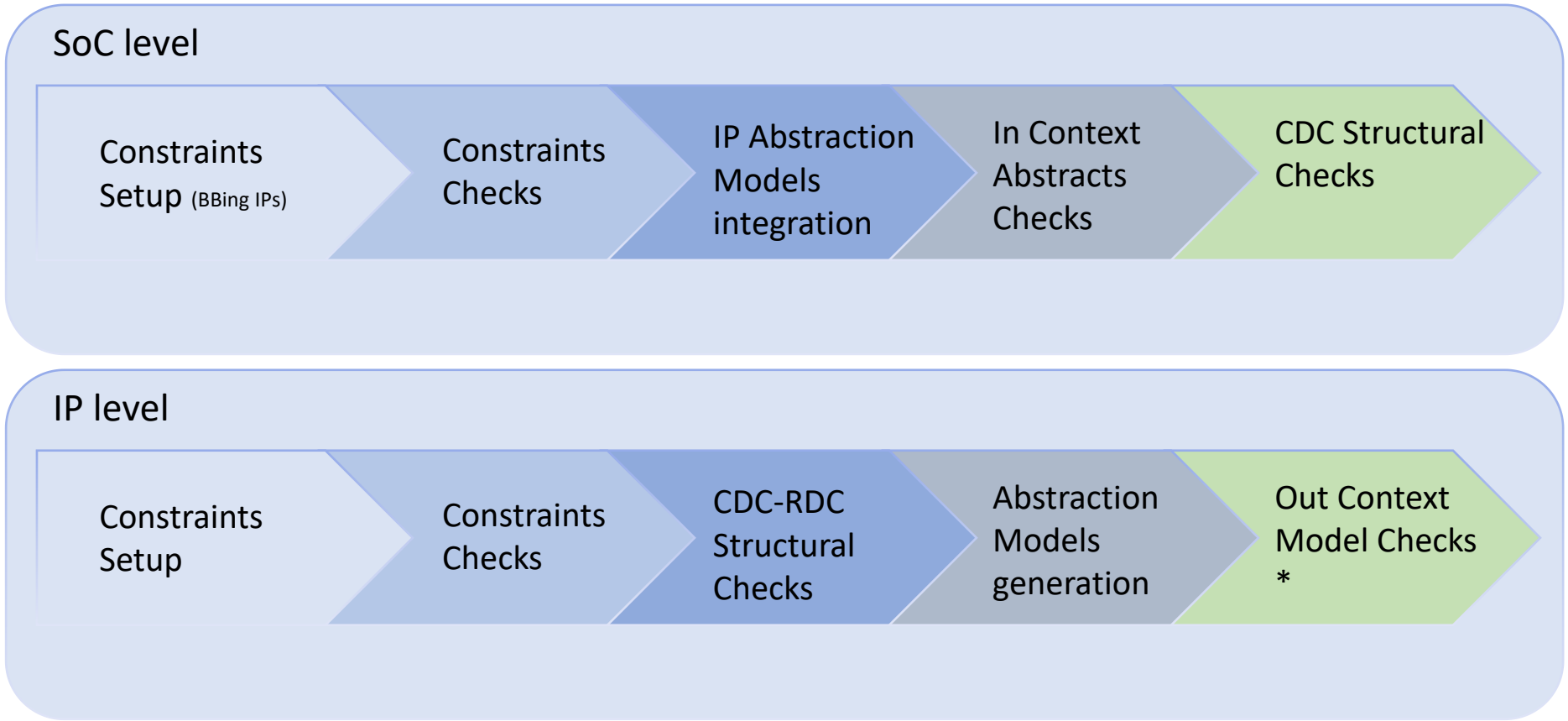
9. CDC-RDC Hierarchical Flow (1/4)

9.1 Why ?

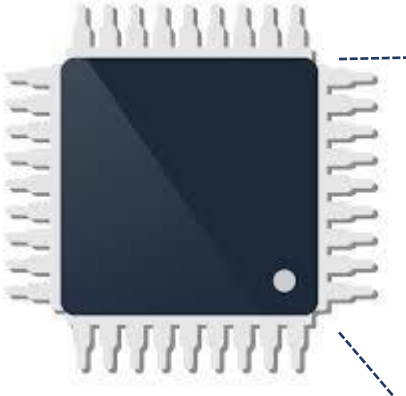
- CDC-RDC Verification at SoC level
  - Flat RTL analysis only possible for small SoC and/or SoC simple clock-reset strategy
- Hierarchical strategy means first identify sub-blocks to be analyzed separately then modeled
  - Each CDC-RDC model being integrated at SoC level
- Allows parallelization of sub-blocks analysis and noiseless analysis at SoC level
- Challenges:
  - Dependency to sub-blocks provider to deliver CDC-RDC model
  - Compatibility of CDC-RDC models in case of multiple EDA tools usage

9. CDC-RDC Hierarchical Flow (1/4)

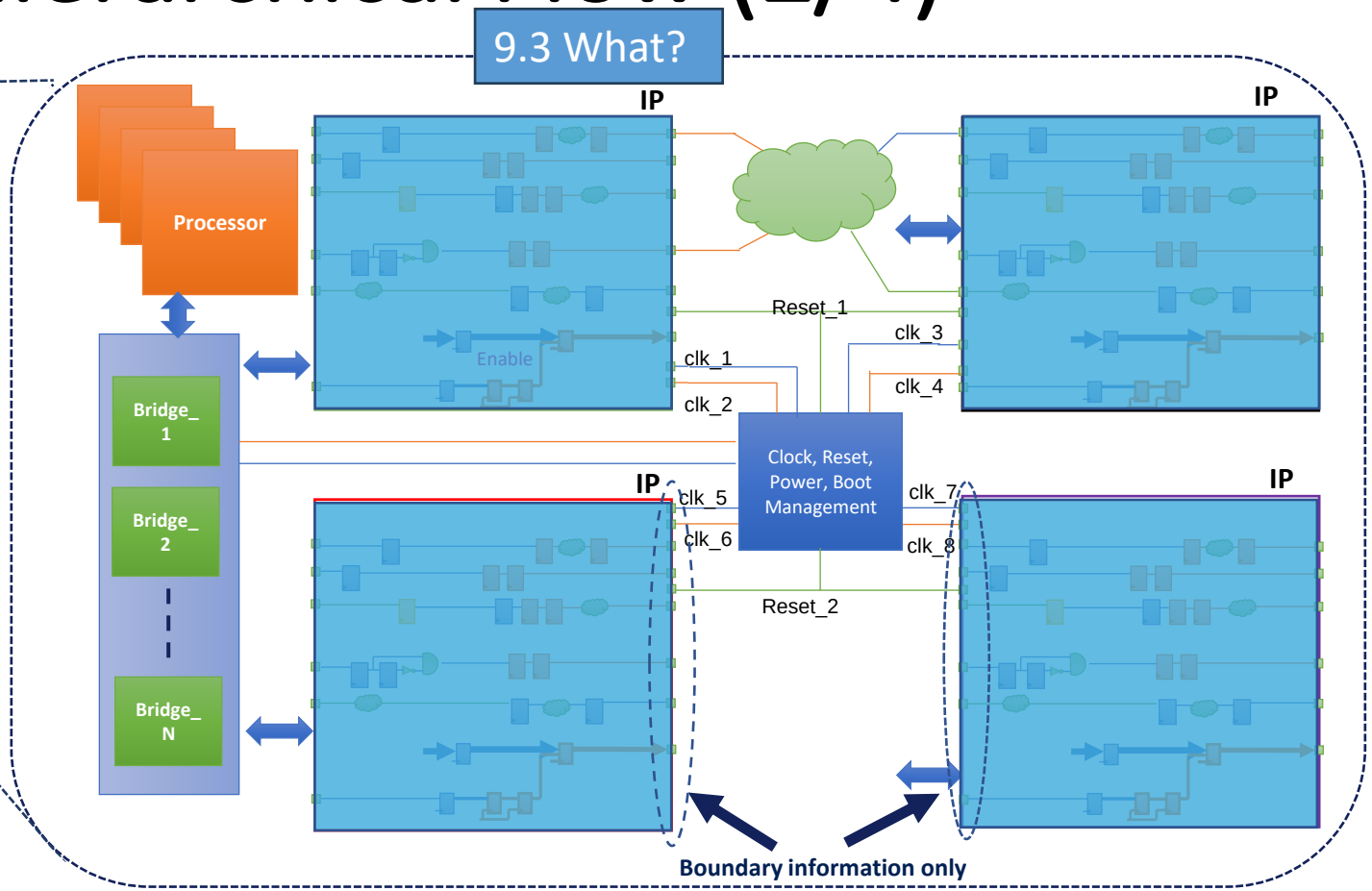
9.2 How ?



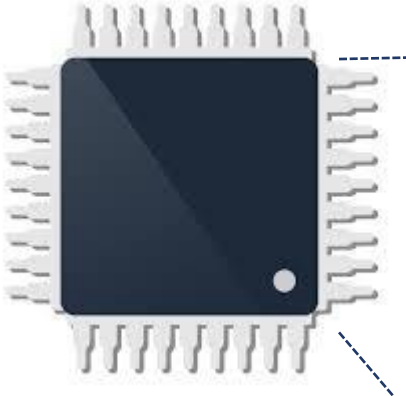
9. CDC-RDC Hierarchical Flow (2/4)



- Abstraction Models enable all Boundary related CDC-RDC info required at SoC integration Checks
- Much better approach compared to Black Box

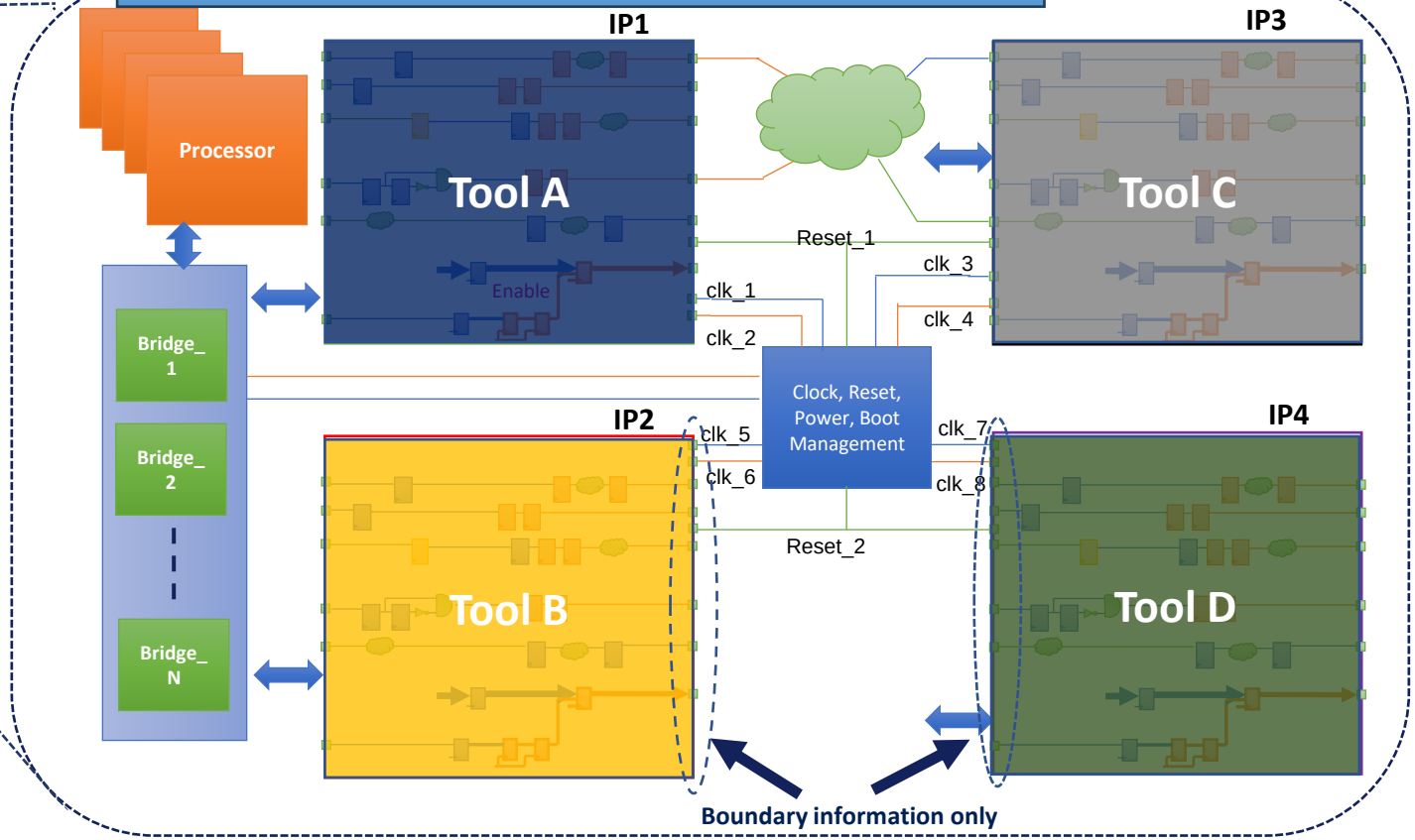


9. CDC-RDC Hierarchical Flow (3/4)



- CDC models are currently lacking standardization.
- CDC models from different tools are not compatible.

9.4 CDC models from different tools



2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SAN JOSE, CA, USA
MARCH 4-7, 2024

Accellera CDC Working Group

- Presentation
- The five sub working groups
- Call for contribution

accellera SYSTEMS INITIATIVE

What was the problem?

- As we move from monolithic designs … to IP/SOC with IPs sourced from a small/select providers … to sourcing IPs globally (to create differentiated products) …
- We must maintain quality as we drive faster time-to-market
- In areas where we have standards (SystemVerilog, OVM/UVM, LP/UPF), the integration is able to meet the above (quality, speed)
- But in areas where we don’t have standards (in this case, CDC), most options trade-off either quality, or time-to-market
- Creating a standard for inter-operable collateral addresses this gap

Accellera CDC WG initiative

- Pre-WG launched Sep ‘22 to evaluate need. WG launched Jan ’23
  - 123 members from 23 companies (as of Jan 29 ‘24)
  - 5 active sub-groups: Format, Output-Collateral, Assertions, Testing, Training
- Timeline
  - Phase1: CDC: Oct ‘23 to Feb ‘24 release
  - Phase2: RDC & Assertions: Jun ‘24 to Aug ‘24 release
  - Phase3: Complexities & Extensions: Dec ‘24 to Feb ’25 release
  - Complete/Final LRM release: **Mar ‘25**

Agnisys	Aldec	AMD	AMS	Analog Devices	ARM	Arteris	Blue Pearl Software
Cadence	Infineon	Intel	Marvel	Microsoft	Nvidia	NXP	Qualcomm
Renesas	Robert Bosch	Siemens	ST Micro	Synopsys	Texas Instruments	Verilab	

Accellera CDC WG Scope

- Tool-agnostic interoperable collateral
- Supporting hierarchical CDC/RDC/Glitch structural analysis
- Human readable, and machine parseable
- LP/UPF compliant
- Multi-mode/param/instance compppliant
- Covering majority of common interface protocols (e.g. AMBA, UCIe, etc.)
- Constraints/Assumptions can be verified with SVAs
- Can meet the needs of FPGA and Analog blocks

Accellera CDC WG initiative

Format

- Sub-working groups presentation

- **Goal**

  1. Determine exact format for domain specific language that can be used to capture required attributes/data from input/output/verification collaterals.
  2. Ensure quality in terms of compliance to spec.

- **Methodology**

  1. List different options like IP-XACT, TCL, Excel, JSON, etc.
  2. Experiment with populating the formats to ascertain the ability to meet the requirements.
  3. Determine pros and cons for each option of format.
  4. Recommend a final format post CDC Workgroup approval

Accellera CDC WG initiative

Format

- A limited feasibility study for CDC
  - conducted on a subsystem with multiple IPs connected by AMBA interfaces
  - across three different vendor tools
  - limited support from the vendors
- Results:
  - 99.5% of what was identifiable in a flat run was also identifiable if the native abstraction collateral was replaced with an XML representation and translated across the vendor tools.

Accellera CDC WG initiative

Format

- Describing IP
  - static or semi-static
  - IP-XACT is industry standard for IP definition and packaging
  - Use models of IP and Product companies
- Integration of IP
  - Dynamic environment requires programmability for CDC definition
  - Tcl is preferred and widely used in industry
  - Use models for Product and EDA companies

Accellera CDC WG initiative

Format

- IP-XACT vs Tcl
  - IP-XACT is perfect for static representation
  - Useful for IP Delivery and SoC Integration
  - Infrastructure required for converting existing proprietary formats to IP-XACT
  - Tcl handles dynamic and conditional CDC scenarios better
  - EDA companies currently supports proprietary formats that are Tcl like
  - Human readability issue

- CDC Workgroup voted to use combination of both Tcl and IP-XACT

Accellera CDC WG initiative

Format

- Format Subgroup is working on
  - Defining API commands for Tcl
  - Defining schema for IP-XACT as extensions to the standard
    - Power is part of standard
    - CDC can be adopted as part of the standard by the IP-ACT WG

Accellera CDC WG initiative

Format

- EDA companies to provide transformers for Tcl to/from IP-XACT
  - Also provide translators to and from its native format from and to the standard format
- Standard is tool agnostic
- IP providers have option to choose tools
  - to verify and produce collateral
  - to generate the standard format for SoCs that use a different tool

Accellera CDC WG initiative

Output-Collateral

- Sub-working groups presentation

- **Goal**

  1. Set of attributes for CDC-RDC model block
  2. The set of attributes are to be sufficient to address a specific pre-defined set of known industry standard interfaces and be able to objectively test the completeness and claim usefulness of the defined set of attributes for those interfaces.
  3. Define the limitation of said set of attributes, by identifying the CDC and RDC schemes for which the set of attributes is sufficient and the CDC and RDC schemes for which the defined set of attributes is not guaranteed to support.

Summary of Changes To Attribute Table

Old

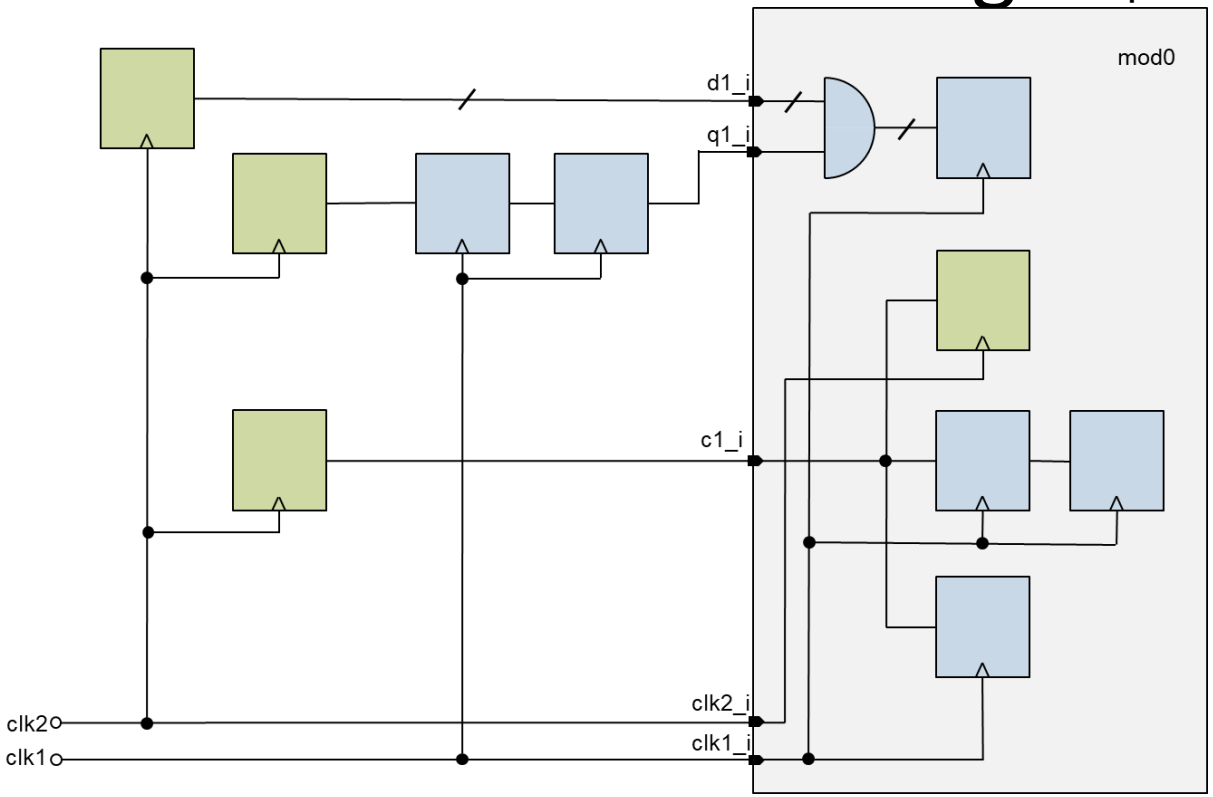
#	Attribute	Switch	data_type	values
1.00	module	name	string	{module name}
2.00	parameter	name	string	
2.01	parameter	value	range-list	{ values }
2.02	parameter	type	string	{ int, string, boolean }
2.03	parameter	ignore	string	{ignore}
3.00	port	name	string	{ name }
3.01	port	direction	defined set	{ input, output, inout, internal }
3.02	port	type	defined set	{ data, clock, async reset, qualifier, supply, analog }
3.03	port	logic	defined set	{ combo, buffer, inverter, internal, glitch-free-combo, internal-sync }
3.04	port	qualifier_from_clock	; separated list	{ clock-names }
3.05	port	associated_clocks	; separated list	{ clock-names }
3.06	port	associated_resets	; separated list	{ reset-names }
3.07	port	associated_inputs	; separated list	{ ports }
3.08	port	associated_outputs	; separated list	{ ports }
3.09	port	clock_group	string	{clock group}
3.10	port	reset_group	string	{reset group}
3.11	port	qualifiers	; separated list	{ associated-ports }
3.12	port	polarity	defined set	{ high, low, both }
3.13	port	ignore	string	{ blocked, hanging }
3.14	port	quasi_static	; separated list	{ be any, <clocks to which it is quasi_static> }
3.15	port	set_case_analysis	; separated list	{binary, hex, and of any length}
3.16	port	gray_coded	boolean	{ true, false:default }
3.17	port	clock_period	string	{clock period}
4.00	tool	name	string	{EDA name}
4.01	tool	version	string	{Tool Version}
5.00	design	version	string	{Design Milestone}
5.01	design	date	string	{Collateral Generation Date}
5.02	design	username	string	{user/tool who generate the collateral}
5.03	design	description	string	
6.01	set_cdc_clock_group	async	string	–group {logical_ck1, logical_ck2} | | –group {logical_ck1} –group {logica
6.01	set_cdc_clock_group	sync	string	–group {logical_ck1, logical_ck2} | | –group {logical_ck1} –group {logica

Existing

	Attribute	Switch	data_type	values
1.00	module	name	string	{module name}
2.00	parameter	name	string	{parameter name}
2.01	parameter	value	range-list	{ values }
2.02	parameter	Type	defined set	{ string, boolean, number [hex, decimal, oct, binary] }
2.03	parameter	ignore	boolean	{True or False}
3.00	port	name	string	{port name}
3.01	port	direction	defined set	{ input, output, inout }
3.02	port	type	defined set	{ data, clock, async reset, cdc_control }
3.03	port	logic	defined set	{ combo, buffer, inverter, glitch-free-combo, internal-sync }
3.04	port	cdc_control_from_clock	string	{ clock-name }
3.05	port	associated_from_clocks	; separated list	{ clock-names }
3.06	port	associated_to_clocks	; separated list	{ clock-names }
3.07	port	associated_resets	; separated list	{ reset-names }
3.08	port	associated_inputs	; separated list	{ ports }
3.09	port	associated_outputs	; separated list	{ ports }
3.10	port	clock_group	string	{clock group}
3.11	port	reset_group	string	{reset group}
3.12	port	cdc_control	; separated list	{ associated-ports }
3.13	port	polarity	defined set	{ high, low, **low_high** }
3.14	port	ignore	defined set	{ blocked, hanging }
3.15	port	cdc_static	; separated list	{ be any, <clocks to which it is quasi_static> }
3.16	port	constant	; separated list	{binary, hex, and of any length}
3.17	port	gray_coded	boolean	Deffer to LRM 0.2
3.18	port	clock_period	string	Deffer to LRM 0.2
4.00	tool	name	string	{tool name}
4.01	tool	version	string	{Tool Version}
5.00	design	version	string	{Design Milestone}
5.01	design	date	string	{Collateral Generation Date}
5.02	design	username	string	{user/tool who generate the collateral}
5.03	design	description	string	
6.01	set_cdc_clock_group	clocks	string	; separated list{clock-names}
6.02	set_cdc_clock_group	name	string	{group-name]

1. Parameter
2. Port Attribute Modelling
3. Removal of keywords that are not EDA-Tool-Agnostic
4. Removal of clock_group and reset_group
5. Set_cdc_clock_group
6. Adding more examples to describe usage of reset, polarity, ignore,cdc_static and constant
7. Deferring usage of gray_code and clock_period to LRM 0.2

Port Attribute Modelling - input interface



Ext: port -name d1_i -type data -direction input -associated_from_clock clk2_i
Int: port -name d1_i -associated_to_clock clk1_i -logic combo -cdc_control q1_i

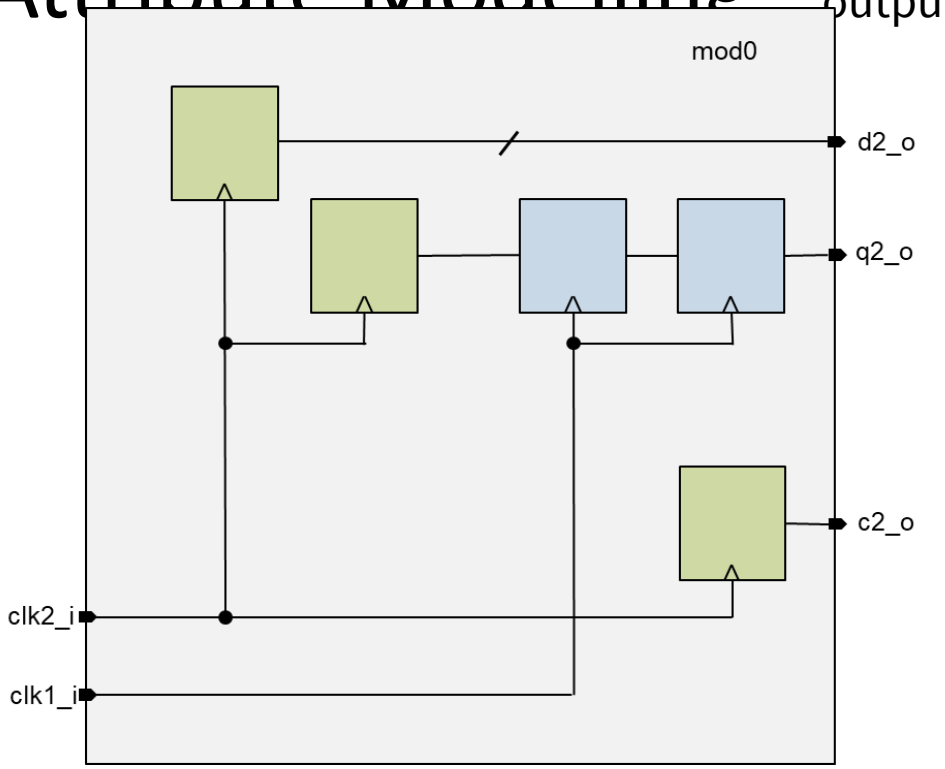
Ext: port -name q1_i -type cdc_control -direction input -cdc_control_from_clock clk2_i -associated_to_clock clk1_i -associated input d1_i
Int: port -name q1_i -logic combo

Ext: port -name c1_i -type data -direction input -associated_from_clock clk2_i
Int: port -name c1_i -associated_to_clock clk2_i
port -name c1_i -associated_to_clock clk1_i -logic internal_sync
port -name c1_i -associated_to_clock clk1_i

Summary of Changes

1. Ext View (the driver) is mandatory
2. Int is optional
3. One port can have multiple lines
4. Uses of EDA-Tool-Agnostic keyword
5. Associated_clocks becomes associated_from_clocks and associated_to_clocks

Port Attribute Modelling - output interface



Summary of Changes

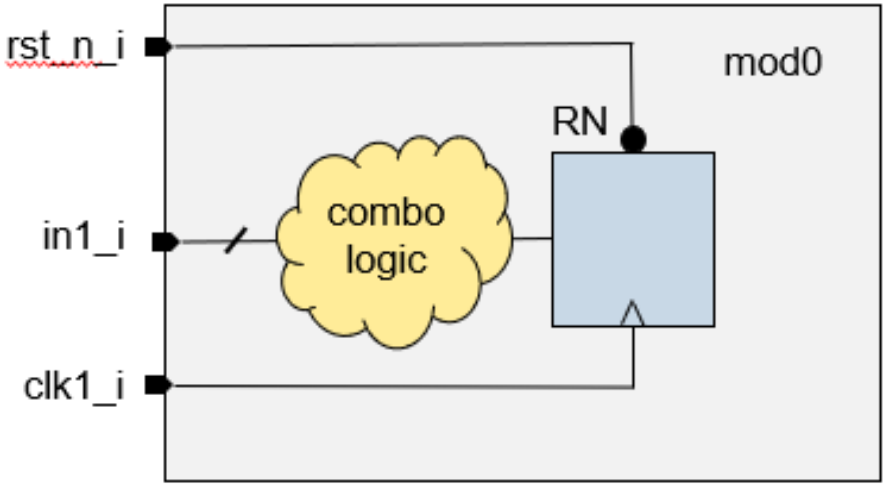
1. Driver is mandatory
2. Uses of EDA-Tool-Agnostic keyword

```
port -name d2_o -type data -direction output -associated_from_clock clk2_i -cdc_control q2_o

port -name q2_o -type cdc_control -direction output -cdc_control_from_clock clk2_i
-associated_from_clock clk1_i -associated_output d2_o

port -name c2_o -type data -direction output - associated_from_clock clk2_i
```

Port Attribute Modelling – virtual clock



With the current specification in LRM 0.1, a virtual clock is assumed when there is no corresponding “port -name” defined.

port -name clk1_i -type clock -direction input

port -name in1_i -type data -direction input -asssociated_from_clocks vclk

port -name in1_i -asssociated_to_clocks clk1_i -logic combo

port -name rst_n_i -type async_reset direction input -asssociated_from_clocks clk1_i

Set_cdc_clock group

- Default clocks are asynchronous to each other unless specified as synchronous.
- Supports 2 use models, with -name or without -name

6.01	set_cdc_clock_group	clocks	string	; separated list{clock-names}	mandatory
6.02	set_cdc_clock_group	name	string	{group-name]	optional

- Use model without –name (

```
set_cdc_clock_group -clocks {clk1;clk2;clk3;clk4}
```

```
set_cdc_clock_group -clocks {clk1;clk2;clk3}
set_cdc_clock_group -clocks {clk2;clk3;clk4}
```

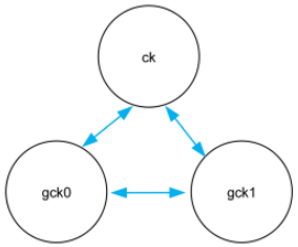
- `set_cdc_clock_group -clocks {clk1;clk4}`

- An abstracted block model shall not contain both use models together

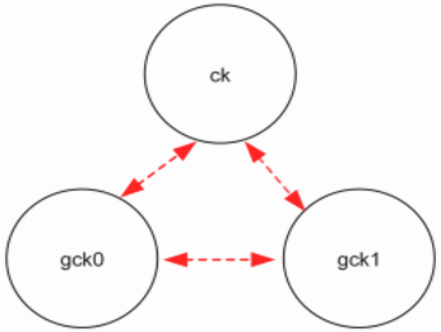
```
set_cdc_clock_group -clocks {clk1;clk2;clk3;clk4} -name sys_clk
```

Set_cdc_clock group usage example

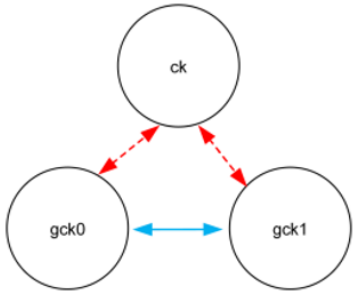
```
set_cdc_clock_group -name common_domain -clocks {ck;gck0;gck1}
```



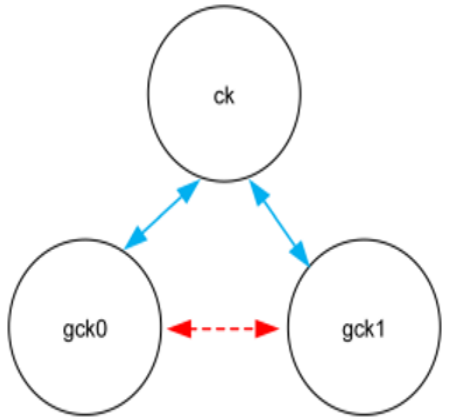

```
set_cdc_clock_group -name domain_c -clocks {ck}
set_cdc_clock_group -name domain_0 -clocks {gck0}
set_cdc_clock_group -name domain_1 -clocks {gck1}
```



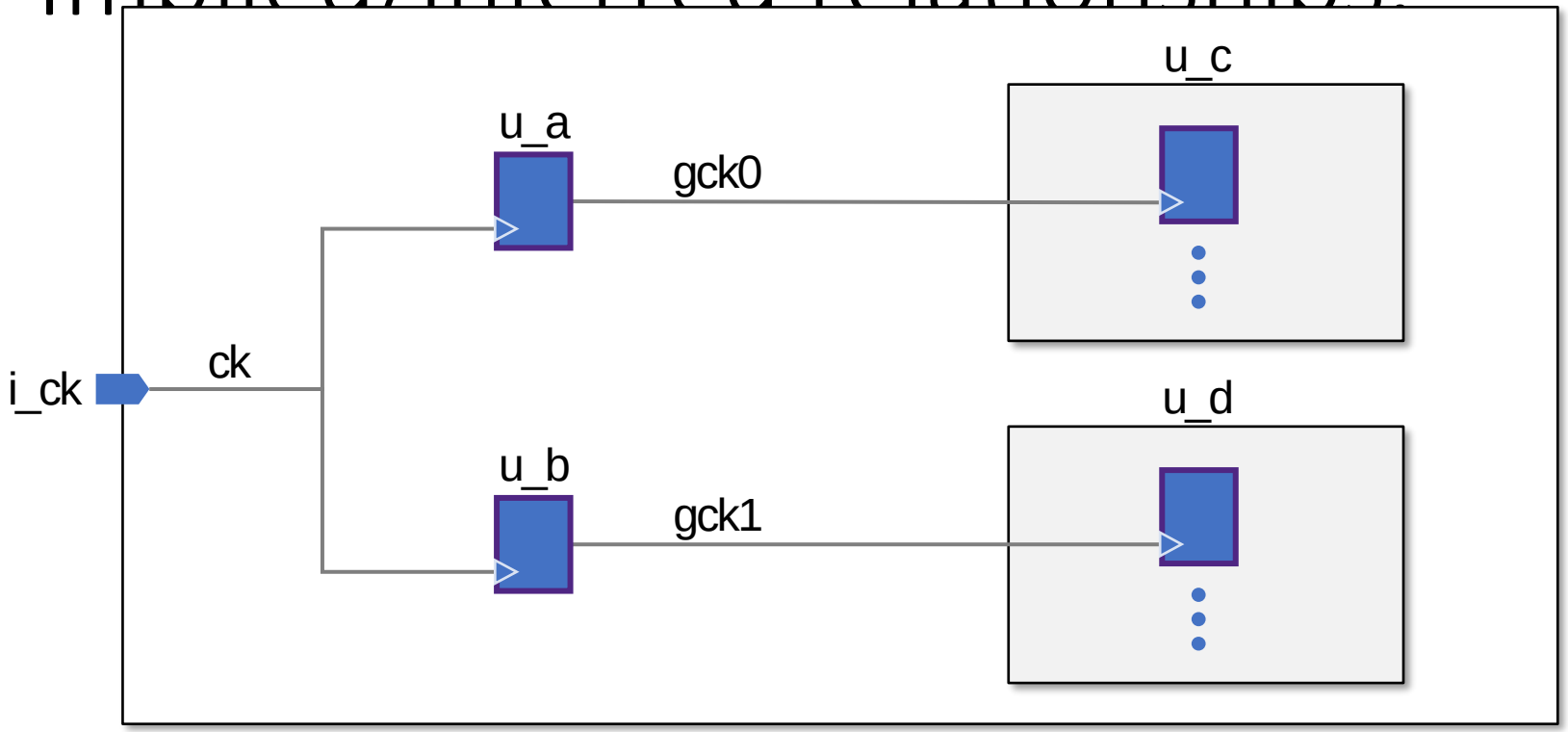

```
set_cdc_clock_group -name small_domain -clocks {ck}
set_cdc_clock_group -name large_domain -clocks {gck0;gck1}
```

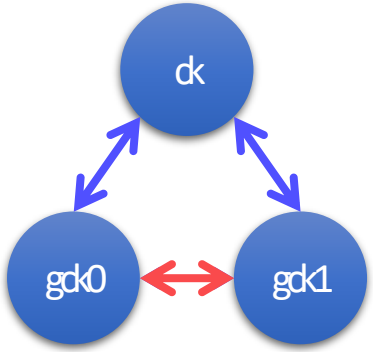



```
set_cdc_clock_group -name clk_branch0 -clocks {ck;gck0}
set_cdc_clock_group -name clk_branch1 -clocks {ck;gck1}
```



Implied/Inferred relationships






```
create_clock -name {ck} -period 10 [get_ports i_ck]
create_generated_clock -name {gck0} -source {i_ck} [get_pins u_a/Q]
create_generated_clock -name {gck1} -source {i_ck} [get_pins u_b/Q]

set_clock_groups -async -group {gck0} -group {gck1}
```

"ck" is not defined so it is inferred as synchronous to both clocks

Defining clock-to-clock relationships

- By default, all clocks have synchronous relationships between them

```
set_clock_groups -async -group {gck0} -group {gck1}
```

  - In above example, **gck0** is asynchronous to **gck1**
  - All remaining clock relationships are synchronous
- Command is distributive

```
set_clock_groups -async -group {ck gck0} -group {gck1}
```

  - Above command is same as these two

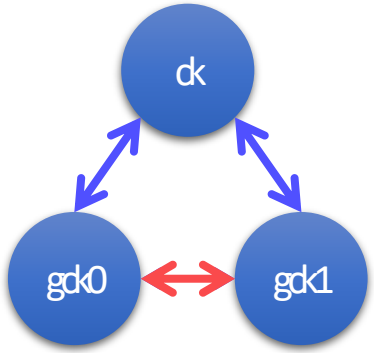
```
set_clock_groups -async -group {ck} -group {gck1}
set_clock_groups -async -group {gck0} -group {gck1}
```

- If only one group is specified, a second "default" group is implied

```
set_clock_groups -async -group {ck} -group {gck0 gck1}
```

- Same clock can appear only once in each command but can appear in as many commands as needed

- **Myth:** clocks assigned to same group are synchronous among them
- **Truth:** command only defines groups that are asynchronous between them



Accellera CDC WG initiative

Output-Collateral

- Sub-working groups presentation

- Examples of set_cdc_clock_group usage
- Defining clock-to-clock relationships

Accellera CDC WG initiative

Output-Collateral

- Sub-working groups presentation

Accellera CDC WG initiative

Output-Collateral

- Sub-working groups presentation

Accellera CDC WG initiative

Assertion

- Sub-working groups presentation

- **Goal**

  1. Produce Language Reference Manual (LRM) addendum for Assertions.
  2. Enable all EDA vendors in developing tools that meet specification for generating System Verilog Assertions (SVA) along with collateral.
  3. Enable Intellectual Property (IP) companies to generate SVA along with collateral using various vendors/tools.
  4. Enable System On Chip (SOC) companies to consume generated SVA from any vendor/tool into their tool of choice.

Accellera CDC WG initiative

Assertion

- Sub-working groups presentation

Accellera CDC WG initiative

Testing

- Sub-working groups presentation

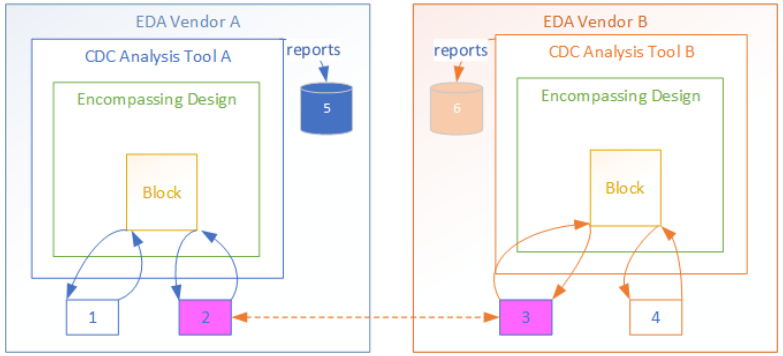
- **Goal**

  1. Quantify the completeness of the Accellera set of CDC attributes against a pre-defined set of interface protocols across a pre-defined set of CDC analysis tools by different vendors.
  2. Use the set of attributes defined by the output collateral subgroup captured in the format defined by the format subgroup.
  3. Use objective qualitative measures to qualify the set of attributes against the pre-defined set of interface protocols and the associated IP and encompassing designs.

Goals & Requirements

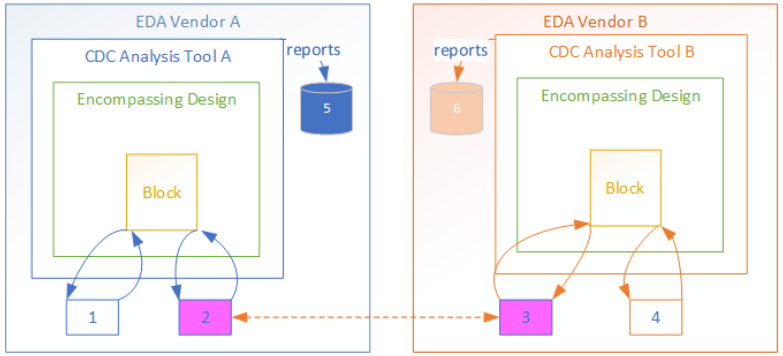
- Quantify the completeness of the Accellera set of CDC attributes against a pre-defined set of interface protocols across a pre-defined set of CDC analysis tools by different vendors
- Use the set of attributes defined by the output collateral subgroup captured in the format defined by the format subgroup
- Use objective qualitative measures to qualify the set of attributes against

Methodology (#1)



- Testing by Tool Vendor A
  - step#1 -Perform static flat CDC using Vendor A tool, creating the native block model (1) and Accellera abstract model (2)
  - step#2 - Perform static hierarchical CDC using native block model (1) & using Accellera abstract model (2)
  - step#2.1 - Compare results flat vs hierarchical with native block model
  - step#2.2 - Compare results of hierarchical native vs Accellera abstract model(s)
  - step#3 - Accellera to facilitate exchange of Accellera abstract model (2) with another tool for the same IP
  - step#3.1 - Perform static hierarchical CDC using Accellera abstract model by another tool vendor (3)
  - step#3.2 - Compare results of Accellera abstract model (2) and another tool vendor's Accellera abstract model (3)

Methodology (#1) [cont…]



- Step#4 – Report fault model grading per step#2.2 and step#3.2. Fault grading information to be provided by Accllera CDC WG

- This process is of course symmetrical, and Vendor B performs tests to the above description for Vendor A

Methodology (#2)

- Testing by Non-tool Vendors
  - Participants with access to more than one required CDC tool can perform cross tool testing
  - Design IP per list of required interface protocol can be either an inhouse design if available or borrowed for the testing purpose (Accellera to facilitate).
  - EDA vendors to provide their tool support to participants

Accellera CDC WG initiative

Testing

- Sub-working groups presentation

Accellera CDC WG initiative

Training

- Sub-working groups presentation

- **Goal**
  1. Provide a generic documentation to let the CDC-RDC IP model user understand :
     - 1.1 CDC-RDC basic knowledge
     - 1.2 List of attributes & definition (related to IP CDC-RDC features/properties) as defined and agreed by the main CDC WG
  2. Presentation of the hierarchical flow
     - 2.1 tool dependency issue
     - 2.2 necessity to get an inter operational CDC-RDC model
  3. Raising awareness on the importance of defining a standard CDC-RDC model
  4. Inter operational CDC-RDC model integration manual

Accellera CDC WG initiative

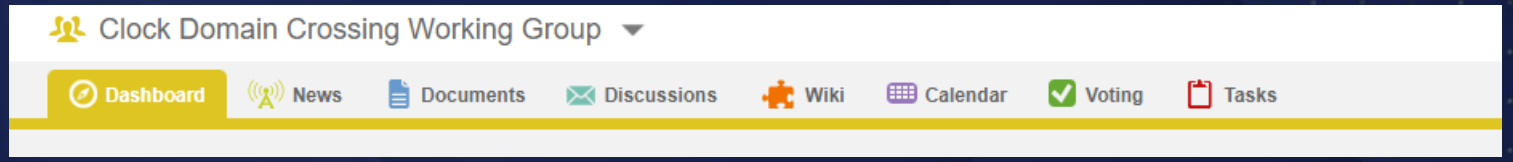
Training

- Sub-working groups presentation

- Accelera CDC WG work promotion through conferences
  - Past/current conferences
    - DVCON Europe 2023
    - DVCON US 2024
  - Targeted conferences (To Be Confirmed)
    - DVCON Japan / India / China / Taiwan / Europe 2024
    - DAC 2024
    - DATE 2025
    - VLSI 2025

Accellera CDC Working Group

Call for Contribution !



https://workspace.accellera.org/wg/CDC

accellera
SYSTEMS INITIATIVE

2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

Questions ?

accellera
SYSTEMS INITIATIVE