POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria Elettronica

Tesi di Laurea Magistrale

CDC, RDC &
INIEZIONE DELLA METASTABILITÀ

Relatore/i

Prof. Maurizio Martina

Ing. Alessandro Giuliano Locardi

Candidato

Andrea Michele Lopinto

Anno Accademico 2020-2021

Ringraziamenti

Voglio dapprima ringraziare il mio relatore, il Prof. Maurizio Martina, per la sua sempre presente disponibilità e attenzione che ha avuto nei miei confronti e che mi ha accompagnato in tutto il corso dell'attività di tesi. Lo ringrazio per la passione che ha fatto nascere in me verso il mondo dell'elettronica digitale.

Ringrazio il mio supervisore, l'Ing. Alessandro Giuliano Locardi, che mi ha dato l'opportunità di svolgere tale attività di tesi, di imparare e di migliorare ogni giorno condividendomi le sue conoscenze ed il suo tempo. Ringrazio il gruppo di verifica digitale, ovvero Alessandro, Paola, Laura, Davide ed Adele per avermi dato l'opportunità di conoscere l'affascinante mondo della verifica digitale e di comprendere l'importanza di un team all'interno di una realtà aziendale.

Ringrazio coloro che mi hanno fatto rendere conto di avere gusti orrendi per la scelta dei colori.

Ringrazio tutti i miei amici che mi hanno sempre sostenuto, in particolare: i miei fratelli acquisiti Peppe, Antonino e Danilo, la sempre presente Eleonora, la rassicurante Carla e il paziente Nick. Ringrazio i miei ex coinquilini Riccardo e Andrea e l'intrusa Roberta. Ringrazio Alessio, Maurizio, Marco, Beatrice e Alessia.

Ringrazio tutti i miei parenti per il loro sempre costante supporto e per le loro parole incoraggianti.

Un ringraziamento speciale per coloro che hanno consentito tutto questo e che hanno posto sempre anima, cuore e fiducia in me:

Mio Padre, mia Madre e mia Sorella che mi hanno sempre sostenuto, incoraggiato ed avuta un'immensa pazienza ogni giorno non lasciando mai l'opportunità di abbattermi. Li ringrazio per avermi dato sempre la forza di continuare e di avermi insegnato a rendere piccoli gli enormi ostacoli della vita.

Indice

Tabella delle figure

1 Introduzione generale al processo di Design e Verifica

Nei moderni dispositivi è sempre più comune l'implementazione di nuove funzionalità, processo che si riflette nell'avere dispositivi competitivi ma al tempo stesso più complicati. I moderni SOC (system on chip) tendono quindi a diventare più complessi e a richiedere un maggiore investimento di tempo per poter verificare il loro corretto funzionamento. Uno degli obiettivi principali nello sviluppo e produzione di design digitali è quello di ridurre al minimo gli errori funzionali e strutturali. Dai meno impattanti che possono risvoltarsi in errori di lieve entità, ma non per questo di minore importanza, ai più pericolosi, che possono portare al completo malfunzionamento del dispositivo. Realizzare design sempre più complessi si ripercuote nell'aumento della semplicità con cui nascono errori come anche nell'aumento della difficoltà nella loro identificazione. La verifica digitale svolge quindi un ruolo fondamentale nella produzione di un dispositivo, con lo scopo di garantire prodotti di qualità al momento del loro tape-out. L'obiettivo principale è quello di rendere la verifica digitale più efficiente, essendoci sempre meno tempo disponibile nell'immettere nuovi dispositivi nel mercato e sempre più efficace, al fine di avere una maggiore affidabilità di quello che è stato prodotto. I nuovi design sono caratterizzati da un numero di domini di clock che tende ad aumentare e questa tendenza è giustificata dallo scopo di ottenere un'ottimizzazione sempre più aggressiva della potenza dinamica, dall'esigenza di integrare sempre più interfacce di I/O e dalla necessità di gestire in maniera essenziale il continuo crescere della loro complessità. I metodi standard di verifica, come ad esempio l'analisi statica del timing o la simulazione a livello RTL, risultano non essere più sufficienti e diventa sempre più difficile raggiungere un'ampia copertura di validazione delle diverse funzionalità dei moderni dispositivi, contrastando quindi l'obiettivo di avere sempre più una maggiore sicurezza del corretto comportamento dei design. Per questo motivo che nasce l'esigenza di affiancare alla fase di verifica digitale quella che è un'ulteriore metodologia: il flusso di analisi "CDC, RDC e Iniezione della Metastabilità".

1.1 Introduzione al lavoro di tesi

In tale sezione viene fornita un'introduzione riguardante al lavoro di tesi svolto. Viene riportato un riassunto dei passaggi affrontati per poter raggiungere l'obiettivo di formare un flusso consistente di verifica CDC-RDC.

L'obiettivo dell'attività di tesi è quello di progettare un nuovo flusso di verifica che consenta di identificare tutte le problematiche dovute alla presenza di Clock-Domain-Crossing e Reset-Domain-Crossing sin dai primi stadi di produzione di un design. Il flusso CDC progettato, approfondito nel lavoro di tesi, ha l'obiettivo di validare tutti i percorsi di Clock-Domain-Crossing e di Reset-Domain-Crossing presenti in un design ed ha lo scopo di controllare che tali percorsi siano gestiti in maniera corretta, come anche assicurare che le strutture di sincronizzazione presenti nel DUT siano implementate correttamente.

La necessità di integrare tale flusso consiste nel non dover attendere la fase di verifica di Post-Layout per poter riscontrare eventuali problematiche CDC-RDC presenti nel design e consente di avere una maggiore consapevolezza di esse sin dalle prime versione del codice RTL. Senza l'introduzione di tale

flusso CDC, la possibilità di identificare le problematiche CDC e RDC risiede nella fase di verifica di Post-Layout attraverso le PLS (post layout simulation). Purtroppo, attraverso di esse non è possibile garantire che vengano identificate tutte le problematiche CDC-RDC presenti all'interno del design e questo perché si è fortemente dipendenti dalla qualità dei testcase che vengono eseguiti, come anche si è fortemente dipendenti dalla percentuale di copertura di verifica raggiunta nella validazione del design.

Identificare problematiche CDC e RDC durante la fase di verifica Post-Layout si può riflettere nel dover attuare modifiche RTL. Tali modifiche richiedono una nuova esecuzione del flusso di progettazione di design e diventa necessario affrontare nuovamente gli stadi di produzione di Sintesi, verifica Post-Sintesi, Place&Route e verifica Post-Layout a causa delle problematiche CDC-RDC riscontrate nella Post-Layout verification. Tali stadi di produzione risultano essere onerosi sia dal punto di vista di tempo di esecuzione che dal punto di vista di elaborazione. Dunque, rilevare la presenza di problematiche CDC-RDC all'interno della fase di verifica Post-Layout si riflette in una sottrazione di tempo significativo per la produzione di un design.

Con l'introduzione del flusso CDC-RDC progettato diventa invece possibile identificare tutte le problematiche legate alle asincronie sin dalle prime versione del codice RTL e dunque ancor prima di eseguire gli onerosi stadi di produzione di Sintesi, Place&Route e relative verifiche. Il flusso CDC-RDC consente quindi di analizzare e rimuovere le criticità presenti nel design non appena è disponibile il codice RTL. Viene attuata un'analisi strutturale e funzionale, la quale è basata sull'utilizzo di un engine formale e simulazioni dinamiche. Il flusso CDC-RDC introduce anche la possibilità di iniezione della metastabilità e di glitches, attraverso la verifica formale e verifica funzionale con simulazioni dinamiche, consentendo di validare la robustezza del design all'occorrenza di metastabilità e di impulsi spuri come anche validare che l'introduzione di strutture e protocolli di sincronizzazione non causino errori funzionali.

Al fine di avere una migliore comprensione di quello che è stato eseguito, viene riportato l'elenco degli step che ho intrapreso per poter raggiungere l'obiettivo del lavoro di tesi:

- **studio dell'architettura digitale e della struttura di clocks e resets di un dispositivo MEMS**: è stato eseguito lo studio del design su cui è stato eseguito il flusso CDC-RDC. Il motivo è legato alla necessità di conoscere le informazioni strutturali necessarie per l'esecuzione del flusso e per poter effettuare una più accurata analisi delle strutture e protocolli di sincronizzazione implementati;
- **studio delle strutture di sincronizzazione per trasferimento dati/segnali e reset:** è stato eseguito uno studio delle comuni violazioni CDC e RDC che possono essere incontrate in un design asincrono, come anche uno studio delle comuni strutture di sincronizzazione implementabili per affrontare tali problematiche;
- **studio dello stato dell'arte degli strumenti di verifica ed analisi strutturale CDC ed RDC:** è stato effettuato uno studio degli strumenti disponibili per poter effettuare l'analisi strutturale del design come anche uno studio del linguaggio SVA (System Verilog Assertion) atto alla scrittura di asserzioni, assunzioni e covers per poter effettuare una validazione formale di protocolli di sincronizzazione presenti nel design;
- **studio della verifica formale e della verifica funzionale con simulazioni dinamiche**;
- **progettazione ed applicazione del flusso di verifica CDC e RDC ad un dispositivo MEMS;**
- **analisi dei risultati ottenuti dall'applicazione del flusso:** sono state ricavate le violazioni CDC e RDC presenti nel design ed è stata eseguita un'analisi di ognuna di esse al fine di indentificare se tali violazioni risultino essere reali problematiche CDC e RDC;

- **applicazione dell'iniezione della metastabilità in verifica formale ed in verifica funzionale con simulazioni dinamiche**: è stata verificata la robustezza dei protocolli e schemi di sincronizzazione del design in presenza di metastabilità;
- **applicazione dell'iniezione di glitches in verifica funzionale con simulazioni dinamiche**: è stata effettuata una verifica della robustezza dei protocolli di sincronizzazione anche in presenza di un campionamento di un impulso spurio ed una verifica del loro impatto sulle funzionalità del design;
- **analisi dei risultati ottenuti dall'applicazione dell'iniezione della metastabilità e glitches**;
- **progettazione di possibili miglioramenti per irrobustire il design in analisi**: sono state progettate delle soluzioni architetturali, come anche procedure da seguire al fine di non incorrere in errori funzionali dovuti ai Clock-Domain-Crossing e Reset-Domain-Crossing;

Viene adesso fornito un riassunto dei capitoli illustrati nel lavoro di tesi al fine di riportare come è stato strutturato quest'ultimo ed anche fornire i risultati che si vogliono ottenere da ognuno di essi:

Nel capitolo 1 viene fornita una introduzione generale al flusso di progettazione digitale come anche un'introduzione alle più comuni metodologie di verifica presenti. L'obiettivo che si vuole raggiungere in tale capitolo è quello di contestualizzare dove si pone il lavoro di tesi all'interno di un flusso di progettazione di design e come il flusso CDC-RDC si identifica all'interno del contesto di verifica digitale.

Nel capitolo 2 viene illustrato il lavoro di tesi e viene effettuata un'introduzione ai tool utilizzati per poter eseguire il flusso CDC-RDC progettato sul design in analisi. Viene quindi illustrato dove si pone il lavoro di tesi all'interno del flusso di progettazione di un design e viene illustrata una base delle problematiche CDC che vuole affrontare tale flusso e come quest'ultimo possa irrobustire un design.

Nel capitolo 3 viene fornita un'introduzione al design sotto analisi e ad una illustrazione delle sue principali caratteristiche necessarie per la corretta esecuzione del flusso. Tale capitolo ha lo scopo di introdurre un'idea generale delle informazioni tipiche da conoscere di un design per poter eseguire il flusso CDC-RDC nel modo più accurato possibile.

Nel capitolo 4 viene introdotto il flusso di analisi CDC eseguito. Sono illustrate alcune delle comuni problematiche CDC che possono essere incontrate durante un'analisi strutturale di un design come anche vengono introdotte le più comuni soluzioni di schemi di sincronizzazione adottabili per poter risolvere tali problematiche CDC. L'obiettivo che si vuole raggiungere da tale capitolo è quello di introdurre le tipiche problematiche CDC che possono essere incontrate a fronte di asincronie ed introdurre quelle soluzioni più comuni per poter affrontare tali problemi. Il capitolo ha anche lo scopo di introdurre gli step che vengono eseguiti per poter impostare un ambiente di analisi CDC e per poter ottenere risultati accurati dall'esecuzione del flusso.

Nel capitolo 5 viene quindi applicato il flusso CDC illustrato nel capitolo 4 al design sotto analisi e quindi sono applicate le informazioni strutturale necessarie per una più accurata analisi CDC. Dopo aver illustrato quindi i passaggi del flusso CDC nel capitolo 4, si procede con applicarli al design.

Nel capitolo 6 sono approfondite alcune delle violazioni CDC riscontrate nell'analisi CDC del design. Tali violazioni sono ottenute dall'esecuzione dello step di identificazione degli schemi di sincronizzazione. Vengono quindi illustrate ed analizzate alcune delle problematiche identificate dall'applicazione del flusso CDC sul design in analisi.

Nel capitolo 7 è introdotto il flusso di analisi RDC. Vengono illustrate le tipiche violazioni RDC che possono essere riscontrate nell'analisi di un design come anche le possibili soluzioni comuni per poter affrontare tali violazioni. L'obiettivo che si vuole raggiungere da tale capitolo è quello di introdurre le tipiche problematiche RDC che possono essere incontrate in un design ed introdurre le soluzioni e schemi di sincronizzazione più comuni per poter affrontare tali problemi. L'analisi RDC eredita gran parte delle informazioni strutturali da parte dell'analisi CDC, perché eseguita in parallelo; dunque, non è presente la necessità di un ulteriore esecuzione di steps.

Nel capitolo 8 viene applicato il flusso RDC al design in analisi e vengono quindi fornite le informazioni strutturali necessarie. Il flusso RDC viene eseguito in parallelo al flusso CDC ed essenzialmente eredita informazioni da parte dei passaggi CDC.

Nel capitolo 9 sono illustrate alcune delle violazioni RDC identificate dall'esecuzione del flusso sul design in analisi. Vengono quindi illustrate ed analizzate alcune delle problematiche identificate dall'applicazione del flusso RDC sul design in analisi.

Nel capitolo 10 viene introdotta la verifica formale. Vengono fornite le basi necessarie per poter comprendere ed utilizzare il linguaggio SVA per raggiungere l'obiettivo di eseguire una validazione dei protocolli presenti nel design e comprendere come è stata eseguita la validazione di tali protocolli. In tale capitolo si vuole quindi illustrare come viene applicata la validazione formale dei protocolli di sincronizzazione riscontrati durante l'analisi delle violazioni strutturali riportate nei capitoli 6 e 7.

Nel capitolo 11 viene introdotta come viene modellata ed iniettata la metastabilità con approccio formale e come vengono modellati ed iniettati metastabilità e glitches in simulazione dinamica RTL. In tale capitolo si vuole raggiungere l'obiettivo di illustrare di come sia possibile verificare il comportamento di un protocollo o schema di sincronizzazione a fronte di metastabilità e glitches e validare se il percorso CDC-RDC, in cui quest'ultimi sono stati iniettati, possono essere definiti Metastability e Glitch safe.

Nel capitolo 12 vengono quindi riportati i risultati ottenuti dall'iniezione della metastabilità ad alcune delle strutture del design illustrate nel capitolo 6.

Nel capitolo 13 vengono riportate le conclusioni del lavoro di tesi e quindi forniti gli obiettivi raggiunti dall'attività.

1.2 Flusso di design

Prima di procedere con l'introduzione dell'analisi "CDC, RDC e Formale" e delle diverse metodologie di verifica presenti, si procede con il rivedere quali sono gli step necessari allo sviluppo di un sistema digitale. Al fine di avere una migliore visione del flusso di progettazione, si riportano i diversi step schematizzati in un diagramma di flusso a partire dalle specifiche del design fino ad arrivare alla sua immissione nel mercato. Tale flusso di progettazione è ovviamente una versione semplificata e generalizzata. Quest'ultimo viene riportato nella Figura 1-1 e viene descritto nei punti successivi.

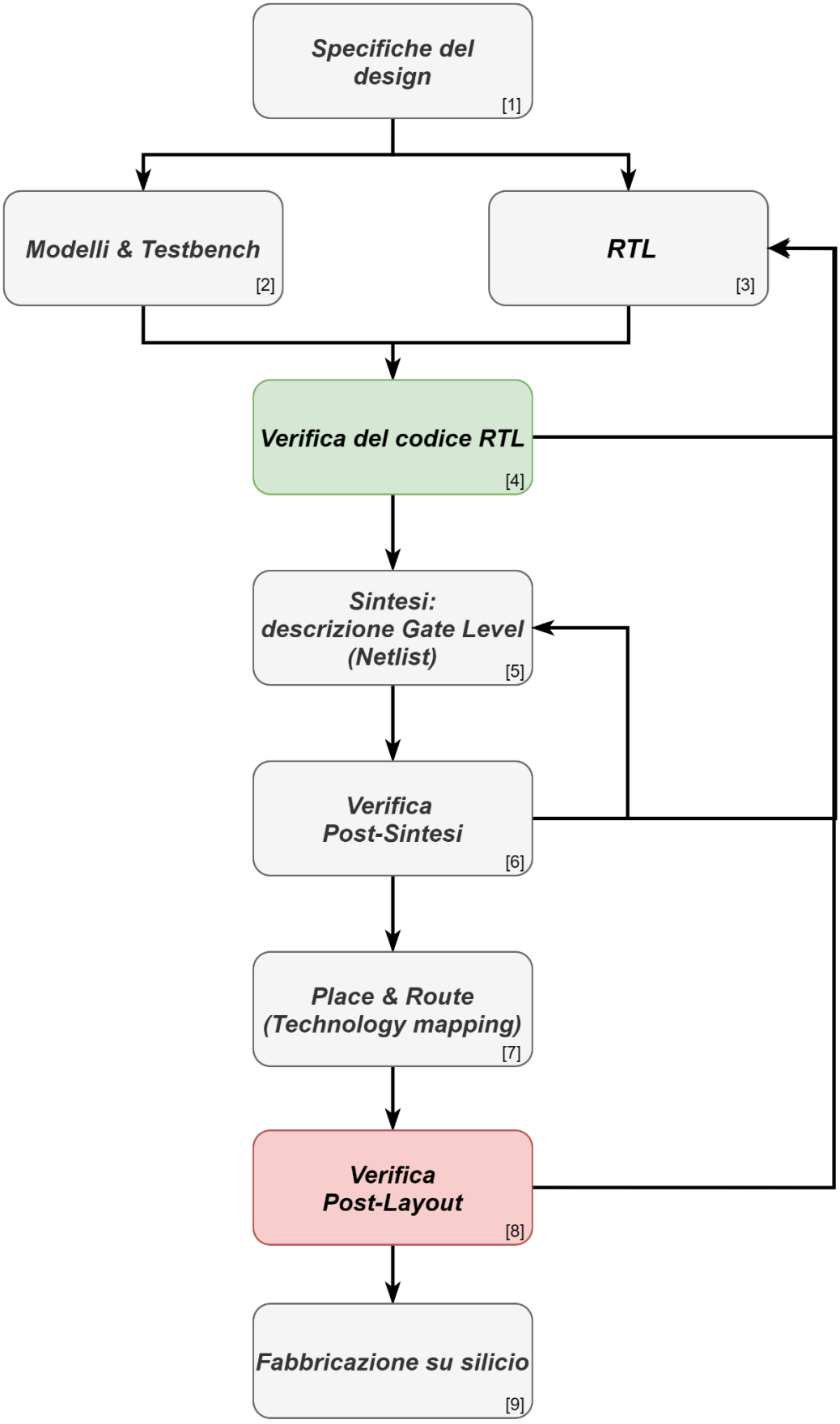

Figura 1-1: Flusso di produzione di un sistema digitale

1) ***Specifiche del design***: Il punto di partenza è caratterizzato dal set di specifiche di cui il design deve essere composto. Esse descrivono il set di comportamenti e funzionalità che il sistema deve essere in grado di soddisfare ed anche il set di constraints che deve essere rispettato durante la sua progettazione: area, potenza, frequenza ed ovviamente il costo complessivo.

2) ***Modelli e Testbench***: Lo step successivo è la descrizione funzionale del design e la relativa progettazione dei Testbench atti alla verifica del design RTL, la cui descrizione viene portata in parallelo a tale step (come è possibile notare nel diagramma di flusso riportato nella Figura 1-1). In questo step si punta a realizzare il modello del design, al fine di soddisfare le sue specifiche ed anche verificare la loro implementazione RTL. Dunque, tale modello si concretizza in una possibile soluzione del design che si vuole progettare. Spesso si ricorrono a linguaggi di programmazione come il C, Python, script Matlab o altre soluzioni dello stesso genere al fine di simulare e verificare il comportamento del design ed assumerlo come modello di riferimento per gli step successivi. Viene affiancata anche la progettazione di modelli di Testbench descritti in HDL e vengono definiti i differenti pattern di stimoli utili a verificare le funzionalità del dispositivo, pattern che vengono definiti Testcase. I modelli in questione vengono scritti al fine di stimolare il funzionamento del design nello step di produzione "Verifica RTL".

3) ***RTL***: Dalle specifiche definite nel primo passaggio di produzione si procede con la sua realizzazione a livello RTL, "*Register Transfer Level*". Si ricorre ad un linguaggio di descrizione Hardware ovvero l'HDL "Hardware Description Language" come, ad esempio, il linguaggio "VHDL" o "Verilog".

4) ***Verifica del codice RTL***: Una volta descritto il design a livello RTL e definiti i relativi modelli di riferimento e Testbench, si procede con la fase di verifica dell'RTL. Questo è uno dei passaggi produttivi atti a verificare il corretto funzionamento del dispositivo ed esso ha il compito di verificare che il design rispetti le specifiche di progettazione richieste comparando il suo comportamento con il modello di riferimento. Si ha quindi la necessità di identificare e risolvere all'interno di questo step i problemi funzionali emersi durante la verifica del codice RTL. Ad ogni errore funzionale o specifica non rispettata viene eseguita una modifica, questo col fine di rimuovere i problemi identificati ed eseguire nuovamente una sua validazione. All'interno della fase di verifica RTL sono effettuate verifiche funzionali, formali e strutturali col fine di ottenere una maggiore conferma della correttezza di funzionamento del design. Vengono utilizzati i modelli Testbench e i pattern specifici dei segnali di ingresso col fine di verificare il corretto comportamento delle diverse funzionalità che compongono il design e per poter effettuare la validazione che i segnali di uscita dell'RTL del DUT rispecchino il comportamento aspettato. In tale verifica non sono presi in considerazione i ritardi logici, presenti invece nella verifica post-layout. Sono effettuate verifiche strutturali con lo scopo di avere un codice RTL sempre più pulito da problemi strutturali. Un esempio di verifica strutturale è la verifica CDC-RDC illustrata nel lavoro di tesi. Viene anche effettuata una verifica formale con lo scopo di affiancare alle verifiche precedenti un'ulteriore forma di validazione del comportamento del design. Quindi, questo passaggio produttivo risulta essere fondamentale al fine di proseguire con i processi produttivi seguenti, ovvero la sintesi del design e il suo mapping tecnologico. Questi due passaggi appena citati possono richiedere del tempo non indifferente per la loro elaborazione e quindi vi è la necessità di avviare il loro processo solo nel momento in cui si ha un elevato livello di confidenza di correttezza del proprio design.

5) ***Sintesi***: Lo step successivo è la sintesi dove si procede con il generare la Netlist del design RTL. In questo passo produttivo, in cui si è raggiunti una certa confidenza con il proprio design, si prosegue con il tradurre il codice RTL nella corrispettiva netlist. Quest'ultima descrive il design a livello hardware, ovvero l'implementazione dei gate logici, memorie e relative connessioni ed ogni componente che viene inserito è legato ad una determinata libreria tecnologica caratterizzata da parametri di potenza, ritardi, area ed etc.. . In tale step si esegue anche l'ottimizzazione del design dal punto di vista di area e celle, dove quest'ultime devono essere implementate in funzione dei parametri di potenza e frequenza stabiliti nelle specifiche di sistema. La generazione della netlist può richiedere del tempo non indifferente per poter raggiungere un compromesso tra area, potenza e frequenza. Questo tempo è anche strettamente dipendente dalla complessità del design in questione. Successivamente alla generazione della netlist del design vi è la sua procedura di verifica. L'implementazione gate level non garantisce che le funzionalità del dispositivo siano ancora rispettate dopo la sua generazione ed è quindi necessario attuare una verifica del design dopo la sua sintesi.

6) ***Verifica Post-sintesi***: Nello step precedente si è quindi sintetizzato il design RTL e si è ottenuta la sua relativa netlist. Come citato precedentemente, vi è la necessità di effettuare una verifica post-sintesi al fine di assicurare che il comportamento del design sia ancora effettivamente rispettato. Questa verifica è necessaria perché vi è una possibilità che siano stati introdotti errori funzionali durante la generazione della relativa netlist. Essenzialmente, viene effettuato un confronto tra i risultati ottenuti dalla verifica RTL pre-sintesi con i risultati ottenuti dalla verifica post-sintesi. La quantità di lavoro da effettuare durante questo step non è paragonabile a quello necessario per la verifica RTL pre-sintesi, in quanto è possibile automatizzare il processo di test di uguaglianza dei risultati ottenuti dalle due verifiche. Nel momento in cui vengono identificati degli errori, si deve procedere con una ulteriore sintesi del design ed eventualmente effettuare delle modifiche del codice RTL al fine di ottenere dei risultati migliori e corretti dalla nuova sintesi. Grazie a questo step si ha la conferma che le funzionalità del design non sono state compromesse dalla fase di sintesi.

7) ***Place & Route***: Si procede quindi con il mapping tecnologico ed il place & route del design. Tale passaggio è costituito da differenti punti: Floorplanning, power planning & routing, il cell placing, signal routing ed infine l'analisi del design e del suo timing. Essenzialmente si procede con il definire il modello analogico del design realizzato. Vengono quindi definite le posizioni all'interno di un'area dei componenti e vengono collegati tra di loro.

8) ***Verifica Post-layout***: Successivamente alla fase di Place & Route del design, vi è la necessità di effettuare una sua verifica al fine di validare che sia ancora rispettata la corretta funzionalità del design. Lo stadio di produzione del Place & Route ha come obiettivo quello di ottimizzare il design nella sua interezza e le diverse ottimizzazione che vengono applicate potrebbero portare alla nascita di errori funzionali. Bisogna anche sottolineare che in questo stadio di verifica si hanno anche a disposizione i ritardi che caratterizzano le celle e le connessioni, diventa quindi possibile validare che i vincoli temporali necessari per il corretto funzionamento del design siano rispettati. Dunque, in questo stadio di verifica è possibile identificare eventuali errori legati a violazioni temporali, come ad esempio la violazione dei tempi di setup ed hold dei FlipFlop.

9) ***Fabbricazione***: Fabbricazione del design su silicio, il prodotto viene realizzato fisicamente.

Il numero di volte che deve essere eseguito tale flusso per poter raggiungere un design sempre più corretto può essere elevato e il tempo richiesto per ogni modifica, verifica e nuove sintesi diventa sempre maggiore mentre quello richiesto per immettere nuovi prodotti nel mercato viene sempre più ridotto. Con l'aumento della complessità dei nuovi design, richiesta per competere con i diversi prodotti nel mercato, e con sempre meno tempo a disposizione per la produzione, i teams di design e verifica spingono verso nuove tecniche e ottimizzazioni al fine di produrre dispositivi con maggiori funzionalità nel minore tempo possibile.

1.3 Verifica RTL

La seguente sezione ha lo scopo di introdurre ed approfondire lo stadio di produzione definito Verifica RTL, introdotto nel flusso di progettazione nella Figura 1-1 ed illustrato nella Figura 1-2. Si procede con il descrivere nel dettaglio quali sono gli obiettivi che si vogliono raggiungere da tale passaggio produttivo e da quali tipi di verifica esso è costituto. Vengono anche riportate le caratteristiche che a loro volta costituiscono le tipologie di verifica presenti al suo interno. Tra i diversi stadi di produzione presenti nel flusso di design, viene posta l'attenzione su questo specifico passaggio produttivo perché lo stadio di verifica RTL costituisce lo step di produzione preso in considerazione dal lavoro di tesi.

La verifica RTL si suddivide principalmente in tre macrocategorie: verifica strutturale, verifica funzionale e verifica formale. Tali categorie vengono introdotte successivamente in tale sezione.

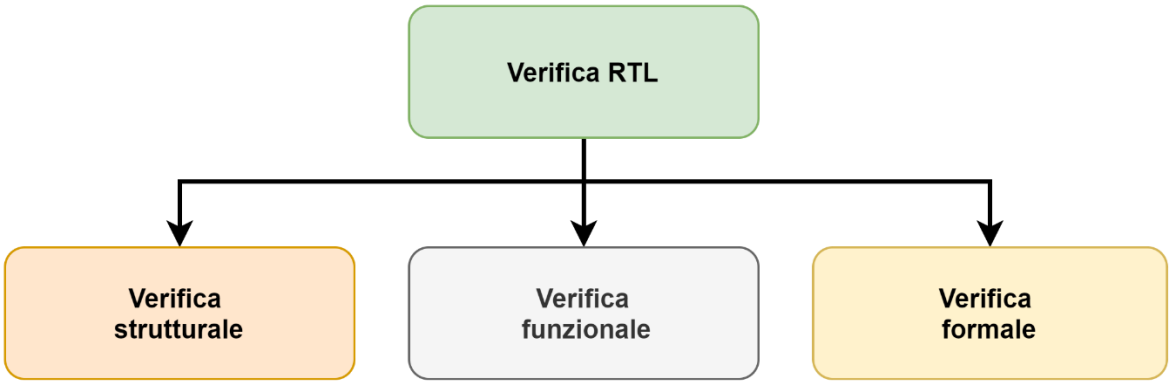

Figura 1-2: Categorie in cui si suddivide la verifica RTL

La verifica RTL svolge quindi un ruolo fondamentale nel flusso di progettazione di un sistema digitale e viene dunque sottoposta a costanti miglioramenti in tutte le sue categorie di analisi ed arricchita con nuove metodologie. Nonostante ciò, la verifica RTL è ancora adesso è una delle attività più onerose in termini di tempo e di risorse[2]. Come citato nella sezione precedente, tale verifica è la prima dei diversi step che viene effettuata nel corso della progettazione di un sistema digitale e ha il compito di individuare errori funzionali, strutturali e verificare che vengano soddisfatte le specifiche di progettazione richieste ed assicurare che sia rispettato il corretto comportamento del design. La verifica RTL ha quindi come obiettivo quello di minimizzare gli errori presenti all'interno dei design e per poter raggiungere tale obiettivo si ha la necessità di massimizzare la Coverage. Per Coverage, si intende quella metrica che misura la parte di design che viene coperta nella verifica del DUT "Design Under Test". Quindi, la coverage identifica quanto effettivamente del design è stato verificato e quanto di tutte le possibili configurazioni del DUT è stato coperto e validato dalle verifiche effettuate. [2]

1.3.1 Verifica funzionale

Il tradizionale flusso di verifica funzionale utilizzato per la verifica di un sistema digitale è tipicamente suddiviso nei seguenti passaggi: [2]

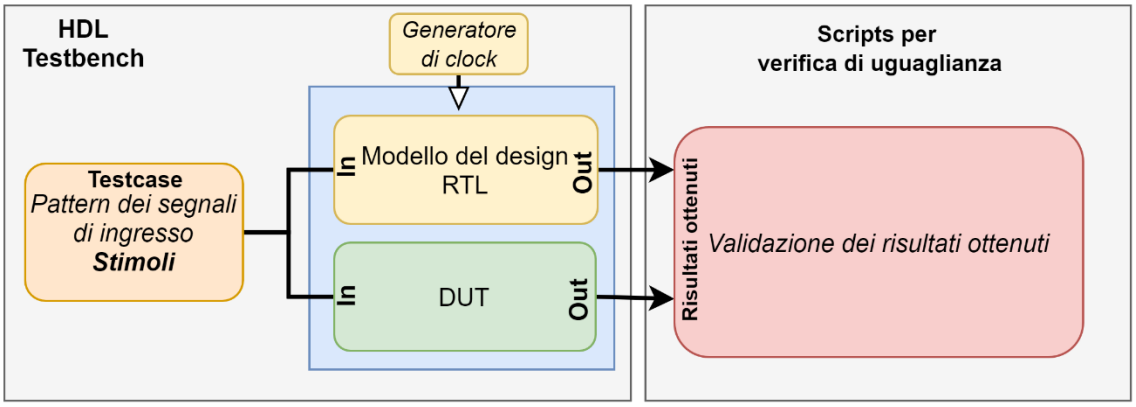

Figura 1-3: Verifica funzionale

1) **Testbench**: Utilizzo di testbench specifici per il design in questione descritto a livello RTL. Tali testbench possono essere descritti per i moduli costituenti il design o per il design completo.

2) **Stimoli**: Utilizzo di pattern specifici di segnali di ingresso atti a stimolare le varie funzionalità del design e che quindi consentono una verifica puntuale del comportamento del design. Tali pattern possono essere riferiti a funzionalità di moduli specifici o atti allo stimolare il comportamento del design nella sua interezza.

3) **Simulazione**: Simulare, attraverso i testbench e i pattern di ingresso, il modello del design scritto in codice RTL. Avviene quindi la verifica dei valori assunti dal design colpo di clock per colpo di clock.

4) **Validazione**: Identificare quindi eventuali malfunzionamenti o comportamenti non aspettati e validare il corretto funzionamento del dispositivo. Da questo step vi è la possibilità di identificare le parti del codice RTL, del design o testbench, che devono essere modificate al fine di correggere il comportamento del design e convergere verso il corretto funzionamento di quest'ultimo.

5) Eseguire quindi in maniera ciclica la simulazione e la correzione del codice RTL al fine di ottenere un design esente da errori e verificare che i risultati ottenuti dall'RTL coincidano con il modello di riferimento del design descritto ad alto livello.

La verifica funzionale è dunque basata sullo stimolare il DUT con determinati segnali di ingresso e dati, anche definiti Testcase, ed i risultati ottenuti sono riferiti a quei pattern specifici applicati. Come è possibile notare dalla Figura 1-3, la simulazione standard richiede diverse componenti per poter verificare il DUT, dove il testbench ed i testcase costituiscono il cuore della verifica dinamica RTL. Una volta ottenuti i risultati, stimolati dai pattern di ingresso colpo di clock per colpo di clock, si può procedere con il confrontare i risultati ottenuti con quelli desiderati.

- **Universal Verification Methodology - UVM**

Una metodologia di descrizione testbench ampiamente utilizzata è la UVM "Universal Verification Methodology". Con l'aumento della complessità dei progetti diventa sempre più comune che la loro progettazione e verifica si suddivida su più teams. Dunque, questo implica l'implementazione di diverse metodologie di verifica che possono però risultare incompatibili tra i diversi teams, generando quindi una discontinuità e limitazione della produttività. La Universal Verification Methodology, come è possibile intendere dal suo nome, è un metodo di verifica standardizzato che riesce a gestire la complessità e la comunicazione tra i diversi team di verifica. La UVM ha come obiettivo quello di ottenere testbench sempre più riutilizzabili. Quest'ultimi vengono caratterizzati da componenti di verifica riutilizzabili che consentono, quindi, un minore investimento di tempo nell'ottenere un ambiente di verifica per il DUT. I testbench UVM riescono quindi a diminuire i tempi di verifica e di migliorare le qualità dei design prodotti consentendo anche un più semplice riutilizzo di componenti di verifica già presenti. [1]

Da come si è potuto percepire dai passaggi appena spiegati, l'obiettivo di avere una Coverage completa risulta essere sempre più complicato con l'aumento della complessità dei design. Lo spazio di verifica di un design che viene validato attraverso tale verifica funzionale tende sempre più a restringersi e quindi vi è la necessità di affiancare sempre nuove metodologie alternative al fine aumentare l'area di copertura di verifica dei moderni design. È possibile intuire che ogni simulazione che viene effettuata copre solo una frazione del comportamento del design e una frazione di tutte le possibili combinazioni di ingresso e di tutti possibili stati da cui è caratterizzato. L'obiettivo ambito è sempre quello di raggiungere una copertura completa, il 100%, delle funzionalità del DUT.

1.3.2 Verifica formale

La verifica formale è basata sull'utilizzo di engine matematici che consentono di analizzare lo spazio completo di tutti i possibili comportamenti del design. [2]

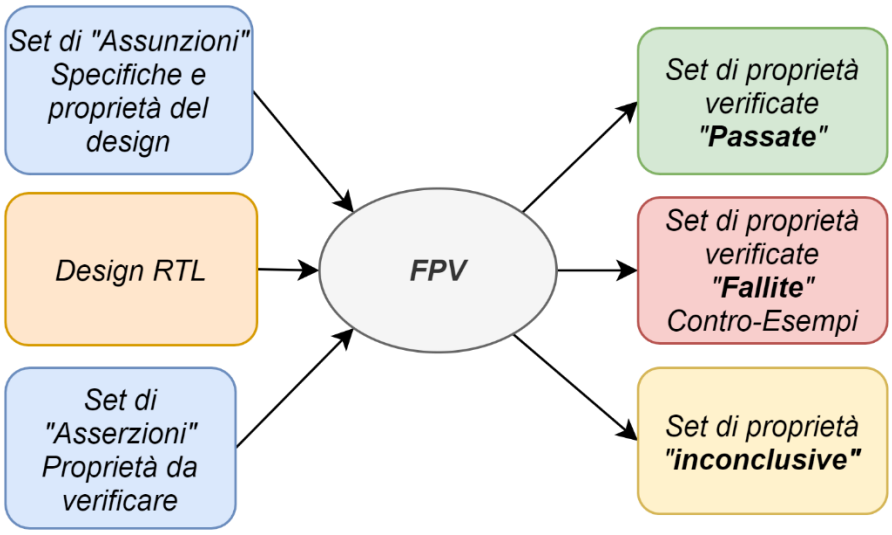

Figura 1-4: FPV-Formal Property Verification

Essenzialmente, la verifica formale ha il compito di prendere in considerazione lo spazio di tutti i possibili comportamenti all'interno del cono di logico coinvolto nella verifica del design, questo grazie all'utilizzo di tecniche matematiche per poter considerare tutte le possibilità presenti. Dunque, l'obiettivo è quello di verificare matematicamente tutto lo spazio possibile definito dai comportamenti del codice RTL e fornire un contro-esempio per validare un eventuale comportamento non rispettato.

La verifica formale e la simulazione sono tecniche complementari, la verifica formale aggiunge un'ulteriore conferma nell'analisi ma non può sostituire completamente la simulazione dinamica.

Affiancando quindi la verifica formale al codice RTL che si vuole analizzare, si introduce quindi la tecnica nominata FPV "Formal Property Verification". Essenzialmente, quest'ultima consiste nel verificare le proprietà che si vogliono validare nel proprio design, attraverso l'utilizzo di asserzioni, assunzioni e cover descritte in SVA "System Verilog Assertions" o altri linguaggi della stessa tipologia.

- **Gli elementi necessari per la FPV sono:**
  1) L'RTL del DUT;
  2) Il set di proprietà che rappresentano i comportamenti del design che ci si aspetta che vengano rispettati dal codice RTL;
  3) Il set di condizioni che descrivono il comportamento del design: Assunzioni, clock e reset;

La verifica FPV ha il compito di verificare le proprietà del design di cui si vuole validare il funzionamento attraverso tutti i possibili comportamenti dei segnali in funzione delle condizioni imposte.

Come è possibile notare dalla Figura 1-4, una volta forniti gli ingressi necessari alla FPV, essa fornisce in uscita:

1) **Proprietà passate**: Le proprietà che sono state validate positivamente e quindi "passate", ovvero il comportamento del design di cui si voleva avere conferma è confermato essere corretto.
2) **Proprietà fallite**: Le proprietà che sono state validate ma non sono "fallite", ovvero il comportamento del design che ci si aspettava non è corretto ed è stato quindi identificato un "contro-esempio" che spiega il motivo per il quale non è possibile il funzionamento atteso.
3) **Proprietà inconclusive**: Proprietà di cui non è stato possibile validare il comportamento, quindi risultati "inconclusivi". Sono quindi elencate quelle serie di proprietà di cui non è possibile confermare che siano corrette come anche che non siano corrette.

1.3.3 Verifica strutturale

Nel contesto della verifica RTL, la verifica strutturale non ha il compito di effettuare una validazione funzionale del comportamento del design ma ha il compito di identificare eventuali errori di struttura presenti nel codice RTL. Tra le macrocategorie di verifica RTL illustrate, la verifica strutturale è il ramo principale trattato dal lavoro di tesi. Esistono diverse tipologie di verifica strutturale e ne vengono riportate tre, dove le ultime due, il Clock Domain Crossing e il Reset Domain Crossing, costituiscono il cuore del lavoro di tesi.

- **Linting**: La verifica "Linting" è un processo di analisi del DUT descritto in codice RTL e quest'ultima fa parte della famiglia della verifica strutturale. Attraverso il linting, si procede con il verificare la qualità del codice scritto utilizzando come riferimento le diverse linee guida e regole di buona prassi per poter scrivere in maniera corretta del codice RTL. Il linting accompagna i team di verifica ancor prima della simulazione, ovvero nel momento stesso in cui il codice RTL è pronto. Il processo di linting consente di proseguire con gli step successivi del flusso di design con un codice RTL "pulito", ovvero un processo che consente di ridurre al minimo errori di sintesi, evitare codice non sintetizzabile ed evitare eventuali problemi funzionali dovuti ad errori di scrittura del codice. Esempi di errori di scrittura che possono riflettersi in problemi di sintesi possono essere, ad esempio, collegamenti dichiarati ma non connessi, loop combinatori non voluti, costrutti non sintetizzabili ed etc.. La verifica linting consente quindi di risaltare sin da subito errori nel codice, dando quindi la possibilità di effettuare correzioni o miglioramenti. In un breve riassunto, la verifica linting è un'analisi statica, che non esegue il codice scritto, ma lo analizza facendo riferimento ad una serie di regole di buona scrittura di codice. Grazie ad esso, si possono evitare errori non voluti o codice scritto in modo non corretto, il quale può riflettersi in errori funzionali non facili da gestire negli stadi successivi.

- **Clock Domain Crossing**: Altra verifica strutturale, di cui si approfondisce l'argomento nei capitoli successivi, è la verifica CDC. La verifica CDC avviene e viene conclusa sul design descritto a livello RTL. Dunque, appena è disponibile il codice RTL, è possibile procedere con la sua analisi CDC. L'acronimo CDC sta per "Clock-Domain-Crossing" e come è possibile intendere dal suo nome, esso si riferisce all'attraversamento di segnali/dati tra domini di clock differenti. L'obiettivo della verifica strutturale CDC è quello di identificare i domini di clock presenti nel design, analizzare i percorsi CDC e le strutture di sincronizzazione presenti nel design, come anche rilevarne la loro eventuale assenza. Il compito principale di tale verifica è quindi di identificare tutti i percorsi CDC, ovvero, tutti quei segnali che vengono trasmessi da un dominio di clock per essere ricevuti da un altro e che possono essere soggetti ad errori funzionali dovuti

ad assenze di schemi di sincronizzazione, utilizzo non corretto dei sincronizzatori implementati, casi di convergenza, divergenza e glitch strutturali. Grazie all'analisi CDC, vi è la possibilità di minimizzare gli errori dovuti alla metastabilità e alla sua propagazione come anche problemi legati alla possibile propagazione di impulsi spuri. Nasce dunque l'esigenza di introdurre tale verifica strutturale nel processo di verifica standard. Questo perché, con l'aumento del numero di domini di clock di un sistema digitale, anche solo dovuto per la presenza di diverse interfacce di I/O, possono aumentare il numero di problemi legati alla nascita della metastabilità.

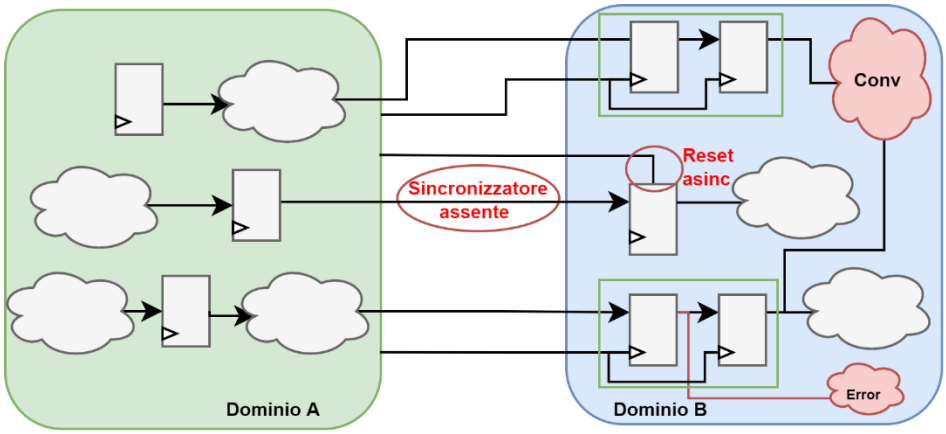

Figura 1-5: Verifica Strutturale CDC & RDC

- **Reset Domain Crossing**: È anche necessario introdurre la verifica RDC, anch'essa approfondita nei capitoli successivi, che si affianca alla verifica CDC. L'acronimo RDC sta per "Reset Domain Crossing" ed è anch'essa una verifica strutturale. L'analisi RDC ha il compito di identificare tutti i segnali di resets presenti nel design che attraversano domini di clock differenti come anche segnali di reset che appartengono ad uno stesso dominio di clock, ma che possono portare ad errori funzionali per via della loro errata implementazione. Durante tale verifica strutturale si procede quindi con l'identificazione di tutti i percorsi RDC al fine di analizzare le strutture di sincronizzazione di reset implementate nel design e se quest'ultime sono implementate in modo corretto, come anche identificare quei percorsi RDC che ne sono sprovvisti. Anche la verifica RDC, come la CDC, ha come compito principale il minimizzare gli errori funzionali dovuti alla metastabilità. La verifica RDC risulta anch'essa necessaria per poter assicurare che il proprio design non soffra di errori funzionali dovuti alla propagazione della metastabilità. Con l'aumento del numero di domini di clock all'interno di un sistema digitale si affianca spesso anche l'aumento del numero di domini di reset o del numero di segnali di reset asincroni che possono causare, se non correttamente gestiti, malfunzionamenti e comportamenti non aspettati.

2 Lavoro di tesi

Il seguente capitolo ha lo scopo di illustrare l'obiettivo del lavoro di tesi eseguito, dove il flusso di verifica progettato si localizza all'interno dei passaggi produttivi di un design e dove può essere classificato all'interno della verifica RTL. Viene anche introdotto il tool di analisi strutturale e formale utilizzato.

2.1 Lo scopo di tesi

Il flusso di analisi CDC-RDC si localizza all'interno del passaggio produttivo di design definito Verifica RTL ed ha quindi lo scopo di introdurre una metodologia il cui compito è quello di ridurre il numero di errori strutturali e funzionali CDC-RDC presenti all'interno dell'RTL, errori causati da un'errata gestione dei ClockDomainCrossing e dei ResetDomainCrossing. Dunque, si vuole progettare un flusso di analisi che consente di irrobustire sin da subito il design in produzione, ovvero riuscire a identificare già nei primi stadi di produzione i problemi che possono sorgere in presenza di molteplici asincronie. Come descritto nella sezione 1.3, la verifica RTL si distingue principalmente in tre categorie: verifica funzionale, verifica strutturale e verifica formale. L'obiettivo di tesi si incentra su tutte e tre le categorie di verifica appena riportate. È necessario abbinare alla verifica strutturale le capacità introdotte dagli engine formali e dalle simulazioni dinamiche per poter raggiungere l'obiettivo di progettare un flusso completo di verifica CDC-RDC strutturale e funzionale, col fine di escludere dai propri design errori legati alla presenza di asincronie.

Lo studio eseguito ha quindi come traguardo quello di dar vita ad un flusso di verifica strutturale e funzionale CDC e RDC con un approccio formale, questo al fine di contrastare il continuo aumento di errori funzionali dovuti alla presenza di sempre più molteplici domini di clock e domini di reset all'interno dei moderni design. Si vuole dunque illustrare un metodo di verifica, suddiviso in un'analisi strutturale e in un'analisi funzionale, che ha lo scopo di identificare e rimuovere sin dalle prime forme dei modelli di design RTL la serie di problemi funzionali introdotti dalla presenza dei Clock-Domain-Crossing e dei Reset-Domain-Crossing.

- **Il problema**

I moderni SOC puntano quindi ad essere GALS complessi: Globally Asynchronous Locally Synchronous[3], quindi sistemi caratterizzati da molteplici domini di clock asincroni. La complessità presente nei moderni design è dovuta alla loro continua ottimizzazione in termini di area, consumi e prestazioni, con l'obiettivo di raggiungere dispositivi sempre più performanti ma con dimensioni e consumi sempre più ridotti. Al fine di avere consumi inferiori ed al tempo stesso frequenze di clock tali da ottenere le prestazioni volute, si progettano i design per poter lavorare con frequenze diverse, tipicamente veloci per le operazioni che necessitano di velocità e lente per quelle operazioni di cui non si ha necessità di ottenere risultati nel minor tempo possibile. In tali sistemi sono anche presenti differenti tipologie di interfaccia di I/O, principali cause di asincronia, ed è possibile intuire che tali design sono caratterizzati da un continuo trasferimento di segnali e dati tra i differenti domini di clock asincroni. Uno dei primi problemi che possono sorgere dal passaggio di segnali tra tali domini è la continua violazione dei tempi di setup/hold, violazioni che si riflettono nel rendere i segnali in questione metastabili.

- **Le problematiche CDC e RDC riscontrabili in un design asincrono**

All'interno di un design asincrono, a causa di una gestione errata dei clock domain crossing e dei reset domain crossing, possono sorgere le seguenti problematiche:

  - **Metastabilità**: Oscillazioni non volute da parte di FlipFlop a fronte dei segnali di ingresso che effettuano transizioni con relativa violazione dei tempi di setup/hold;
  - **Glitches**: La presenza di logica combinatoria nei percorsi CDC comporta ad una generazione di impulsi spuri e ad una loro possibile acquisizione da parte dei domini di destinazione con relativi risultati non corretti;
  - **Incoerenza nei dati**: Convergenza di segnali correlati nel tempo ma sincronizzati in maniera indipendente;
  - **Perdita di dati**: Transizione di dati non controllata durante un meccanismo di sincronizzazione oppure non corretta valutazione del tempo necessario al fine del campionamento da parte del dominio di clock di destinazione;

- **Identificare i problemi CDC sin dall'RTL**

L'obiettivo è quindi quello di eseguire e terminare tale flusso già all'interno dello stadio della verifica RTL ed evitare la presenza di problemi CDC-RDC nello stadio di verifica post-layout. Il problema di riscontrare errori CDC-RDC, nello stadio produttivo della verifica post-layout, risiede nell'investimento di tempo necessario per poter risolvere i problemi riscontrati. Come è possibile notare nella Figura 2-1 a sinistra, nel momento in cui vengono identificati tali problemi nella fase di verifica post-layout, è necessario eseguire nuovamente il flusso di design dai suoi primi stadi di produzione. Dunque, riscontrati i problemi CDC-RDC, è necessario eseguire nuovamente i passaggi di sintesi, Place&Route e relative verifiche. È possibile intuire quanto sia oneroso riscontrare un problema CDC-RDC in tale stadio produttivo e di quanto tempo è necessario investire per poter verificare di aver risolto tali problemi. È anche necessario sottolineare che non è possibile garantire che durante la verifica post-layout siano identificati tutti i problemi CDC-RDC presenti nel design, in quanto con tale verifica non è possibile coprire in maniera esaustiva i problemi CDC e RDC.

Invece, riferendosi alla Figura 2-1 a destra, è possibile notare che, introducendo il flusso progettato di verifica CDC-RDC all'interno dello stadio di verifica RTL, si riduce drasticamente il tempo necessario per poter identificare e risolvere la presenza di tali problemi come anche si riduce il numero di passaggi da eseguire nuovamente a fronte di tali errori. Essenzialmente, è possibile rilevare i problemi CDC-RDC già all'interno della verifica del codice RTL, consentendo quindi di avere una riduzione del tempo necessario da investire per poter identificare, risolvere e controllare che i problemi riscontrati siano stati risolti. Dunque, introducendo il flusso progettato all'interno dello stadio produttivo di verifica RTL, è possibile identificare sin da subito tutti i problemi CDC-RDC presenti all'interno del design, effettuare la modifica al codice RTL per poter risolvere i problemi identificati ed è possibile avere all'interno dello stesso stadio produttivo il riscontro di aver risolto tali problemi. È quindi possibile riuscire a identificare i problemi CDC-RDC presenti nel proprio design ancora prima di iniziare la fase di sintesi di quest'ultimo, consentendo quindi di intraprendere il flusso di design con un RTL "pulito" da errori CDC-RDC. Quindi, tramite il flusso CDC, RDC & Metastability Injection è possibile spostare nello stadio di Verifica RTL, con più efficacia, la verifica inerente ai problemi dovuti alle molteplici asincronie.

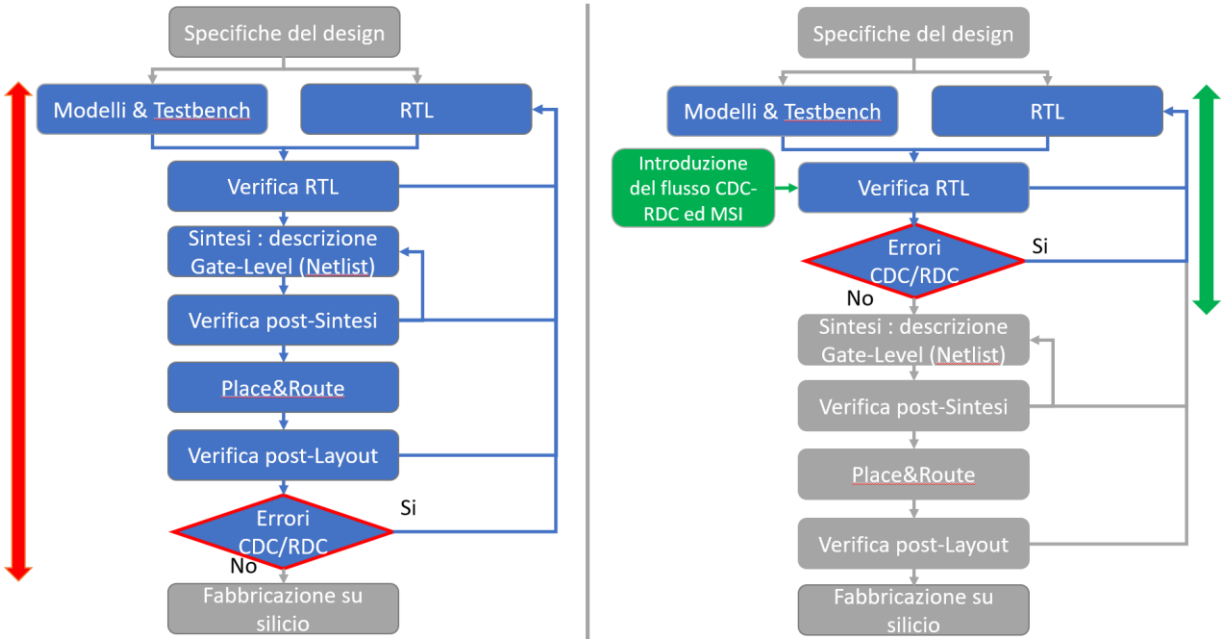

Figura 2-1: Confronto dei passaggi produttivi da rieseguire nel momento in cui si riscontrano problemi CDC-RDC. A sinistra non è presente il flusso CDC-RDC progettato; A destra è presente il flusso CDC-RDC progettato;

Quindi, il lavoro di tesi ha lo scopo di formare un flusso di verifica strutturale CDC e RDC che consenta di raggiungere un "CDC-Clean RTL Signoff" integrabile sin da subito nei primi passaggi del flusso di design RTL.

- **Perché l'approccio formale?**

Come spiegato precedentemente, non ci si pone solamente all'interno della verifica strutturale ma per poter raggiungere l'obiettivo di dar vita ad un flusso completo di signoff RTL CDC, sia dal punto di vista strutturale che dal punto di vista funzionale, si introduce anche il supporto della verifica formale.

All'analisi strutturale CDC-RDC viene affiancata la verifica formale, le cui capacità consentono di andare ben oltre la sola verifica strutturale di un design, andando ad integrare la possibilità di effettuare verifiche funzionali grazie all'ausilio degli engine formali.

La verifica formale viene quindi utilizzata con lo scopo di validare le assunzioni effettuate nel corso dell'analisi strutturale e con l'obiettivo di giustificare le violazioni strutturali, che non si ritengono essere problematiche CDC/RDC, riportate durante l'esecuzione del flusso. Attraverso la verifica formale si ha quindi l'obiettivo di validare i protocolli CDC/RDC adottati per le varia strutture di sincronizzazione e si ha lo scopo di consentire una maggiore completezza dell'analisi strutturale, la quale da sola non può osservare la corretta implementazione dei protocolli e proprietà adottate. Ad esempio, può essere verificata una corretta implementazione di una struttura FIFO attraverso la verifica strutturale, ma non risulta possibile effettuare una validazione delle sue proprietà (L'applicazione della codifica gray, gestione dei puntatori, il sistema di "full" e "empty") se non attraverso la verifica funzionale. Dunque, nasce la possibilità di effettuare asserzioni riguardanti schemi di sincronizzazione e proprietà non convenzionali che possono validare la corretta sincronizzazione attraverso l'ausilio degli engine formali.

- **Una maggiore robustezza del design in presenza di metastabilità**

Infine, si prosegue con l'aggiungere all'analisi CDC-RDC un'ulteriore forma di validazione della robustezza funzionale del design in caso di metastabilità ed impulsi spuri nel trasferimento segnali/dati tra domini di clock. Si introduce al flusso la possibilità di iniettare metastabilità e glitch nei percorsi CDC/RDC identificati. Questo consente di avere una maggiore confidenza del funzionamento dei protocolli e schemi adottati per la sincronizzazione di segnali/dati. Essenzialmente, è possibile validare che sia i protocolli che le strutture implementate sono effettivamente in grado di affrontare e superare gli effetti della metastabilità e glitch. Questo è possibile grazie alla modellazione degli effetti di quest'ultimi attraverso la loro iniezione nelle coppie CDC/RDC identificate, con approccio formale e con simulazione dinamica.

Dunque, si vuole raggiungere l'obiettivo di validare che non vi siano problemi funzionali nei protocolli adottati anche in funzione della variazione di latenza che viene introdotta dall'iniezione della metastabilità.

Essenzialmente, si introduce la possibilità di modellare gli effetti della metastabilità come anche l'iniezione di glitches nei percorsi CDC/RDC che possono causare problemi funzionali al design. Viene quindi introdotta la possibilità di portare allo scoperto eventuali problemi che si ritengono essere risolti tramite i protocolli adottati. Questa modellazione consente quindi di disporre di un ulteriore sistema per poter far emergere e risolvere errori, consentendo di avere una maggiore robustezza ed affidabilità del design dal punto di vista CDC-RDC.

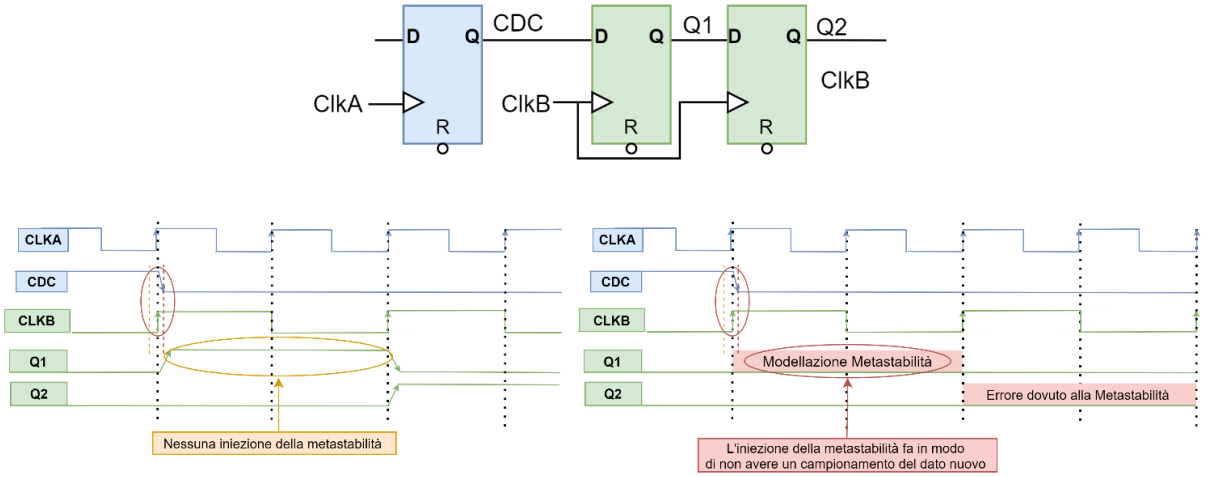

Figura 2-2: Esempio 1 di iniezione della metastabilità

Al fine di modellare la metastabilità, nel momento in cui vi è un trasferimento di segnali CDC, viene impostato un valore casuale tra il dato precedente e il dato attuale e viene dunque modellata un'incertezza di un colpo di clock nel percorso CDC in questione. Nel momento in cui avviene effettivamente un errore funzionale dovuto quindi alla latenza modellata attraverso l'iniezione della metastabilità, si conferma di avere un percorso CDC non robusto.

Nella Figura 2-2, si prende in considerazione uno schema di sincronizzazione standard MultiFlop caratterizzato da due FlipFlop. Dal punto di vista strutturale non vi sono problemi nel percorso CDC, essendo presente una struttura di sincronizzazione. Si suppone di aver scritto e validato, attraverso la verifica formale, le proprietà e asserzioni che descrivono il comportamento dei segnali e protocolli coinvolti nel percorso CDC e che quindi si è ottenuto il risultato riportato nella parte di sinistra della Figura 2-2. Successivamente, si vuole avere una maggiore sicurezza della robustezza di tali protocolli

procedendo con l'iniezione della metastabilità, al fine di avere una validazione anche in presenza della latenza introdotta dalla sua modellazione. Dunque, nella parte di destra è possibile notare che la modellazione della metastabilità fa in modo di non avere il campionamento del nuovo dato e che l'uscita del primo flip flop della struttura di sincronizzazione converga al valore precedente anziché al successivo. Dunque, si nota che per via della metastabilità non vi è stato il campionamento del valore logico 1, come invece avviene nel comportamento aspettato. Dunque, il dato è stato perso portando quindi ad avere un errore funzionale.

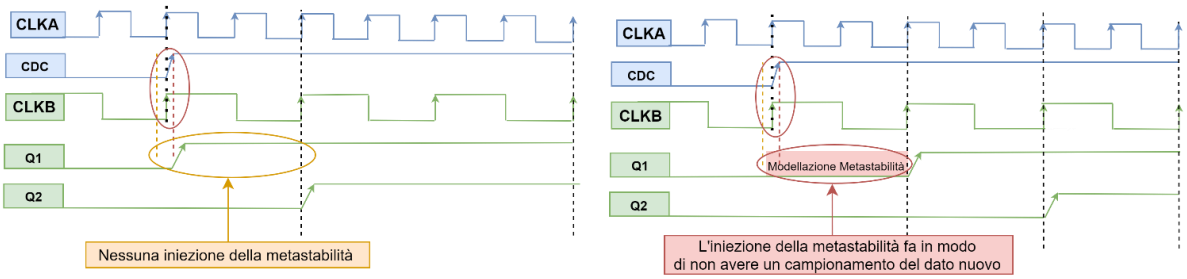

Figura 2-3: Esempio 2 di iniezione della metastabilità

Nella Figura 2-3 viene riportato un caso differente. È possibile notare che nel caso in questione, l'iniezione della metastabilità porta ad avere l'introduzione di un colpo di clock di latenza rispetto al comportamento assunto senza metastabilità. Dunque, nel caso in questione, il protocollo adottato per il segnale CDC trasmesso da parte del dominio del clock A può essere considerato valido anche in presenza di metastabilità. Questo perché l'unica possibilità che può essere introdotta da parte della metastabilità è quella di dover aspettare una latenza pari ad un colpo di clock di sistema.

- **Flusso completo CDC, RDC e iniezione della metastabilità**

Viene adesso riportato, nella Figura 2-4, il diagramma di flusso realizzato e seguito per poter identificare e risolvere problemi CDC-RDC all'interno del DUT in questione. Tale diagramma rappresenta il flusso affiancato alla fase produttiva della Verifica RTL.

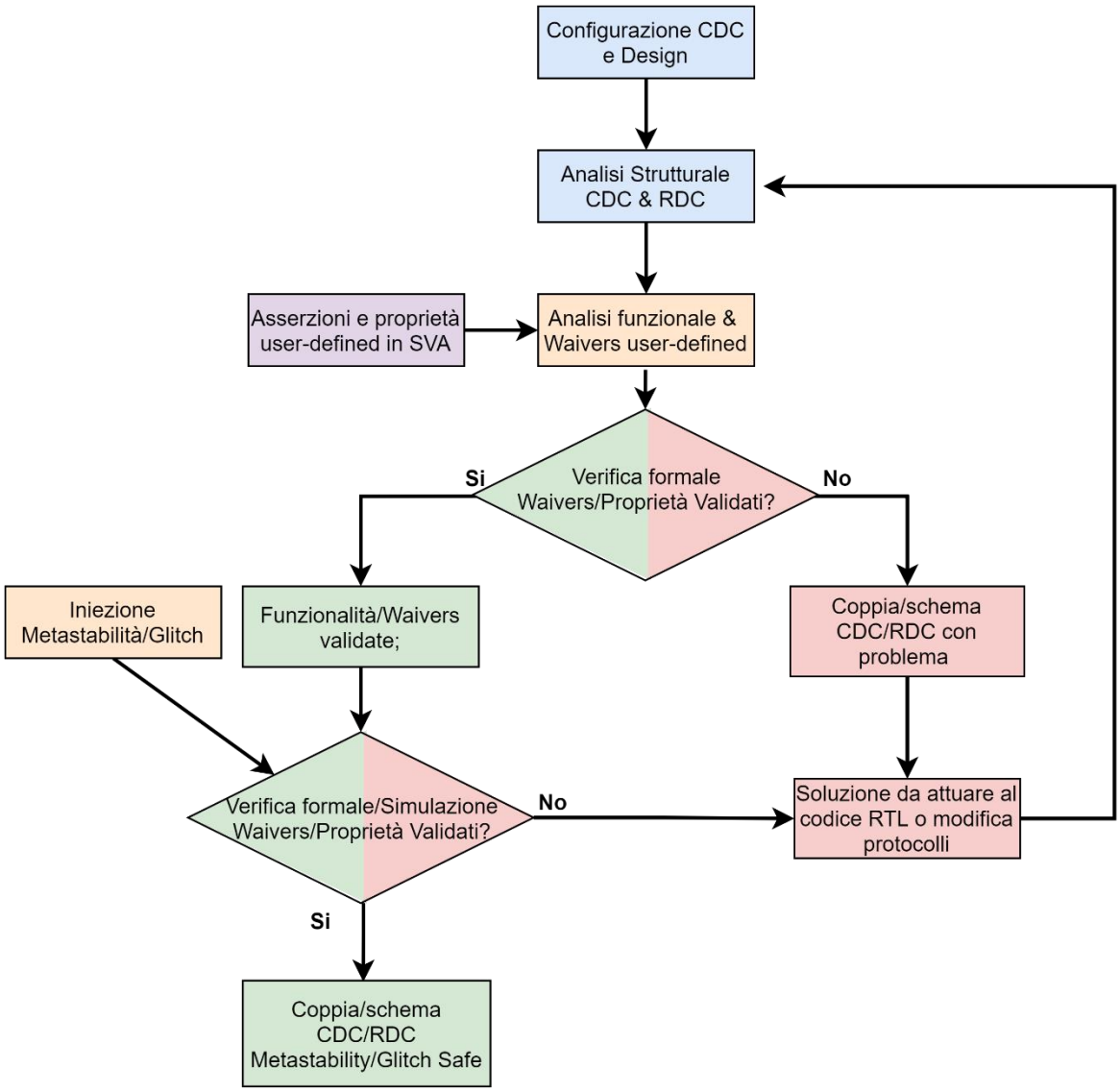

Figura 2-4: Flusso progettato

Il flusso riportato è costituito dai passaggi di analisi strutturale e passaggi di analisi funzionale e viene integrata l'iniezione della metastabilità, sia con approccio formale che in simulazione dinamica. Le diverse fasi del flusso vengono riprese e spiegate nel dettaglio nei capitoli successivi, mentre adesso viene riportata un'introduzione del ruolo svolto da ognuno dei passaggi presenti nel diagramma.

Configurazione CDC e Design: Esso rappresenta il primo passaggio strutturale CDC-RDC. Tale fase svolge un ruolo fondamentale all'interno del flusso in quanto consente di avere un'analisi più accurata del design come anche ottenere risultati più accurati. Essenzialmente, si procede con l'effettuare la configurazione delle porte di ingresso e di uscita, la configurazione dei segnali interni al design di cui si conosce il comportamento e la configurazione dei reset e clock presenti. Questa configurazione consente di avere una corretta identificazione dei domini di reset e domini di clock presenti nel DUT.

Effettuare una sempre più precisa configurazione del design consente l'esclusione di casi CDC-RDC non problematici dai risultati riportati dall'analisi, ottenendo quindi l'identificazione di reali violazioni presenti all'interno del design.

Analisi strutturale CDC & RDC: Fase in cui avviene l'analisi strutturale CDC e RDC effettiva. In tale passaggio del flusso vengono identificate le varie problematiche strutturali come l'assenza di meccanismi di sincronizzazione, casi di convergenza, divergenza, glitch strutturali, strutture di sincronizzazione non correttamente implementate, tutti problemi CDC-RDC eventualmente presenti all'interno del design. Essenzialmente, vengono identificati i domini presenti, le coppie CDC-RDC, gli schemi di sincronizzazione e vengono riportate le violazioni risultanti dall'analisi strutturale. In tale passaggio vi è quindi la necessità di analizzare le violazioni ottenute e discriminare quali tra le violazioni risultanti sono effettivamente dei problemi CDC/RDC e quali in realtà possono non essere considerate problematiche. Una volta identificate le violazioni non ritenute problematiche all'interno del design, si effettuano i Waive di quest'ultime (Waiver: La rimozione di una violazione strutturale. Ovviamente, tale rimozione deve essere giustificata dalla presenza di un protocollo per quel percorso CDC/RDC identificato).

Analisi funzionale & Waivers user-defined: Esso rappresenta il primo passaggio dell'analisi funzionale all'interno del flusso. Dal passaggio precedente sono state quindi identificate le coppie CDC-RDC presenti all'interno del design, i relativi schemi di sincronizzazione implementati oppure le eventuali assenze di quest'ultimi. Vengono anche identificate quelle violazioni che non risultano essere reali problemi all'interno del design, questo perché possono essere presenti protocolli o configurazioni previste dalle specifiche. Durante la verifica strutturale non sono presi in considerazioni i protocolli adottati all'interno del design e questi devono essere validati nella fase dell'analisi funzionale. Dunque, in questo step vengono definite e descritte manualmente, e in minor parte descritte automaticamente da parte del tool, le proprietà e asserzioni che caratterizzano i protocolli di sincronizzazione all'interno del design e tali proprietà devono essere validate attraverso l'ausilio della verifica formale. Questo viene effettuato col fine di confermare che le violazioni strutturali identificate non sono un reale problema. L'analisi funzionale deve quindi essere sempre accompagnata all'analisi strutturale, poiché quest'ultima non garantisce la corretta implementazione dei protocolli adottati ma identifica solamente i problemi di implementazione strutturale nei percorsi CDC-RDC presenti. Dall'analisi funzionale è possibile ottenere due risultati:

1) **Funzionalità/Waivers validate**: Il waive effettuato di una violazione risulta essere corretto in seguito alla validazione del protocollo di sincronizzazione adottato. Dunque, la funzionalità delle proprietà descritte e il waiver effettuato risultano essere corretti.

2) **Coppia/Schema CDC/RDC con problema**: Il waive effettuato di una violazione risulta essere non corretto in seguito ad un esito negativo della verifica del protocollo di sincronizzazione adottato. Quindi, è stata trovata almeno una condizione tale per cui il protocollo adottato non rispetta il comportamento aspettato ed è quindi necessario effettuare delle modifiche RTL al fine di risolvere il problema identificato. Bisogna quindi ricorrere ad effettuare una modifica del protocollo adottato oppure ricorrere ad una corretta implementazione di uno schema di sincronizzazione per poter gestire il percorso CDC/RDC critico.

Iniezione della metastabilità e glitch:

Dopo aver validato il corretto comportamento delle proprietà e protocolli di sincronizzazione adottati all'interno del proprio design attraverso l'utilizzo di assunzioni e asserzioni in verifica formale, si prosegue con l'avere una maggiore sicurezza del funzionamento di tali protocolli attraverso la loro validazione in presenza di metastabilità e glitch. Essenzialmente, si procede con il confermare che le

asserzioni e proprietà descritte e validate nello step precedente siano ancora valide anche in presenza di metastabilità. Attraverso la verifica funzionale con simulazioni dinamiche è possibile procedere con il confermare che i protocolli siano validi anche in presenza di glitch. Dunque, si procede con l'iniettare gli effetti della metastabilità attraverso un approccio formale e in simulazioni dinamica e l'iniezione dei glitch in simulazione dinamica, con lo scopo di verificare che i protocolli di sincronizzazione adottati siano effettivamente validi. I risultati ottenibili da parte di tale validazione sono due:

1) **Funzionalità e Coppia CDC/RDC Metastability/Glitch Safe:** Il protocollo e struttura adottati lungo il percorso CDC e RDC in questione può essere ritenuto robusto in presenza della metastabilità ed in presenza di glitch.
2) **Coppia/Schema CDC/RDC con problema in presenza di Metastability/Glitch:** Il protocollo e struttura adottati lungo il percorso CDC e RDC rispettano il comportamento aspettato attraverso la validazione in verifica formale ma non risultano essere robusti in presenza di glitch e metastabilità. È dunque necessario effettuare una correzione del protocollo e dell'RTL al fine di avere una maggiore robustezza a fronte di tali problemi.

2.2 Introduzione ai tool utilizzati

L'analisi effettuata sul design in questione è stata possibile grazie ad un tool di analisi CDC e un tool di analisi formale ed un simulatore digitale.

Il tool utilizzato per poter effettuare l'analisi CDC-RDC fornisce una soluzione completa per poter effettuare una corretta analisi dell'RTL che definisce il DUT, comprensiva di analisi strutturale, funzionale e convergenza (quest'ultima sempre compresa all'interno della tipologia di analisi strutturale). Quindi, è possibile effettuare un'analisi completa dei problemi CDC-RDC e rendere il proprio codice RTL "pulito" da tali problemi.

I passaggi madre che offre il tool sono principalmente sei, suddivisi in analisi strutturale e funzionale.

- **Analisi strutturale**
  1) Definizione del design e gestione dell'ambiente CDC-RDC
     - Impostazione dei clock e reset;
     - Configurazione porte e segnali;
     - Identificazione dei domini di clock, coppie CDC e RDC;

Tali passaggi definiscono uno stadio importante del flusso di analisi, in quanto una corretta configurazione del design consente di avere risultati più "puliti", ovvero si esclude un numero elevato di false violazioni che possono portare solamente ad avere risultati "rumorosi".

  2) Analisi strutturale CDC
     - Identificazione della presenza o meno di schemi di sincronizzazione standard;
     - Possibilità di definire schemi di sincronizzazione non convenzionali;
     - Identificazione di percorsi di convergenza e divergenza;
  3) Analisi strutturale RDC
     - Identificazione della presenza o meno di schemi di sincronizzazione di reset;

- **Analisi Funzionale**
  1) Generazione e validazione funzionale dei protocolli che definiscono le strutture di sincronizzazione;
  2) Validazione dei protocolli e proprietà non convenzionali, meglio definite “user-defined”, al fine della validazione dei waivers effettuati manualmente;
  3) Processo di iniezione della metastabilita’ attraverso engine formale e metastabilità e glitch in simulazione dinamica;

Il tool di verifica formale rende invece possibile avere un approccio formale nella validazione di protocolli e proprietà non convenzionali definite da parte dell’utente, meglio definite “user-defined”. Essenzialmente, il tool consente di effettuare una validazione dei protocolli e proprietà presenti all’interno di un design, consentendo quindi di ottenere una conferma della corretta rimozione di quelle violazioni riportate durante l’analisi strutturale CDC-RDC che si ritengono non essere reali problematiche.

Al fine di procedere con l’analisi dei risultati ottenuti dall’iniezione della metastabilità e glitch in simulazione dinamica e verificare il corretto comportamento dei protocolli è stato utilizzato un tool di simulazione RTL.

3 Introduzione al design sotto analisi

Il design sotto analisi è caratterizzato da un ASIC, un sensore MEMS (Micro Electro-Mechanical Systems) e dalla presenza di 3 interfacce di uscita digitali I3C, I2C e SPI per poter comunicare con il mondo esterno. Nella Figura 3-1 viene rappresentato lo schema a blocchi del dispositivo.

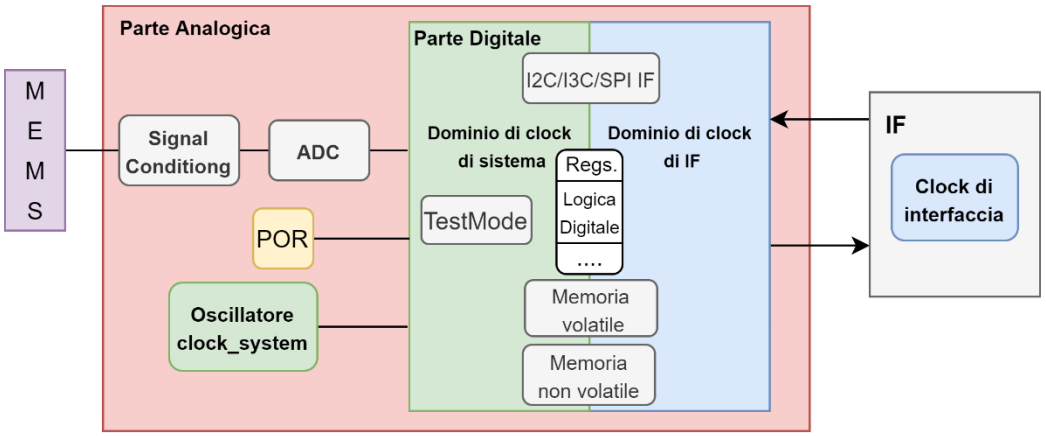

Figura 3-1: Schema a blocchi del dispositivo sotto analisi

Il punto di partenza del dispositivo è il sensore MEMS (Micro Electro-Mechanical Systems) da cui avvengono le misurazioni meccaniche che vengono processate dalla parte digitale del dispositivo. Le misurazioni meccaniche ottenute sono condizionate e adattate dalla parte di analogica del sistema, questo per poter effettuare successivamente la loro conversione dal mondo analogico al mondo digitale e quindi avere il loro processamento da parte del design. Dunque, i segnali analogici ottenuti sono forniti al convertitore analogico/digitale ADC. Da quest'ultimo sono dunque ottenuti i dati digitali pronti per essere processati dalle diverse catene di lettura presenti nel dispositivo. Il sistema è caratterizzato da due tipologie di memorie, una memoria non volatile ed una memoria volatile ed è caratterizzato da un banco di registri utili allo scambio di informazioni tra le interfacce presenti e il dispositivo. Il banco di registri è anche utilizzato per poter effettuare le configurazioni delle diverse funzionalità del design. Il dispositivo ha il compito di effettuare le misurazioni necessarie ed elaborare i dati misurati, questo col fine di ottenere i risultati desiderati da parte delle diverse funzionalità presenti nel design ed effettuare una comunicazione attraverso una delle interfacce digitali presenti nel design.

- **Complessità del design identificata dal tool**

Dall'elaborazione ed analisi CDC del design in questione sono stati riportati alcuni dei parametri utili a fornire un'idea della complessità del design su cui è stata eseguita l'analisi. Sono stati identificati:

1) 1390 FlipFlop;
2) 2 domini di clock asincroni, dominio di clock di sistema e dominio di clock di interfaccia;
3) 187 segnali in ingresso al pin di clock dei FlipFlop, ovvero la somma di tutti quei segnali atti alla funzione di clock ed appartenenti al dominio di clock di sistema e dominio di clock di interfaccia;
4) 254 coppie CDC, ovvero il numero di coppie di FF sorgente-destinazione appartenenti a domini di clock asincroni tra loro;
5) 1 segnale di reset asincrono ai domini di clock presenti e nominato POR;

3.1.1 Caratteristiche del design utili all'analisi CDC

Da come è possibile intuire dallo schema a blocchi riportato nella Figura 3-1, il dispositivo è quindi caratterizzato da un oscillatore che costituisce il clock di sistema. Il dispositivo comunica con un'interfaccia di I/O caratterizzata invece da un clock di interfaccia asincrono a quello di sistema. Viene adesso riportato un elenco di caratteristiche e componenti del design:

- **Domini di clock:** Si identificano due principali domini di clock asincroni all'interno della parte digitale del dispositivo: il dominio di clock di sistema e il dominio di clock di interfaccia. Quest'ultima ha lo scopo di comunicare con i registri presenti all'interno del sistema ed è quindi una delle principali motivazioni di asincronia presenti nel dispositivo.

- **Interfacce**: Sono presenti tre tipologie di interfacce: SPI, I2C e I3C. Le interfacce presenti non possono essere utilizzate contemporaneamente, ovvero è possibile selezionarne una sola. Tali interfacce vengono utilizzate al fine di effettuare scritture/letture dei dati nel banco di registri presente nel design. Le interfacce rappresentano la principale causa di asincronia nel design.

- **Blocco registri di configurazione**: Il design è caratterizzato da un blocco di registri utili alla configurazione delle diverse funzionalità che costituiscono il cuore del dispositivo. Tale blocco viene scritto sia attraverso il clock di interfaccia che attraverso il clock di sistema ed i suoi segnali di uscita sono dislocati nei diversi componenti del design. Tali componenti appartengono sia al dominio di clock di sistema che al dominio di clock di interfaccia.
  Sono presenti diverse tipologie di registri:

    - **Registri di test**: utilizzabili per poter abilitare test digitali/analogici ed anche per poter impostare la DFT presente nel dispositivo.
    - **Registri di configurazione**: utilizzabili per poter configurare le diverse funzionalità analogiche/digitali del dispositivo e impostare le caratteristiche della logica digitale.
    - **Registri di output;**

- **Memorie**: Sono presenti all'interno del design due tipologie di memoria: una memoria volatile e una memoria non volatile. Esse sono Proprietà Intellettuali IP integrate nel design e quest'ultime non sono soggette ad alcun tipo di verifica CDC-RDC essendo già esternamente validate. Essenzialmente non si è interessati al loro funzionamento interno e alla loro validazione CDC-RDC interna, ma si è esclusivamente interessati ai loro segnali di ingresso e di uscita e come questi interagiscono all'interno del design. Deve essere esclusivamente validata la loro integrazione e la loro connettività all'interno del design.

- **TestMode**: Sono presenti diverse modalità funzionali nel dispositivo, ma sono anche presenti diverse modalità di test, sia digitali che analogiche, che vengono chiamate "TestMode". Quest'ultime non rappresentano delle modalità funzionali ma sono modalità utilizzate per poter effettuare diverse verifiche all'interno del dispositivo e per poter rilevare eventuali errori nel design. Tali modalità di test sono abilitabili e configurabili attraverso i diversi registri di configurazione presenti nel design.

- **POR**: Il sistema è caratterizzato da un solo segnale di reset esterno chiamato POR. Come è possibile intuire dal nome, tale circuito analogico ha il compito di asserire il segnale di reset al

fine di mantenere l'intera logica digitale del design ferma al valore di reset per tutto il tempo di avvio del sistema, ovvero per il tempo necessario per cui vengono raggiunte le alimentazioni corrette per il funzionamento del dispositivo. Tale segnale di reset è anche asserito nel momento in cui il livello di tensione scende al di sotto deli valori definiti per avere un corretto funzionamento del design. Nel momento in cui esso è asserito, il dispositivo non dispone di un clock di sistema e dunque è possibile considerare che i Flip Flop presenti nel design non ricevono il clock per tutto il tempo nel quale il segnale POR è asserito. Il POR è un segnale di reset asincrono sia al clock di sistema che al clock di interfaccia. Viene presa in considerazione tale caratteristica durante l'analisi RDC del design, questo col fine di rimuovere il numero più elevato possibile di violazioni legate al segnale di reset POR che non risultano essere reali problematiche RDC.

Sono state sottolineate queste caratteristiche e le componenti del dispositivo per poter gestire le prime configurazioni dell'ambiente di verifica CDC-RDC.

4 Flusso di analisi CDC - Clock Domain Crossing

Nel seguente capitolo viene approfondita e descritta nel dettaglio la verifica CDC. Vengono quindi descritte le problematiche che tale verifica ha il compito di affrontare e risolvere. Vengono adesso descritti nel dettaglio i passaggi che costituiscono l'intero flusso di analisi CDC.

4.1 I possibili problemi CDC

I problemi che possono sorgere dal Clock-Domain-Crossing non sono solamente legati alla metastabilità, ma sono presenti problemi legati alla nascita di glitch, incoerenza nei dati e la possibile perdita di dati. Dunque, i design caratterizzati da molteplici domini di clock asincroni devono essere strutturati correttamente al fine di affrontare tutte le problematiche CDC. Vi è la necessità di procedere con l'implementazione a livello RTL di strutture come handshake, sincronizzatori, codifiche gray ed altri possibili meccanismi di sincronizzazione al fine di non avere errori funzionali CDC e non soffrire della propagazione della metastabilità e glitch attraverso il design.

Si procede dunque con l'elencare e introdurre i possibili problemi CDC che possono essere riscontrati all'interno di un design caratterizzato da molteplici asincronie:

- **Metastabilità**

Come citato precedentemente, la metastabilità accade nel momento in cui avviene una violazione dei tempi di setup e dei tempi di hold di un FlipFlop.[3]

La metastabilità in un FlipFlop rappresenta essenzialmente uno stato di indecisione nel valore assunto dalla sua uscita Q. Una volta in cui il FF è entrato in tale stato metastabile, esso può rimanerci per un tempo non definito prima di convergere verso un valore logico stabile 0 o 1. Essenzialmente, la tensione di uscita di un FlipFlop entrato in questo stato può assumere un valore di tensione intermedio tra il valore di tensione assunto nello stato logico 0 e il valore di tensione assunto nello stato logico 1.

Effettuando un breve riepilogo, per il tempo di "setup" si intende il tempo minimo che deve intercorrere tra l'ultima variazione del segnale di ingresso e il fronte attivo del clock del FF in questione. Mentre, per il tempo di "Hold" si intende il minimo intervallo di tempo per cui il segnale in ingresso deve rimanere stabile dall'avvenuto fronte attivo del clock.

Nel momento in cui vengono rispettati tali tempi è possibile garantire che l'uscita del FF raggiunga il valore logico stabile dopo un tempo ben definito nominato "Clock to Output". Nel momento in cui non vengono rispettati tali tempi, non è possibile garantire che venga rispettato tale "Clock to Output" e si richiede un maggiore tempo, non definito, per poter muoversi verso un valore logico stabile dell'uscita del FlipFlop.

Quindi, una variazione del segnale all'interno di questo intervallo definito da questi due tempi comporta ad avere la generazione della metastabilità.

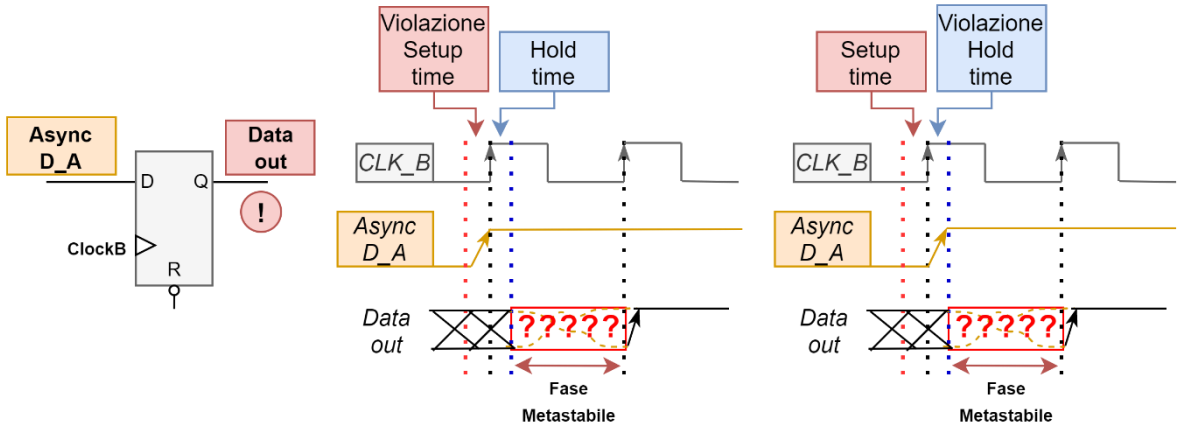

Figura 4-1: Violazione dei tempi di setup/hold

È possibile limitare gli effetti negativi della metastabilità grazie all'utilizzo di sincronizzatori ed è necessario sottolineare che tali strutture non sono legate esclusivamente ad evitare la propagazione della metastabilità nella logica, ma hanno anche lo scopo di evitare perdite e corruzione di dati che si rifletterebbero nel non funzionamento del dispositivo. Diventa obbligatoria l'implementazione di schemi di sincronizzazione, convenzionali o meno, per poter gestire segnali di controllo, dati e resets presenti in un sistema. Un metodo tradizionale come l'analisi statica del timing, che consiste nel verificare che i parametri temporali vengano rispettati nella rappresentazione del design attraverso la netlist, non consente la modellazione dell'effetto della metastabilità risultando quindi non sufficiente per la validazione del trasferimento dei dati tra i domini di clock asincroni. In questi casi, l'unico modo effettivo per rilevare eventuali errori CDC funzionali è la "post-layout verification stage", ovvero lo stadio di verifica del dispositivo su cui è stato effettuato il Place&Route. Ovviamente, identificare un errore in questo stadio del design è qualcosa che si vuole evitare. Effettuare modifiche in tale passo produttivo ha un costo decisamente significativo in termini di risorse e di tempo. Per poter identificare in tempo utile problemi effettivi nel passaggio di segnali tra domini di clock asincroni, è necessario spostarsi verso l'analisi CDC strutturale e funzionale.

- **Glitch**

Sorge la possibilità di avere il campionamento nei domini di destinazione asincroni di eventuali glitch, dove quest'ultimi possono insorgere a causa della presenza di logica combinatoria nei cammini CDC-RDC in questione. Il glitch è essenzialmente un impulso spurio la cui formazione è dovuta dalla presenza di percorsi combinatori caratterizzati da porte logiche con ritardi differenti ed è anche dovuta dalla presenza di percorsi di interconnessione con ritardi di propagazione non bilanciati. I glitch possono essere filtrati su cammini sincroni, essendo tali impulsi spuri esclusivamente legati ai ritardi di propagazione delle porte logiche e ai ritardi di propagazione, come anche è possibile verificare la loro risoluzione all'interno di un colpo di clock attraverso l'analisi statica del timing. Purtroppo, questo non è possibile garantirlo nel mondo CDC, mondo in cui vi è la possibilità di avere il campionamento di un glitch in un dominio di destinazione. Ovviamente, la formazione di un glitch porta ad avere errori funzionali di cui può essere difficile rilevare la causa.

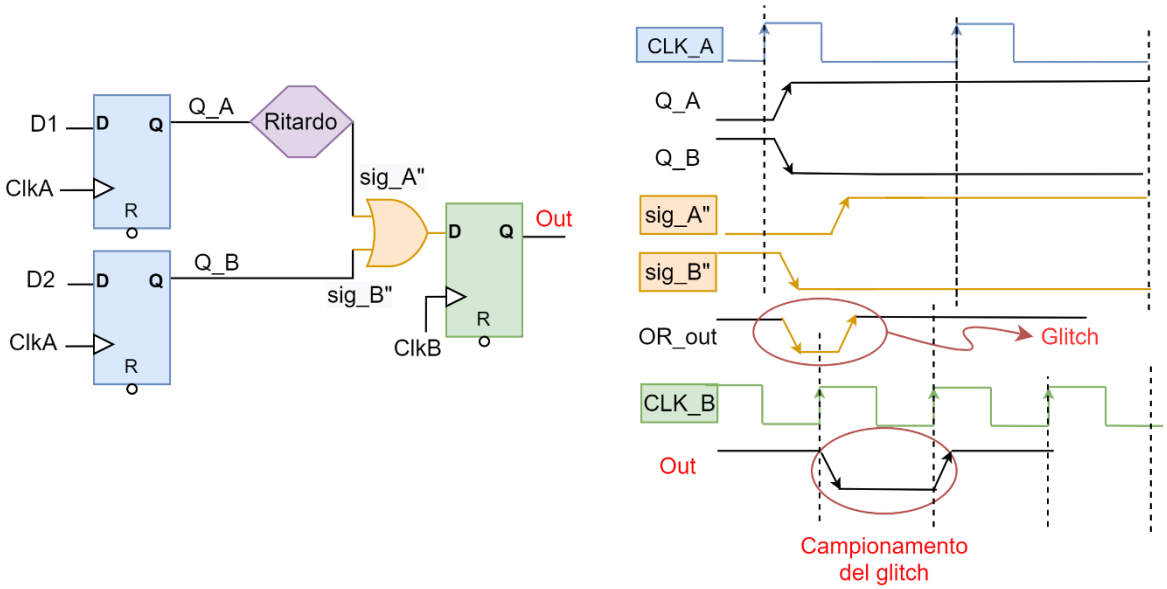

Figura 4-2: Glitch

- **Incoerenza dati**

Vi è la possibilità di avere problemi di incoerenza nel campionamento dei dati. Si può anche solamente pensare ai bit di un bus dati ed ogni bit dispone di un proprio sincronizzatore standard, ad esempio a due FF. Non vi è modo di assicurare che tutti i sincronizzatori riescano a campionare correttamente una commutazione dei bit, è possibile anche solo pensare alla presenza di un ritardo tra QA[0] e QA[1], dovuto ad esempio ad un differente tempo di clock to output o differente tempo di propagazione lungo le linee, il quale potrebbe non consentire un corretto campionamento dei segnali commutati contemporaneamente. Quindi, questo implica che è possibile avere alcuni sincronizzatori il cui "Q" riporta un dato nuovo mentre altri potrebbero riportare un dato precedente. Dunque, vi è la possibilità di avere la generazione di un risultato non coerente con una relativa propagazione di un dato non corretto attraverso il design. Sorge quindi la necessità di porre attenzione al tipo di meccanismo di sincronizzazione adottato in un percorso CDC, questo col fine di non avere la propagazione di dati non coerenti che possono portare ad un errore funzionale. Quindi, alcuni errori funzionali possono essere dovuti ad una non corretta implementazione di uno schema di sincronizzazione. Per poter avere la sincronizzazione di data bus è possibile, ad esempio, ricorrere a strutture come i "mux synchronizer". Tale struttura viene trattata nella sezione 4.2.2.2.

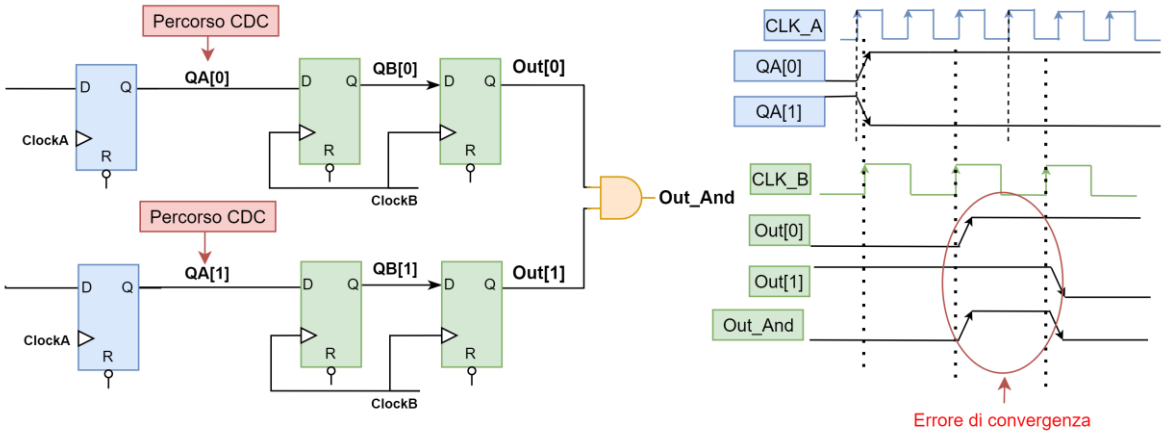

Figura 4-3: Incoerenza

- **Perdita dati:** Altro possibile problema CDC risiede nella possibilità di non avere il campionamento di un dato. Essenzialmente vi è la possibilità di perdere un dato trasmesso da parte del dominio di trasmissione perché non "visto" da parte del dominio di ricezione. È dunque necessario porre l'attenzione sui tempi minimi necessari per poter garantire il campionamento dei dati trasmessi e non avere la loro perdita. Dunque, vi può essere la possibilità di avere una trasmissione continua di dati tale da non riuscire a garantire che tutti i dati trasmessi siano correttamente campionati dal dominio di ricezione. Ecco perché nascono strutture di sincronizzazione ad anello chiuso, ovvero basate su un segnale di "Ack" per poter stabilire quando è possibile effettuare una trasmissione dati e quando non è possibile.

4.2 Analisi CDC

Per analisi CDC strutturale si intende l'analisi completa del design al fine di identificare tutti i domini di clock presenti all'interno del design, identificare tutti i segnali e relativi percorsi CDC in cui è assente uno stadio di sincronizzazione, rilevare la presenza di logica combinatoria illegale all'interno degli stadi di sincronizzazione ed ovviamente risaltare i diversi problemi legati agli stadi sincronizzatori non implementati in maniera corretta o mancanti. Durante l'analisi strutturale CDC avviene anche la classificazione del comportamento dei segnali come anche la classificazione dei segnali CDC all'interno dei diversi protocolli presenti nel design. Vengono quindi determinati il comportamento e proprietà dei segnali. Ad esempio, viene definito se tali segnali sono appartenenti a sincronizzatori handshake oppure a strutture FIFO, come anche viene definito se i segnali presenti sono statici o meno. Da quanto quindi descritto, è possibile intuire che dall'analisi strutturale vi è solo la possibilità di validare la presenza o meno di strutture di sincronizzazione e i loro relativi errori, ma la verifica strutturale non può confermare la corretta funzionalità degli schemi implementati. Dunque, essa si limita esclusivamente alla ricerca, per l'appunto strutturale, dei percorsi CDC. Per poter verificare che i dati e segnali siano correttamente campionati dagli schemi di sincronizzazione implementati lungo i percorsi CDC è necessario ricorrere alla verifica CDC funzionale. Essenzialmente, grazie ad essa, si verificano che i protocolli di sincronizzazione attuati siano corretti per il tipo di percorso CDC in analisi e tali controlli funzionali sono possibili grazie all'ausilio della verifica formale.

4.2.1 Introduzione ad un comune caso di violazione CDC

Il dominio di clock è una parte del design che viene gestita da un singolo clock oppure da clock differenti che dispongono di una relazione di fase costante. Clock sincroni dispongono di una relazione di fase costante e conosciuta, mentre i clock asincroni sono caratterizzati da relazioni di fase e frequenza variabili. Nella Figura 4-4 è possibile notare, nella parte superiore, un esempio di dominio di clock.

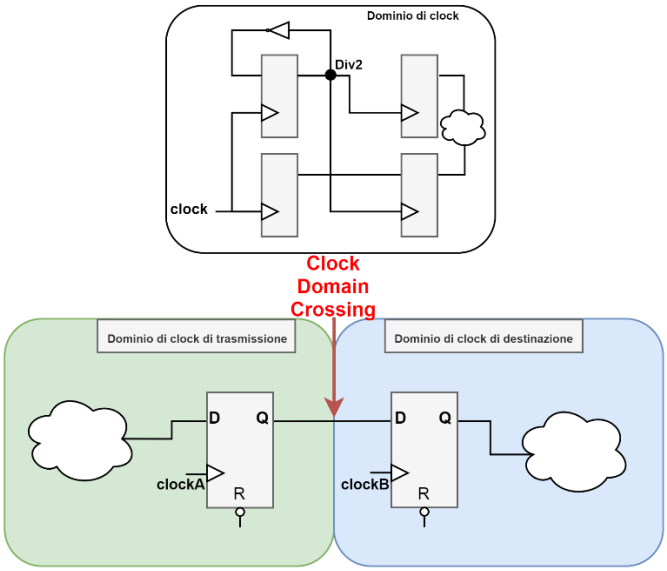

Figura 4-4: Dominio di clock rappresentato nella parte superiore. Passaggio CDC rappresentato nella parte inferiore.

Esso è caratterizzato da un singolo clock di riferimento, chiamato “clock”, da cui viene ricavato un secondo clock “Div2” che corrisponde alla sua metà ed è caratterizzato da una relazione di fase costante ed è sincrono a “clock”. Il Clock-Domain-Crossing avviene nel momento i cui vi è il passaggio di dati/segnali tra domini di clock asincroni, come è possibile notare nella parte inferiore della stessa figura, ed il segnale in questione è definito come “Coppia CDC”.

Nell’esempio riportato nella Figura 4-5 viene rappresentato un caso comune di clock domain crossing in cui sono presenti due FlipFlop, A e B, appartenenti ai rispettivi domini di clock asincroni. È possibile notare che nel momento in cui avviene la trasmissione del segnale Q_A dal dominio di clock A, il segnale di uscita Q_B del FF_B entra in uno stato di metastabilità in cui impiega del tempo indefinito per poter convergere ad uno stato logico stabile. Il motivo è ovviamente legato ad una violazione dei tempi di setup/hold che portano per l’appunto alla metastabilità ed ovviamente tale violazione è dovuta alla commutazione del segnale Q_A che può avvenire in qualsiasi momento per il dominio di clock B. Il tempo necessario per poter convergere ad uno stato logico stabile non è calcolabile in maniera deterministica, quindi risulta essere non facilmente predicibile. Al fine di evitare tale problema CDC, vi è la necessità di introdurre un sincronizzatore. L’obiettivo principale dei sincronizzatori è quello di evitare la propagazione di un segnale metastabile illegale nella logica combinatoria presente a valle del FF, così da non avere una sua propagazione attraverso il design. Un esempio di come poter evitare la propagazione della metastabilità è ricorrere all’implementazione del più comune stadio di sincronizzazione nominato Multiflop. Quest’ultimo può essere caratterizzato da 2 FF guidati dal clock del dominio di destinazione come presentato nella Figura 4-6 e spiegato nella sezione 4.2.2.1 (è possibile anche utilizzare un sincronizzatore a 3 o più FF).

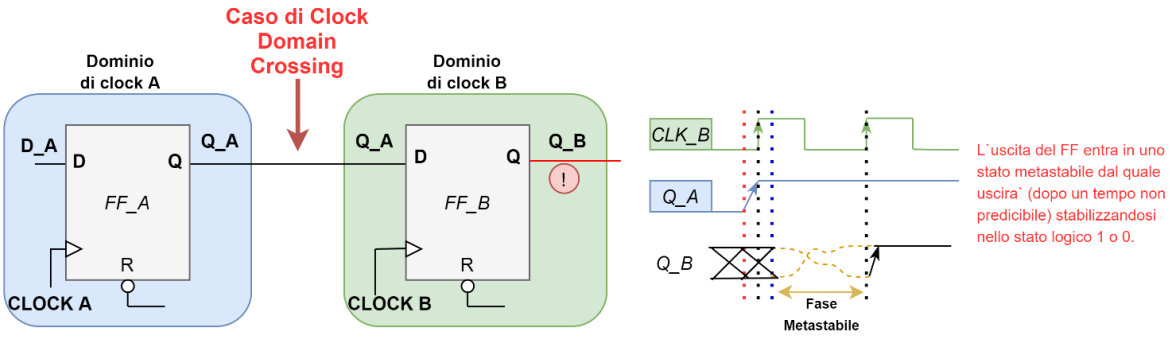

Figura 4-5: Semplice caso CDC in cui è assente una struttura di sincronizzazione

4.2.2 Schemi di sincronizzazione

La metastabilità non può quindi essere evitata all'interno di design asincroni ma può essere gestita con l'implementazione di schemi di sincronizzazione. I sincronizzatori raggiungono l'obiettivo di limitare gli effetti negativi della metastabilità isolando la propagazione attraverso il design di segnali metastabili. Essendo la verifica strutturale basata anche sull'identificazione delle strutture di sincronizzazione implementate all'interno del design, vengono introdotti i più comuni schemi implementabili nei design. In tale sezione vengono quindi riportati alcuni esempi di schemi di sincronizzazione standard più comuni. Ovviamente, non ci si limita solo all'implementazione di schemi di sincronizzazione standard ma possono anche essere realizzate strutture "custom", ovvero pensate per determinate implementazioni architetturali. Infatti, nei tool di verifica strutturale CDC-RDC è possibile aggiungere anche moduli "custom" all'elenco dei sincronizzatori rilevabili automaticamente.

4.2.2.1 Sincronizzatore standard

La prima struttura di sincronizzazione che viene descritta è il sincronizzatore standard a più FlipFlop. Esso rappresenta lo schema tipicamente utilizzato per i Clock-Domain-Crossing[3] presenti all'interno di un design asincrono e può essere definito "multi-Flop". Essenzialmente, esso è caratterizzato da due o più FlipFlop disposti in serie. Tale schema di sincronizzazione è basato sulla metodologia "Open-loop", ovvero esso è una struttura non basata sulla presenza di un segnale di "Acknowledgement" da parte del dominio di destinazione.

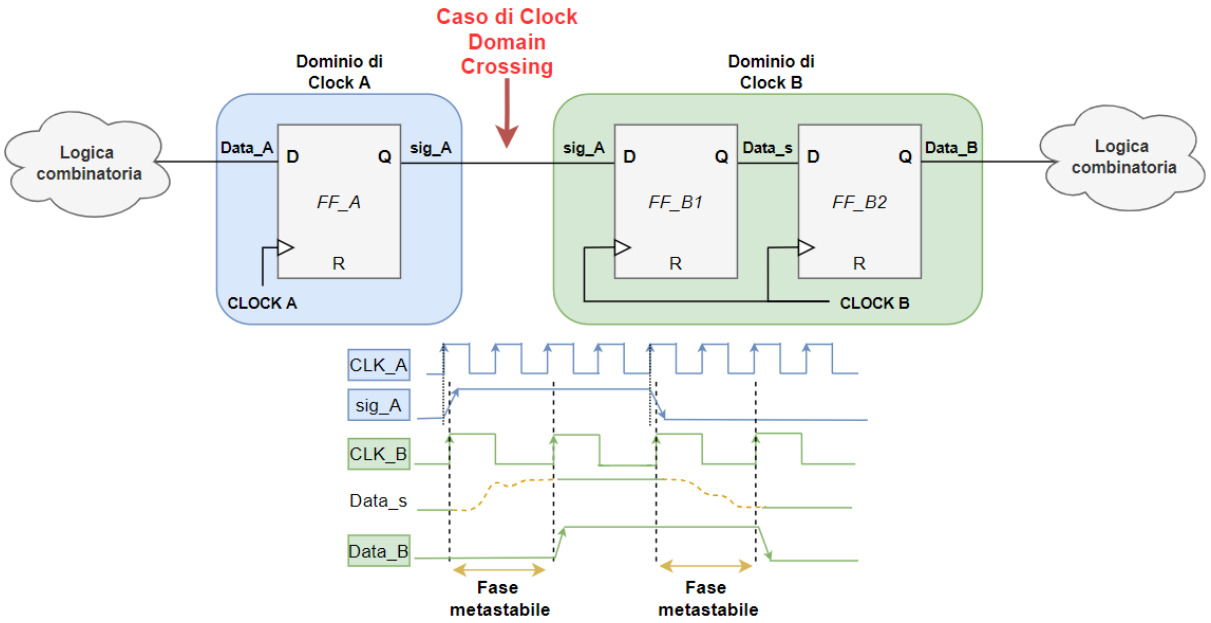

Figura 4-6: Semplice caso CDC con sincronizzatore a 2FF

Nella Figura 4-6 è riportata una coppia CDC in cui è presente un sincronizzatore standard a due FF. Esso è caratterizzato da due FlipFlop disposti in serie nel dominio di clock B di destinazione con lo scopo di campionare il segnale trasmesso da parte del dominio di clock A di trasmissione. Con tale schema di sincronizzazione è possibile mitigare la propagazione della metastabilità che può insorgere come nel caso tipico di violazione CDC affrontato nella sezione 4.2.1 in Figura 4-5. Tale struttura, ovviamente, si riferisce al passaggio CDC di un singolo bit e viene spesso implementata per i segnali di controllo. Il più semplice sincronizzatore è caratterizzato da soli due FlipFlop, ma ovviamente possono esserne aggiunti altri in serie.

4.2.2.2 Mux synchronizer

I mux synchronizers sono tipicamente usati nei design per i percorsi CDC di dati. Grazie a tale sistema di sincronizzazione è possibile campionare correttamente i bits di un dato che commutano contemporaneamente. Tale schema è basato sull'utilizzo di un segnale come selettore del multiplexer in questione e tale segnale è generato dal dominio di trasmissione e sincronizzato per il dominio di destinazione. Essenzialmente, viene impiegato un comune sincronizzatore a due o tre FF al fine di sincronizzare il segnale di Enable proveniente dal dominio di trasmissione. Con tale struttura è quindi possibile campionare i dati in maniera corretta nel dominio di destinazione.

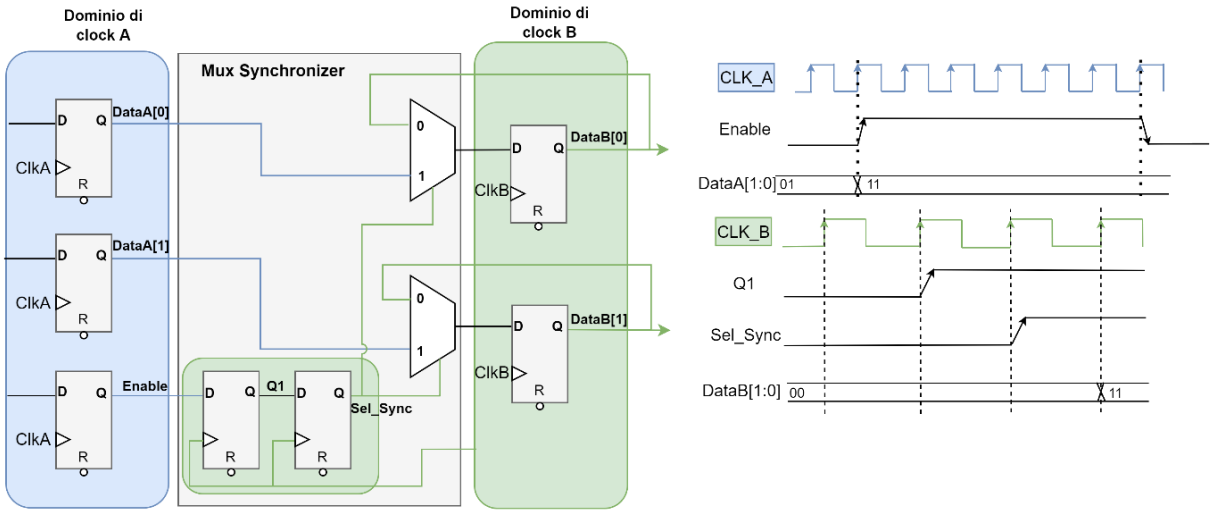

Figura 4-7: Mux synchronizer

Al fine di utilizzare correttamente tale struttura di sincronizzazione ed avere un corretto campionamento da parte del dominio di destinazione del dato trasmesso, quest'ultimo deve essere stabile per tutto il tempo in cui il segnale di Enable è asserito oppure fino a quando il dato non è stato campionato correttamente dal dominio di destinazione. Bisogna nuovamente sottolineare che questa tipologia di struttura funziona correttamente solo nel momento in cui avviene la propagazione di un dato stabile nel periodo in cui è asserito il segnale di Enable.

4.2.2.3 Sincronizzatore Handshaske

Altra tipo di struttura di sincronizzazione molto comune è la struttura “Handshake” [4]. Tale struttura è basata sulla metodologia “Closed-loop”, ovvero strutture basate sulla necessità di aspettare un segnale di “Acknowledgement” da parte del dominio di destinazione al fine di poter apprendere che il dato CDC sia stato campionato correttamente. Il sincronizzatore handshake è una delle metodologie più ampiamente utilizzate nel trasferimento di vettori di dati tra domini di clock asincroni. Un esempio di struttura viene riportata nella Figura 4-8. Con questa tipo di struttura è possibile effettuare un passaggio dati tra domini di clock asincroni in sicurezza attraverso un semplice protocollo di “Request” e “Acknowledgement”. Questo è essenzialmente il motivo per il quale tale protocollo è chiamato “Handshake”.

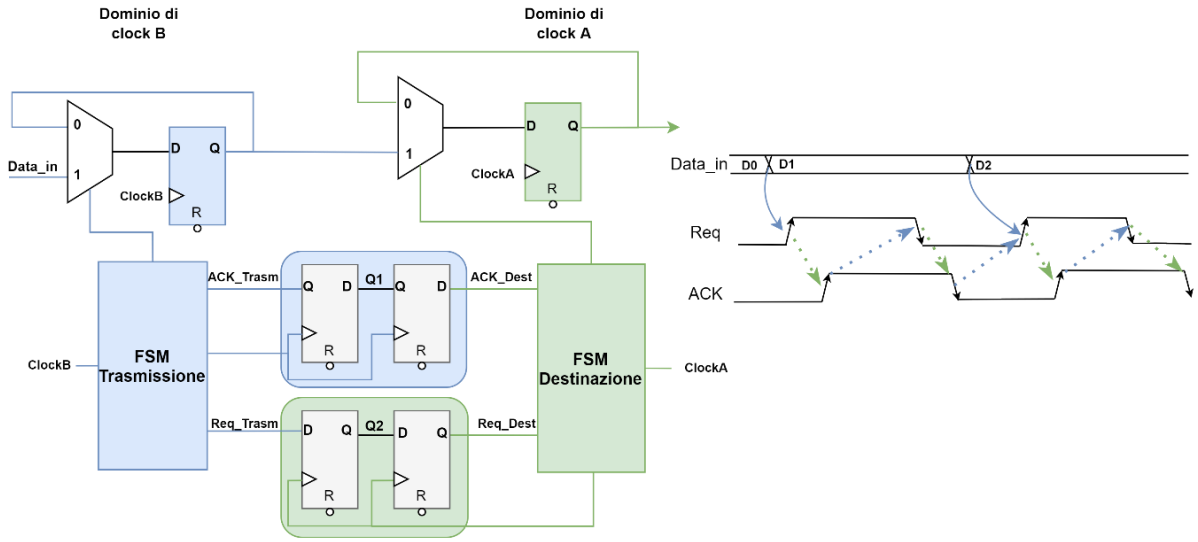

Figura 4-8: Struttura di sincronizzazione Handshake

I dati vengono trasmessi da parte del dominio di trasmissione, gestito dal “clockB”, e successivamente viene anche asserito il segnale di “Request” per poter far partire il protocollo di trasmissione. Tale segnale, una volta sincronizzato per il dominio di destinazione attraverso uno schema di sincronizzazione comune a due FF, viene acquisito dall’unità di controllo di destinazione. Quest’ultima ha quindi conoscenza di avere sul bus un dato valido e stabile e che può quindi procedere con il suo campionamento. L’unità di controllo di destinazione risponde con l’asserzione del segnale di “ACK”, ovvero informa l’unità di controllo di ricezione di aver capito e di aver campionato i dati trasmessi sul bus. Una volta acquisito il segnale di “ACK” da parte dell’unità di controllo di trasmissione, dove ovviamente tale segnale viene anche lui sincronizzato per il corretto campionamento in tale dominio, quest’ultima può quindi procedere con la rimozione del segnale di “Request”. Per poter poi procedere con la trasmissione di un ulteriore dato sul bus, l’unità di controllo di trasmissione deve aspettare un ulteriore segnale di “ACK” da parte dell’unità di controllo di destinazione, al fine di poter capire che il ricevitore ha terminato ed è quindi pronto per ricevere un altro dato sul bus. Dunque, nel momento in cui il ricevitore acquisisce la de-asserzione del segnale di “Request”, risponde a sua volta con la de-asserzione del segnale di “ACK”. A questo punto, il ricevitore è quindi disponibile a ricevere un’altra richiesta di trasmissioni dati.

È possibile capire di come sia efficace questa tipo di struttura di sincronizzazione all'interno di un design asincrono. Per poter avere un corretto funzionamento di tale struttura bisogna quindi rispettare due semplici regole: il trasmettitore deve mantenere asserito il segnale di "Request" fino a quando il ricevitore non risponde con l'asserzione del segnale di "ACK" e non deve asserire un nuovo segnale di "Request" fino a quando non viene recepita la de-asserzione del segnale di "ACK" riferita all'ultima trasmissione. Possono ovviamente essere implementati altri tipi di protocolli con questa tipo di struttura. La procedura appena descritta è basata su un protocollo "handshake four phases" che risulta essere uno dei protocolli più sicuri, ma come è possibile intuire dal protocollo appena spiegato, la velocità di comunicazione che può essere raggiunta non è elevata. Un'altra possibilità è utilizzare un protocollo "handshake two phases". Quest'ultimo è più veloce in quanto si hanno metà delle transizioni ma ha come svantaggio di essere meno sicuro rispetto al precedente. Ogni transizione dei segnali di "Request" e "ACK" ha sempre i seguenti significati: trasmissione di un nuovo dato e aver recepito ed accettato il nuovo dato. Essenzialmente, il ricevitore deve essere sufficientemente veloce ad essere pronto nel ricevere un nuovo dato da parte del trasmettitore, in quanto il trasmettitore non attende di ricevere informazioni da parte del ricevitore se quest'ultimo ha terminato l'acquisizione.

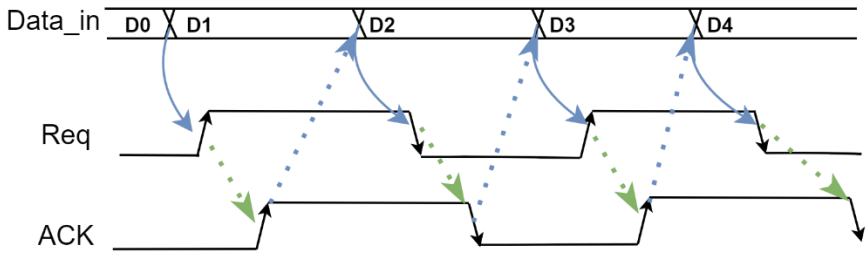

Figura 4-9: Timing protocollo a due fasi

4.2.2.4 FIFO asincrona

Altra struttura di sincronizzazione comune spesso implementata per poter effettuare un trasferimento dati tra domini di clock asincroni è la “FIFO synchronizer” [3] a due clock. Viene riporta una sua tipica struttura nella Figura 4-10.

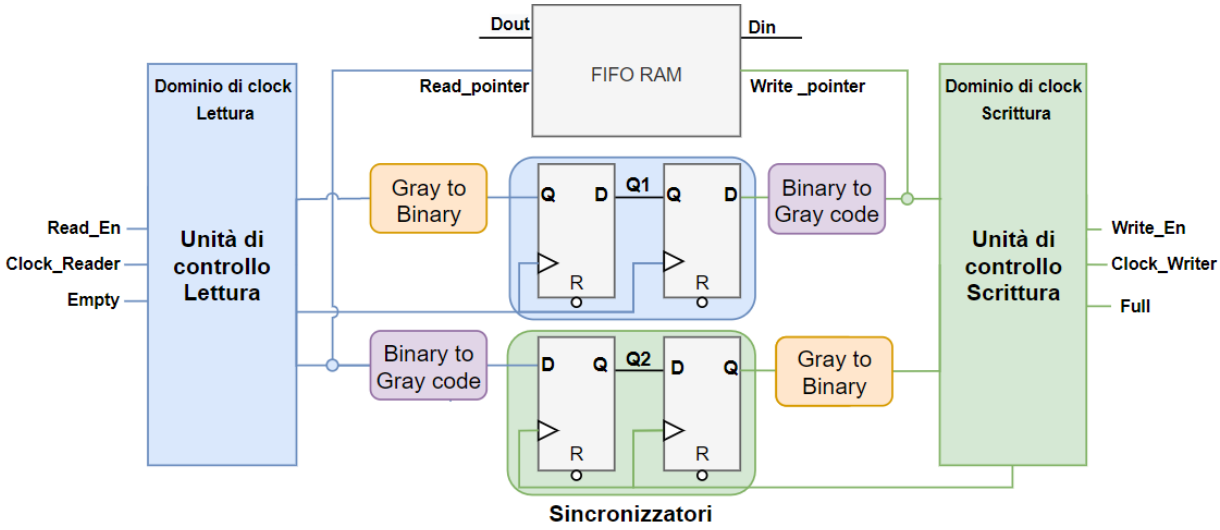

Figura 4-10: Sincronizzatore FIFO

Essenzialmente, la FIFO in questione è caratterizzata da due clock, uno di scrittura e uno di lettura, che sono i domini asincroni in questione ed ecco perché viene chiamata FIFO asincrona. L’unità di controllo di scrittura effettua una richiesta attraverso il segnale “Write_En” e la scrittura può avvenire solo nel momento in cui la FIFO non è nello stato di “Full”. Per poter invece effettuare la lettura, l’unità di controllo di lettura effettua una richiesta attraverso il segnale “Read_En” ed ovviamente questa può avvenire solo nel momento in cui la FIFO non è nello stato di “Empty”. Il funzionamento è gestito dai puntatori di lettura e scrittura ed essi sono ottenuti rispettivamente dall’unità di controllo di lettura e di scrittura con i propri clock. Quindi, una volta effettuata una scrittura o una lettura nella memoria, i rispettivi puntatori vengono incrementati. È possibile notare la presenza di schemi di sincronizzazione, tipicamente strutture standard a due o tre FF, che hanno il ruolo di sincronizzare i puntatori da un dominio di clock all’altro col fine di effettuare le elaborazioni necessarie per poter valutare se la FIFO è nello stato di “Full” o nello stato di “Empty”. La sincronizzazione è necessaria per poter effettuare lo scambio dei puntatori tra i domini asincroni in questione e possono essere utilizzati i sincronizzatori più comuni a due o tre FF. I puntatori vengono codificati secondo la codifica gray, ottenendo il vantaggio di effettuare la sincronizzazione di un puntatore in cui commuta un solo bit alla volta. Ovviamente, è possibile intuire che il problema principale di tale struttura è dettato dalla latenza. Nel momento in cui avviene una prima scrittura nella FIFO, che quindi è inizialmente nello stato di “Empty”, bisogna attendere un numero di colpi di clock addizionali affinché il puntatore di scrittura raggiunga l’unità di controllo di lettura per poter de-asserire lo stato di “Empty”. Questo perché bisogna tenere conto dell’implementazione degli schemi di sincronizzazione, i quali introducono un numero di colpi di clock di latenza a seconda della loro struttura. Quindi, la struttura FIFO introduce una certa latenza che dipende dalla presenza di sincronizzatori dei puntatori, ma essa consente di avere un veloce trasferimento dati tra domini di clock asincroni evitando problemi CDC.

4.2.3 Violazioni CDC tipiche

Dopo aver quindi introdotto l'analisi strutturale, che ha il compito di rilevare tutte le violazioni CDC presenti nel design dovute ad errori di implementazione o ad eventuali assenze, si procede con prendere in analisi alcuni esempi di violazioni che possono essere incontrate durante la verifica strutturale all'interno di un DUT. Le seguenti violazioni che vengono riportati non rappresentano tutte le violazioni CDC possibili ma vengono riportate le più comuni.

4.2.3.1 Assenza di sincronizzazione

La prima violazione CDC che comunemente viene identificata è l'assenza di un sincronizzatore. La violazione è stata definita e presentata nella Figura 4-5 ed è anche spiegata la sua gravità. Essenzialmente, il tool di verifica CDC riporta l'errore di non avere identificato nessuno schema di sincronizzazione lungo il percorso CDC. Viene riportato il percorso ed il suo relativo punto di partenza da cui il valore metastabile può propagarsi attraverso il design, causando quindi errori funzionali. La possibile soluzione che può essere attuata è stata definita e presentata nella Figura 4-6.

4.2.3.2 Logica combinatoria nel cammino CDC – Insorgenza di glitch

La metastabilità non è l'unico problema che può presentarsi durante un passaggio CDC ed infatti si affianca anche il problema dell'insorgenza e propagazione di glitches nel trasferimento di segnali da un dominio di clock asincrono ad un altro. Questo può portare a problemi di diverso genere: dal non corretto funzionamento del dispositivo alla non coerenza e perdita di dati. Un semplice caso di schema di sincronizzazione in cui ci si può imbattere in una propagazione di un glitch è presentato nella Figura 4-11. Ovviamente, viene rappresentato un caso in cui vengono riportati dei ritardi generici e questi possono essere di diversa natura: dalla differenza di tempo di clock to output dei FF di trasmissione in questione ad una differenza nei ritardi di propagazione delle linee in ingresso alla porta logica. Si può anche solo pensare alla presenza di un'altra logica combinatoria tra gli ingressi della suddetta porta e l'uscita dei FF di trasmissione. La presenza di logica combinatoria nel percorso CDC può portare ad avere problemi funzionali anche se presente uno stadio di sincronizzazione, come è possibile notare dal diagramma del timing riportato nella Figura 4-11. Infatti, è sempre consigliato avere percorsi CDC "puliti" tra l'uscita di un dominio di trasmissione e l'ingresso del dominio di ricezione asincrono, al fine di ridurre al minimo l'insorgenza di errori funzionali.

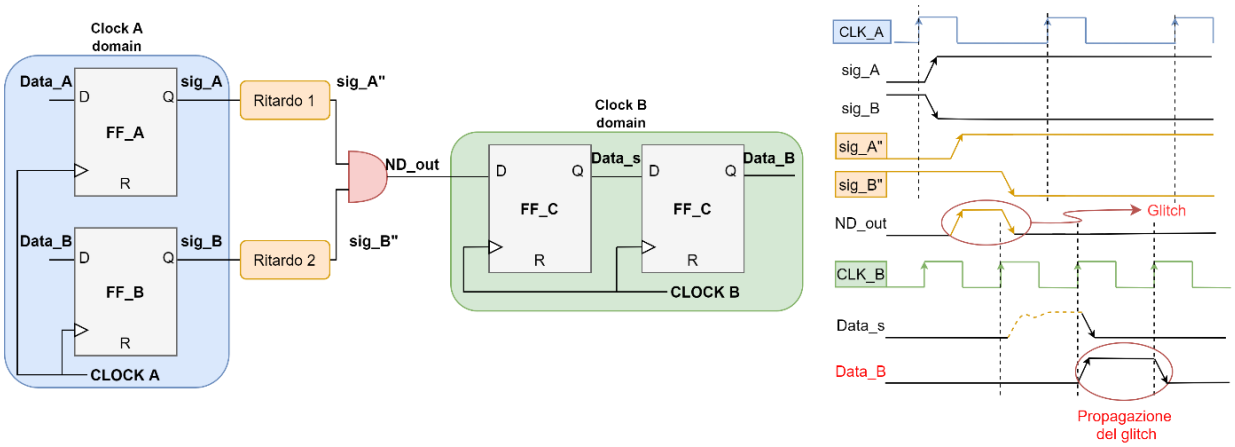

Figura 4-11: Caso di propagazione di un glitch in un percorso di convergenza CDC

Questo è ovviamente solo uno dei tanti esempi di schemi che possono portare ad avere errori funzionali. Una possibile soluzione attuabile per poter evitare l'insorgenza di glitch nell'esempio riportato è quello di non avere una commutazione contemporanea dei due segnali in questione. Procedendo con il far commutare i due segnali in colpi di clock diversi, si elimina la possibilità di generare un glitch.

4.2.3.3 Violazione di convergenza

Altra tipo di violazione CDC è riferita alla Figura 4-12 ed è definita come violazione di convergenza. In prima analisi la struttura in questione potrebbe sembrare esente da problemi, in quanto abbiamo una struttura di sincronizzazione tra il dominio di clock A e il dominio di clock B e non vi è alcuna presenza di logica combinatoria all'ingresso della struttura di sincronizzazione come neanche tra i FlipFlop costituenti lo schema di sincronizzazione. La convergenza è definita come un gruppo di segnali oppure differenti bits di uno stesso segnale che provengono dallo stesso o differenti domini di clock che convergono in una logica combinatoria, dopo la loro sincronizzazione, verso il clock di destinazione. Prendendo in considerazione questo caso, è possibile pensare ad un Bus dati.

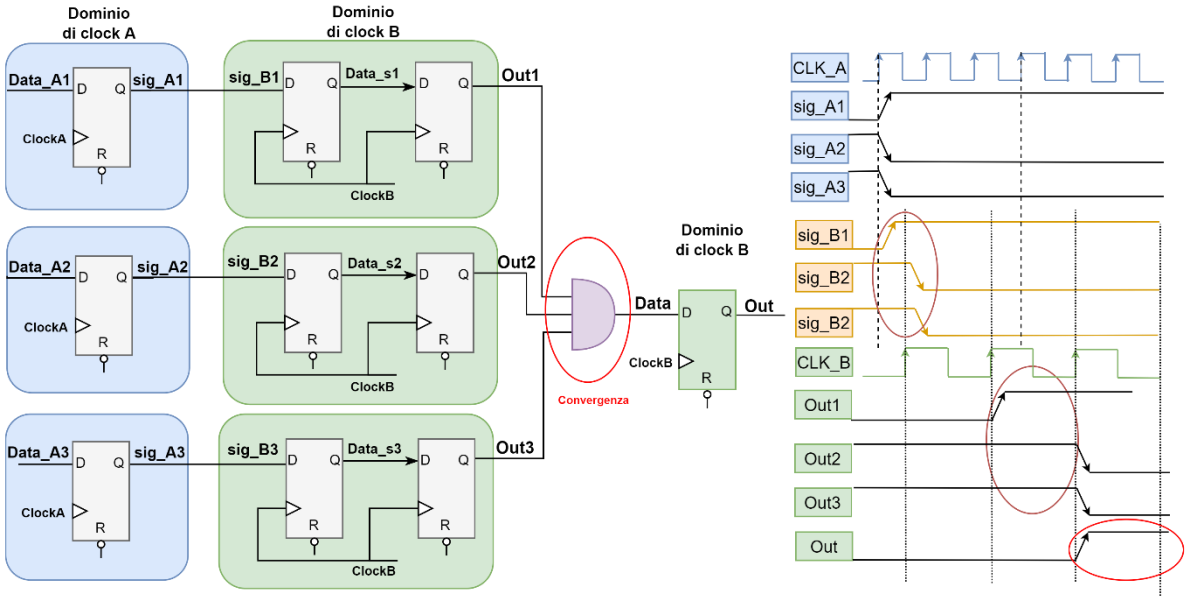

Figura 4-12: Errore CDC di convergenza

Questa tipo di configurazione può rappresentare problemi CDC. È sufficiente pensare alla presenza di un ritardo tra sig_B1, sig_B2 e sig_B3, dovuto ad esempio ad un differente tempo di clock to output o differente tempo di propagazione lungo le linee che non consente un campionamento corretto dei segnali commutati contemporaneamente. Vi è la possibilità che alcuni dei sincronizzatori campionino un dato precedente mentre altri campionino il dato nuovo, riflettendosi quindi nell'avere un risultato non coerente. Dunque, non è possibile assicurare che avvenga il campionamento corretto di tutti i segnali commutati contemporaneamente. Questa struttura potrebbe portare ad eventuali errori o malfunzionamenti nel momento in cui viene, ad esempio, implementata per generare un segnale di "Start" per una macchina a stati o per poter incrementare/decrementare un contatore senza ricorrere ad alcun tipo di codifica.

Tale struttura invece non presenta alcun tipo di problema nel momento in cui vengono implementate configurazioni "mutually exclusive", caso in cui i segnali possono commutare uno alla volta, oppure se vengono implementate codifiche come la codifica gray. In tale caso, muovendosi un solo segnale alla volta, è possibile assicurare il corretto campionamento dei segnali in questione e si può considerare la struttura correttamente funzionante.

4.2.3.4 Violazione di divergenza

Altra tipo di violazione molto comune è la violazione di “Divergenza” e l’esempio di struttura viene riportata nella Figura 4-13. Tale violazione viene definita di divergenza in quanto un unico segnale CDC viene collegato a più sincronizzatori riflettendosi in possibili errori funzionali. È possibile pensare, ad esempio, di utilizzare tale struttura per poter gestire contatori oppure macchine a stati diverse che devono essere incrementate/decrementate contemporaneamente o partire nello stesso momento grazie ai segnali di “Start1” e “Start2”.

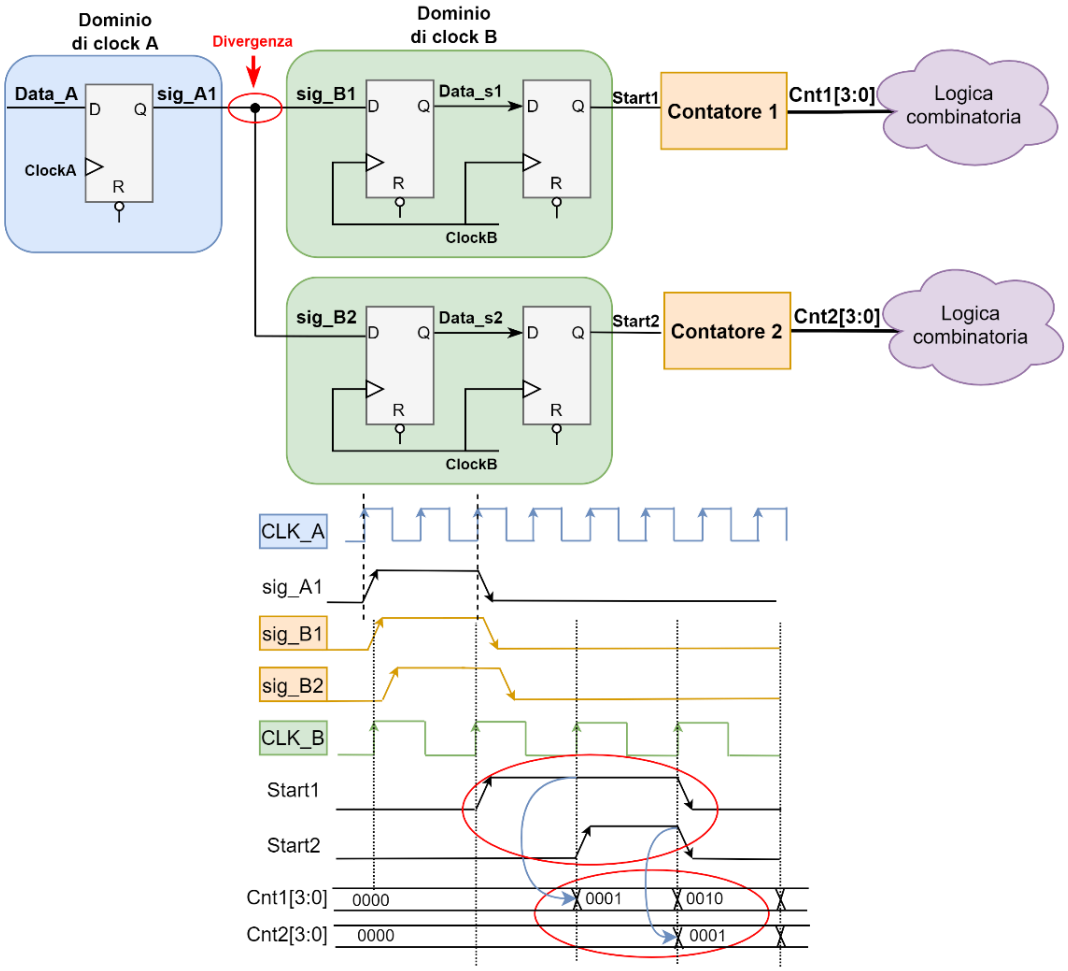

Figura 4-13: Violazione di Divergenza

Purtroppo, ritardi di propagazione differenti possono portare ad avere una commutazione dei segnali “Start1” e “Start2” in colpi di clock diversi, così portando ad avere un errore funzionale. Le macchine a stati gestite da tali segnali non riescono quindi a partire in contemporanea oppure i contatori vengono gestiti in maniera non corretta. Questa tipo di struttura deve essere evitata provando a adattare il proprio design a lavorare con un solo segnale di start.

4.3 Introduzione al flusso di analisi CDC

L'analisi CDC, come riportato nella Figura 4-14, può essere divisa in 3 macro-passaggi che a loro volta sono suddivisi in differenti steps. Nei prossimi punti vengono quindi descritti i passaggi da eseguire nei differenti stadi dell'analisi. Tali punti caratterizzano il flusso comune da seguire sui design su cui bisogna effettuare un'analisi CDC-RDC. Dalla sua esecuzione vengono estratte tutte le violazioni presenti nel design su cui è potenzialmente presente un errore, RDC o CDC o di entrambe le tipologie. Una volta analizzate le violazioni riportate e identificati gli errori effettivi presenti nel design, è possibile procedere con la rivisitazione del codice RTL, o con soluzioni definite dall'utente, al fine di risolvere i problemi identificati e quindi rieseguire l'analisi per poter avere conferma di avere eliminato le problematiche identificate. Non tutte le violazioni riportate sono da considerarsi sempre come errori effettivi, questo perché si è fortemente dipendenti dai protocolli presenti all'interno del proprio design. Quindi, è importante identificare e risolvere gli errori risaltati dall'analisi, presenti nel proprio design, ma anche identificare e giustificare le violazioni che non sono ritenute reali problematiche. Si introduce quindi il concetto di "waiver", ovvero la rimozione di una violazione identificata durante l'analisi CDC-RDC che non è ritenuta essere tale. Il concetto di giustificazione dei waivers, attraverso la verifica formale, viene approfondito nei capitoli successivi.

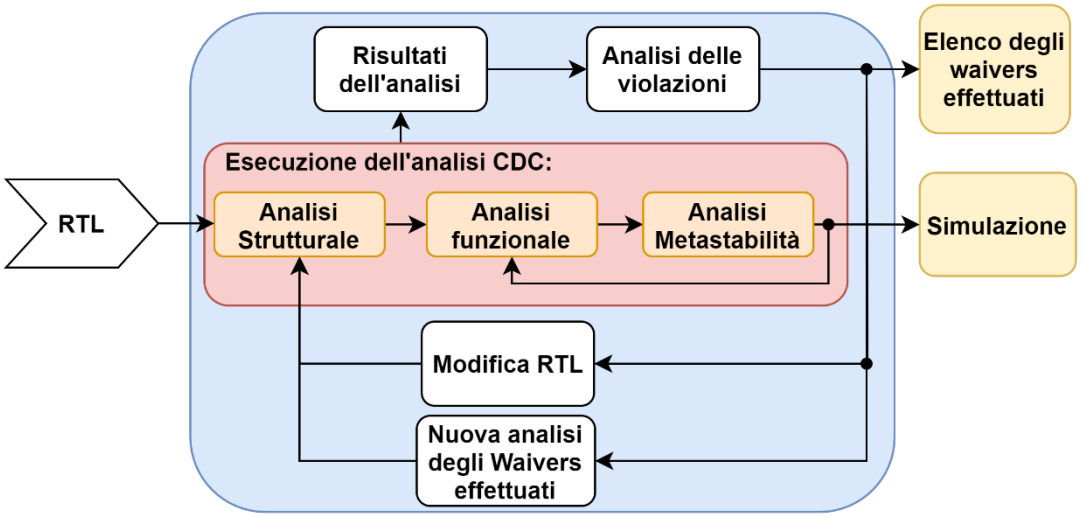

Figura 4-14: Flusso dell'analisi CDC

4.3.1 Passaggi iniziali dell'analisi strutturale

Nello schema riportato nella Figura 4-15 vengono illustrati gli step da seguire per impostare ed eseguire l'analisi strutturale su un design.

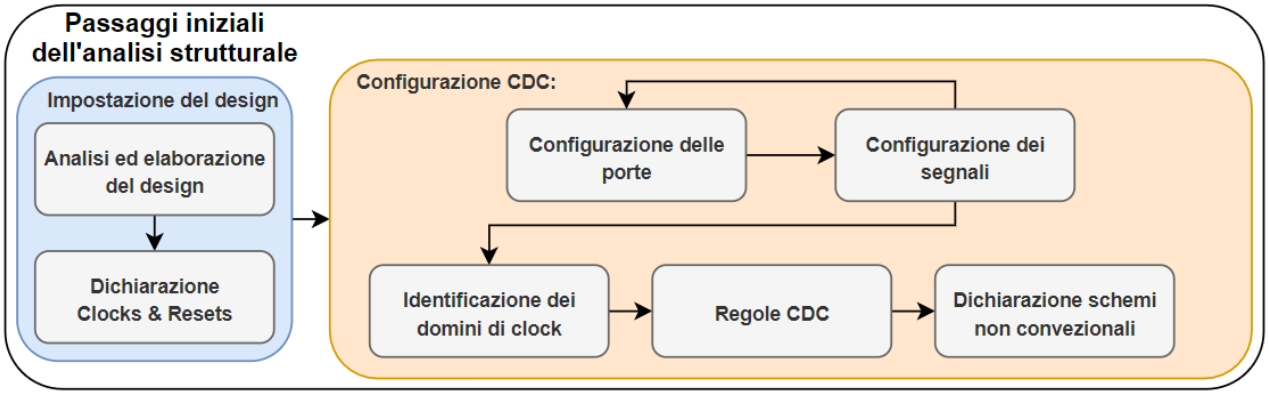

Figura 4-15: Schema passaggi iniziali dell'analisi strutturale

4.3.2 Impostazione del design

Il punto di partenza di questo step è l'analisi ed elaborazione del design a livello RTL. Il primo punto chiave per poter procedere con l'analisi strutturale è la definizione dei clocks e resets presenti nel design. Quindi, questo è uno step necessario al fine di identificare successivamente i domini di clocks presenti nel proprio sistema.

1. Si procede con la definizione dei clocks, dei loro fronti attivi e delle relazioni costanti di fase e di frequenza, se queste sono presenti. Essenzialmente, vengono dichiarati i clocks principali presenti all'interno del design sotto analisi e si forniscono le relazioni che sussistono tra i clock dichiarati e se essi sono sincroni oppure asincroni tra loro. Viene adesso riportato un esempio di dichiarazione dei clock e delle loro relazioni all'interno di un design.

    a) Ad esempio, se sono presenti tre clock all'interno del DUT, si procede dapprima con il dichiararli:

        1) Clk -ClockA - Entrambi i fronti
        2) Clk -ClockB - Entrambi i fronti
        3) Clk -ClockC - Entrambi i fronti

    b) I clock così dichiarati risultano essere tutti e tre asincroni e i loro fronti attivi sono sia il fronte di salita che il fronte di discesa. Non è presente alcuna tipo di relazione tra loro. Si suppone adesso che il ClockB sia un clock derivato da ClockA, e che dunque sia sincrono ad esso, e si suppone che sia anche la metà più lento.

        1) Clk -ClockA -Entrambi i fronti
        2) Clk -ClockA -ClockB 2 - Entrambi i fronti
        3) Clk -ClockC - Entrambi i fronti

c) Ecco dunque dichiarato che il ClockB è sincrono al ClockA e che la sua frequenza è la metà della frequenza del ClockA. Dunque, il ClockA ed il ClockB sono sincroni tra loro e sono asincroni rispetto al ClockC.

2. Dopo la definizione dei clock si procede con la definizione dei reset e come questi vengono asseriti. È anche necessario definire se i reset presenti sono:
   - **Sincroni**, il che vuol dire che sia la loro asserzione che de-asserzione è sincrona ad un clock;
   - **Asincroni**, ovvero che sia la loro asserzione che de-asserzione è asincrona al clock;
   - **Sincronizzati**. Quest'ultima tipo di configurazione ha come significato che il reset in questione è l'uscita di un sincronizzatore di reset, il che vuol dire che la sua asserzione è asincrona ma la sua de-asserzione è sincrona al clock.

Questo passaggio di definizione delle caratteristiche del reset è fondamentale per lo step di analisi RDC, la quale viene introdotta e approfondita nel capitolo 7. Nella Figura 4-16 vengono riportati alcuni dei possibili schemi convenzionali per la definizione appena riporta dei segnali di resets.

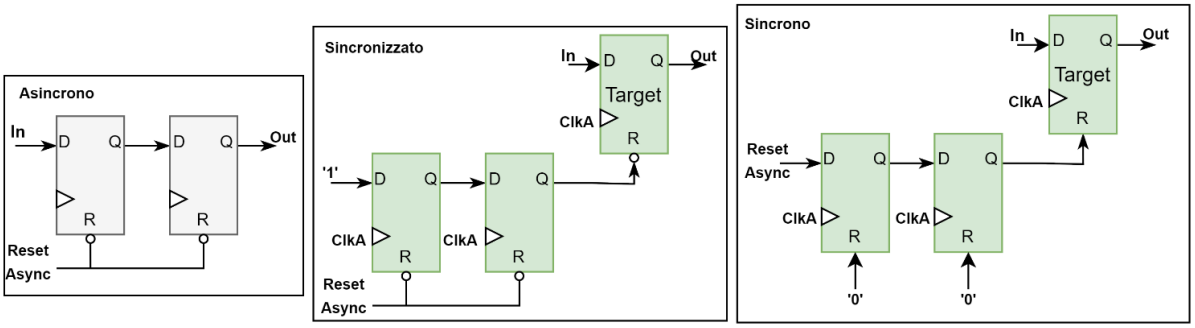

Figura 4-16: Configurazione dei resets

A sinistra viene riportato un caso di segnale di reset asincrono in cui non è presente alcun tipo di schema di sincronizzazione del reset. Questa tipo di configurazione può portare a problemi RDC legati alla violazione dei tempi di removal/recovery dei FF come anche ad un problema di asserzione asincrona in prossimità dei tempi di setup/hold dei FF presenti a valle. Al centro viene riportato un esempio di segnale di reset sincronizzato. Il segnale di reset viene asserito in maniera asincrona e procede con il resettare lo schema di sincronizzazione di reset presente e anche il FF target gestito da tale schema. Nel momento in cui avviene la de-asserzione asincrona del segnale di reset, quest'ultima viene sincronizzata da parte dello schema di sincronizzazione. Infine, a destra, viene riportato un caso di segnale di reset in cui sia la sua asserzione che de-asserzione sono resi sincroni al clock del FF target grazie allo schema di sincronizzazione standard multiflop presente.

3. All'interno dello step di impostazione del design si può procedere con il definire i componenti di cui non si vuole effettuare un'analisi CDC-RDC e di cui si vuole prendere in considerazione esclusivamente l'interazione dei segnali di uscita e di ingresso con il design, ovvero si effettua un "BlackBoxing" del modulo in questione. Tipicamente, all'interno dei design possono essere presenti delle proprietà intellettuali "IP" o componenti già esternamente validate dal punto di vista CDC-RDC e di cui quindi non si ha necessità di effettuare un'ulteriore analisi del blocco. Ovviamente, vi è la necessità di effettuare una verifica della loro interazione con il sistema in

cui sono integrati. È quindi necessario rappresentare tali componenti come dei BlackBox al fine di escluderli dall'analisi ed evitare la segnalazione di eventuali violazioni di cui si ha già la conferma che non rappresentano un problema reale. È ovviamente possibile associare i segnali di ingresso e di uscita dei moduli BlackBox ai loro rispettivi domini di clock ed effettuare una configurazione del loro comportamento. Essenzialmente, viene identificata la presenza di un determinato modulo di cui si ha a conoscenza esclusivamente dei suoi segnali di ingresso,di uscita, dei loro rispettivi domini di clock e comportamento.

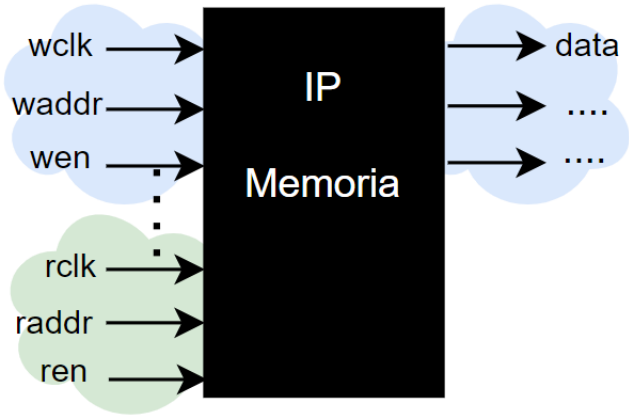

Figura 4-17:BlackBox

Nella Figura 4-17 viene riportato un esempio di come viene visualizzato un elemento su cui è stato effettuato il blackboxing. In questo caso è quindi necessario effettuare la configurazione dei segnali di ingresso e di uscita, ovvero è necessario associare le porte di ingresso e di uscita che costituiscono la blackbox ai domini di clock relativi e descrivere il comportamento assunto dai segnali in questione. Questo viene effettuato col fine di avere una sua corretta configurazione all'interno del design ed escludere eventuali violazioni che non risultano essere reali problemi CDC-RDC.

4.3.3 Configurazione CDC

4.3.3.1 Configurazione delle porte del design

In questo primo step della configurazione CDC viene fornito il legame di parentela delle porte di ingresso e di uscita, presenti nel top digital, con i domini di clock di appartenenza. Questo passaggio può essere eseguito, almeno in parte, automaticamente dal tool di verifica ma non è possibile avere un'associazione automatica per tutte le porte presenti. Dunque, è necessario eseguire un'associazione manuale. Il tool è in grado di eseguire automaticamente determinate associazione grazie all'analisi del codice RTL. Essenzialmente, esso effettua l'associazione di un clock alla porta di ingresso prendendo in considerazione il dominio di clock del primo FF situato nel percorso effettuato da parte della porta di ingresso. Quindi, viene effettuata l'associazione del dominio controllando il clock del FF a cui arriva la porta di ingresso in questione. Le porte non identificate sono in questo caso chiamate "Unclocked" ed il motivo è dovuto alla presenza di più FF appartenenti a domini di clock differenti. In questi casi il tool non è in grado di effettuare un'associazione automatica, riportando dunque il problema con il termine Unclocked. È necessario gestire tali segnali unclocked in quanto non effettuare una loro configurazione può portare ad errori di analisi. Per le porte di uscita avviene un meccanismo di associazione automatica simile, basandosi sui domini di clock dei registri da cui vengono generati tali segnali di

uscita. Determinate porte unclocked possono essere associate ad un dominio di clock solo dopo aver effettuato lo step della "configurazione dei segnali", step che viene approfondito nella sezione successiva. Vi è quindi la necessità di passare attraverso lo step successivo per poter completare questo stadio di analisi, in quanto, nella maggior parte dei casi, è possibile associare un dominio ad una porta, o ad un segnale, solo dopo aver configurato quei segnali di cui si conosce il comportamento.

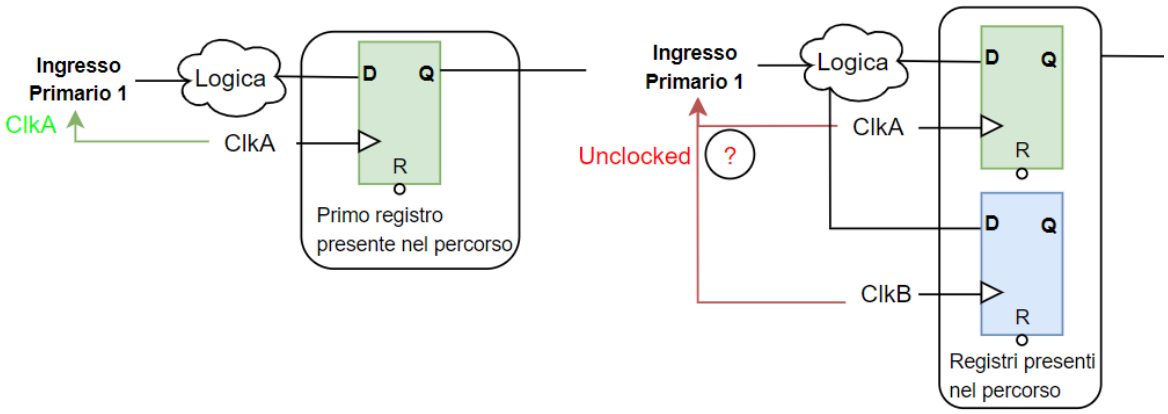

Figura 4-18: Associazione del dominio di clock alle porte primarie di ingresso

Nella Figura 4-18 viene riportato l'esempio dell'associazione automatica effettuata da parte del tool descritta precedentemente. Come è possibile notare, alla sinistra viene riportato il caso in cui il tool effettua automaticamente l'associazione, questo perché sul percorso seguito da parte dell'"Ingresso Primario 1" è presente un FF appartenente al dominio di clock ClkA. La porta "Ingresso Primario 1" viene quindi associata al dominio di clock ClkA. Ovviamente, è sempre necessario effettuare un controllo dell'associazione automatica effettuata, in quanto non è sempre garantito che il segnale proveniente dalla porta "Ingresso Primario 1" sia effettivamente sincrono al dominio di clock associato in automatico. Invece, nella parte di destra, viene riportato il caso in cui nel percorso seguito da parte della porta "Ingresso Primario 1" sono presenti due FF appartenenti a due domini di clock differenti ed asincroni. In questo caso, il tool non effettua un'associazione automatica e vi è dunque la necessità di effettuare una configurazione manuale.

4.3.3.2 Configurazione dei segnali

Per poter procedere con l'analisi CDC si effettua lo step definito "Configurazione dei segnali". In questo stadio dell'analisi si effettua la configurazione di tutti quei segnali utili all'impostazione della modalità di funzionamento del design. Vengono dunque fornite le informazioni strutturali necessarie per poter ottenere una più accurata analisi CDC.

Si descrive il comportamento di quell'insieme di segnali di cui si ha conoscenza della loro attività o da specifiche o da design. È possibile impostare determinate configurazioni di segnali con lo scopo di escludere configurazioni non contemplate o comportamenti non realistici, perché non previsti dalle specifiche o dalle funzionalità del design. È anche possibile effettuare la configurazione di quei segnali utili a discriminare o escludere determinate modalità di funzionamento del dispositivo e dunque suddividere in task più semplici l'analisi CDC. Essenzialmente, si tende a descrivere nel dettaglio quell'insieme di segnali di cui si ha conoscenza del loro comportamento da specifiche o da design e che consentono la rimozione di eventuali violazioni CDC identificate dall'analisi. Ovviamente, si intende

l'esclusione di eventuali violazioni che non costituiscono un reale problema CDC proprio per via di un determinato comportamento assunto dai segnali coinvolti nella violazione identificata. Riassumendo, in tale passaggio viene quindi data una configurazione a quei segnali di cui si vuole descrivere il loro comportamento e che possono sempre più definire i contorni all'interno del quale il design può funzionare, eliminando gradi di libertà non contemplati.

- **Esempio**: si è a conoscenza che una porta di ingresso del proprio design non può assumere valori qualsiasi ma può assumere solo una determinata configurazione di valori. Dunque, non descrivere tale configurazione può portare all'identificazione di eventuali violazioni CDC che in realtà non risultano essere un reale problema, perché i segnali coinvolti possono assumere solo quella determinata configurazione. Non descrivere tale comportamento può portare all'identificazione di reali problemi CDC ma può anche portare all'identificazione di numerose violazioni che risultano essere in realtà solamente "rumore". Invece, andando a descrivere il comportamento corretto di tali segnali, l'analisi CDC può portare a non identificare violazioni o può portare all'identificazione di violazioni che possono risultare essere reali problematiche CDC.

Le configurazioni tipiche assumibili da parte di un segnale o gruppi di segnali sono le seguenti:

1. **Statico**: Non viene presa una decisione sul valore logico assunto da parte del segnale ma si afferma che quest'ultimo non può commutare durante il funzionamento del dispositivo. Ad esempio, è possibile anche solo pensare ad un trasferimento CDC di un segnale statico. Tale trasferimento non si riflette in problemi di metastabilità non essendoci la possibilità di violare i tempi di setup/hold di un FF, questo perché il segnale in questione non può cambiare il suo valore. Configurando la serie di segnali che hanno un comportamento come quello appena descritto, vi è la possibilità di escludere possibili violazioni che in realtà non risultano essere reali problemi CDC.
2. **Costante**: Il comportamento è simile a quello statico, ma viene assegnato al segnale un valore logico ben definito e quest'ultimo non può cambiare durante il funzionamento del dispositivo. Anche in questo caso il trasferimento CDC di tale segnale non può portare ad una violazione dei tempi di setup/hold di un FF, in quanto quest'ultimo non può cambiare il suo valore. Grazie a tale configurazione è anche possibile impostare determinate funzionalità del dispositivo che sono discriminate in base al valore logico assunto, ad esempio, da un segnale di configurazione.
3. **Mutualmente esclusivi**: I segnali possono anche essere settati come mutualmente esclusivi, il che vuol dire che i segnali, all'interno di un gruppo definito, non possono commutare contemporaneamente ma solo uno per volta.
4. **Codificato**: Possiamo anche definire segnali che seguono una determinata codifica, come ad esempio la "codifica gray".
5. Infine, è anche possibile eliminare parte del proprio design che non deve essere analizzato durante l'analisi, quindi identificare "Falsi cammini" in cui non si devono effettuare controlli CDC.

Ovviamente, le configurazioni appena riportate sono le configurazioni più comuni, ma è sempre possibile definire manualmente altre modalità di comportamento dei segnali che lo richiedono. È necessario sottolineare l'importanza di questo step perché nel momento in cui viene descritto il più possibile il comportamento del design attraverso i suoi segnali, si riesce ad avere una sempre più accurata identificazione degli errori CDC effettivi del dispositivo ed eliminare la segnalazione di eventuali violazioni che nella realtà non risultano essere reali problematiche CDC.

4.3.3.3 *Identificazione dei domini di clock*

In questo step si procede con l'identificazione dei domini di clock presenti nel design sotto analisi. Vengono estrapolati tutti i segnali che fungono da clock per i registri/FF presenti nel design e che quindi popolano i loro domini di appartenenza. Dunque, si prendono in considerazione i segnali di clock derivati dai clock principali presenti all'interno di un design. Bisogna anche introdurre il concetto di "dominio di clock combinato", ovvero un dominio che è combinazione di più domini di clock. È possibile prendere in considerazione una porta logica OR: si supponga che gli ingressi di tale porta sono segnali appartenenti a due domini di clock differenti e questo implica che l'uscita di tale porta logica caratterizza un terzo dominio di clock che è combinazione dei due domini presenti in ingresso ad essa. In questo caso, il dominio combinato ottenuto viene definito come dominio "Combinatorio", perché proveniente da una logica combinatoria. Anche il segnale di uscita di un multiplexer può caratterizzare un dominio di clock combinato, in questo caso definito "Mux", nel momento in cui il selettore non è ad un valore logico costante ma dipende dal funzionamento del design.

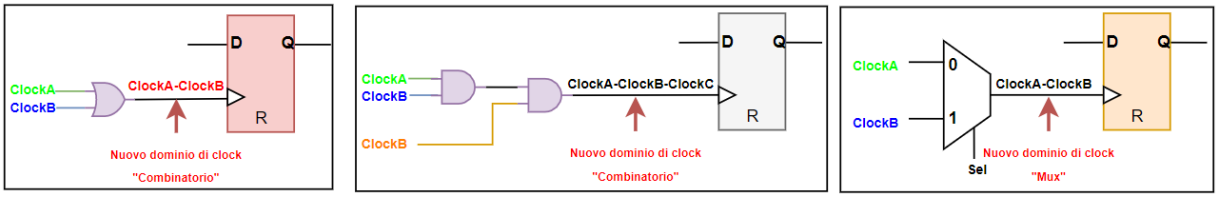

Figura 4-19: Dominio combinato

Nella Figura 4-19 vengono riportati degli esempi di generazione di domini di clock combinati dovuti dalla presenza di logica combinatoria nel cammino del segnale di clock. Nel caso riportato a sinistra è presente una porta logica OR i cui ingressi sono dei segnali di clock differenti. Il risultato ottenuto dalla porta OR è dunque un segnale di clock ottenuto dalla combinazione di quest'ultimi. Al centro viene riportata una situazione simile alla precedente. A destra, invece, viene riportato un caso di generazione di un dominio di clock combinato dovuto dalla presenza di un mutiplexer. In questo caso, come descritto precedentemente, non essendoci alcun tipo di configurazione sul selettore del multiplexer, non è possibile definire a quale dominio di clock appartenga il segnale presente in uscita al mux e viene dunque generato un dominio di clock combinato. Nel momento in cui il selettore del mux assume invece una configurazione "costante" ad un valore logico ben definito, non viene più generato un dominio di clock combinato.

Viene adesso riportato nella Tabella 1 un esempio di visualizzazione dell'elenco dei domini di clock presenti in un design ed i segnali di clock che popolano i rispettivi domini.

Domini di clock	***Nome segnali***
Clock_system	Clk_from_Mux1
	Clk_afterBuffer
	…
Clock_interface	clk_from_Mux2
	OR2_clk
	…

Tabella 1: Domini di clock

4.3.3.4 Regole CDC

È ovviamente possibile introdurre delle proprie regole nell'analisi CDC. Per regole si può intendere ad esempio:

- Modificare la criticità di determinate tipo di violazioni. Può essere quindi impostata una priorità degli errori CDC che vengono identificati, mostrando principalmente i problemi che si vogliono analizzare del proprio design.
- Modificare, ad esempio, il numero di FF che devono essere disposti in serie per poterli identificare come sincronizzatore standard multi-Flop. Ad esempio, se le proprie strutture di sincronizzazione sono caratterizzate da tre FF in serie, vi è la possibilità di modificare le regole CDC al fine di rilevare tre FF in serie come un sincronizzatore standard.
- ….

Vengono quindi introdotte delle particolari condizioni al fine di avere una migliore gestione e visualizzazione delle violazioni riportare durante l'analisi del DUT.

4.3.3.5 Dichiarazione schemi non convenzionali

Come già introdotto nella sezione 4.2.2, dove sono stati introdotti alcuni degli schemi di sincronizzazione comuni, vi è la possibilità di implementare nei propri design strutture di sincronizzazione "custom" o meglio definite "user-defined". Durante l'analisi strutturale si procede con l'identificare gli schemi di sincronizzazione implementati nel design in maniera automatica e dunque determinate strutture di sincronizzazione potrebbero non essere identificate essendo strutture non convenzionali. Al fine di ridurre il numero di violazioni che non risultano essere delle reali problematiche CDC, vi è la possibilità di aggiungere ai tool di verifica strutturale i vari sincronizzatori custom implementati all'interno del design. Ovviamente, devono essere aggiunti tali schemi user-defined all'elenco dei possibili schemi predefiniti ma devono anche essere aggiunti i loro relativi protocolli di funzionamento "custom", al fine di poter effettuare una loro verifica funzionale con l'ausilio della verifica formale.

4.4 Fasi CDC

Dopo aver quindi terminato la fase di configurazione CDC del design, si procede quindi con gli step caratterizzanti le "Fasi CDC".

4.4.1 Identificazione delle coppie CDC

In questo step dell'analisi si procede con l'identificare le "Coppie" CDC presenti nel sistema. Essenzialmente, vengono identificati tutti i percorsi di Clock-Domain-Crossing presenti nel sistema e vengono raggruppati e classificati in funzione del loro dominio di partenza e del loro dominio di destinazione. Viene quindi presentata una tabella all'interno del quale sono presenti tutte le coppie "Dominio di partenza-Dominio di destinazione" CDC presenti nel DUT e per ogni coppia viene riportato l'elenco di unità "Sorgente-Destinazione" coinvolte nel percorso identificato. Per poter avere una migliore visione del compito eseguito in tale step, viene riportato un esempio semplificato di quanto viene visualizzato dall'analisi nella Tabella 2. In tale passaggio non sono ancora identificati gli schemi di sincronizzazione presenti, ma solamente i percorsi CDC con i loro relativi FF di partenza, arrivo, i relativi segnali di clk ed i componenti all'interno dei quali sono presenti le unità di sorgente e destinazione.

Coppia CDC	***Unità sorgente***	***Componente Sorgente***	***Unità di destinazione***	***Componente Destinazione***
Clock_interface -> Clock_system	FF1 – clk_if	Component1	FF_dest1 – clk_sys	Digital1
	FF2 – clk_if	Comp_IF2	FF_dest2– clk_sys	Comp_Sys2
	…	…	…	…
Clock_system -> Clock_interface	FF3 – clk_sys	DigitalLogic	Reg_IF – clk_if	IFLogic
	FF4 – clk_sys	Comp2	FF_Dest3 – clk_if	Comp_IF3
	…	…	…	…

Tabella 2: Coppie CDC

4.4.2 Identificazione degli schemi di sincronizzazione e violazioni CDC

In questo punto dell'analisi, grazie all'esecuzione dello step precedente, si prosegue con l'identificare gli schemi di sincronizzazione presenti all'interno del design. Alcune tipologie comuni sono state introdotte nella sezione 4.2.2. I sincronizzatori presenti possono comunque portare ad errori funzionali, se utilizzati in maniera non corretta, nonostante la loro implementazione. Il primo obiettivo è quindi quello di identificare gli schemi di sincronizzazione implementati non correttamente. Il secondo obiettivo di tale step è analizzare l'assenza di elementi di sincronizzazione lungo i cammini CDC. Dunque, in questo punto dell'analisi si identificano tutti i percorsi CDC in cui potenzialmente si possono avere problemi di propagazione di metastabilità e di glitch e vengono elencate tutte le violazioni CDC presenti all'interno del DUT. Tali violazioni sono classificate in funzione della loro tipologia, questo perché, come spiegato in precedenza, non è solamente presente la violazione dell'assenza di schemi di sincronizzazione. Le più comuni tipologie di violazione sono state introdotte nella sezione 4.2.3. In questo punto dell'analisi bisogna manualmente analizzare tutte le violazioni riportate al fine di verificare se quest'ultime risultano essere reali problematiche CDC. Una volta identificate le reali violazioni, si può procedere con il modificare il codice RTL e quindi ri-eseguire la verifica CDC col fine di rilevare se tali problematiche sono state risolte. Nel momento in cui vengono

identificate violazioni CDC che non risultano essere dei reali problemi, si procede con effettuare il loro Waive, ovvero rimuovere tali violazioni. Al fine di avere una corretta rimozione di queste false violazioni, bisogna definire una giustificazione della rimozione effettuata e verificare attraverso l'ausilio della verifica formale che sia stata effettuata correttamente la rimozione della violazione.

4.4.3 Analisi CDC di Convergenza

In questo step dell'analisi si procede con l'identificare ed analizzare tutti i possibili percorsi di convergenza presenti nel design, come anche identificare le strutture implementate che possono riflettersi nella generazione di glitch, i quali possono propagarsi nei domini di destinazione e attraverso la logica del design. Come spiegato nella sezione 4.2.3.2 e 4.2.3.3 delle violazioni tipiche CDC, errori di convergenza e glitches possono portare a malfunzionamenti o comportamenti inaspettati del dispositivo, malfunzionamenti dovuti alla perdita di dati o al campionamento di impulsi spuri. Viene quindi preso in considerazione questo passaggio dell'analisi CDC per poter effettuare la ricerca di tali problemi ed ovviamente classificare le violazioni identificate in uno dei due casi, convergenza o glitches.

4.4.4 Analisi CDC funzionale

L'analisi CDC strutturale consente di identificare la presenza o l'assenza di strutture di sincronizzazione nei percorsi CDC presenti nel design ed anche verificare la loro corretta implementazione. L'analisi strutturale però non consente di assicurare che gli schemi di sincronizzazione implementati siano corretti dal punto di vista funzionale. La sola presenza di un sincronizzatore non assicura sempre il corretto passaggio CDC. In parole semplici, l'aver implementato tutte le strutture di sincronizzazione all'interno del proprio design non garantisce l'aver rimosso tutti i problemi CDC possibili se tali strutture non vengono controllate in maniera corretta. La verifica strutturale non prende in analisi i protocolli adottati per i segnali CDC ed ecco il motivo per il quale viene effettuata un'analisi CDC funzionale.

- Prendendo in considerazione una struttura standard come lo schema handshake introdotto nella sezione 4.2.2.3, questo può effettivamente funzionare in modo corretto se vengono rispettate le regole fondamentali riguardanti i segnali di "Request" e "Acknowledgement". Viene quindi creato un protocollo di verifica per poter validare il corretto funzionamento di tale struttura: il ricevitore non può asserire il segnale di "Acknowledgement" fino a quando il trasmettitore non ha asserito il segnale di "Request".

- Un segnale CDC può attraversare un comune sincronizzatore a due FF, ma nel momento in cui il dato/segnale in questione non rimane stabile all'ingresso del sincronizzatore per il tempo necessario al suo campionamento, quest'ultimo potrebbe non essere in grado di acquisire il dato per il dominio di destinazione, riflettendosi quindi nell'avere una perdita di dati. Quindi, bisogna assicurare che ogni commutazione di un segnale, trasmesso da parte del dominio di trasmissione, rimanga stabile per un tempo sufficiente al campionamento da parte del dominio di ricezione. Viene quindi creato un protocollo di verifica che si assicura che venga rispettato tale comportamento.

Per poter verificare il corretto campionamento del dato bisogna quindi ricorrere all'analisi CDC funzionale, al fine di verificare che il protocollo associato al sincronizzatore implementato sia corretto

per il passaggio CDC in questione. Essenzialmente, i controlli CDC funzionali hanno il compito di verificare che il dato/segnale che si sta propagando verso il dominio di clock di destinazione venga campionato correttamente, ovvero che non vi siano perdite o corruzioni di quest'ultimo e che vengano rispettate le procedure caratterizzanti i protocolli delle strutture implementate.

Il tool di verifica CDC può generare alcune delle proprietà formali utili per verificare il corretto funzionamento di alcuni dei sincronizzatori standard identificati ma le proprietà generate sono limitate esclusivamente alle strutture di sincronizzazione standard.

Ad esempio, prendendo in analisi l'esempio di struttura di sincronizzazione illustrata nella sezione 4.2.2.2, la proprietà che può essere generata è basata sul segnale di enable e sulla stabilità del dato.

- Se il segnale di "Enable" è al valore logico 1, allora i dati trasmessi "Data[0]" e "Data[1]" devono essere stabili ad ogni colpo di clock del dominio di trasmissione.
  (@posedge CLKB) Enable |=> stable(Data)

Dunque, questo step dell'analisi funzionale è principalmente basato sulla necessità di descrivere manualmente le proprietà, asserzioni e assunzioni, definite "user-defined", necessarie per poter validare i protocolli adottati all'interno del design che di cui vuole avere la conferma del loro corretto funzionamento e dunque validare i comportamenti aspettati delle strutture e protocolli di acquisizione presenti nel design.

L'analisi CDC funzionale è possibile grazie all'ausilio della verifica formale, approfondita nel capitolo 10.

5 Applicazione del flusso di analisi CDC

Viene adesso applicato il flusso CDC sul design in analisi e vengono riportate le configurazioni effettuate utili all'impostazione dell'ambiente CDC e ad una corretta verifica del design. Le procedure effettuate sono illustrate seguendo il flusso introdotto nel capitolo 4.

5.1 Impostazione del design-CDC

- **Clock e Reset**
  - Il design in analisi non viene analizzato in blocchi specifici ma viene analizzato nella sua interezza e vengono dichiarati il clock di sistema "Clock_System" e il clock di interfaccia "Clock_Interface". Essi sono definiti con entrambi i fronti (salita e discesa) attivi, essendo il design caratterizzato da FF sensibili ad entrambi i fronti. Non viene definita nessun tipo di relazione tra i 2 clock del design, essendo questi asincroni.

  - Il dispositivo in questione dispone solamente di un reset esterno, POR, attivo basso e asincrono. Tale reset è un Power On Reset ed è quindi necessario definire in maniera corretta tale caratteristica, in quanto quando esso è asserito, non sono attivi i clocks presenti nel design. Questa caratteristica deve essere tenuta in considerazione durante l'analisi RDC, al fine di rimuovere false violazioni e far emergere errori RDC effettivi. Viene approfondita la sua impostazione nel flusso operativo RDC nel capitolo 8.

- **BlackBox**
  - Nel design sotto analisi sono state effettuate le blackbox sulle componenti di memoria, in quanto come citato nella sezione 3.1.1, quest'ultime sono IP validate CDC-RDC esternamente di cui non si ha la necessità di effettuare una loro analisi. Le memorie non volatili vengono quindi impostate come BlackBox e non ne viene effettuata un'analisi interna CDC-RDC ma vengono effettuate le configurazioni delle loro porte di ingresso e di uscita, al fine di avere una corretta interazione con il design. Nello step successivo di configurazione dei segnali del design, sezione 4.3.3.2, si procede con il configurare il comportamento dei loro segnali di ingresso e di quelli di uscita.

5.2 Configurazione delle porte del design

- Vengono riportati, nella Tabella 3 sottostante, alcuni esempi di segnali configurati in maniera automatica oppure tramite ausilio dell'utente.

Segnale	Tipo	Configurazione	Dominio di clock
Scan_Test_Enable	*Uscita Primaria*	*Costante == 1'b0*	*Sincrono a tutti*
SPI_Enable	*Ingresso Primario*	*Costante == 1'b1*	*Sincrono a tutti*
PowerDown_Mem	*Uscita Primaria*	*Associazione Manuale*	*Clock_System-Clock_Interface*
Pad_sign1	*Ingresso Primario*	*Statico*	*Sincrono a tutti*
Pad_sign2	*Ingresso Primario*	*Associazione Autom.*	*Clock_Interface*
PowerDown_ADC	*Uscita Primaria*	*Associazione Autom.*	*Clock_System*
……	……	……	……

Tabella 3: Tabella segnali configurati

Come è possibile notare dalla tabella riportata, molte configurazioni di porte di ingresso e di uscita sono descritte come "Associazione Automatica". Bisogna sottolineare che questa tipo di associazione non implica che non vi è stata nessuna tipo di configurazione da parte dell'utente, ma semplicemente indica che il tool ha dedotto la relazione che vi è tra il segnale in questione e il suo dominio di clock, come è stato introdotto nella sezione 4.3.3.1 . Determinate associazioni automatiche sono possibili solamente dopo aver eseguito una configurazione dei comportamenti dei segnali, configurazione ovviamente effettuata da parte dell'utente in base a specifiche e analisi che voglio essere intraprese. Quest'ultime contribuiscono a fornire informazioni al tool per poter effettuare un'associazione corretta dei domini del design alle porte unclocked in questione. Ad esempio, una vera e propria associazione automatica da parte del tool è riferita al segnale di uscita "PowerDown_ADC". Quest'ultimo è automaticamente associato al dominio di clock di sistema senza nessun tipo di configurazione da parte dell'utente, in quanto tale uscita è direttamente dipendente dal FF "PowerDown_Adc" avente come clock "Clock Sistema", come è possibile notare nella Figura 5-1 nella parte di sinistra. Ovviamente, il segnale "ScanTestEn" non contribuisce in alcun modo essendo costante.

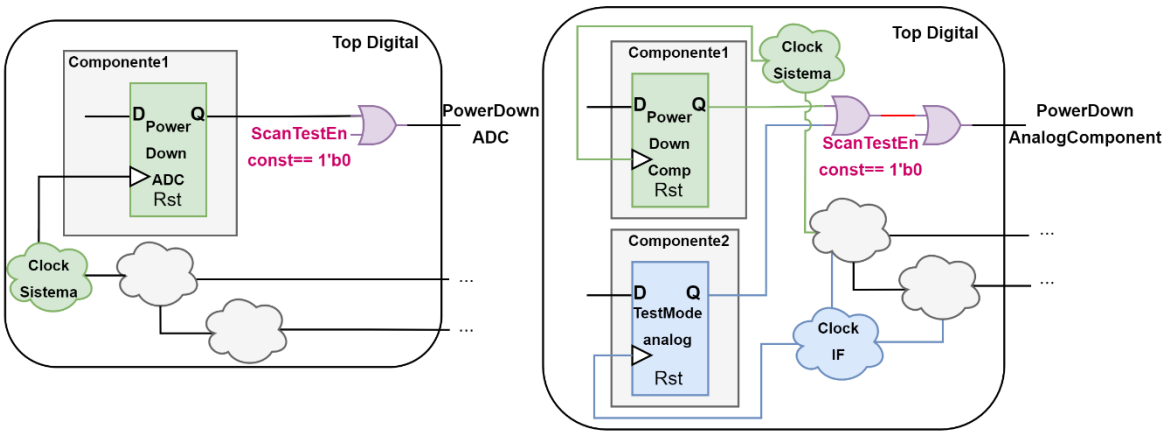

Figura 5-1: Esempi di associazione

Invece, per quanto riguarda il segnale di uscita "PD_AnalogComponent", quest'ultimo è dipendente da entrambi i clock, "Clock Sistema" e "Clock Interface", essendo l'uscita della porta logica OR avente

come ingressi i segnali “PD_AnalogComponent” e “TestMode_AnalogComponent”. Il segnale “TestMode_AnalogComponent” è un segnale che viene configurato come Statico, ovvero un segnale che non effettua commutazioni una volta impostato, in quanto non è utilizzato durante il funzionamento del dispositivo. Dunque, dopo tale configurazione, il tool esegue in maniera automatica l’associazione del segnale di uscita “PD_AnalogComponent” al dominio di clock “Clock sistema”. Quindi, questo è un esempio di associazione automatica sotto ausilio dell’utente.

Nel momento in cui non ci fosse stata una configurazione statica del segnale “TestMode_AnalogComponent”, il tool avrebbe automaticamente effettuato un’associazione ad un dominio di clock ottenuto dalla combinazione dei due domini presenti all’ingresso della porta OR. Infatti, il segnale “TestMode_AnalogComponent” è proveniente dal dominio di clock di interfaccia, mentre il segnale “PD_AnalogComponent” è proveniente dal dominio di clock di sistema. In un caso del genere viene creato un terzo dominio di clock “Clock_system-Clock_interface” che è ottenuto dalla combinazione dei due clock in questione.

È infatti possibile notare, nella Tabella 3, che i segnali “TestMode1” e “PowerDown_Memoria” sono appartenenti al dominio di clock combinato ottenuto dal clock di sistema e dal clock di interfaccia. In questo caso non viene effettuata alcun tipo di correzione non essendo possibile effettuare configurazioni dei segnali che coinvolgono tali porte.

È anche possibile notare che alcuni segnali vengono definiti come “Sincroni a tutti”, come ad esempio i segnali costanti “ScanTestEnable” e “SPI_Enable”. In questo caso, definendo tali segnali come sincroni a tutti i domini di clock, si sta implicitamente definendo tali segnali come completamente asincroni ai domini di clock presenti. Il suo nome può essere fuorviante, ma il suo significato è definire un determinato segnale come asincrono ai domini di clock presenti nel sistema. Tali segnali sono però definiti costanti o statici e dunque non vi è necessità di effettuare un’associazione ad un dominio di clock specifico.

Dunque, nonostante il tool di verifica può proseguire con associazioni automatiche delle porte del design analizzando il codice RTL, l’utente deve verificare e validare che queste associazioni sono state effettuate in maniera corretta e deve ovviamente effettuare la configurazione di determinati segnali di cui si conosce il comportamento, al fine di avere un’associazione automatica sempre più corretta ed escludere eventuali impostazioni che non sono contemplate da specifiche.

5.3 Configurazione dei segnali

Si procede quindi con la configurazione dei segnali il cui comportamento è noto:

1) La prima configurazione riguarda il segnale “*ScanTestEnable*”, responsabile per l’attivazione della DFT nel design e anche per la scelta del segnale di Reset corretto per il funzionamento del dispositivo. Il segnale *ScanTestEnable* viene settato costante a 0, per poter per l’appunto entrare nella funzionalità normale del dispositivo ed escludere la DFT.

La task attuale è dunque principalmente impostata dalle due seguenti assunzioni:

- *“Assume -env {ScanTestEnable ==1’b0}”*;
- *“Assume -env {SPI_Enable== 1’b1}”; --Abilitazione della IF SPI*

Bisogna sottolineare che per “Assunzione” si intende una configurazione effettuata dall’utente, necessaria per l’impostazione del dispositivo. Le assunzioni sono per loro natura considerate come

corrette. Quindi, quest'ultime non sono soggette a controlli durante l'analisi CDC e di conseguenza non viene verificata la loro validità. Bisogna porre molta attenzione al tipo di assunzioni CDC che vengono effettuate, in quanto, nel momento in cui queste risultano essere sbagliate oppure è stata eseguita una configurazione errata, possono riflettersi in un sovra-condizionamento del design che può portare a nascondere reali violazioni CDC. La configurazione "-env", delle assunzioni appena descritte, riporta come significato che le assunzioni sono valide anche durante il periodo in cui il reset è asserito. Quindi, dalla loro impostazione in poi, esse vanno a definire e condizionare tutte le future assunzioni da effettuare e quindi vanno a definire la task di analisi corrente.

2) Le prossime assunzioni hanno come ruolo l'impostare la disabilitazione completa delle interfacce I3C/I2C che, come spiegato precedentemente, vengono escluse dall'analisi attuale. Con le assunzioni di seguito riportate, si sta assicurando che i clock legati alle interfacce di I2C e I3C sono stabili per tutto il funzionamento del dispositivo.
    - *Assume -env {I2C_Enable== 1'b0}; Disabilitazione della IF I2C*
    - *Assume -env {I3C_Enable== 1'b0}; Disabilitazione della IF I3C*

Sono quindi state definite le assunzioni utili alla definizione della task di analisi che si vuol effettuare sul DUT in questione.

- Come citato nella sezione 4.3.2, le memorie presenti nel design sono state impostate come BlackBox e dunque, allo scopo dell'analisi della loro integrazione e connettività del design, viene effettuata una configurazione dei loro segnali di ingresso e dei loro segnali uscita e viene effettuata l'associazione ai domini di clock rispettivi. La memoria non volatile è operativa esclusivamente nella fase di avvio del dispositivo, ovvero durante la fase di "Boot". Questa fase di avvio del dispositivo non viene considerata nell'analisi CDC. La verifica CDC che viene effettuata prende solamente in analisi il dispositivo quando esso è nella sua fase funzionale. La memoria non volatile non viene quindi configurata durante il funzionamento del design e non è attiva e dunque i suoi segnali di ingresso e di uscita non sono soggetti a commutazioni ma rimangono stabili ai valori configurati nella fase di avvio del dispositivo.

Si prosegue con l'impostazione della configurazione dei segnali della memoria e viene riportato un esempio di quanto effettuato nella Tabella 4.

Segnali memoria non volatile	**Configurazione**
boot_running	Costante== 1'b0
boot_ running direct	Costante== 1'b0
Out1[7:0]	Statico
……	Statico

Tabella 4: Segnali della memoria non volatile

Da quanto spiegato precedentemente, bisogna prestare particolare attenzione ai segnali necessari per poter escludere la fase di boot del design dall'analisi CDC ed è quindi necessario settare valori costanti ai segnali "boot_running" e "boot_wen_sel". Di seguito la loro descrizione con le rispettive assunzioni:

- *Assume -env {boot_running == 1'b0};* Viene prestata particolare attenzione a questa tipo di assunzione, in quanto il segnale "boot_running" ha lo scopo di discriminare la fase di boot dalla fase funzionale. Essenzialmente, configurando tale segnale al valore logico 0, si sta affermando che il design non è in fase di boot ma è in fase di funzionamento. L'obiettivo di tale verifica è quella di effettuare un'analisi CDC esclusivamente nella fase operativa del dispositivo. Dunque, attraverso la configurazione di tale segnale, viene presa esclusivamente in considerazione il funzionamento del dispositivo dopo aver terminato la sua fase di boot.

- *Assume -env {boot_running_direct == 1'b0};* Quest'assunzione è legata alla precedente e consente di prendere in considerazione il clock di interfaccia che deve essere utilizzato durante il funzionamento del dispositivo. Nella task corrente in questo caso si tratta del clock di interfaccia SPI.

Viene riportato nella Figura 5-2 un estratto del design, al fine di comprendere di come le assunzioni effettuate influiscano sulla logica. È possibile notare che attraverso la configurazione del segnale di "ScanTesteEnable" e del segnale "boot_running_direct" si è andati ad escludere il segnale di uscita "boot" dal funzionamento del design, essendo non di nostro interesse la sua analisi nella fase di boot. È anche possibile notare che viene propagato nel design il clock di interfaccia SPI, mentre le interfacce I2C e I3C vengono riportate in rosso come "Costante", essendo il loro funzionamento disabilitato e quindi il loro clock di uscita stabile.

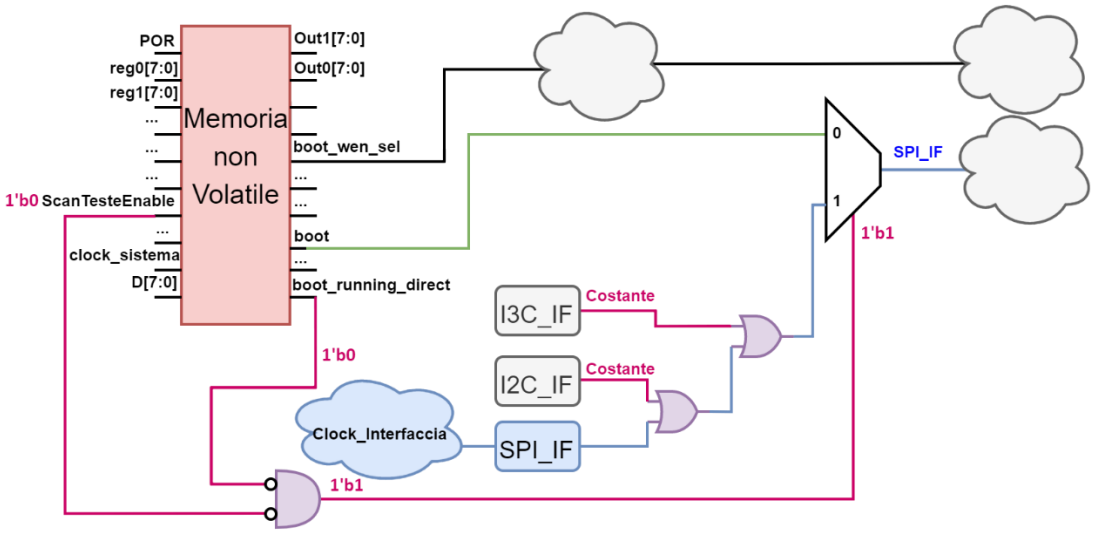

Figura 5-2: Esempio di struttura su cui sono state applicate delle assunzioni

Successivamente, nel corso della verifica del design, sono state effettuate ulteriori assunzioni che sono state derivate durante l'analisi delle violazioni CDC incontrate, al fine di continuare ad escludere quell'insieme di violazioni che non rappresentano reali problematiche CDC ed avere una descrizione sempre più completa ed accurata del DUT.

5.4 Identificazione dei domini di clock, regole CDC, dichiarazione schemi non convenzionali e identificazione delle coppie CDC

Come riportato nella sezione 4.3.3.3, nello step di "identificazione dei domini di clock" viene effettuata l'identificazione automatica di tutti i domini di clock presenti nel design e di tutti i segnali di clock che popolano tali domini. Dunque, vengono identificati tutti i segnali di clock, derivati dai clock principali, presenti all'interno del design e quest'ultimi vengono associati ai relativi domini di appartenenza. Essenzialmente, viene riportata una tabella caratterizzata dai domini di clock presenti nel design e per ogni dominio viene presentata una lista di segnali di clock derivati dai clock principali.

Domini di clock	***Nome segnali***
Clock_system	Clk_ Or2 Clk_Buffer …
Clock_interface	clk_gated I2C_scl …

Tabella 5: Elenco domini di clock del DUT

La Tabella 5 appena riportata rappresenta solamente una parte della lista di segnali di clock presenti nel design ed è riportata solo col fine di fornire un'idea di cosa viene ottenuto dall'esecuzione dell'analisi. L'elenco di segnali ottenuto è eccessivamente lungo da riportare nel lavoro di tesi e non porta informazioni aggiuntive rispetto all'esempio appena riportato.

Gli ultimi due step della configurazione dell'ambiente CDC, "regole CDC e dichiarazione degli schemi non convenzionali", non sono stati presi in analisi per il design in questione, essendo quest'ultimo non caratterizzato da schemi di sincronizzazione non convenzionali da dichiarare e non vi è stata la necessità di integrare regole CDC.

Come riportato nella sezione 4.4.1, nel primo step delle fasi CDC definito "identificazione delle coppie CDC" viene effettuata l'identificazione automatica di tutte le coppie CDC presenti nel design.

Coppia CDC	***Unità sorgente***	***Componete Sorgente***	***Unità di destinazione***	***Componente Destinazione***
Clock_interface -> Clock_system	RST_CFG– clk_if Config – clk_if …	Instance1 Comp_IF2 …	Start – clk_sys pd_memD1– clk_sys …	Digital1 Comp_Sys2 …
Clock_system -> Clock_interface	Reg3 – clk_sys FF4 – clk_sys …	DigitalLogic Comp2 …	Reg2_IF – clk_if FF_Dest3 – clk_if …	IFLogic Comp_IF3 …

Tabella 6: Elenco Coppie CDC presenti all'interno del DUT

Anche la Tabella 6 rappresenta solo una parte della lista delle coppie CDC presenti nel design ed è riportata solo col fine di fornire un'idea di cosa viene visualizzato durante l'analisi. L'elenco reale di segnali di trasmissione e di destinazione, con le relative componenti sorgente e destinazione, è eccessivamente lungo da riportare nel lavoro di tesi e, come la tabella precedente, non porta informazioni aggiuntive rispetto all'esempio appena riportato.

6 Risultati CDC

Dopo aver eseguito le configurazioni dell'ambiente CDC e del design, è possibile procedere con il ricavare i risultati CDC ottenuti dall'esecuzione delle Fasi del flusso di analisi. Si procede quindi con l'analizzare nel dettaglio i risultati ottenuti dallo step di "Identificazione degli schemi di sincronizzazione e violazioni CDC".

6.1 Identificazione degli schemi di sincronizzazione e violazioni CDC

6.1.1 Violazioni CDC – TestMode

Sono state identificate, all'interno della struttura presente nella Figura 6-1, una serie di violazioni CDC che possono portare a diverse tipologie di errori funzionali. La struttura in questione rappresenta una parte della logica del design legata ad una delle modalità di test e dunque rappresenta una logica che non viene utilizzata durante la fase funzionale del dispositivo. La violazione riportata in questa sezione rappresenta una delle principali problematiche CDC riscontrate all'interno del design dove è stato necessario implementare delle strutture di sincronizzazione ed applicare un protocollo per poter rimuovere i problemi CDC presenti in tale struttura. Quest'ultima non crea problemi per le funzionalità del dispositivo, ma vengono comunque applicate delle soluzioni per irrobustirla e rendere tale parte della modalità di TestMode più corretta. Viene successivamente mostrata, nella sezione 12.2.1 in iniezione della metastabilità e glitch in simulazione, l'effetto delle problematiche riscontrate in tale struttura.

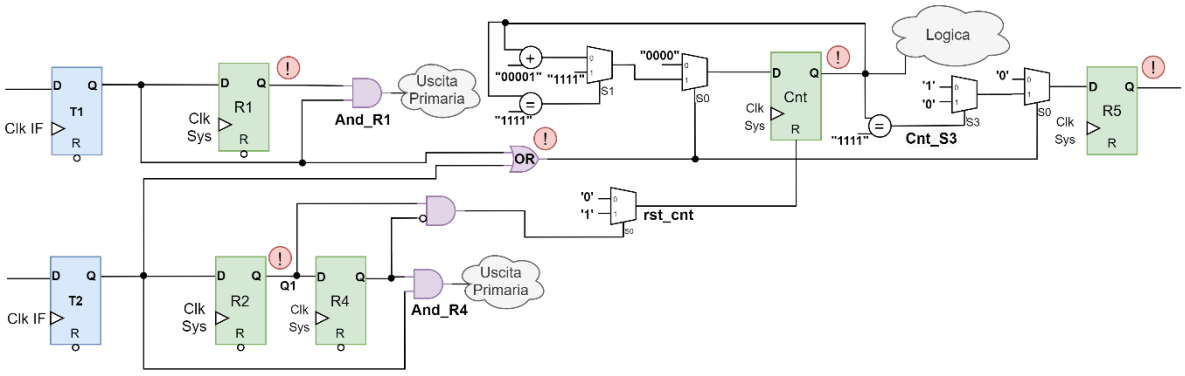

Figura 6-1: Schema del TestMode

- **Violazioni CDC identificate:**

Nello schema riportato è possibile notare la presenza di differenti percorsi CDC in cui sono assenti strutture di sincronizzazione. È possibile notare anche la presenza di logica combinatoria nel cammino CDC che può portare al problema di generazione e campionamento di eventuali glitch nel dominio di destinazione.

1) La prima violazione CDC ha come punto di partenza il FF TestMode1(T1), appartenente al dominio di clock di interfaccia, e come punto di destinazione il FF R1, appartenente al dominio di clock di sistema.

2) La seconda violazione CDC coinvolge il FF di trasmissione TestMode2(T2) e il FF di destinazione R2. È possibile notare la presenza di un ulteriore FF R4 in serie al FF R2 e questo rappresenta un sincronizzatore standard a due FF. Il problema CDC è legato alla connessione di una logica combinatoria in uscita al registro R2 che rappresenta il primo elemento della catena di sincronizzazione.

3) Le ulteriori due violazioni hanno un ugual punto di trasmissione, costituito dai FF TestMode1(T1) e TestMode2(T2), e come punti di destinazione i FF Cnt ed R5.

- **Descrizione delle violazioni CDC identificate:**

Si procede dunque col descrivere le possibili problematiche che possono essere incontrate lungo i percorsi CDC identificati.

1) Non essendo presente un sincronizzatore nel percorso CDC tra il FF TestMode1(T1) ed il FF R1, vi è la possibilità di una violazione dei tempi di setup/hold di quest'ultimo portando dunque ad una possibile propagazione della metastabilità attraverso la logica del design e verso l'esterno di esso. In Figura 6-2 viene riportato un esempio di violazione.

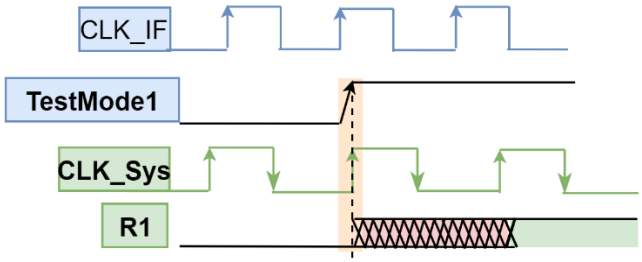

Figura 6-2: Esempio metastabilità

2) La seconda violazione riferita al percorso CDC tra il FF TestMode2(T2) e il FF R2 è legata alla presenza di una logica combinatoria all'interno della struttura di sincronizzazione. Il collegamento di una logica combinatoria più o meno complessa all'interno di uno schema di sincronizzazione è sconsigliato poiché è possibile avere la propagazione della metastabilità attraverso tale logica.

3) Infine, nel percorso CDC identificato tra i FF di trasmissione TestMode1(T1) e TestMode2(T2) ed i FF di destinazione Cnt e R5, vi è la possibilità di avere una violazione dei tempi di setup e di hold di quest'ultimi e dunque una possibile propagazione di metastabilità. La presenza di logica combinatoria in tale percorso CDC può anche portare alla possibile generazione di

glitches, i quali possono riflettersi in un movimento non corretto dei selettori dei multiplexer presenti in ingresso ai FF Cnt e R5. Il movimento non corretto di tali selettori può portare ad avere un errore all'uscita del contatore Cnt come anche all'uscita del FF R5. Essenzialmente, vi è la possibilità di avere un errore durante l'operazione di incremento del contatore come anche un errore all'uscita del FF R5, in quanto vi è la possibilità di generare una commutazione non voluta del selettore del mux in ingresso a quest'ultimi, a causa di un possibile glitch, che può quindi portare l'uscita del contatore al valore 0000 e l'uscita del FF R5 al valore logico 0. Ovviamente, in caso di tale errore viene perso lo stato corretto del FF R5 e del contatore, dove quest'ultimo prosegue in maniera errata col contare nuovamente dal valore 0000. In Figura 6-3 viene riportato un esempio di errore che può essere ottenuto a causa della generazione di un glitch in prossimità del fronte di salita del clock di sistema.

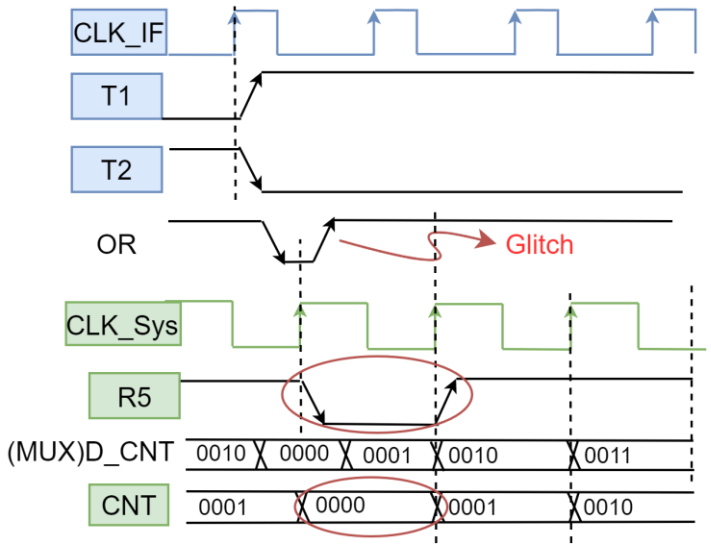

Figura 6-3: Esempio di errore dovuto alla generazione di un glitch

Tale problema viene anche mostrato in simulazione RTL con l'iniezione della metastabilità e glitch nella sezione 12.2.1 validando la problematica CDC che può essere incontrata in tale struttura.

- **Descrizione delle soluzioni attuabili per irrobustire la struttura:**

Viene adesso riportata, nella Figura 6-4, una possibile soluzione di implementazione per poter raggiungere una migliore robustezza della struttura appena introdotta a fronte della metastabilità. Essenzialmente, non vengono intaccati i FF che costituiscono i ritardi necessari alla funzionalità della struttura nei due rami definiti da TestMode1(T1) e TestMode2(T2), ma vi è ovviamente l'introduzione di una latenza aggiuntiva dovuta all'implementazione dei sincronizzatori standard multi-Flop. Questa introduzione della latenza non risulta essere un problema rilevante per tale struttura, essendo quest'ultima implementata per una modalità di test del dispositivo.

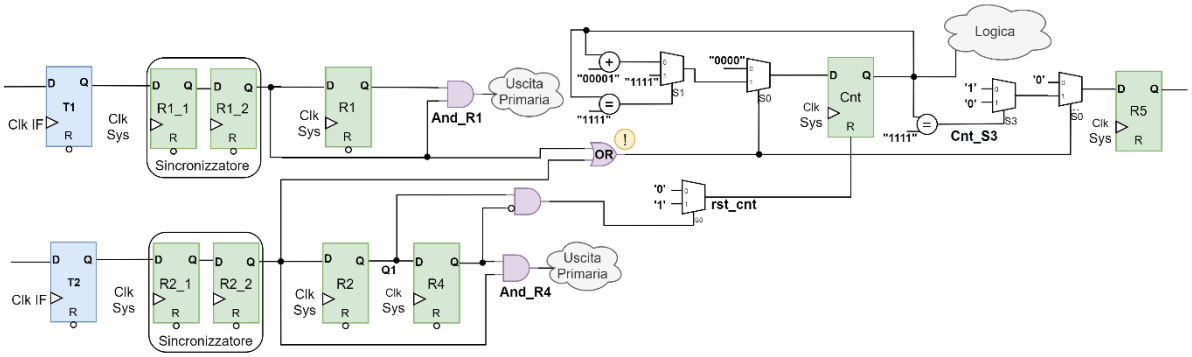

Figura 6-4: Struttura modificata per ridurre problemi CDC nel TestMode

- Con questa modifica si risolvono i percorsi CDC problematici che erano presenti nella struttura di partenza.

- È possibile notare però la presenza di un errore di convergenza, problema illustrato nella sezione 4.2.3.3, dovuto alle uscite dei due sincronizzatori che vengono convogliate verso la porta logica OR. L'uscita di tale porta logica procede con il selezionare l'ingresso dei multiplexer. Questo errore di convergenza è effettivamente presente e risulta essere un problema al momento della commutazione dei segnali TestMode1(T1) ed TestMode2(T2) dai valori logici 01 a 10 come anche 10 a 01. Dunque, per una possibile differenza nei ritardi di propagazione che possono essere presenti all'uscita dei FF TestMode1(T1) e TestMode2(T2), vi è la possibilità di ottenere in uscita a due sincronizzatori, durante tale tipo di commutazione, i valori logici 00 e dunque generare un errore funzionale. Al fine di evitare questa possibile problematica, bisogna ricorrere all'utilizzo di una configurazione dei segnali "mutualmente esclusiva" o utilizzare una codifica che non consenta la commutazione contemporanea dei due segnali. Tale problematica di convergenza viene illustrata attraverso l'iniezione della metastabilità in simulazione RTL nella sezione 12.2.1.

Al fine di risolvere completamente i problemi legati a tale struttura, si è deciso di proporre e adottare una procedura tale da non consentire l'emergere di problematiche CDC.

- **Descrizione della procedura da attuare:**

La procedura progettata richiede quindi di seguire un determinato timing specifico ed una corretta configurazione dei segnali al fine di procedere con tale TestMode senza incorrere in problematiche CDC. Questo implica che non è contemplato un comportamento differente da quello che sta per essere descritto.

La procedura consiste principalmente nella configurazione da seguire dei segnali TestMode1(T1) ed TestMode2(T1) ed il tempo necessario nel quale tali segnali devono rimanere costanti ai valori imposti. Dunque, avendo come punto di partenza i segnali TestMode1(T1) e TestMode2(T2) costanti al valore logico 00, essi devono rispettare la seguente serie di passaggi:

1) Il segnale TestMode1(T1) e il segnale TestMode2(T2) devono essere rispettivamente impostati ai valori logici 0 ed 1, successivamente è necessario attendere un tempo pari ad almeno diciotto colpi di clock di sistema prima di effettuare la prossima operazione;

2) Il segnale TestMode1(T1) e il segnale TestMode2(T2) devono essere rispettivamente impostati ai valori logici 1 ed 1, questo con lo scopo di non avere una commutazione simultanea dei due bit di TestMode. Può essenzialmente essere vista come una configurazione tale da non avere una commutazione contemporanea dei due segnali ai valori logici opposti;

3) Il segnale TestMode1(T1) e il segnale TestMode2(T2) devono essere rispettivamente impostati ai valori logici 1 e 0, successivamente è necessario attendere un timing specifico per poter procedere con la lettura dei risultati ottenuti dalla struttura di TestMode;

La seconda operazione descritta ha semplicemente lo scopo di non avere la generazione di un errore di convergenza dovuto dalla presenza della porta logica "OR", i cui ingressi sono le uscite dei due sincronizzatori implementati e l'uscita di tale porta logica pilota i selettori dei mux in ingresso al contatore ed in ingresso al FF R5.

Viene riportato nella Figura 6-5 il diagramma di timing della procedura appena descritta. Tale procedura consente quindi di rimuovere la problematica CDC di convergenza.

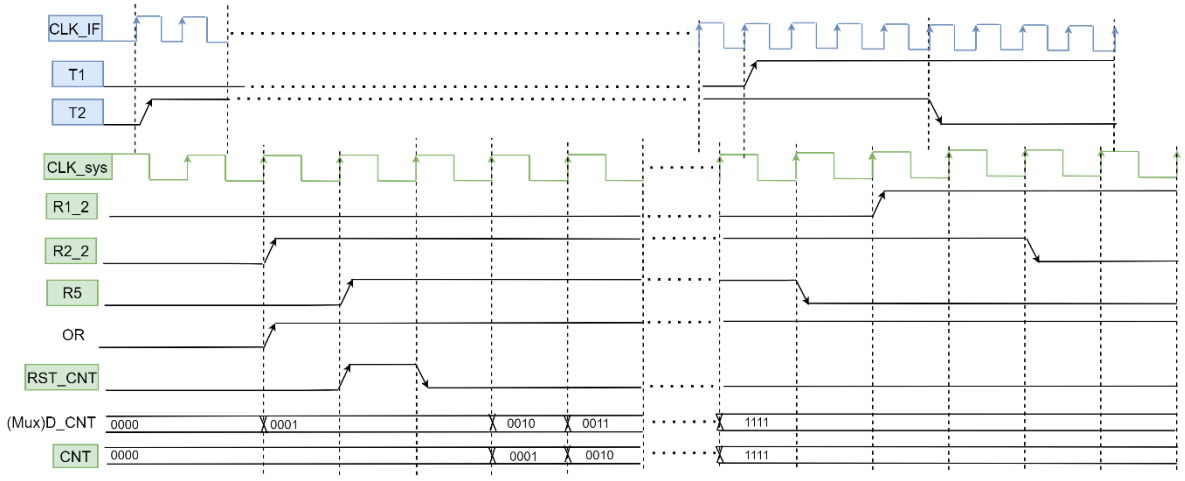

Figura 6-5: Diagramma di timing della procedura da attuare al fine di risolvere i problemi CDC

6.1.2 Violazione CDC – Registri di configurazione

Tra i risultati ottenuti dall'analisi sono state identificate violazioni CDC riferite ai segnali di configurazione presenti nel design. Le violazioni coinvolgono bit singoli e gruppi di bits atti alla configurazione della logica digitale presente nel design, come anche vengono presentate violazioni CDC che coinvolgono bus dati che effettuano un passaggio tra domini di clock asincroni.

La logica digitale presente è caratterizzata da uno o più registri di configurazione ad otto bit e sono anche presenti bus dati di diverse dimensioni. I registri di configurazione consentono di impostare le caratteristiche che compongono la logica digitale costituente il design. Ad esempio, attraverso la scrittura di questi registri è possibile settare le soglie oppure gli assi che devono essere coinvolti nella generazione dei vari interrupt. Attraverso i bus possono essere, ad esempio, trasferiti i diversi valori di soglia e di durata. Viene riportato un esempio di schematico estrapolato e semplificato dal design nella Figura 6-6 al fine di semplificare la visione delle possibili violazioni.

- **Violazione CDC identificata e descrizione:**

Essenzialmente, sono state identificate un numero elevato di assenze di strutture di sincronizzazione lungo i percorsi CDC riferiti ai registri di configurazione e bus dati. La violazione CDC ha come punto di partenza il dominio di clock interfaccia, essendo questi scritti attraverso l'interfaccia, e come punto di destinazione più componenti del design appartenenti al dominio di clock di sistema. Ognuno dei bit di configurazione viene dislocato in diversi parti del design attraverso della logica combinatoria come anche verso componenti atti alla gestione della logica digitale, dove quest'ultima è appartenente al dominio di clock di sistema. La violazione identificata è dunque caratterizzata dall'assenza di una struttura di sincronizzazione per ognuno di questi bit di configurazione e di dato. Dunque, nel momento in cui viene effettuata una scrittura attraverso il clock di interfaccia, si potrebbero avere violazioni dei tempi di setup/hold dei FF appartenenti al dominio di clock di sistema con conseguente generazione e propagazione dimetastabilità attraverso la logica costituente il design, come anche sono possibili perdite di dato e campionamenti di impulsi spuri (glitch) che portano ad avere risultati non coerenti. Come è possibile intuire, questo può portare a diversi problemi di funzionamento. Ad esempio, è possibile pensare ad una variazione non voluta dei contatori oppure ad una modifica dei valori di soglia come anche ad un comportamento non corretto della logica digitale che costituisce il cuore del design.

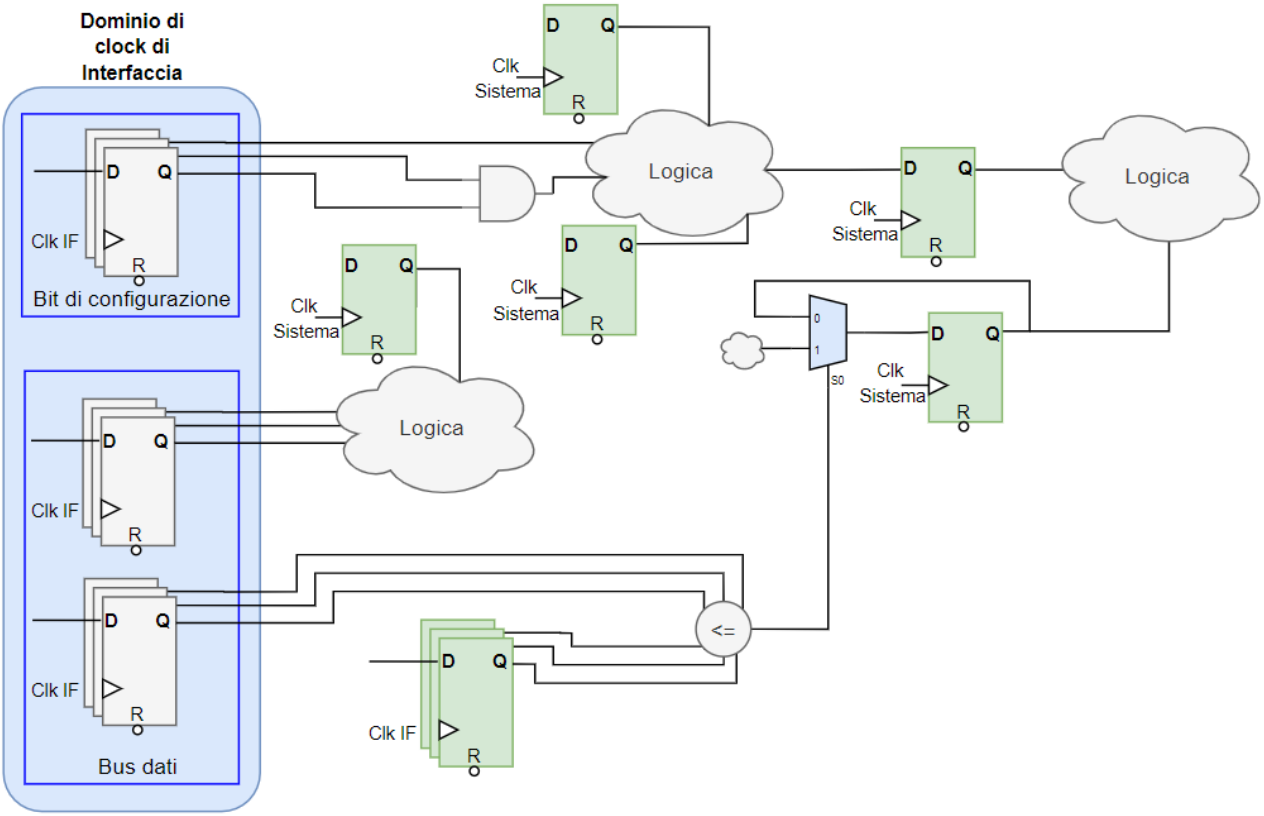

Figura 6-6: Estrapolazione del design dalle violazioni CDC riportate

- **I possibili problemi in cui è possibile incorrere sono:**
  - **Metastabilità**: Vi è ovviamente la possibilità di incorrere nella violazione dei tempi di setup e di hold dei registri di destinazione e dunque avere la propagazione della metastabilità attraverso il design. Quindi, se non vengono introdotti protocolli di acquisizione o schemi di sincronizzazione, si può incorrere in errori funzionali.

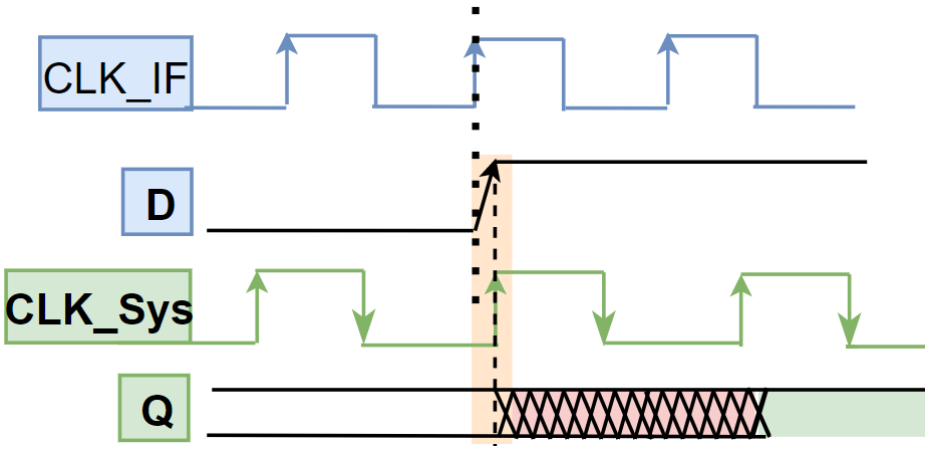

Figura 6-7: Possibile propagazione della metastabilità

  - **Glitch**: Non è solo possibile avere una generazione e propagazione della metastabilità ma vi è anche la possibilità di campionare nel dominio di destinazione un glitch dovuto ai differenti ritardi di propagazione dei segnali coinvolti in una logica combinatoria. Dunque, potrebbe non sorgere un problema di metastabilità ma potrebbe comunque

sorgere un impulso spurio che può essere acquisito e dunque generare comportamenti non voluti, soprattutto trattandosi di segnali di configurazione.

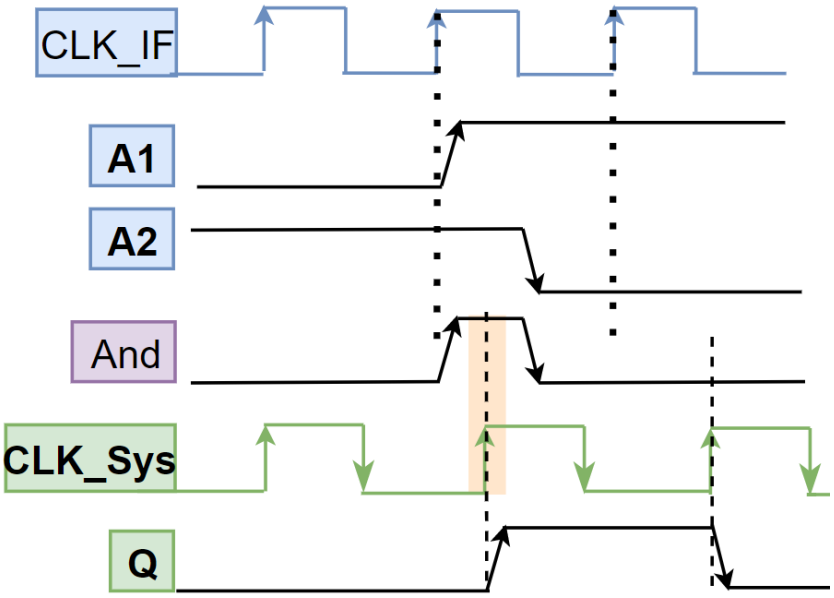

Figura 6-8: Possibile campionamento di un glitch

- **Descrizioni delle soluzioni a tale violazione CDC:**

Si può subito pensare di inserire delle strutture di sincronizzazione standard a due o tre FF per ognuno dei bit coinvolti nelle violazioni CDC. Questa è effettivamente una soluzione ma non attuabile per tutti i bits presenti. Devono essere implementate le strutture di sincronizzazione corrette a seconda della logica combinatoria presente nei percorsi CDC e a seconda del comportamento dei segnali coinvolti. Lo stesso discorso deve essere anche applicato sul come implementare strutture di sincronizzazione corrette per i bus presenti, questo per assicurarsi di non incorrere in problemi CDC nonostante l'implementazione di schemi di sincronizzazione. Bisogna effettuare uno studio corretto di implementazione dei sincronizzatori per poter evitare la generazione di una delle violazioni CDC tipiche elencate nella sezione 4.2.3.

  - **Soluzioni possibili per i bus dati:**

Per poter gestire i bus dati non è possibile utilizzare sincronizzatori standard a due FF per ognuno dei bit presenti nel bus, poiché con tale tipo di implementazione di sincronizzazione si incorre in un problema di incoerenza nella trasmissione dei dati. Non è possibile garantire che ognuno dei sincronizzatori implementati per il bus dati campioni correttamente il nuovo dato, ma alcuni di essi potrebbero commutare al valore del nuovo dato ed altri commutare al valore del dato precedente. In più, si potrebbe anche incorrere nel problema CDC di "Convergenza", ovvero tali bits possono convergere verso una logica combinatoria che può quindi generare un'uscita temporaneamente non corretta. Problema introdotto nella sezione 4.2.3.3.

La soluzione che può essere attuata per i bus dati è basata sulle strutture di sincronizzazione caratterizzate da un protocollo di trasmissione. Quindi si può ricorrere ad utilizzare strutture come "Mux Synchronizer" e "Handshake", introdotti nella sezione 4.2.2. Per l'esempio dei bus dati riportato

in Figura 6-6, può essere implementata la struttura “Mux Synchronizer”, introdotta nella sezione 4.2.2.2. È necessario implementare un segnale di “Enable” che deve essere sincronizzato con un multi-Flop standard a due FF per poter avere un campionamento corretto dei dati posti sul bus ed ovviamente quest’ultimi devono essere stabili per tutto il tempo in cui il segnale di enable è asserito. Dunque, una volta posti sul bus i dati e asserito il segnale di enable, quest’ultimi possono essere correttamente campionati nel dominio di destinazione ed è quindi possibile ottenere dei risultati coerenti.

- **Soluzioni possibili per i registri di configurazione**

Per quanto riguarda i registri di configurazione è possibile implementare strutture standard di sincronizzazione a due o tre FF. Bisogna però porre attenzione dove tali segnali CDC convergono o se vi è la presenza di logica combinatoria nel loro percorso che non può essere rimossa. Bisogna porre attenzione al non incorrere nelle violazioni CDC di convergenza e di glitch, introdotte nelle sezioni 4.2.3.2 e 4.2.3.3.

- Nel caso in cui non fosse presente alcuna logica combinatoria critica nel percorso CDC e quindi quest’ultimo risulta essere “pulito”, è possibile implementare una semplice struttura di sincronizzazione standard multi-Flop.

- Nel momento in cui fosse presente della logica combinatoria nel percorso CDC, bisogna prestare attenzione al comportamento dei segnali coinvolti in tale logica. Se quest’ultimi possono commutare contemporaneamente potrebbero portare alla generazione di un impulso spurio, il quale potrebbe essere campionato dalla struttura di sincronizzazione presente a valle della logica combinatoria. Si potrebbe sempre incorrere ad un problema di violazione di convergenza nel momento in cui vengano implementate strutture di sincronizzazione a monte della logica combinatoria. In questi casi, bisogna fare in modo di avere un comportamento “mutualmente esclusivo” dei segnali di configurazione che possono convergere in una logica combinatoria o applicare una determinata codifica.

- **Descrizione delle soluzioni attuabili nel design:**

Nel design in questione non possono però essere inseriti dei sincronizzatori per ognuno dei registri presenti nel sistema, poiché questo si rifletterebbe in un numero fin troppo elevato di sincronizzatori solo per poter gestire tali registri di configurazione e dunque questo si rifletterebbe su un incremento di area non accettabile. Sono presenti circa dieci registri da otto bit, questo si riflette nel dover implementare ottanta sincronizzatori standard e dunque centosessanta FF con il solo scopo di sincronizzazione. In più, è presente una logica combinatoria più o meno complessa nei percorsi CDC identificati e dunque questo non consente di applicare solamente dei sincronizzatori standard, ma vi è anche la necessità di implementare delle codifiche e dei comportamenti dei segnali che porterebbe anche ad un aumento della complessità del design. Questa non è quindi una soluzione efficace per poter solamente avere un corretto campionamento dei segnali di configurazione.
Tipicamente, la configurazione di un dispositivo non viene modificata più volte durante la sua fase funzionale e quindi questo è un ulteriore motivo per il quale non è giustificato un aumento di complessità e di area del design con il solo scopo di avere un sistema di sincronizzazione per i segnali di configurazione.

Vi è anche la necessità di implementare strutture per i bus dati e questo implica dover inserire, se viene utilizzata la struttura mux synchronizer, un multiplexer e un FF per ogni bit presente nel bus e in più vi è la necessità di implementare un segnale di enable ed un sincronizzatore standard multi-flop per quest'ultimo.

La soluzione identificata per il design sotto analisi consiste nell'effettuare la configurazione dei registri attraverso l'interfaccia con il clock di sistema inibito. È possibile effettuare una configurazione dei registri del dispositivo attraverso il clock di interfaccia disattivando il clock di sistema. In questo modo è possibile effettuare la scrittura dei registri di configurazione evitando la possibilità di violare i tempi di setup/hold dei FF appartenenti ai domini di sistema e non portando ad avere eventuali generazioni di metastabilità e problemi CDC legati a glitch e convergenza. Quindi, una volta effettuata la scrittura dei registri di configurazione, si procede con ritornare nella modalità operativa del dispositivo desiderata tra quelle disponibili e quindi fornire nuovamente il clock di sistema. Nel momento in cui il clock di sistema ritorna attivo, i segnali provenienti dai registri di configurazione risultano essere stabili al momento del loro campionamento da parte del dominio di clock di sistema e quindi possono essere acquisiti correttamente evitando i possibili problemi CDC. Al fine di rappresentare questa soluzione durante l'analisi del design, sono stati settati statici tutti i segnali che caratterizzano i registri di configurazione come anche i bus coinvolti nelle violazioni CDC. Con questa tipo di regola dei segnali è possibile rimuovere le violazioni CDC identificate riguardanti quest'ultimi. Possono essere settati statici poiché, come appena spiegato, nel momento in cui il clock è attivo quest'ultimi non possono commutare e l'unico modo per poter effettuare una loro nuova scrittura è quella di passare dalla fase in cui il clock del sistema viene disattivato. Vengono quindi aggiunte le seguenti configurazioni dei segnali coinvolti nelle violazioni identificate e tali configurazioni vengono riportate nello step della configurazione dei segnali introdotto nella sezione 4.3.3.2 nel capitolo 4.3. Ovviamente, lo svantaggio introdotto nella soluzione appena attuata è quello di dover effettuare una configurazione di tali segnali esclusivamente con il clock di sistema disattivato.

Viene adesso riportato un esempio di elenco rappresentativo dei segnali di quei registri sui cui è stata impostata una configurazione statica.

- Assume -env static {CFG1[7:0]} –Registri di configurazione;
- Assume -env static {CFG2[7:0]} –Registri di configurazione;
- Assume -env static {INTERRUPT_CFG [6:0]} –Registri di configurazione Interrupt;
- Assume -env static {threshold[7:0]} –Registri di configurazione della soglia;
- …..

6.1.3 Violazione CDC – Dati DSP

Vengono adesso riportati tre tipi di violazioni CDC identificate durante il corso dell'analisi. Queste violazioni sono state identificate come problemi CDC non reali per via del protocollo di acquisizione presente e ne viene riportato uno schema rappresentativo estrapolato dal design e semplificato, nella Figura 6-9, che rappresenta le diverse tipologie di violazioni presenti.

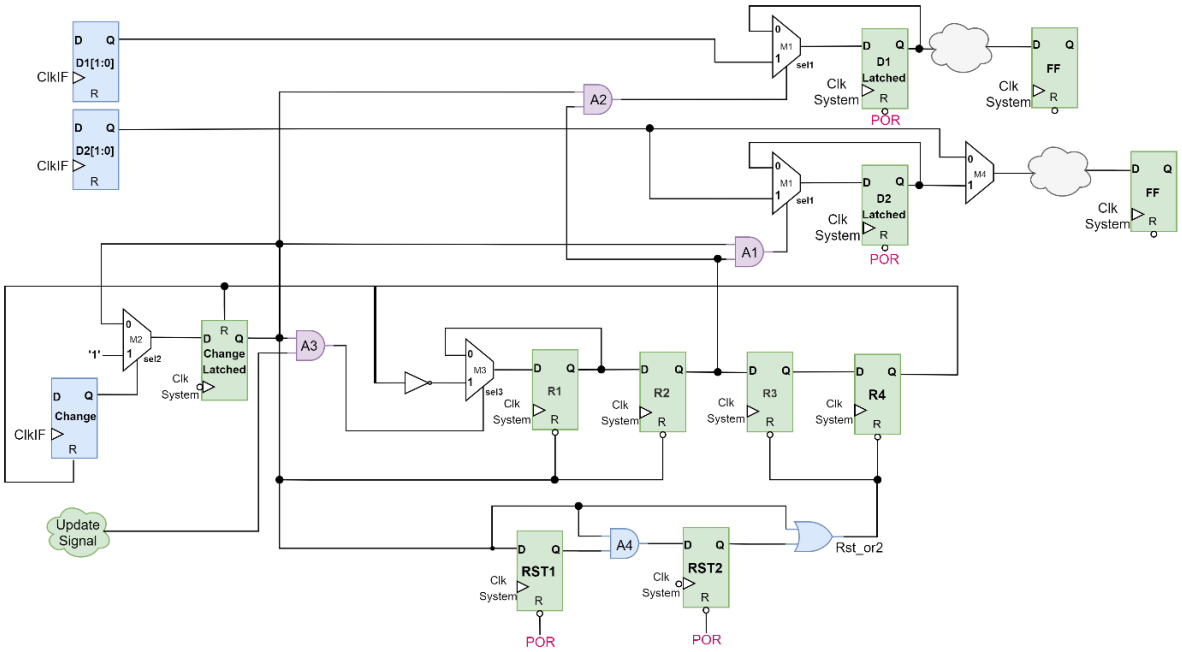

Figura 6-9: Struttura della violazione CDC Dati DSP

- **Violazioni CDC identificate:**

Le violazioni sono incentrate sui percorsi CDC che collegano i registri di trasmissione D1[1:0] e D2[1:0], appartenenti al dominio di clock di interfaccia, e i registri di ricezione "D1_Latched[1:0]" "D2_Latched[1:0]", appartenenti al dominio di clock di sistema. L'ultima violazione è invece incentrata sul percorso CDC tra il FF "Change" ed il FF "Change Latched". Vengono riportate tali violazioni in quanto sono effettivamente assenti, nei percorsi CDC in questione, strutture di sincronizzazione atte al trasferimento di tali dati. Le violazioni presenti sui percorsi CDC tra D1[1:0], D2[1:0] e D1_Latched[1:0], D2_Latched[1:0] vengono quindi classificate come assenza di sincronizzazione nel momento in cui i multiplexer "M1" hanno il selettore impostato al valore logico 1. Quindi, viene segnalata la possibilità di avere una violazione dei tempi di setup/hold dei registri di destinazione D1_Latched[1:0] e D2_Latched[1:0], come riportato in Figura 6-10, ed una propagazione della metastabilità. Anche la violazione CDC tra il FF Change e il FF Change Latched viene classificata come assenza di struttura di sincronizzazione.

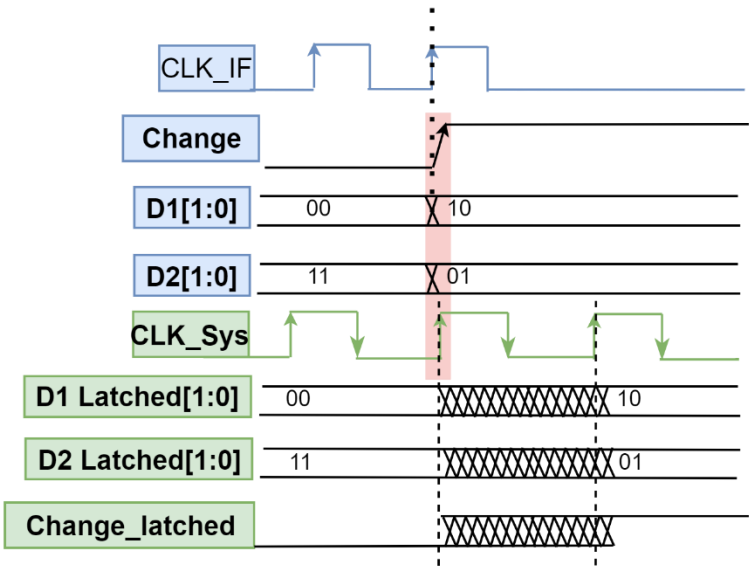

Figura 6-10: Possibile metastabilità

- **Descrizione della violazione CDC:**

Queste violazioni vengono identificate come tali poiché i dati D1[1:0], D2[1:0] e Change sono trasmessi dal dominio di clock di interfaccia "Clk IF", quindi essi possono commutare in qualsiasi momento rispetto al dominio di clock di sistema "Clk Sys" e vi è dunque la possibilità di eventuali violazioni dei tempi di setup/hold dei registri D1_Latched[1:0],D2_Latched[1:0] e FF Change Latched. Questo può ovviamente riflettersi in una propagazione della metastabilità attraverso la logica a valle dei registri in questione. Vi è dunque la possibilità di effettuare un campionamento non corretto dei dati D1[1:0],D2[1:0] e Change quando quest'ultimi commutano in prossimità del fronte attivo del clock di destinazione e portando dunque a generare errori funzionali.

- **Descrizione del protocollo:**

Queste violazioni, a seguito della loro analisi, risultano non essere un reale problema CDC grazie al meccanismo di acquisizione di tali dati presente. Bisogna precisare che l'analisi CDC strutturale effettua solamente il controllo della presenza o meno di schemi di sincronizzazione lungo i percorsi CDC identificati. Dunque, l'analisi strutturale non considera la presenza di protocolli adottati per l'acquisizione dei segnali/dati in questione e questo caso riportato ne è un esempio. Nella violazione in questione viene per l'appunto adottato un protocollo tale da consentire il campionamento dei dati D1[1:0] e D2[1:0] quando essi sono già stabili da diversi colpi di clock di sistema. Viene quindi evitata la possibilità di un'acquisizione durante una loro commutazione, così rimuovendo la possibilità di una violazione dei tempi di setup/hold. Essenzialmente, il dato viene visto come statico al momento del suo campionamento e non comporta alcun problema CDC. Invece, per quanto riguarda il segnale Change, una volta scritto con il clock di interfaccia, esso rimane fisso al valore logico 1 fin quando non resettato da parte della struttura di acquisizione in questione e quindi fin quando non viene confermata la conclusione del meccanismo di acquisizione dei dati. Essenzialmente, vi è la possibilità di avere una propagazione della metastabilità attraverso la logica presente a valle del FF Change Latched, ma al colpo di clock successivo viene garantito il suo corretto campionamento, essendo

quest'ultimo stabile fino al termine della procedura, proseguendo dunque con una corretta esecuzione della procedura di acquisizione, semplicemente traslata di un colpo di clock di latenza rispetto al momento in cui è stato asserito il segnale Change.

Per quanto riguarda i registri D1_Latched[1:0] e D2_Latched[1:0], essi sono inizialmente in configurazione di memoria grazie ai multiplexer "M1" e grazie alla porta And "A1" che guida i selettori di quest'ultimi, impostandoli inizialmente al valore logico "0". Nel momento in cui avviene una scrittura nei registri "D1[1:0]" e "D2[1:0]" non avviene una commutazione dei selettori dei mux "M1", ma avviene la scrittura del FF "Change". Il FF "Change" assieme al FF "update" costituiscono la condizione di avvio per far partire il meccanismo di campionamento dei nuovi dati "D1[1:0]" e "D2[1:0]".

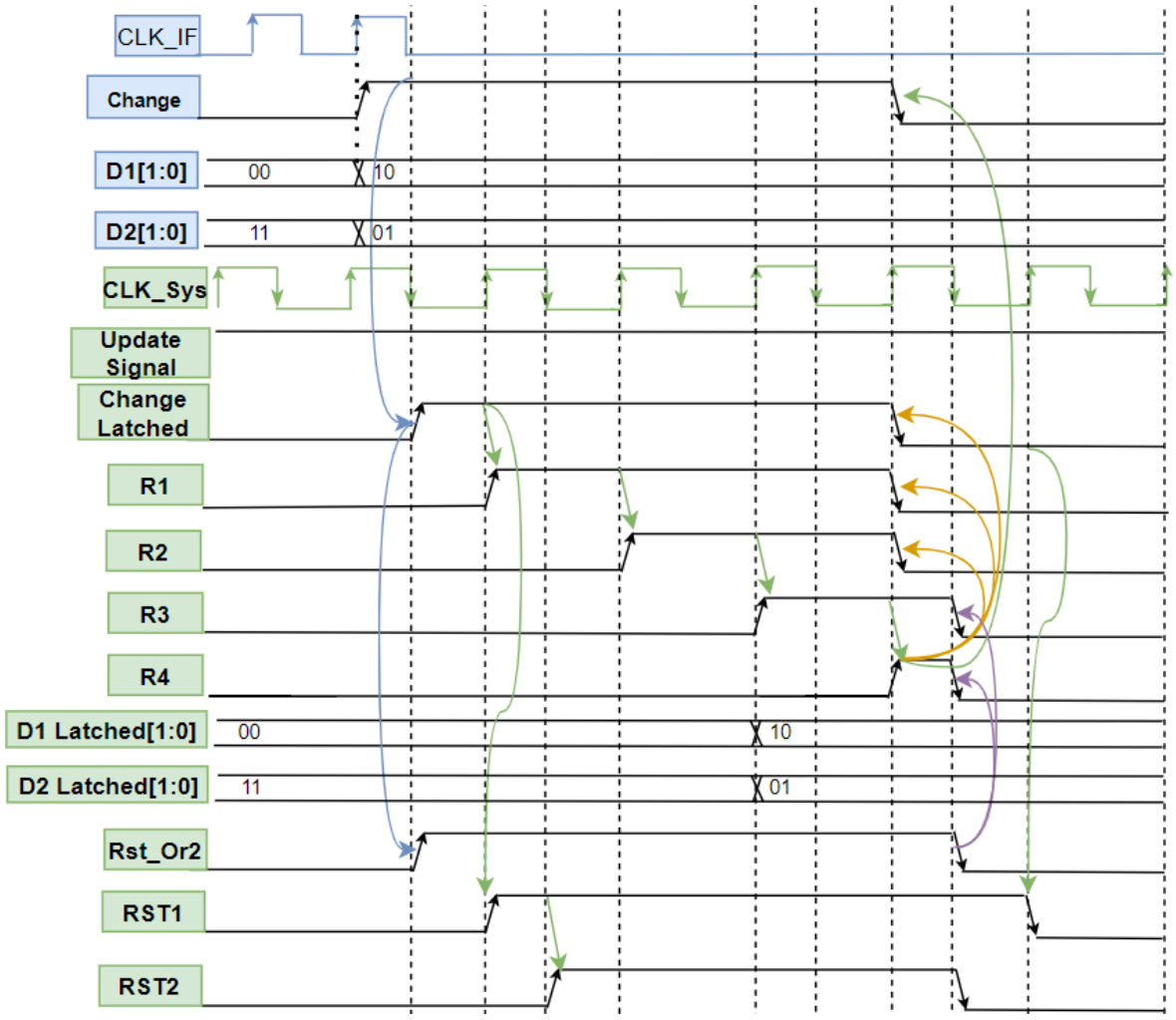

Figura 6-11: Diagramma di timing Dati DSP

Come riportato dal diagramma di timing in Figura 6-11, una volta campionato il Change da parte del FF Change_latched sul fronte di discesa del clock di sistema, il FF R2 viene scritto ad un "1" logico dopo due colpi di clock di sistema e quindi la porta "And1" procede col selezionare l'ingresso 1 del multiplexer, ovvero con il selezionare i nuovi dati "D1[1:0]" e "D2[1:0]". Come è possibile notare, nel momento in cui viene asserito il segnale di Change, avviene anche la commutazione dei dati "D1[1:0]" e "D2[1:0]" ed essi rimangono stabili per tutto il tempo durante il quale il segnale Change è al valore logico "1". Quest'ultimi vengono quindi campionati al colpo di clock di sistema successivo da parte dei registri "D1_latched[1:0]" e "D2_latched[1:0]". I dati "D1[1:0]" e "D2[1:0]" vengono quindi campionati

dopo due colpi di clock di sistema dalla loro ultima commutazione e quindi nel momento in cui essi sono già stabili da tempo. La loro acquisizione avviene in maniera corretta e questo è possibile grazie al selettore dei mux M1 che viene pilotato in maniera sincrona al dominio di destinazione da parte della porta And "A1" gestita dai FF "R2" e "Change_latched". Dunque, dal momento del campionamento del valore logico "1" da parte del FF R2, al colpo di clock successivo si ottiene il campionamento dei nuovi dati da parte dei registri "D1_latched[1:0]" e "D2_latched[1:0]". Tale struttura può essere vista come una sorta di "mux synchronizer". È anche possibile notare che il segnale di Change viene resettato da parte del meccanismo di acquisizione nel momento in cui avviene la commutazione all' "1" logico dell'uscita del FF R4, garantendo quindi di non avere la possibilità che quest'ultimo non venga "visto" da parte del dominio di destinazione. Questo garantisce anche che il segnale di Change viene resettato esclusivamente nel momento in cui la procedura è terminata, segnalando quindi la corretta conclusione dell'acquisizione del nuovo dato e rendendo a tutti gli effetti tale meccanismo un protocollo di acquisizione.

- **Conclusioni: come procedere con il validare il waiver della violazione CDC**

Affinché il completo meccanismo appena spiegato funzioni in maniera corretta, è necessario che il FF Change rimanga asserito per tutta la procedura. Infatti, tale protocollo di acquisizione procede con il reset del FF Change solamente quando tutto il meccanismo è terminato. Ovviamente, altra condizione fondamentale tale per cui l'intero meccanismo venga eseguito in maniera corretta, è che i dati "D1[1:0]" e "D2[1:0]" non devono effettuare ulteriori commutazioni dal loro primo cambiamento durante la procedura di acquisizione, al fine di non vanificare il protocollo adottato. Quindi, queste caratteristiche appena riportate rappresentano la prima condizione che deve essere validata, ovvero che ad ogni colpo di clock di sistema, i dati "D1[1:0]" e "D2[1:0]" devono rimanere stabili fino al corretto campionamento da parte dei registri "D1[1:0]_Latched" e "D2[1:0]_Latched" e deve essere verificata che l'acquisizione avvenga al terzo colpo di clock di sistema da quando è stato asserito il segnale di Change Latched.

- L'intera procedura, come anche le sue condizioni, deve essere validata utilizzando la verifica formale al fine di confermare che tale violazione CDC non sia realmente un reale problema. Il flusso di validazione del protocollo di gestione introdotto viene spiegato nel capitolo 10: Flusso di verifica formale.

6.1.4 Violazione CDC – Reset Configurations

Viene adesso riportata e descritta la violazione CDC identificata riguardante il protocollo di reset dei registri di configurazione del design, definito “Reset configurations”. Tale violazione è risultata non essere un reale problema CDC grazie al protocollo presente.

Il protocollo dichiara che con la scrittura di un registro di controllo specifico “RST_CFG”, attraverso il clock di interfaccia “Clock_IF”, avviene il reset dei registri di configurazione del sistema. Lo schema della violazione in questione è riportato nella Figura 6-12, in versione semplificata al fine di spiegare la violazione incontrata e giustificare il motivo per il quale risulta non essere un problema CDC.

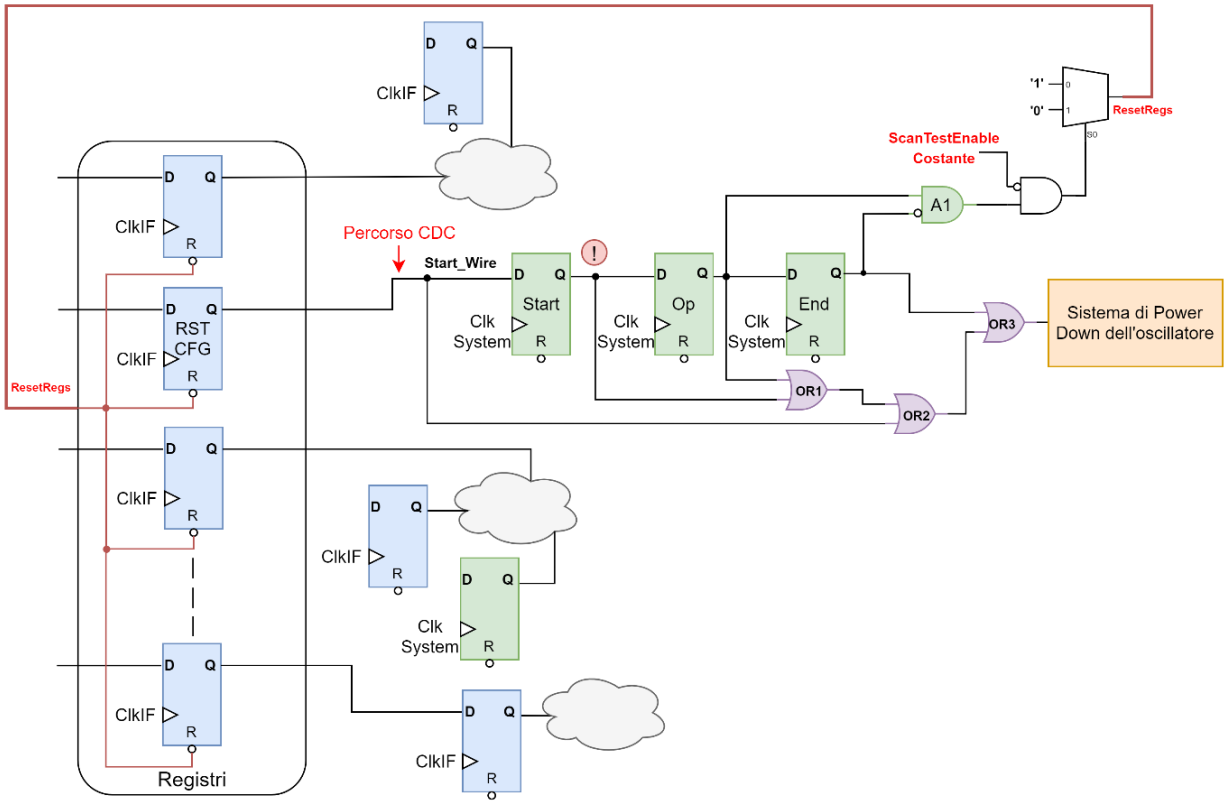

Figura 6-12: Struttura del sistema di reset dei registri di configurazione

- **Violazione CDC identificata:**

Durante l’analisi viene identificata l’assenza di uno schema di sincronizzazione nel percorso CDC che unisce il FF “RST_CFG”, il quale effettua un passaggio dal dominio di clock di interfaccia “ClkIF”, identificato come dominio di trasmissione, al FF “Start”, appartenente al dominio di clock di sistema “Clk System” e identificato come dominio di destinazione. Tale violazione viene classificata come caso di “percorso CDC non sincronizzato” ed il motivo è legato alla presenza di una logica combinatoria collegata all’interno della struttura di sincronizzazione. Questa tipologia di violazione porta quindi a segnalare la possibilità di avere continue violazioni dei tempi di setup e di hold e dunque ad una generazione e propagazione di metastabilità attraverso il design.

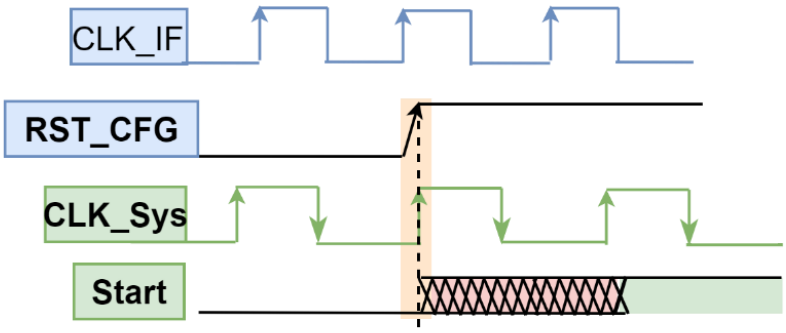

Figura 6-13: Possibile metastabilità

- **Descrizione della violazione CDC:**

La procedura “reset configurations” viene innescata effettuando la scrittura di un “1” logico nel FF “RST_CFG”. Tale FF viene scritto con il clock di interfaccia e può quindi portare ad una possibile violazione dei tempi di setup/hold del FF “Start”, appartenente al dominio di clock di sistema. In prima analisi, potrebbe sembrare errata la violazione riportata, questo perché sono presenti tre FF in cascata appartenenti al dominio di clock di sistema e quindi questi possono rappresentare il più comune schema di sincronizzazione multi-Flop. La violazione è però valida perché viene collegata all’uscita del FF “Start” della logica combinatoria, più precisamente la porta “OR1”. Il collegamento di una logica combinatoria all’uscita del FF costituente il primo elemento della catena di sincronizzazione è sconsigliato, in quanto è possibile avere una propagazione della metastabilità attraverso tale logica combinatoria. Dopo il secondo FF “Op” si può considerare già terminata la catena di sincronizzazione e dunque il problema CDC rimane isolato al primo FF. Dunque, tale configurazione si riflette in un utilizzo non corretto del sincronizzatore impiegato.

- **Descrizione del protocollo:**

Grazie al protocollo adottato in tale struttura, tale violazione risulta non essere un reale problema CDC.

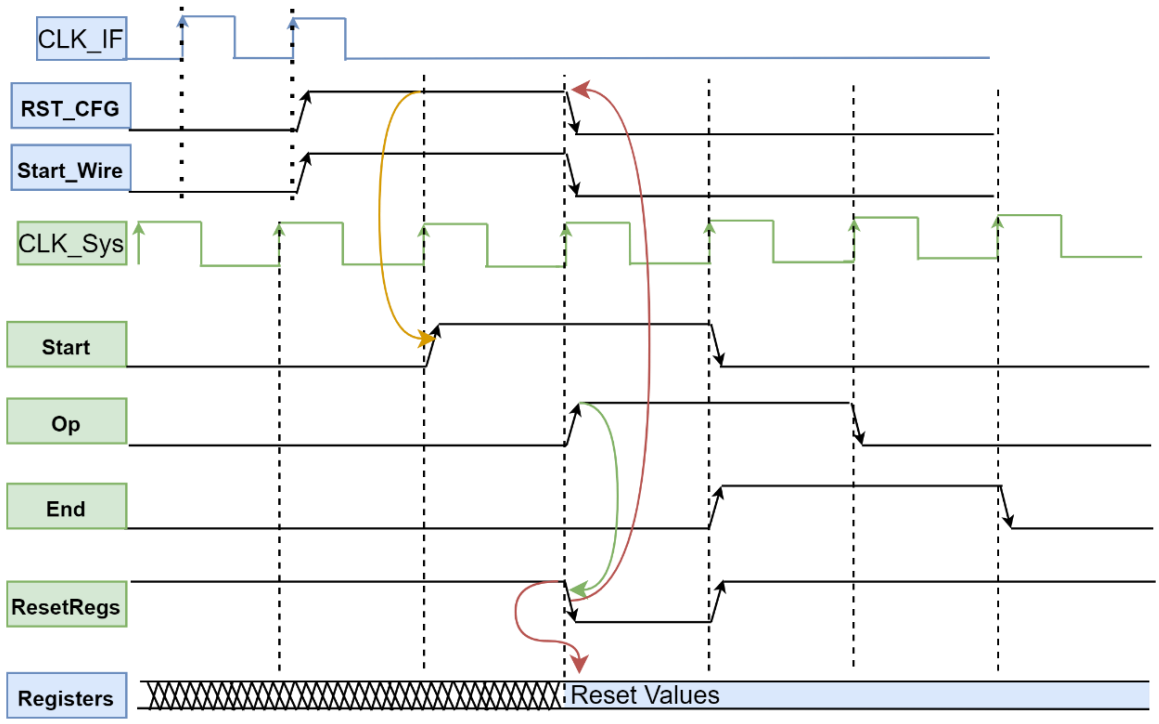

Figura 6-14: Diagramma di timing Reset Configurations

Come è possibile notare dal diagramma di timing riportato nella Figura 6-14, una volta effettuata la scrittura al valore logico “1” del FF “RST_CFG” attraverso il clock di interfaccia, quest’ultimo viene interrotto e si procede quindi con l’aspettare il campionamento da parte del FF “Start” appartenente al dominio di destinazione ed il bit “RST_CFG” rimane asserito al valore logico “1” per tutta la procedura di reset, rimanendo in attesa di essere “auto-resettato”. È possibile notare, dallo schematico in Figura 6-12, che il secondo ingresso della porta “OR2” è direttamente collegato all’uscita del FF “RST_CFG”, il quale sta trasmettendo il valore logico “1” e dunque sta procedendo con l’assicurare che non venga innescata la procedura di “Power Down” dell’oscillatore durante l’operazione del “reset configurations”, dunque si garantisce che il clock di sistema non venga spento e si garantisce di non avere errori funzionali dovuti ad una possibile propagazione della metastabilità dal FF “Start”. Una volta avvenuto il campionamento da parte di “Start”, al colpo di clock successivo vi è il campionamento da parte del FF “OP” e viene dunque innescata la procedura di reset dei registri di configurazione presenti nel design. Questo perché il selettore del multiplexer viene gestito dalla porta And presente nello schema ed avente come ingressi il negato dell’uscita del FF “End” e l’uscita del FF “OP”. Dunque, l’uscita del multiplexer “ResetRegs” viene portata al valore logico “0” per un intero colpo di clock di sistema. I registri di configurazione nominati “Registers” procedono dunque con il commutare ai loro valori di reset ed è anche possibile notare che il reset viene asserito anche per il FF “RST_CFG”, il quale viene quindi riportato al suo valore logico di reset “0” e che dunque segna anche il termine della procedura di reset.

- **Conclusioni: come procedere con il validare il waiver della violazione CDC**

Al fine di dimostrare che tale violazione non risulta essere un problema CDC, è necessario validare la procedura appena riportata con l'ausilio della verifica formale e validare il seguente punto:

  - il bit "RST_CFG" deve rimanere ad "1" per almeno due colpi di clock di sistema al fine di garantire il campionamento da parte del FF "Start" ed una corretta esecuzione della procedura di "reset configurations". Quindi deve essere assicurata una stabilità del dato, ed avendo una validazione di tale stabilità si assicura anche che l'uscita della porta logica "OR2" rimane stabile al valore logico "1" per tutto il tempo necessario per poter effettuare l'acquisizione del segnale "RST_CFG".

6.1.5 Problema riscontrato – Clock Gating

Viene adesso preso in considerazione un problema CDC riscontrato durante l'analisi del design che viene definito "Clock Gating" ed è ovviamente estrapolato dai risultati ottenuti dall'analisi CDC. È necessario sottolineare che questa non è una violazione CDC presente nel design ma è un errore di analisi da parte del tool riscontrato durante l'analisi CDC del design. Essenzialmente, una serie di violazioni CDC riportate durante l'analisi strutturale del design sono risultate non essere reali problemi CDC per via di un errata interpretazione delle strutture di clock gating presenti all'interno del design da parte del tool di analisi. Al fine di fornire una migliore visione del problema riscontrato, viene riportato uno schema rappresentativo e semplificato, nella Figura 6-15, delle strutture su cui è stato identificato tale errore.

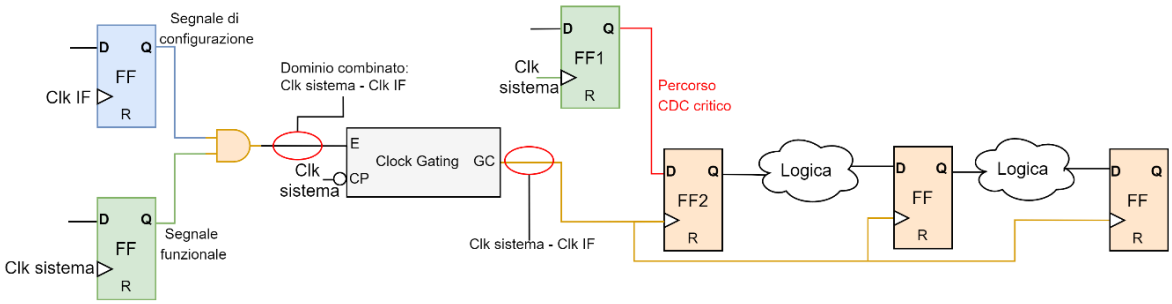

Figura 6-15: Clock Gating

Nello schematico in Figura 6-15 è presente una struttura di "clock gating", la quale consente di pilotare il clock di sistema attraverso un segnale di enable "E". Tale segnale di enable è definito da una porta logica "And" i cui ingressi sono caratterizzati da un segnale di configurazione, generato dal clock di interfaccia, ed un segnale funzionale, generato dal clock di sistema. Il risultato generato da tale porta è quindi un segnale di enable ottenuto dalla combinazione dei due domini di clock dei segnali in ingresso e questo quindi si riflette nel dar vita ad un segnale di clock combinato all'uscita del clock gating. Il risultato che ne segue è che i registri i cui clock sono pilotati da tale clock gating appartengono ad un dominio di clock combinato che può essere definito "Clock sistema- Clock IF".

- **Descrizione del problema riscontrato:**

Come è possibile intuire quindi dallo schematico riportato, nella Figura 6-15, è presente un percorso CDC la cui sorgente è un FF "FF1", appartenente al dominio di clock di sistema, e la cui destinazione è il FF "FF2" appartenente al dominio di clock combinato appena introdotto. La violazione viene riportata in quanto è assente una struttura di sincronizzazione in tale percorso CDC e dunque non è possibile garantire che non vi siano violazioni dei tempi di setup/hold del FF di destinazione "FF2" appartenente al dominio di clock combinato e quindi vi è la possibilità dell'insorgere della metastabilità che può propagarsi lungo la logica combinatoria a valle di tale FF. Ovviamente, tale propagazione attraverso la logica combinatoria può causare errori funzionali che possono quindi propagarsi attraverso il design.

- **Procedura di risoluzione attuata:**

Viene quindi effettuata un'analisi sul segnale di configurazione generato dal dominio di clock di interfaccia posto all'ingresso della porta And. Il primo step consiste nell'analizzare il dominio di clock ottenuto all'uscita del clock gating impostando come costante il comportamento del segnale di configurazione appartenente al clock di interfaccia presente all'ingresso della porta And:

1. Viene quindi impostato il valore logico "1" e definito come "costante". Dunque, non può essere effettuata alcuna commutazione durante il funzionamento del dispositivo. Tale configurazione si riflette nel non avere la generazione del dominio di clock combinato "clock sistema-clock interfaccia" ma nell'ottenere un segnale di clock da parte del clock gating appartenente al dominio di clock di sistema.
2. Nel momento in cui viene posto il valore logico "0" e definito come costante tale segnale di configurazione, non è più presente una violazione CDC non essendoci alcun percorso CDC tra i FF "FF1" e "FF2", in quanto quest'ultimo è adesso appartenente al dominio di clock di sistema. Ovviamente, nel momento in cui viene impostato il valore logico "0" e definito come costante, l'enable del clock gating è sempre disabilitato e dunque non viene fornito alcun segnale di clock da parte di quest'ultimo e come è possibile intuire, anche questa configurazione si riflette nel non avere alcuna violazione CDC.

Viene quindi confermato che il segnale in questione non comporta alcuna generazione di un dominio di clock combinato nel momento in cui il suo valore logico è costante o statico durante il funzionamento del dispositivo.

- **Controllo del comportamento assumibile da parte del segnale**: Ovviamente, non è possibile procedere con l'impostare il segnale di configurazione ad uno dei valori logici assumibili a priori, ma bisogna controllare dalle specifiche del design quale è il suo comportamento durante la fase funzionale di quest'ultimo. Dalle specifiche del design viene confermato che tale segnale, una volta configurato, non può effettuare ulteriori commutazioni durante la funzionalità del dispositivo. Quindi è possibile intuire che tale segnale può essere impostato come "statico", ovvero non viene effettuata alcuna considerazione sul suo valore logico ma si conferma che esso, una volta configurato, non effettua ulteriori commutazioni. Il suo valore logico non può essere però scelto a priori in quanto esso discrimina alcune delle funzionalità del dispositivo

- **Configurazione attuabile:** Grazie quindi all'analisi precedente in cui si è verificato il comportamento ottenuto dai due possibili valori logici e alla possibilità di impostare il suo comportamento come statico, senza incorrere in violazioni delle specifiche del design, si può procedere con il confermare che il segnale di clock generato da parte della struttura di clock gating è sempre appartenente al dominio di clock di sistema. Nella Figura 6-16 è possibile visualizzare il risultato ottenuto impostando il segnale di configurazione come "Statico".

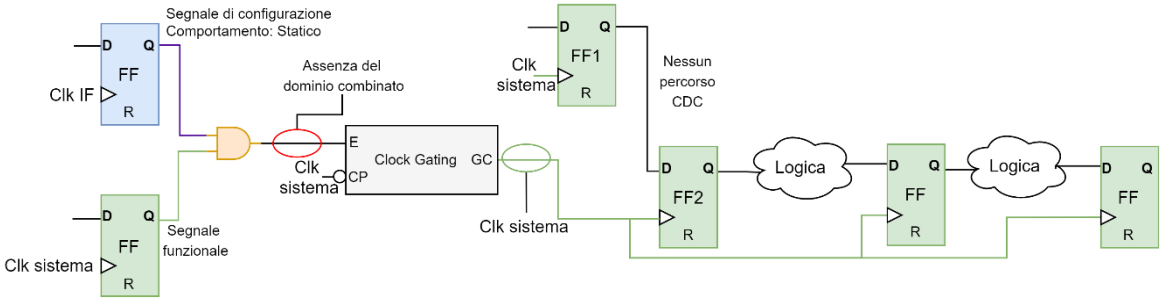

Figura 6-16: Segnale di configurazione Statico, nessun percorso CDC identificato

Il problema è quindi risolto effettuando una corretta configurazione dei segnali coinvolti, in questo caso solamente uno.

6.2 Analisi CDC di Convergenza

Successivamente allo step di "Identificazione degli schemi di sincronizzazione e violazioni CDC" si procede con l'analizzare i risultati ottenuti dallo step della fase CDC definito "Analisi CDC di Convergenza".

6.2.1 Violazione CDC identificata-PowerDown Memoria Volatile – Possibile insorgenza di glitch

Viene adesso riportato un esempio di violazione CDC identificata nel design in analisi in questo step della verifica strutturale. Viene preso in considerazione il seguente caso, semplice, ma al tempo stesso importante perché coinvolge il sistema di "Power Down" della memoria volatile presente nel sistema, ovvero il sistema di spegnimento ed isolamento della memoria. Gli ulteriori casi identificati sono caratterizzati dalla presenza di una logica combinatoria complessa e dunque viene selezionata la seguente struttura per poter illustrare la tipologia delle violazioni che vengono analizzate in tale step di analisi CDC e la loro possibile criticità all'interno del design.

- **Violazione CDC identificata:**

Viene rappresentato lo schema della violazione identificata nella Figura 6-17. È possibile notare la presenza di un sincronizzatore standard nel dominio di ricezione e dunque è possibile intendere che la violazione riportata non è dovuta ad un'assenza di una struttura di sincronizzazione ma è riferita alla presenza di logica combinatoria nel percorso CDC in questione. Tale logica combinatoria può portare alla generazione di un glitch, dove quest'ultimo non è filtrato in alcun modo e può quindi essere campionato nel dominio di destinazione. Essenzialmente, questa tipologia di violazione CDC è stata introdotta nella sezione 4.2.3.2 nell'introduzione alle violazioni tipiche CDC nel capitolo 2.1.

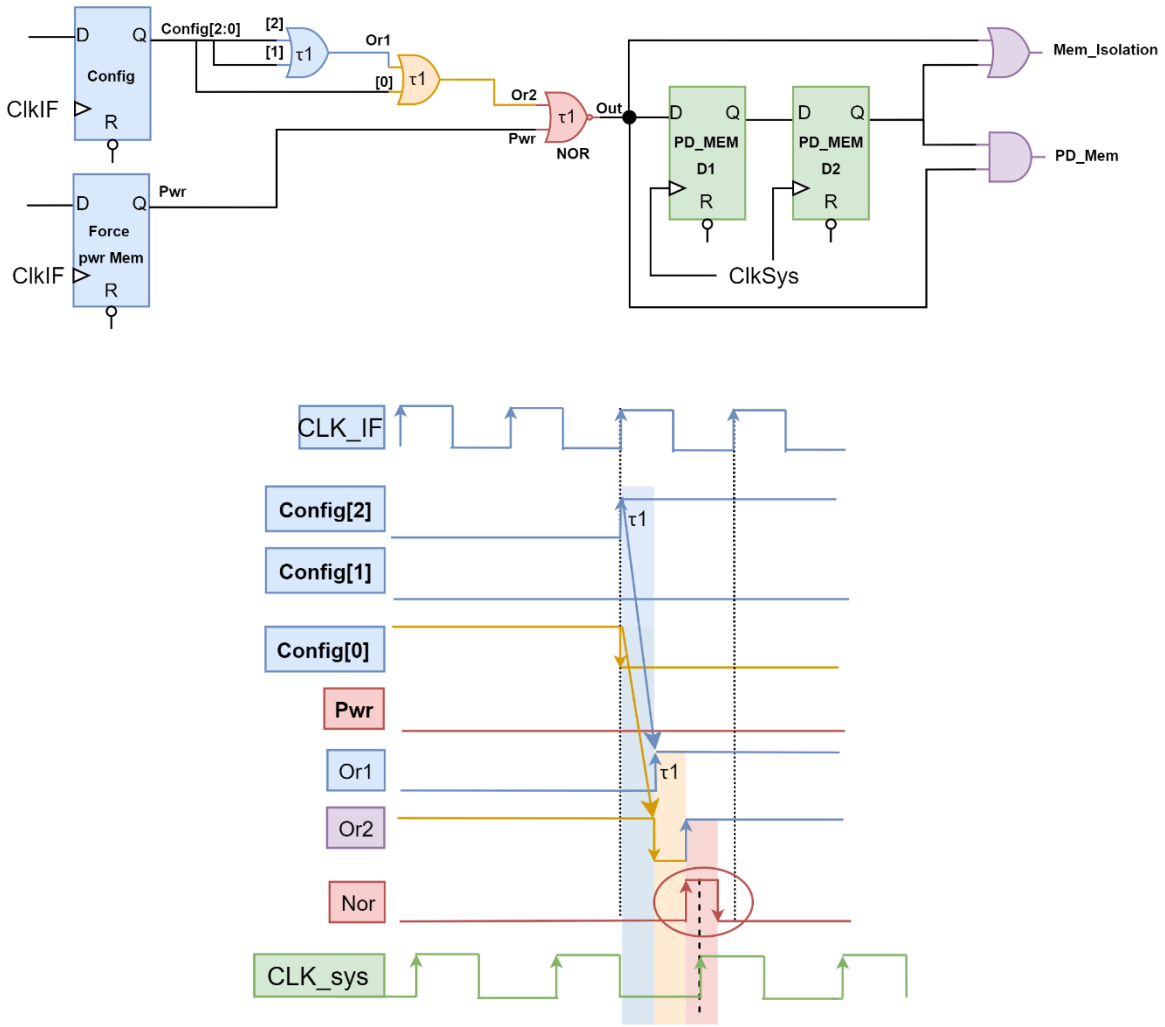

Figura 6-17: Struttura del caso di convergenza identificato

- **Descrizione della violazione CDC:**

Il problema è essenzialmente dovuto alla catena di porte OR e la porta NOR presente nel cammino CDC, il cui dominio di trasmissione è il dominio di clock di interfaccia e il dominio di destinazione è il dominio di clock di sistema. È presente un sincronizzatore standard a due FF ma, nonostante ciò, possono insorgere problemi funzionali CDC. Nel caso specifico vi è la possibilità di una propagazione di glitch nel dominio di destinazione. In tale violazione è coinvolto l'isolamento della memoria volatile, dunque il campionamento di un glitch può portare a malfunzionamenti durante l'attività del sistema. Il problema principale è dettato dai tre bit "Config[2:0]", appartenenti al dominio di clock di interfaccia e costituenti i bit di configurazione della memoria. Essi sono i segnali che permettono di selezionare una delle sue diverse modalità di funzionamento e consentono anche il suo isolamento. Come è possibile notare dal diagramma di timing riportato nella Figura 6-17, nel momento in cui avviene una commutazione contemporanea di più bit del registro "Config[2:0]" , vi è la possibilità di dar vita ad un impulso spurio che può essere campionato dalla catena di sincronizzazione del dominio di destinazione.

È stato preso in considerazione un ritardo di propagazione "τ1" uguale per tutte le porte logiche in questione, questo per poter mostrare come è possibile avere la generazione di un glitch. Si prende in considerazione un caso di partenza in cui i segnali di Config[2:0] sono rispettivamente ai valori logici "001". Successivamente, avviene un cambio di configurazione che porta tali segnali rispettivamente ai valori logici "100". Dunque, avviene una commutazione contemporanea dei segnali "Config[2]" e "Config[0]" ed ovviamente il risultato che deve essere ottenuto all'uscita della porta NOR deve essere sempre uno "0" logico. Essendo però presente un percorso non bilanciato tra i due segnali di Config in questione, vi è la possibilità di ottenere un impulso spurio all'uscita della porta NOR. La porta Or2 ha come ingresso il segnale diretto config[0] al valore logico "0" mentre il secondo ingresso è proveniente dalla porta logica Or1 , la quale non è ancora commutata all' "1" logico. Dunque, dopo un tempo τ1 il segnale di uscita Or2 commuta erroneamente al valore logico "0" per poi ritornare al valore logico "1" dopo un altro tempo τ1, in quanto il segnale Or1 è commutato al valore logico "1". Ovviamente, per via dell'impulso spurio generato dalla porta logico Or2, dopo un tempo τ1 da esso, si ottiene anche un impulso spurio da parte della porta NOR. Quest'ultimo impulso spurio può essere campionato da parte del sincronizzatore presente nel dominio di destinazione.

Dunque, la generazione di un glitch è possibile nel momento in cui avviene un cambio di configurazione della modalità della memoria durante il funzionamento del design ed è principalmente dovuto alla possibilità di avere diversi ritardi combinatori e diversi ritardi di propagazione nei rami che coinvolgono la violazione CDC. Il campionamento di tale glitch porta all'isolamento inaspettato della memoria volatile, la quale procede quindi con il rimanere isolata per un intero colpo di clock di sistema e che dunque comporta ad avere un conseguente malfunzionamento del dispositivo.

- **Descrizione della soluzione adottata per tale violazione CDC:**

Una soluzione per poter evitare la generazione e propagazione di un eventuale glitch è quella di effettuare un cambio della modalità di funzionamento della memoria passando attraverso l'isolamento controllato di quest'ultima. Il passaggio in questa modalità avviene effettuando una scrittura di tutti zeri nel registro "Config[2:0]". Essenzialmente, effettuando tale scrittura dei registri "Config[2:0]" si elimina la possibilità dell'insorgenza di un glitch. Quindi, si elimina la possibilità di ottenere una commutazione errata da parte della porta NOR al valore logico "0" quando invece il risultato aspettato è sempre "1", ovviamente nel momento in cui si è in una modalità di funzionamento differente da quella dell'isolamento. Dunque, effettuando ogni cambio di modalità della memoria passando attraverso il suo isolamento controllato, si elimina la possibilità di avere un isolamento non

atteso dovuto alla generazione di un impulso spurio. Viene riportato nella Figura 6-18 un esempio di diagramma di timing in cui avviene un cambio di modalità effettuando il passaggio attraverso la scrittura di tutti “0“ nel registro Config[2:0].

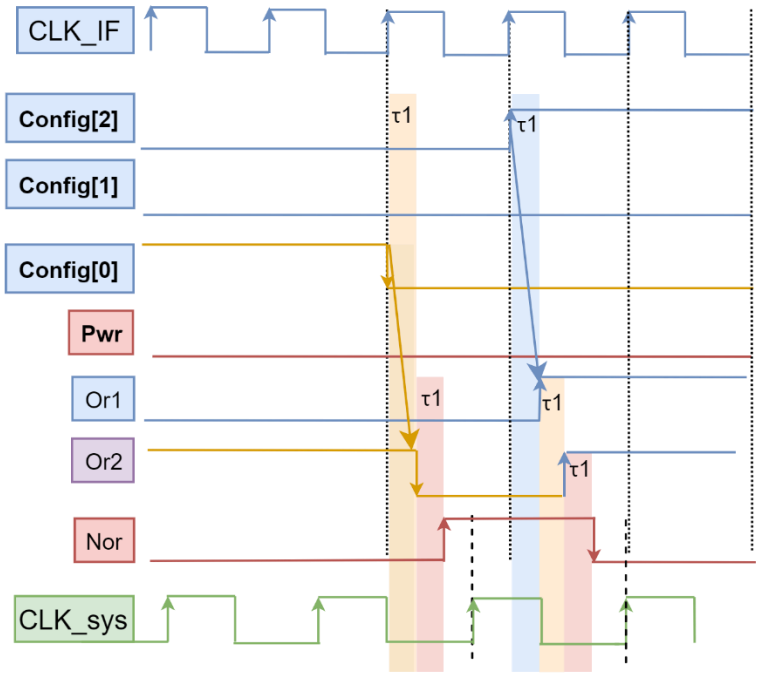

Figura 6-18: Diagramma di timing rappresentativo della soluzione da attuare

Nel momento in cui non fosse possibile effettuare un passaggio dalla modalità di isolamento della memoria, l’ulteriore soluzione attuabile consiste nell’effettuare una prima scrittura del registro di “Force pwr Memory” al valore logico uno, il quale porta a mantenere accesa e non isolata la memoria fin quando questo bit è settato ad uno. In questo modo, la porta NOR non effettua ulteriori commutazioni alla variazione dei bit di controllo del registro Config[2:0], non consentendo la propagazione di eventuali glitches. Quindi, la possibilità è quella di effettuare la scrittura del bit “Force pwr Memory”, effettuare la scrittura per poter cambiare la modalità di funzionamento della memoria e successivamente riscrivere al valore logico “0” il bit di “Force pwr Memory”. In questo modo è possibile non avere la propagazione di impulsi spuri e non effettuare il passaggio dalla modalità di isolamento della memoria.

7 Flusso di analisi RDC -Reset Domain Crossing

7.1 Analisi RDC – Reset Domain Crossing

Nell'aumento della complessità dei moderni SOC non vi è solo l'incremento del numero dei domini di clock asincroni ma si affianca anche l'aumento del numero di domini di reset asincroni. Per questo motivo che all'analisi CDC viene associata l'analisi RDC "Reset-Domain-Crossing". L'analisi CDC, come approfondito nei capitoli precedenti, è principalmente legata all'identificazione di percorsi di segnali e dati tra domini di clock asincroni in cui vi è il rischio di metastabilità o glitches. Invece, come è possibile intendere dal suo nome, l'analisi RDC ha come obiettivo quello di identificare e analizzare i cammini dei segnali di reset presenti nel design che possono "potenzialmente" portare alla nascita della metastabilità e della sua relativa propagazione attraverso il design. Anche in questa analisi viene eseguita l'identificazione della presenza o assenza di schemi di sincronizzazione, ma in questo contesto vengono definiti come "Sincronizzatori di reset". A differenza dell'analisi CDC, in cui si prende in considerazione come unico scenario il trasferimento di segnali e dati attraverso diversi domini di clock asincroni, nell'analisi RDC viene preso in considerazione un primo scenario in cui è presente il trasferimento di segnali di reset tra FF appartenenti allo stesso dominio di clock ed un secondo scenario in cui vi è il trasferimento di reset tra domini di clock asincroni.

Quindi, come nell'analisi CDC, vengono identificati i diversi domini di resets presenti nel design e vengono analizzati i percorsi RDC in cui "potenzialmente" possono essere presenti errori funzionali.

7.1.1 Asserzione e De-Asserzione asincrona del segnale di reset

Molto spesso ci si limita al porre l'attenzione solamente all'asserzione dei segnali di reset, ma in realtà bisogna anche identificare i problemi che possono sorgere dalla loro de-asserzione asincrona.

La de-asserzione asincrona di un segnale di reset può portare all'insorgere di metastabilità. Prendendo quindi in considerazione un segnale di reset asincrono, quindi un Reset Domain crossing, le condizioni che portano ad avere la metastabilità sono le seguenti:

- **Condizioni per avere Metastabilità:**

  1) Il dato presente all'ingresso del FlipFlop è differente dal valore assunto nello stato di reset e dunque può sorgere un problema al momento della de-asserzione del segnale di reset;
  2) Il clock del FlipFlop che sta venendo resettato è attivo;
  3) La de-asserzione del segnale di reset è asincrona e può portare alla violazione dei tempi di recovery/removal;

Quindi, la metastabilità può insorgere con queste condizioni nel momento in cui non vengono rispettati i tempi di "recovery" e "removal", le controparti dei segnali di reset dei tempi di setup e di hold. Nella Figura 7-1 vengono riportati i tempi appena citati, dove è possibile notare che per "recovery" si intende il tempo minimo che deve intercorrere dall'ultima variazione del segnale di reset e il fronte attivo del clock, ecco perché definita controparte del tempo di setup, mentre per il tempo di "removal" si intende il minimo intervallo di tempo per cui il segnale di reset deve rimanere stabile

dal fronte attivo del clock ed ecco perché controparte del tempo di hold. Una violazione dei tempi di removal/recovery si riflette nella nascita di metastabilità proprio come una violazione dei tempi di setup/hold. Nella Figura 7-2 è riportato un esempio di violazione dei tempi.

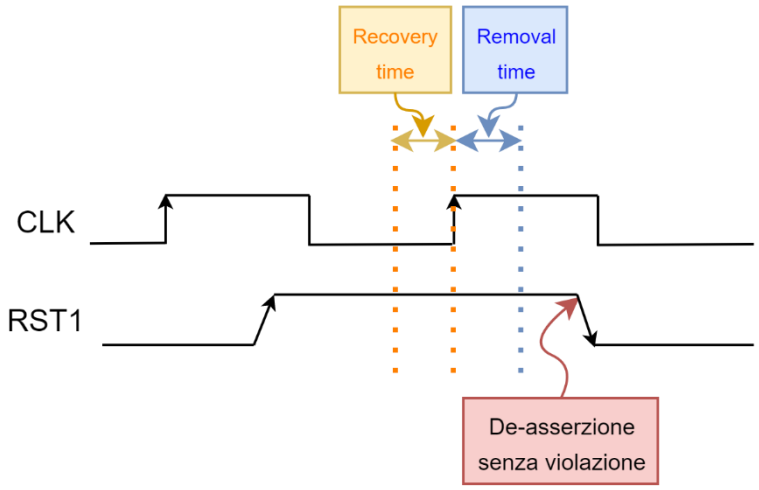

Figura 7-1: Tempi di Recovey e Removal rispettati, nessuna possibilità di metastabilità

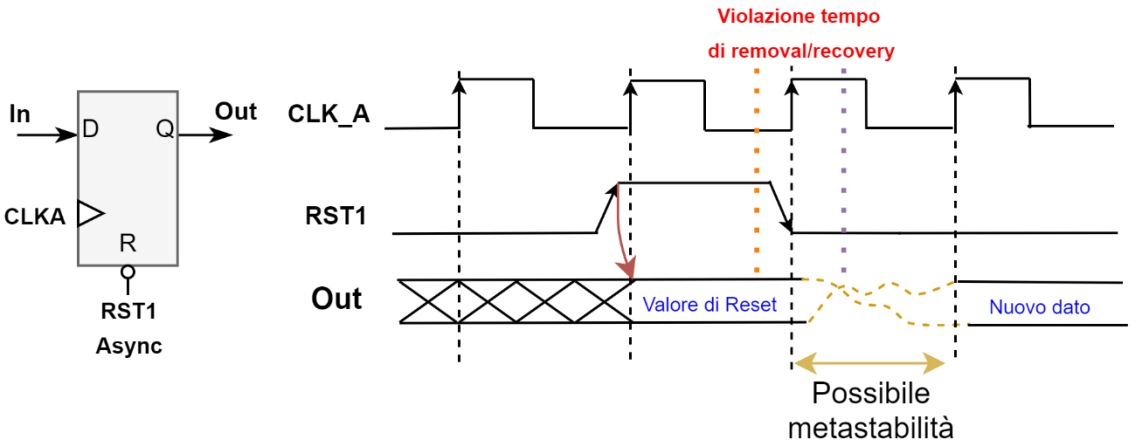

Figura 7-2: Metastabilità dovuta alla violazione dei tempi di removal e recovery

Quindi, in presenza di tali condizioni che possono portare all'insorgenza della metastabilità bisogna attuare delle soluzioni per poter escludere la possibilità di avere una sua propagazione.

- **Soluzioni attuabili:**

1) Assicurarsi che il clock del FlipFlop che sta venendo resettato sia inibito, più precisamente che sia inibito in prossimità dell'evento di de-asserzione del segnale di reset;
2) Imporre che il valore del dato in ingresso al FlipFlop sia uguale al valore assunto durante lo stato di reset;
3) Implementare strutture di sincronizzazione che hanno lo scopo di sincronizzare l'asserzione e de-asserzione del reset oppure solamente la sua de-asserzione;

Ecco quindi spiegata l'esigenza di dover implementare schemi di sincronizzazione dei segnali di reset all'interno di design asincroni.

7.1.2 Sincronizzatori di reset

Esistono quindi anche delle soluzioni comuni di strutture di sincronizzazione per poter gestire i segnali di reset al fine di non incorrere nei problemi di metastabilità. Il più comune sincronizzatore di reset è riportato a sinistra nella Figura 7-3 , dove è possibile notare che il segnale di reset asincrono guida direttamente i pin di reset del sincronizzatore, così portando ad avere un'asserzione asincrona del reset del FF "Target" che quindi commuta al suo valore di reset. Essendo il sincronizzatore appartenente allo stesso dominio di clock del FF "Target", nel momento in cui avviene la de-asserzione del segnale "Reset", il segnale "Reset_S" viene de-asserito dopo due colpi di clock "CLKA" in maniera sincrona al clock di appartenenza e quindi assicurando di rispettare i tempi di removal/recovery. Ovviamente, deve essere considerata la latenza che viene introdotta con questo tipo di schema di sincronizzazione, in quanto il FF "Target" esce dallo stato di reset dopo due colpi di clock addizionali dal momento della de-asserzione del segnale "Reset". Nella Figura 7-3 a destra, viene riportato un esempio di schema di sincronizzatore "Diretto", in questo caso il segnale "Reset" entra direttamente nell'ingresso D del primo FF e tale struttura garantisce di avere sia l'asserzione che la de-asserzione del segnale "Reset_S" sincrona al clock di appartenenza del FF "Target".

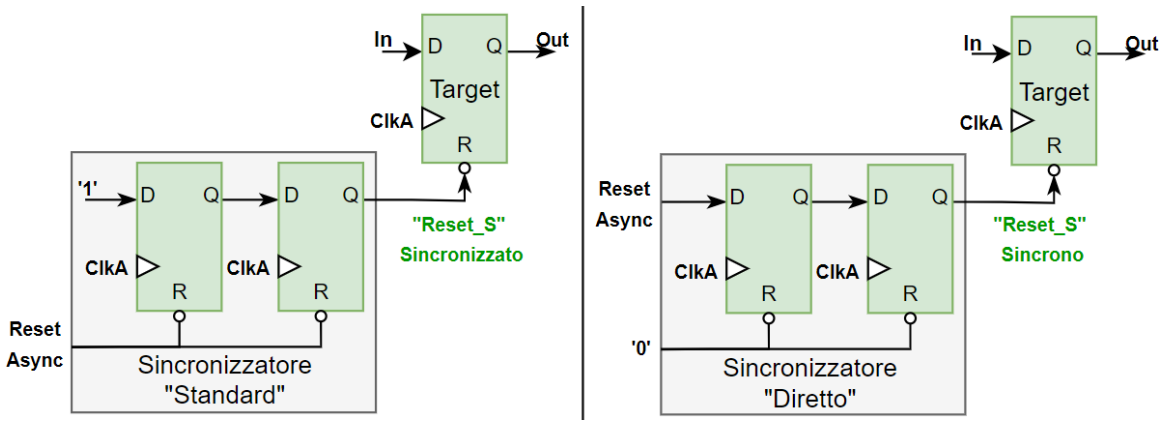

Figura 7-3: Schema comune di sincronizzazione del reset a sinistra, a destra sincronizzatore diretto

7.1.3 RDC tra FF appartenenti allo stesso dominio di clock

Viene riportata nella Figura 7-4 un esempio di possibile violazione RDC dovuta ad una possibile insorgenza di metastabilità nonostante i FF in questione siano appartenenti allo stesso dominio di clock. I reset presenti sono differenti ed asincroni e sono identificati come RST1 e RST2.

In questo caso, le condizioni che consentono l'insorgere della metastabilità sono le seguenti:

- **Condizioni**

  1) Il segnale di reset RST2 rimane de-asserito;
  2) Il segnale di RST1 viene asserito in maniera asincrona al dominio di clock a cui appartengono i FF coinvolti nel percorso e dunque è possibile una commutazione del dato, che ritorna al suo valore di reset, in prossimità del fronte attivo del clock del FF di destinazione;

È possibile notare che nel momento in cui viene asserito in maniera asincrona il reset RST1, il dato "Data_s" viene resettato in maniera asincrona al valore di reset dopo un tempo di propagazione "reset to output". L'asserzione del RST1 in prossimità del fronte attivo del clock si riflette in una violazione del tempo di setup/hold del FF ricevitore, comportando quindi ad una propagazione della metastabilità nella logica a valle. Nasce quindi l'esigenza di porre attenzione alla gestione dei reset all'interno dei design e diventa fondamentale l'esecuzione dell'analisi RDC. Un design strutturato correttamente e verificato positivamente dal punto di vista dell'analisi CDC non può essere considerato esente da problemi se non anche verificato dal punto di vista RDC.

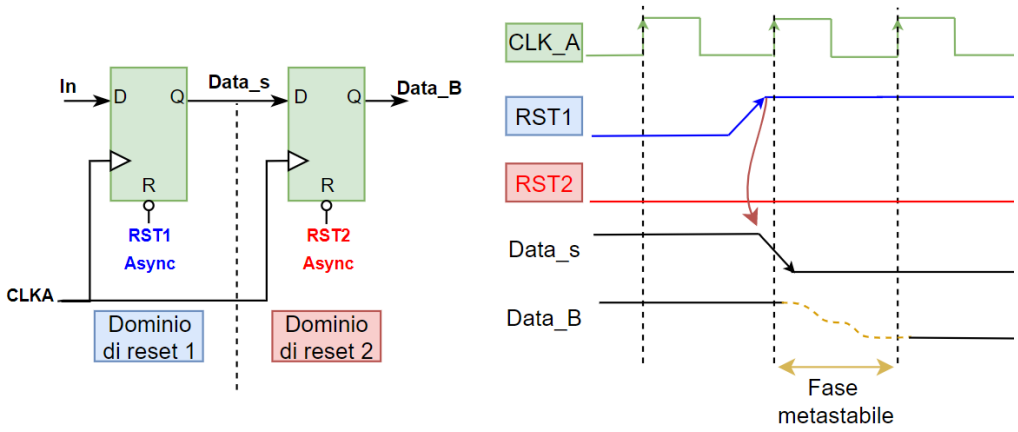

Figura 7-4: Caso RDC semplice in cui è presente lo stesso dominio di clock

É possibile capire dall'esempio riportato che la metastabilità può insorgere nel momento in cui il reset asincrono del FF sorgente è differente dal reset asincrono del FF di destinazione, anche se i FF in questione sono appartenenti allo stesso dominio di clock.

- **Soluzioni Possibili:**

  - Una possibile soluzione per poter gestire il caso RDC appena riportato è la definizione dell'ordine di asserzione dei reset presenti all'interno del design. Per poter evitare l'insorgenza della metastabilità nella Figura 7-4 è sufficiente asserire per primo il RST2 e per secondo il RST1, così da evitare la possibilità di una violazione dei tempi di setup/hold nel FF ricevitore. Viene rappresentata nella Figura 7-5 come dovrebbe

essere gestito l'ordine dei reset dei FF per poter evitare il problema della metastabilità. Ovviamente, questa tipo di soluzione deve essere adottata nel momento in cui non è possibile adottare lo stesso segnale di reset per entrambi i FF. Se entrambi i segnali di reset vengono asseriti nello stesso momento si riesce comunque ad evitare il problema della metastabilità.

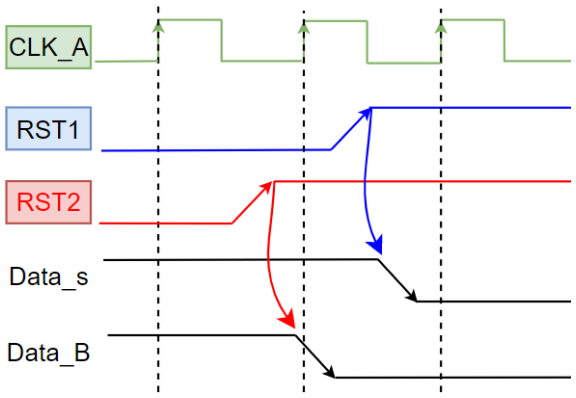

Figura 7-5: Ordine di priorità dei segnali di reset

- Un'ulteriore soluzione per non incorrere nella metastabilità è quello di inibire il clock durante tutto il periodo di reset.

Dunque, nell'analisi RDC si esegue uno step simile alla "configurazione dei segnali" eseguito durante l'analisi CDC, in cui si definisce una "priorità" dei segnali di reset presenti nel proprio design al fine di ridurre al minimo il numero di false violazioni che possono emergere durante l'analisi RDC. Per priorità, si intende l'ordine con cui devono essere asseriti i segnali di reset presenti nel design. Essenzialmente, è possibile definire uno step di "configurazione dei segnali di reset", in cui si descrive il loro comportamento all'interno del sistema.

8 Applicazione del flusso di analisi RDC

In questo punto dell'analisi si procede con l'effettuare la verifica RDC del design ai fini di analizzare le violazioni RDC risultanti.

- Il flusso RDC viene eseguito in parallelo all'analisi CDC ed infatti segue gli stessi step effettuati nel flusso CDC e quindi non vi è necessità di effettuare un'analisi separata ma vengono utilizzate le impostazioni ottenute dal flusso CDC. Anche nel flusso RDC viene eseguita l'identificazione dei domini di reset presenti e degli schemi di sincronizzazione presenti o meno nel design ma questa è ovviamente incentrata sui percorsi RDC e focalizzata nel ricercare schemi di sincronizzazione di reset di cui sono stati riportati degli esempi nella sezione 7.1.1.
- È necessario effettuare ulteriori configurazioni del design ed aggiungere regole RDC al fine di escludere dai risultati dell'analisi false violazioni.

8.1 Impostazione del design-RDC

- Come introdotto nella sezione 5.1 del flusso operativo CDC eseguito, vi è la necessità di eseguire una corretta impostazione del segnale di reset "POR" presente nel design. Non è sufficiente definire come esso viene asserito e che esso è un reset asincrono rispetto ai segnali di clock presenti nel sistema, ovvero che il "POR" è asincrono al clock di sistema e clock di interfaccia, ma è necessario anche definire cosa avviene nel design al momento della sua asserzione e de-asserzione.

Come descritto nella sezione 3.1.1 dell'introduzione al design, il reset POR in questione è un segnale di reset che viene gestito in maniera autonoma da un circuito analogico che ha il compito di monitorare la tensione con cui sta operando il dispositivo ed assicurare un corretto avvio di quest'ultimo. È quindi possibile intuire che tale segnale di reset viene asserito in una di queste condizioni e nel momento in cui esso è attivo il clock di sistema del dispositivo e dell'interfaccia sono spenti. Il POR viene distribuito su tutti i FF resettabili presenti all'interno del design. Deve quindi essere istruito il tool di analisi di tali condizioni al fine di escludere le violazioni RDC possibili che coinvolgono esclusivamente il segnale di reset POR. Le violazioni RDC dovute al segnale di reset POR non portano alcuna informazione ai fine della verifica del design ed è quindi non necessario prendere in considerazione questi tipi di violazioni. È possibile intuire che il numero di violazioni che possono essere dovute al segnale di reset è pressoché simile se non maggiore al numero di registri presenti nel design; dunque, la quantità di false violazioni introdotte sarebbe elevata.

- **Vengono quindi riportati degli esempi di cosa riportare al tool per istruire il design per poter gestire il segnale di POR:**

1. Configurazione del segnale POR = > reset asincrono POR attivo basso
2. Configurazione al momento della sua asserzione = > disabilita il FF di destinazione con segnale di reset POR con tali clocks {clock_system clock_interface}
3. Configurazione al momento della sua de-asserzione = > - disabilita il FF di destinazione con segnale di reset POR con tali clocks {clock_system clock_interface}

Gli ultimi due punti, (2) e (3), corrispondono all'informazione necessaria da fornire per poter istruire il tool riguardo all'asserzione e la de-asserzione del segnale di reset. Nel momento in cui viene asserito o de-asserito il segnale POR, i registri di destinazione di tale reset non dispongono di un clock attivo.

Le violazioni riguardanti il POR sono principalmente legate all'assenza di schemi di sincronizzazione di reset e vengono quindi riportate tali violazioni per poter identificare possibili violazioni dei tempi di removal/recovery dei registri di destinazione come anche possibili errori RDC tra registri non aventi stesso segnale di reset.

- Non sono definiti altri segnali di reset esterni al design e per questo motivo non viene definito un ordine di priorità di asserzione e de-asserzione dei segnali di reset esterni, essendo presente solo il POR. Bisogna sottolineare che, ovviamente, la sola dichiarazione del segnale di POR non implica che esso sia l'unico segnale di reset presente all'interno del design e che non vi siano altri segnali di reset all'interno di quest'ultimo. Il POR corrisponde all'unico reset esterno al design ed è per questo motivo che deve essere dichiarata la sua tipologia ed il suo comportamento.

Durante l'esecuzione operativa del flusso RDC vengono rilevati tutti i segnali di reset presenti all'interno del design, i rispettivi domini di reset di appartenenza e sono identificate le coppie RDC esistenti. Dunque, non devono essere dichiarati singolarmente tutti i segnali di reset presenti all'interno del design ma devono essere dichiarati i segnali di reset esterni che non possono essere identificati da parte del tool.

9 Risultati RDC

Dunque, dopo aver aggiunto le ultime configurazioni necessarie ad avere una corretta analisi delle violazioni RDC, si procede con l'effettuare l'identificazione degli schemi di sincronizzazione di reset presenti o meno nei percorsi RDC individuati. Vengono quindi adesso riportati alcuni dei risultati ottenuti dall'esecuzione del flusso RDC.

9.1 Violazione RDC – Registri non resettabili

Dall'analisi RDC effettuata sono state riscontrate violazioni RDC che coinvolgono registri senza segnali di reset. Essenzialmente, vengono identificate violazioni in cui il secondo FF coinvolto nella coppia RDC identificata non dispone di un segnale di reset e non è quindi resettabile. Il tool esegue una verifica strutturale del DUT e deve quindi segnalare la presenza anche di queste tipologie di registri che non dispongono di un segnale di reset, in quanto la sua assenza potrebbe essere un'omissione non voluta e quindi causare errori funzionali nel design.

Nel DUT sotto analisi questo tipo di registri senza segnali di reset sono voluti all'interno del design e non si deve quindi effettuare un'analisi delle violazioni che coinvolgono questi registri. Dunque, al fine di non avere la segnalazione di tali tipi di violazioni risultanti dall'analisi RDC, si applica una regola RDC, come spiegato nella sezione 4.3.3.4, per poter escludere tali violazioni dai risultati. La regola RDC in questione è la seguente:

- Configurazione della regola RDC => Ignora_FF_NonResettabili nelle violazioni RDC riscontrate

Dunque, come è possible notare dalla regola riportata, si procede con l'ignorare le violazioni RDC in cui i registri coinvolti sono registri che non dispongono del segnale di reset.

9.2 Violazione RDC – Reset Configurations

Viene adesso presa in considerazione la struttura di gestione dei reset dei registri di configurazione "Reset Configurations", già presentata nella sezione 6.1.4, per poter illustrare la violazione RDC identificata che coinvolge tutti i registri di configurazione presenti nel design.

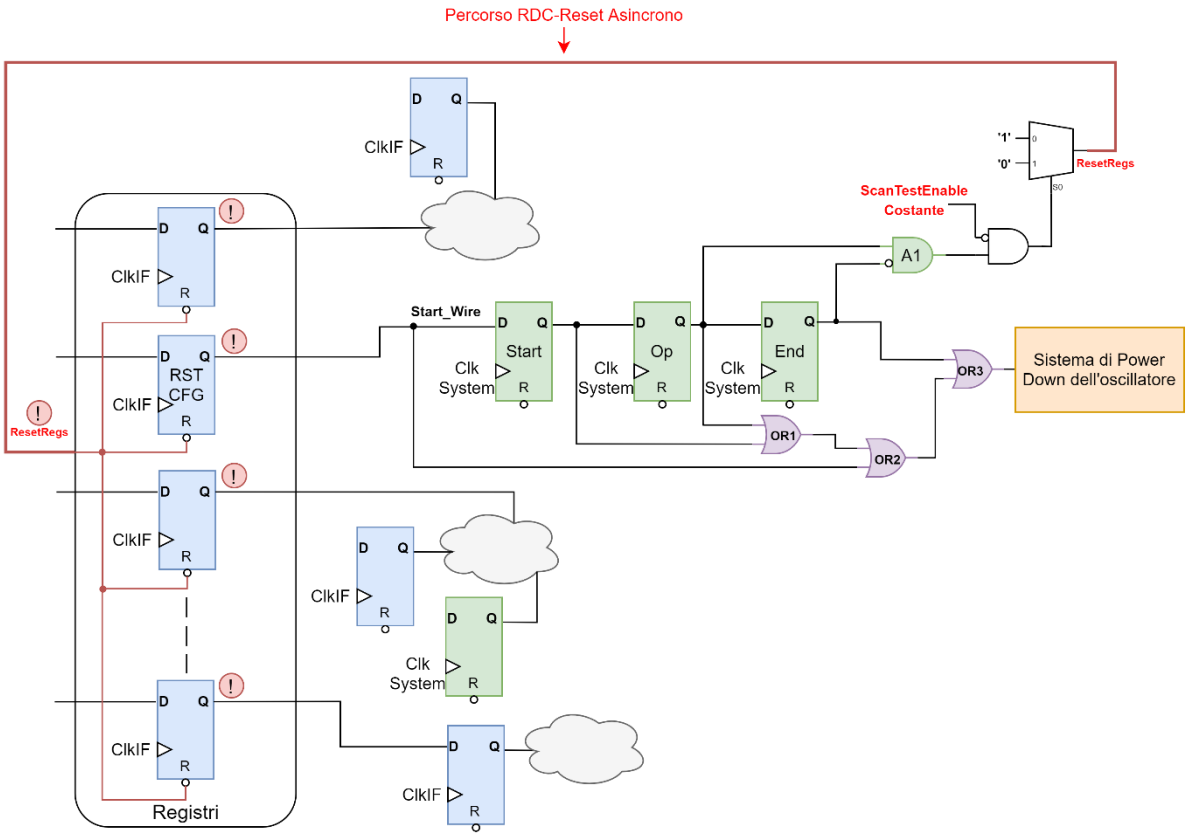

Figura 9-1: Violazione RDC reset configurations

- **Violazione RDC identificata:**

La violazione viene classificata come assenza di schema di sincronizzazione di reset lungo il percorso RDC che collega l'uscita del multiplexer "ResetRegs", pilotato in maniera sincrona al dominio di clock di sistema, e gli ingressi dei reset dei registri di configurazione, appartenenti al dominio di clock di interfaccia. Vi è dunque la possibilità di propagazione di una eventuale metastabilità dovuta ad un'asserzione asincrona del reset che può portare ad avere una violazione dei tempi di setup/hold dei registri appartenenti al dominio di clock di interfaccia come anche dovuta ad una de-asserzione asincrona in prossimità del fronte attivo del clock di interfaccia che può riflettersi in una possibile violazione dei tempi di removal/recovery.

- **Descrizione della violazione RDC:**

Come già introdotto nella sezione 6.1.4, per poter innescare la procedura di reset è necessario effettuare la scrittura del FF "RST_CFG" e nel momento in cui l' "1" logico viene campionato dal secondo FF "Op", viene innescata la procedura di auto-reset del FF "RST_CFG" attraverso la porta And

“A1” ed anche del reset dei registri di configurazione presenti nel design, portandosi quindi al loro valore di reset. Questa procedura si riflette in una violazione RDC, in quanto il segnale di reset generato da tale meccanismo si muove dal dominio di clock di sistema al dominio di clock di interfaccia, risultando quindi in un reset asincrono per il dominio di interfaccia. L’asserzione asincrona del “ResetRegs” potrebbe riflettersi in un problema RDC per i FF appartenenti allo stesso dominio di clock di interfaccia, come riportato nello schema di Figura 9-1. Alcuni dei registri di configurazione appartenenti al dominio di clock di interfaccia trasmettono verso il dominio di clock di sistema ma anche verso altri FF appartenenti al dominio di interfaccia. Quindi, un reset asincrono potrebbe essere asserito in prossimità del fronte attivo del clock di interfaccia portando ad avere una violazione dei tempi di setup/hold dei FF a valle e dunque avere una propagazione di metastabilità attraverso il design, come riportato nella Figura 9-2 . Essendo il segnale di reset asincrono, potrebbe anche insorgere una violazione dei tempi di removal/recovery nel momento della de-asserzione asincrona di tale segnale di reset in prossimità del fronte attivo del clock, come riportato nella Figura 9-3.

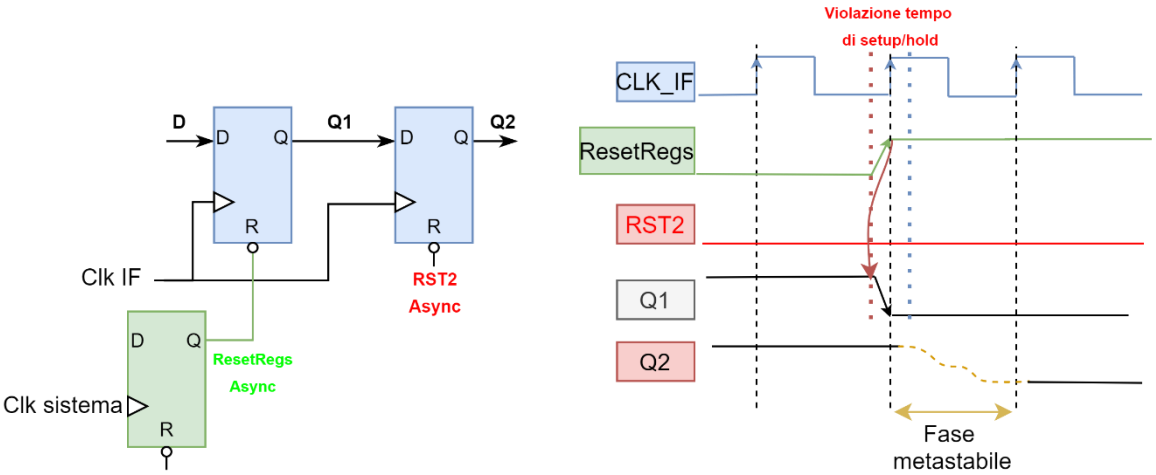

Figura 9-2: Violazione RDC dovuta ad asserzione asincrona reset ResetRegs

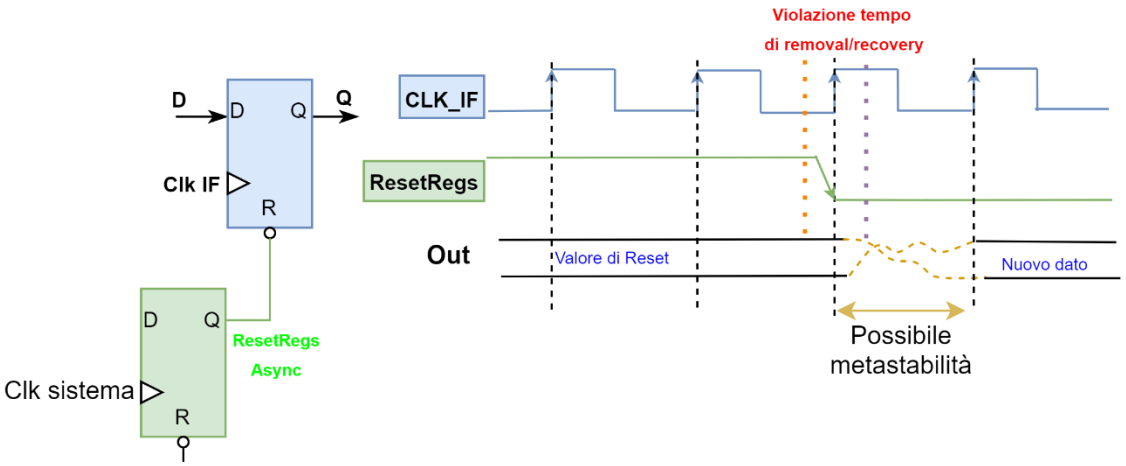

Figura 9-3: Violazione RDC dovuta a de-asserzione asincrona reset ResetRegs

- **Descrizione del protocollo:**

Tale violazione RDC risulta non essere un reale problema RDC per via del protocollo attuato nel design. Come riportato nel diagramma di timing, in Figura 9-4, una volta effettuata la scrittura del FF "RST_CFG", il clock di interfaccia viene inibito. Dunque, una volta avvenuto il campionamento dell' "1" logico da parte del FF "Op" con il clock di sistema, viene innescata la procedura di reset ed è possibile notare che il reset viene asserito per un intero colpo di clock di sistema portando quindi ad avere il reset dei registri e del FF "RST_CFG", il quale viene riportato al suo valore logico di reset "0" segnando il termine della procedura in questione. Una volta innescata tale procedura di reset, è necessario attendere che questa venga completata correttamente prima di poter procedere con una nuova scrittura da parte del clock di interfaccia.

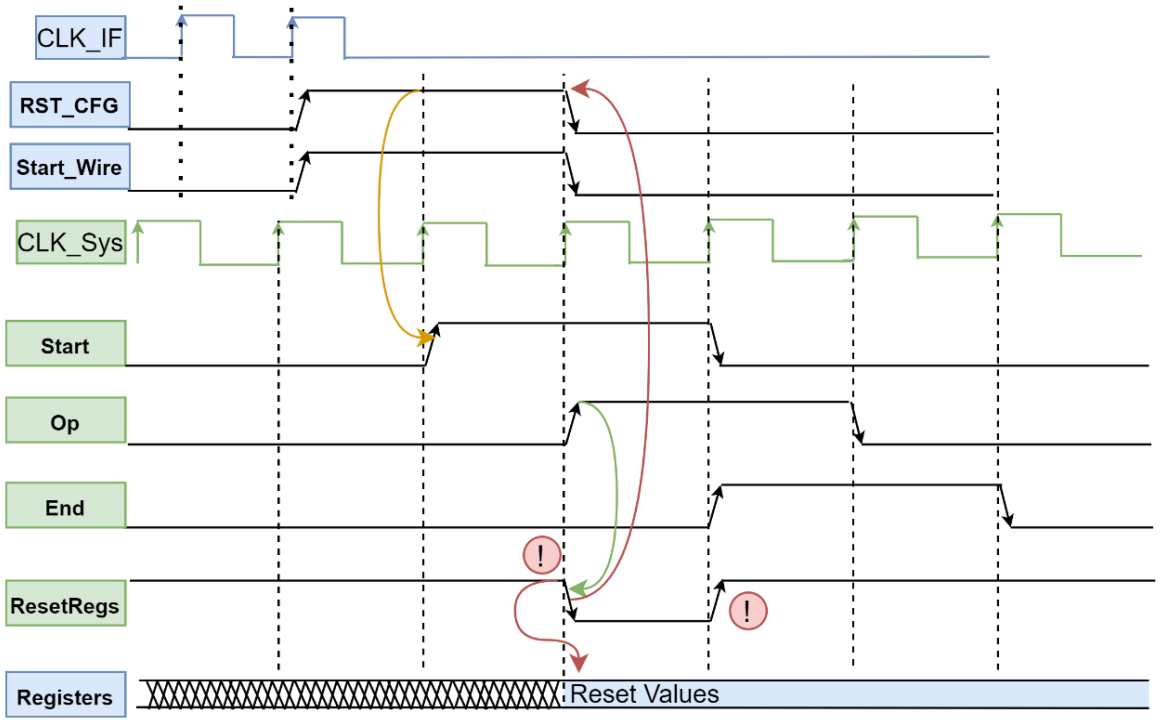

Figura 9-4: Diagramma di timing Reset configurations

Dunque, l'asserzione asincrona del segnale di reset "ResetRegs" gestito dal dominio di clock di sistema non risulta essere un problema RDC per il dominio di clock di interfaccia, in quanto il clock di interfaccia viene inibito e non è dunque possibile avere una violazione dei tempi di setup/hold dei registri appartenenti a tale dominio. Lo stesso discorso è valido per la de-asserzione asincrona, essendo il clock di interfaccia inibito. Nel momento in cui non fosse presente un'interruzione del clock di interfaccia, diverrebbe reale la violazione RDC dovuta alla asserzione e de-asserzione asincrona del "ResetRegs", le quali potrebbero portare ad una violazione rispettivamente dei tempi di setup/hold e removal/recovery, riportati nella Figura 9-2 e Figura 9-3, di tutti i registri di configurazione e quindi ad una propagazione della metastabilità attraverso il design da ognuno di essi.

Il problema dell'asserzione del reset si potrebbe riflettere anche in un problema CDC verso il dominio di clock di sistema, ma essendo il "ResetRegs" sincrono al dominio di clock di sistema, non si ha nessun problema CDC durante la procedura di reset. Ovviamente, vale lo stesso discorso per la sua de-asserzione, questa non è un problema in quanto anche la de-asserzione del "ResetRegs" è sincrona al dominio di clock di sistema.

Dal caso appena riportato, è possibile quindi notare come sia stata adottata una delle soluzioni elencate nella sezione 7.1.1 e nella sezione 7.1.3, ovvero la soluzione in cui si inibisce il clock interessato durante la procedura di reset dei registri.

9.3 Violazione RDC – Dati DSP

Viene adesso presa in considerazione una parte della struttura già presentata nella sezione 6.1.3 al fine di illustrare la violazione RDC identificata che coinvolge il reset del FF "Change" che appartiene al dominio di clock di interfaccia.

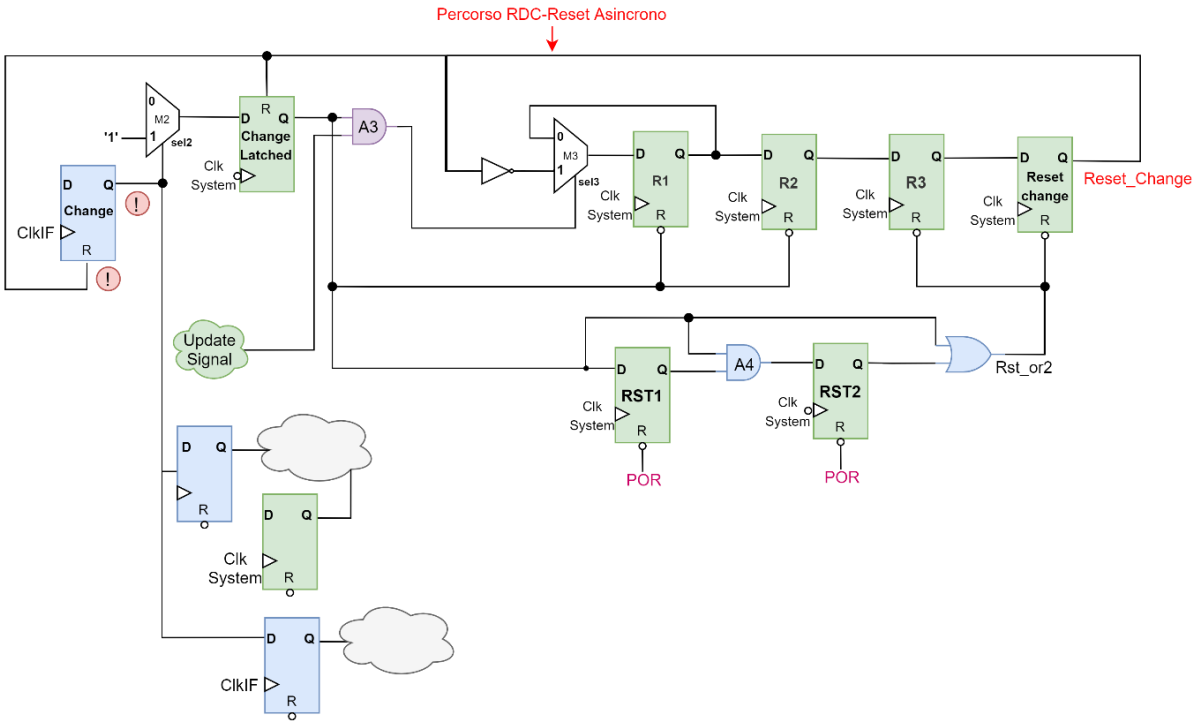

Figura 9-5: Struttura reset del registro Change

- **Violazione RDC identificata:**

La violazione RDC identificata coinvolge il percorso RDC che collega l'uscita del FF "ResetChange", appartenente al dominio di clock di sistema, e l'ingresso del reset del FF "Change", appartenente al dominio di clock di interfaccia. Essenzialmente, viene identificata l'assenza di uno schema di sincronizzazione di reset in tale percorso e vi è dunque la possibilità di propagazione di una eventuale metastabilità, causata da un'asserzione asincrona del reset che porta ad una violazione dei tempi di setup /hold dei registri appartenenti ai domini di clock di interfaccia come anche dovuta ad una de-asserzione asincrona che può riflettersi in una possibile violazione dei tempi di removal/recovery.

- **Descrizione della violazione RDC:**

L'asserzione asincrona del reset "Reset_Change" potrebbe riflettersi in un problema RDC per i FF appartenenti allo stesso dominio di clock di interfaccia, in quanto, come riportato nello schema di Figura 9-5, l'uscita del FF "Change" è collegata ad ulteriori registri appartenenti al dominio di clock di

interfaccia. Il reset asincrono potrebbe essere asserito in prossimità del fronte attivo del clock di interfaccia portando ad avere una violazione dei tempi di setup/hold dei FF a valle appartenenti al dominio di clock di interfaccia e dunque avere una propagazione di metastabilità attraverso il design, caso quindi simile a quello riportato nella Figura 9-2 della violazione RDC precedente. Essendo il segnale di reset asincrono, potrebbe anche insorgere una violazione dei tempi di removal/recovery nel momento della de-asserzione asincrona di tale segnale di reset in prossimità del fronte attivo del clock di interfaccia, dunque, questo si riflette in un caso simile come quello riportato nella Figura 9-3 della violazione RDC precedente.

- **Descrizione del protocollo:**

Come riportato dal diagramma di timing in Figura 9-6, una volta effettuata la scrittura del FF "Change", il clock di interfaccia viene inibito e si attende quindi il campionamento del segnale da parte del domino di destinazione. Avvenuto il campionamento del segnale "Change" da parte del FF "Change_latched", il FF "ResetChange" viene asserito al quarto colpo di clock di sistema e rimane al valore logico "1" per mezzo colpo di clock. Al momento dell'asserzione del segnale di reset "ResetChange" è possibile notare che viene resettato il FF "Change", come anche i FF "R1 R2 ed Change Latched", che vengono riportati al suo valore di reset "0" logico segnando anche la conclusione dell'acquisizione del nuovo dato. Successivamente, la de-asserzione del segnale di "ResetChange" è possibile grazie al segnale "Rst_Or2" che commuta allo "0" logico.

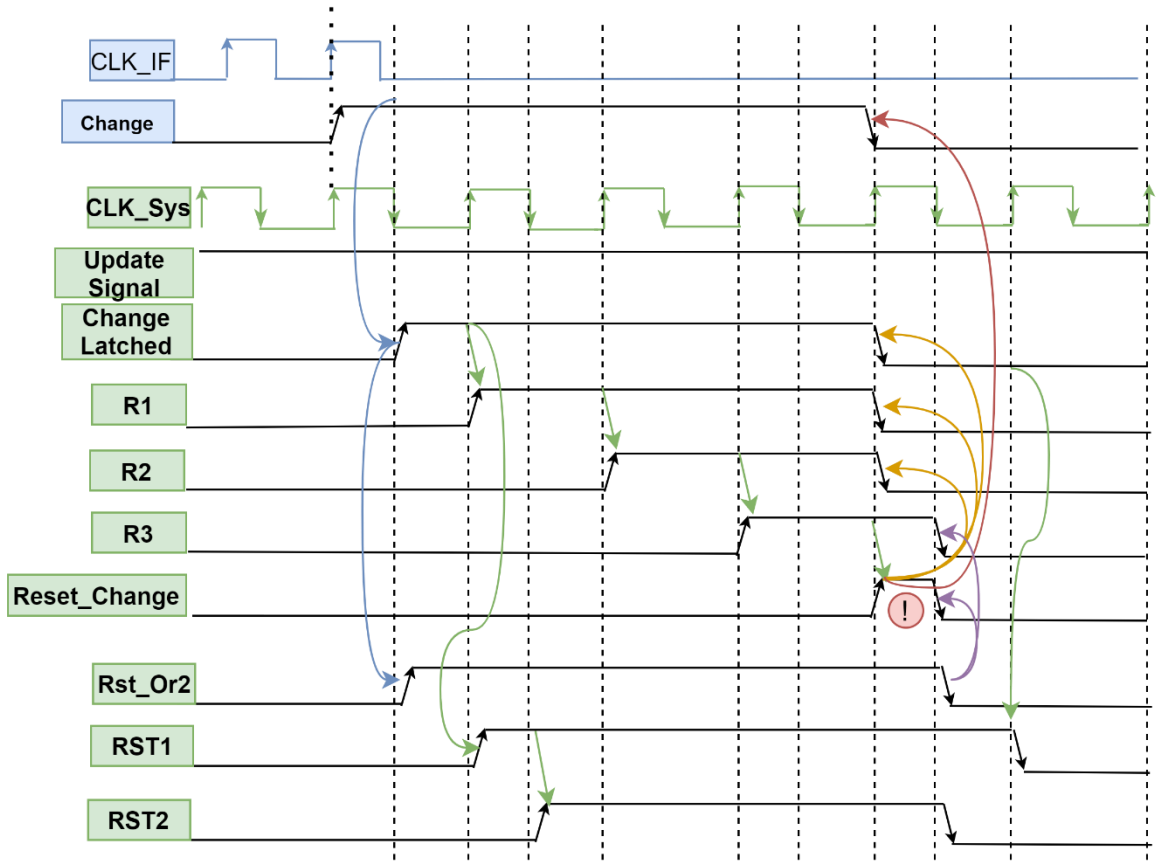

Figura 9-6: Diagramma di timing

È dunque possibile notare che l'asserzione asincrona del segnale di reset "ResetChange" gestita dal dominio di clock di sistema non risulta essere un problema RDC per il dominio di clock di interfaccia, in quanto il clock di interfaccia viene inibito e non è dunque possibile avere una violazione dei tempi

di setup/hold dei registri a valle del FF “Change” ed appartenenti a tale dominio. Lo stesso discorso è da considerarsi valido per la de-asserzione asincrona essendo il clock di interfaccia inibito. Nel caso in cui non avvenisse l’interruzione del clock di interfaccia, tale violazione RDC diverrebbe un reale problema RDC legato sia alla asserzione che de-asserzione asincrona del “Reset_Change”, che potrebbero dunque portare ad una violazione rispettivamente dei tempi di setup/hold dei registri presenti a valle e dei tempi di removal/recovery del registro “Change” e dunque all’insorgere e al propagarsi della metastabilità attraverso il design.

Il problema dell’asserzione del segnale di reset “ResetChange” si potrebbe anche riflettere in un problema CDC verso il dominio di clock di sistema, ma essendo tale segnale di reset gestito dal dominio di clock di sistema, non si ha nessun problema CDC. Lo stesso discorso è valido per la sua de-asserzione, questa non è un problema in quanto anche la de-asserzione del “ResetChange” è sincrona al dominio di clock di sistema.

Anche in questo caso appena riportato è possibile notare come sia stata adottata la soluzione in cui si inibisce il clock interessato durante la procedura di reset, una delle soluzioni elencate nella sezione 7.1.1 e nella sezione 7.1.3.

10 Flusso di verifica formale

Lo scopo che si vuole raggiungere all'interno del capitolo seguente è quello di validare, attraverso l'ausilio della verifica formale, i waivers effettuati sulle violazioni riportate nei capitoli precedenti che sono stati individuate dall'analisi CDC e RDC del design. Si prosegue con seguire il flusso di verifica formale progettato, come riportato nella Figura 10-1, con lo scopo di raggiungere tale obiettivo di validazione. Come è possibile notare dal flusso, una volta descritte le asserzioni, proprietà e assunzioni che descrivono i comportamenti dei protocolli presenti nelle diverse violazioni trovate, si procede con il verificare formalmente che questi protocolli rispettino il comportamento aspettato. In caso contrario, si deve ricorrere ad eventuali modifiche del codice RTL, in quanto i waivers effettuati non sono corretti perché i protocolli, proprietà oppure gli schemi adottati non rispettano il comportamento aspettato e dunque risultano essere reali problematiche CDC o RDC.

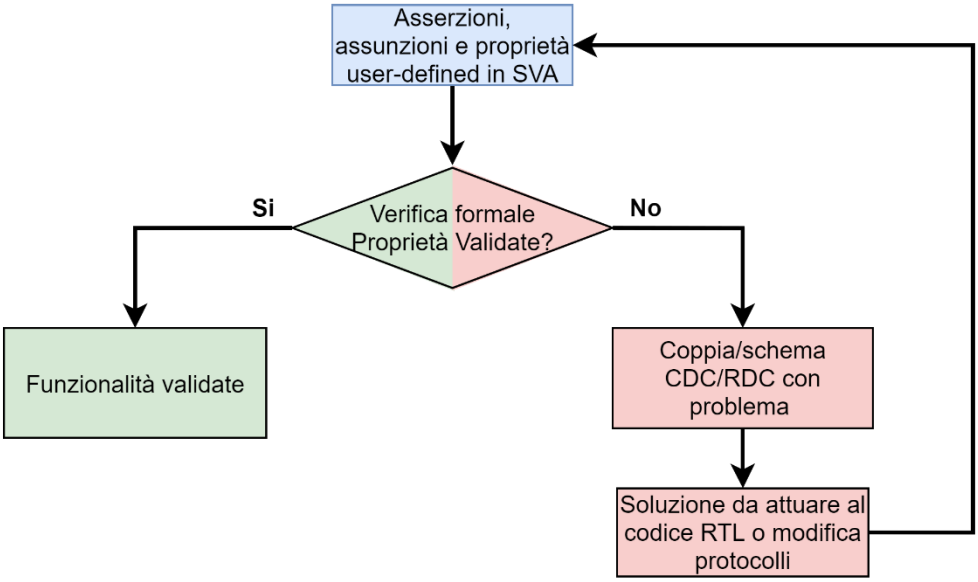

Figura 10-1: Flusso di verifica formale

10.1 Analisi formale

Per poter procedere con la Formal Property Verification e quindi poter verificare il corretto comportamento del proprio design, vi è la necessità di conoscere il modo con il quale è possibile definire il comportamento voluto del proprio design. Bisogna quindi introdurre i concetti di asserzione, assunzione e cover ed imparare a descriverli grazie all'utilizzo del linguaggio SVA "System Verilog Assertions". Ovviamente, non esiste solo questo linguaggio per poter raggiungere tale scopo di verifica, ma vi è anche la presenza di altri possibili linguaggi come, ad esempio, il PSL "Property Specification Language". Viene però preso in considerazione il linguaggio SVA essendo quest'ultimo adottato come standard nel gruppo di lavoro dove è stato svolto il lavoro di tesi.

Si introducono quindi i concetti di:

- **Asserzione**: Quando si procede con il definire un'asserzione, si sta procedendo con il definire un comportamento del proprio design che ci si aspetta essere sempre vero. Essenzialmente, l'obiettivo principale è quello di provare matematicamente che l'RTL del proprio design non possa in alcun modo far fallire l'asserzione definita. Attraverso un'asserzione è possibile verificare che una determinata condizione, sequenza di eventi o comportamento di cui si vuole avere conferma nel proprio design avvenga in maniera corretta. Ovvero, tramite un'asserzione è possibile verificare che il proprio RTL soddisfi le specifiche volute e non vi siano condizioni tali per cui non vengano rispettati tali comportamenti. Le asserzioni possono essere di due tipologie:
    1) **Immediate:** Sono asserzioni che non sono basate su un meccanismo di clock o reset e vengono prese in analisi durante l'analisi procedurale del codice SVA.
    2) **Concurrent**: Sono le asserzioni che vengono tipicamente utilizzate e sono dipendenti da un meccanismo di clock o di reset e consentono la descrizione di proprietà più complicate del comportamento dei segnali nel tempo, ovvero consentono di poter descrivere una sequenza di eventi che possono concatenarsi o meno e succedersi una dopo l'altra.

  Per poter meglio capire il significato di asserzione, si può pensare ad una sequenza di eventi che devono avvenire nel proprio design. Ad esempio, nel momento in cui avviene una transizione di un segnale "Start" dal valore logico "0" al valore logico "1" sul fronte di salita del clock, al prossimo colpo di clock il segnale "A" deve essere al valore logico "1" e deve rimanere asserito a tale valore anche per il prossimo colpo di clock. L'asserzione ha quindi il compito di trovare una possibile combinazione di segnali che non consentano che tale sequenza possa verificarsi, essenzialmente ha il compito di identificare un "Contro-Esempio" che illustra il motivo per il quale non avviene tale sequenza di eventi. In questo semplice esempio, si potrebbe pensare che venga trovato un controesempio in cui il segnale di reset non consenta la sequenza, bisogna quindi escluderlo dall'asserzione in quanto quest'ultima non è più valida nel momento in cui il reset è asserito. È possibile apprezzare l'asserzione nel diagramma di timing riportato nella Figura 10-2 e nella Figura 10-3.

1) “(@posedge CLKB) $Rose(Start) |-> A ##1 A “, trovato contro-esempio “Cex” dovuto al segnale di Reset

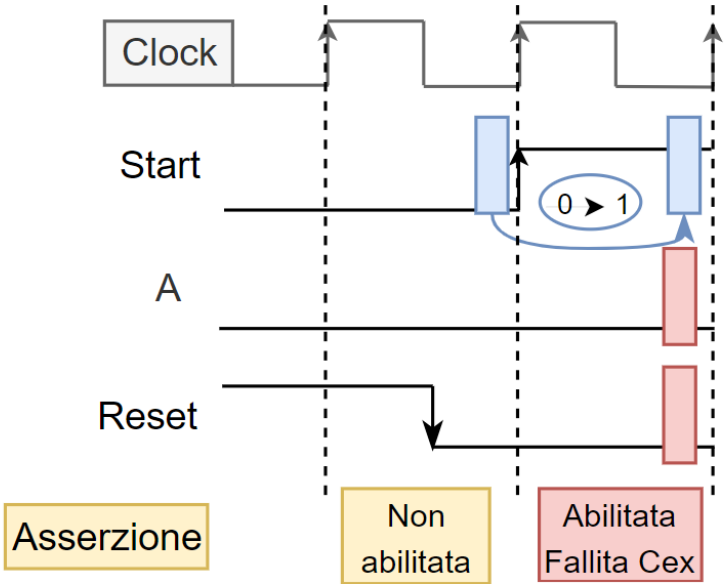

Figura 10-2: Esempio di asserzione 1

2) “(@posedge CLKB) disable iff (~Reset) $Rose(Start) |-> A ##1 A “

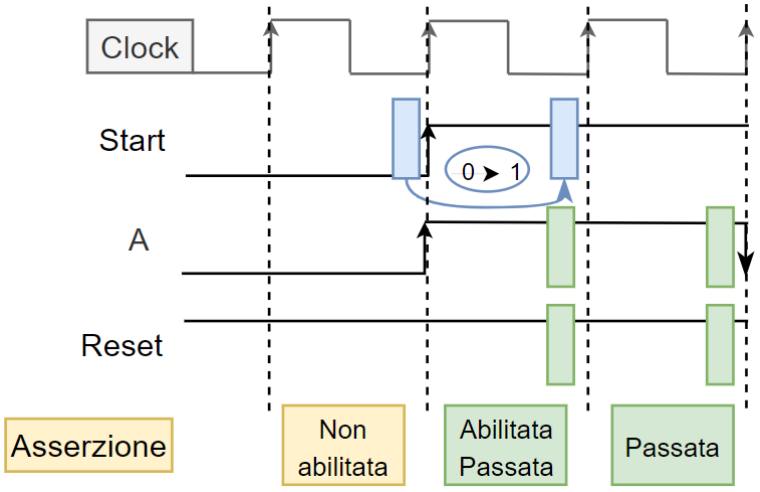

Figura 10-3: Esempio di asserzione 2

- **Assunzione**: L’assunzione, a differenza dell’asserzione, ha il compito di definire un determinato comportamento che caratterizza il funzionamento del design. Come è possibile intuire dal nome, un’assunzione è sempre considerata vera dal tool di verifica formale. L’assunzione viene definita al fine di verificare le asserzioni in modo corretto, meglio ancora in un “contesto” corretto. Senza le assunzioni, un tool di verifica formale procede con il provare formalmente un’asserzione andando a commutare ogni possibile segnale del DUT presente nel cono di logica, ma non è detto che determinati eventi siano contemplati dalle specifiche o che il design sia stato progettato per poterli affrontare. Quindi, attraverso le assunzioni è possibile descrivere il possibile comportamento degli ingressi e dei segnali escludendo quindi eventualità non possibili e non realistiche e quindi eventi che non sono previsti per specifiche del design o per struttura di quest’ultimo.

- **Cover points**: Attraverso le cover vi è la possibilità di definire delle particolari condizioni di cui si vuole essere sicuri che siano prese in analisi durante la verifica formale. Tipicamente, descrivere asserzioni e assunzioni assieme non è sufficiente per poter ottenere una copertura effettiva del design. Vi è la possibilità che alcune delle assunzioni applicate possano riflettersi in un "Over-Constraint" non voluto del comportamento del design. Essenzialmente, si potrebbe escludere in maniera non volontaria degli stati o combinazioni del design che in realtà dovrebbero essere "coperti" al momento della verifica dell'asserzione. Grazie alle "Cover", si riesce a forzare la copertura di stati o combinazioni del design durante la verifica dell'asserzione, sottolineando anche possibili errori nella verifica dovuti ad assunzioni non effettuate correttamente o risultanti troppo limitative. Le cover, dunque, assicurano che vengano coperti formalmente sequenze di eventi, stati o combinazioni all'interno del design. I risultati ottenuti dalla cover sono essenzialmente due: Provata o Fallita. Ma tali risultati non devono essere confusi con quelli delle asserzioni. Una cover provata porta ad avere una rappresentazione della sequenza di eventi di cui si è voluto verificare che fosse possibile avere, ma non implica direttamente che quest'ultima sia corretta. Una cover fallita non implica direttamente che vi è un errore, ma implica che la sequenza di eventi descritta non può avvenire e non c'è alcun modo per poterla far avvenire.

10.2 Applicazione del flusso di verifica formale

Nella seguente sezione si procede con il verificare, attraverso l'ausilio della verifica formale, che i waivers effettuati e le proprietà che descrivono i protocolli presenti siano effettivamente rispettati.

10.2.1 Verifica formale effettuata riguardante la disabilitazione dell'interfacce I2C e I3C

Nella seguente sezione si procede con il giustificare, attraverso la verifica formale, la corretta disabilitazione delle interfacce di I/O I2C e I3C.

- **Descrizione del comportamento che si vuole verificare:**

Come citato nella sezione 3.1.1, l'obiettivo di analisi in questione non prevede il funzionamento dell'interfaccia I2C e I3C durante l'attività di analisi del sistema, ma prevede solo il funzionamento dell'interfaccia SPI. Si assume che i clock che gestiscono tali interfacce siano stabili e che non effettuino per alcun motivo una commutazione. Questo comportamento è dovuto alle assunzioni effettuate, introdotte nella sezione 5.3 nella configurazione dei segnali, sui bit di controllo dei registri che gestiscono la disabilitazione delle interfacce I2C e I3C, come anche l'assunzione effettuata sull'ingresso primario "SPI_Enable" che consente di selezionare l'interfaccia SPI. Al fine di verificare effettivamente che non possano esserci commutazioni da parte della logica e dei registri che coinvolgono le interfacce escluse, viene verificato formalmente che i clock I2C e I3C rimangano stabili per tutto il funzionamento del dispositivo.

- **Descrizione dei passaggi dell'asserzioni:**

Vengono adesso riportate le asserzioni effettuate in codice SVA-System Verilog Assertions che riguardano i clock e i segnali di controllo dell'interfaccia I3C:

- *"Assert {register_inst.I3C_CFG[1] |-> I3C_DIS && ~enable_i3c_scl } -name i3cInterfaceDisable".* Tale asserzione procede con il verificare che nello stesso colpo di clock in cui il bit di disabilitazione dell'interfaccia I3C assume il valore logico "1", i segnali di "I3C disable" e il segnale di gating "enable_I3C_scl" del clock assumano rispettivamente i valori logici "1" e "0".

- *"Assert { @(posedge if_ck) disable iff (~register_inst.I3C_CFG[1]) $rose(register_inst.I3C_CFG[1]) && ~ if_ckI3C |=> if_ckI3C } -name ClkGatedI3C"*

Tale asserzione ha il compito di verificare che il clock di interfaccia utile per la I3C rimanga stabile nel momento in cui il bit di disabilitazione dell'interfaccia "I3C_CFG[1]" viene configurato al valore logico "1". Essenzialmente, si prende come riferimento il clock di interfaccia "if_ck" e nel momento in cui viene rilevato sul suo fronte di salita una transizione "0->1" del bit di controllo "I3C_CFG[1]" e il valore attuale del clock dell'interfaccia I3C è al valore logico "0", viene attivata l'asserzione. Il rilevamento della transizione è possibile attraverso la funzione $rose(). Da quando quest'ultima è attiva, viene verificato al colpo di clock successivo, l'operatore "|=>", che il valore assunto da parte di "if_ck_i3c" è uguale al valore logico "1", dunque si verifica che non è più avvenuta la commutazione del clock di interfaccia. Con il simbolo "~" viene indicato lo "0" logico. Viene riportato, nel diagramma di timing nella Figura 10-4, il comportamento dell'asserzione in questione e tale diagramma viene riportato per una più semplice comprensione di come si comporta tale asserzione. È possibile anche notare la presenza della funzione "disable iff (~register_inst.I3C_CFG[1])" dove quest'ultima ha il compito di disabilitare l'asserzione nel momento in cui il bit di controllo in questione commuti al valore logico "0. Essenzialmente, si sta affermando che nel momento in cui il segnale "I3C_CFG[1]" assume il valore logico "0", non ha più validità tale asserzione e quindi non si vuole controllare il suo comportamento sotto tale condizione. Tale funzione viene aggiunta in quanto è necessario studiare il comportamento del design nel momento in cui il segnale in questione assume esclusivamente il valore logico "1" ed evitare che venga fornito un contro-esempio in cui viene mostrato che l'asserzione non viene verificata perché si riporta al valore logico "0" il bit "I3C_CFG[1]".

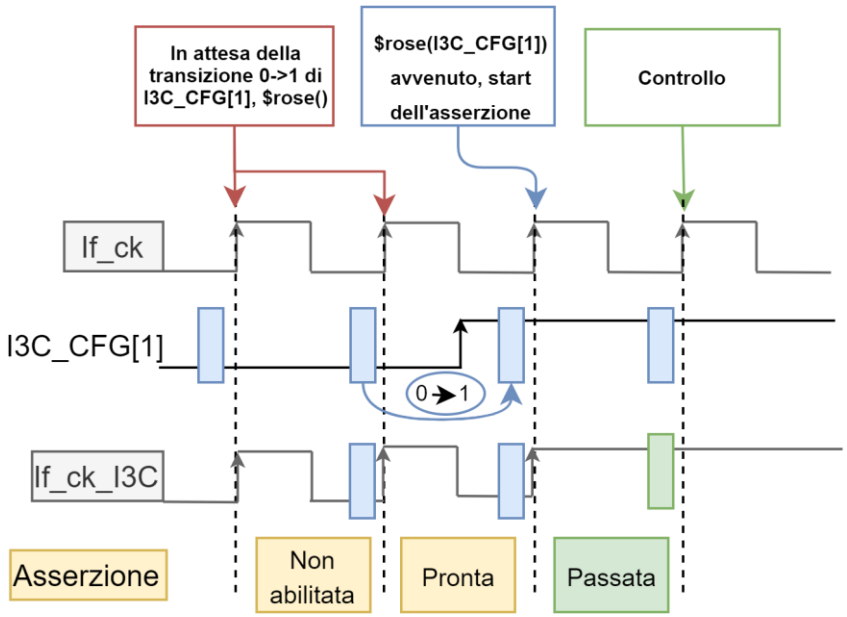

Figura 10-4: Diagramma di timing per asserzione I3C

Vengono adesso riportate le asserzioni effettuate che riguardano i clock e i segnali di controllo dell'interfaccia I2C:

- *Assert { register_inst.I2C_CFG_i[0] |-> ~enable_i2c && ~enable_sda_i2c } -name i2cInterfaceDis*
- *"Assert { @(posedge if_ck) disable iff (~register_inst.I3C_CFG_i[0]) $rose(register_inst.I2C_CFG_i[1]) && ~ if_ckI2C |=> if_ckI2C } -name ClkGatedI2C"*

10.2.2 Verifica formale del waiver effettuato riguardante Dati DSP

Nella seguente sezione si procede con il giustificare, attraverso la verifica formale, che la violazione CDC del caso “Dati DSP”, spiegato nella sezione 6.1.3, non risulta essere un reale problema CDC per via del protocollo di acquisizione adottato.

- **Descrizione del comportamento che si vuole verificare:**

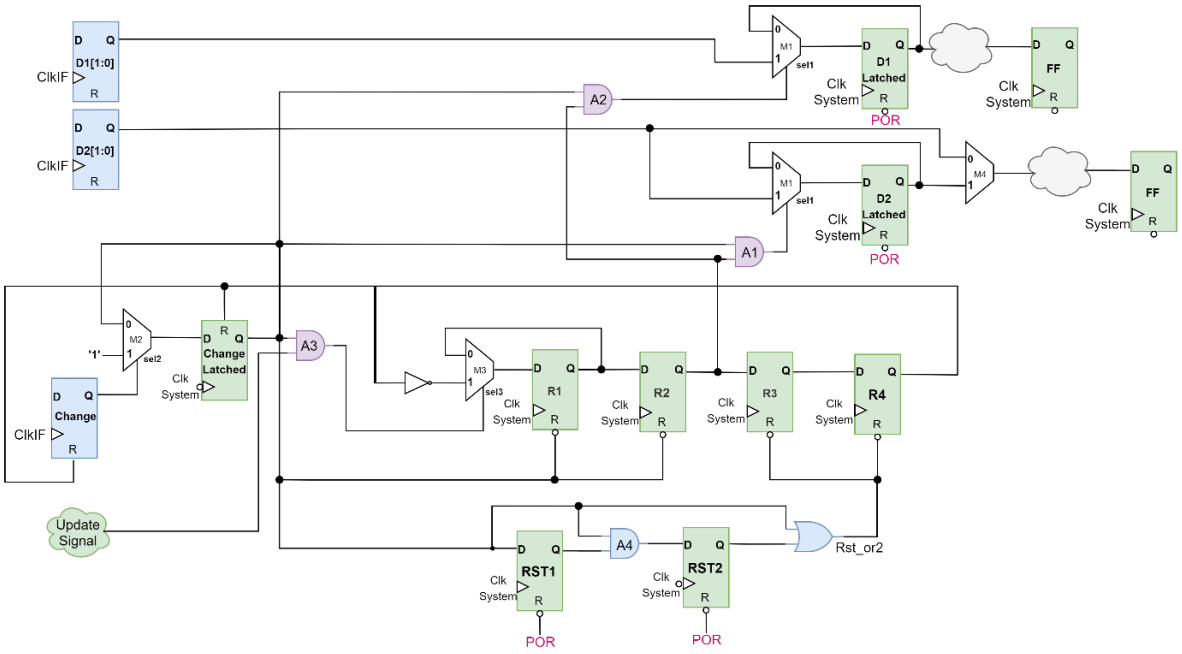

Figura 10-5: Struttura del caso Dati DSP

Come spiegato nella sezione 6.1.3, la violazione in questione non risulta essere un reale problema CDC, in quanto il campionamento dei dati D1[1:0] e D2[1:0], provenienti dal dominio di clock di interfaccia, non può avvenire al momento della loro commutazione ma avviene solo nel momento in cui essi sono stabili. Dunque, l’obiettivo principale è quello di verificare che i dati “D1[1:0], D2[1:0]” non effettuino alcuna commutazione durante la procedura di acquisizione e che il campionamento di essi avvenga da parte dei registri “D1_Latched[1:0]” “D2_Latched[1:0]” quando essi sono già stabili da diversi colpi di clock di sistema. In questo modo, si garantisce di non avere alcuna violazione dei tempi di setup/hold di tali registri del dominio di clock di sistema. Al fine di avere una migliore comprensione dei procedimenti attuati per poter effettuare la validazione del waiver, viene riportata la struttura di acquisizione nella Figura 10-5.

- **Descrizione dei passaggi costituenti l’asserzione:**

Con la seguente asserzione viene verificata completamente la procedura di acquisizione dei dati “D1[1:0], D2[1:0]” da parte dei registri “D1_Latched[1:0]” “D2_Latched[1:0]” e viene riportata, nella Figura 10-6, il timing dell’asserzione al fine di avere una migliore comprensione del comportamento che deve essere validato. L’asserzione viene abilitata nel momento in cui vi è stata una transizione 0-1 del FF “Change_Latched”. Al tempo stesso, l’operatore “&&” (And), il bit “Update” deve esser al valore logico “1” e deve essersi verificata una commutazione qualsiasi dei dati “D1[1:0], D2[1:0]” sul fronte di salita del clock di sistema, condizione espressa attraverso l’operatore $changed(“..”).

Viene riportato il diagramma di timing, inerente all'asserzione effettuata, nella Figura 10-6. È possibile notare che non è presente un solo clock di interfaccia, ma è presente un ulteriore clock asincrono definito Clock_IF_WR. Tale clock è anch'esso asincrono a quello di sistema ed è derivato e più lento del clock di interfaccia. Esso rappresenta il clock con cui viene effettuata la scrittura dei registri in questione.

Viene adesso riportato il codice scritto in SVA al fine di validare tale caso:

L'asserzione viene abilitata nel momento in cui avviene l'evento di partenza definito dal termine *@(posedge Clock) ("..")*, ovvero quando è presente una determinata condizione sul fronte di salita del clock in questione. Con il termine "$rose("..")", si prende in considerazione la transizione da "0" ad "1" del bit in questione. Con il simbolo "~" viene indicato lo "0" logico.

1) *Assert { @(posedge Clock di sistema)* ***(Start asserzione)*** *$rose(Change_latched) && Update && $changed(D1[1:0]) && $changed(D2[1:0]) |-> ~ R1 && ~R2 && D1_Latched[1:0] ==$past(D1_Latched[1:0]) && D2_Latched[1:0] [1:0]==$past(D2_Latched[1:0]) ##1* ***(1.0)*** *R1 && ~ R2 && D1[1:0] == $past(D1[1:0]) && D2[1:0] == $past(D2[1:0]) && D1_Latched[1:0]== $past(D1_Latched[1:0]) && D2_Latched[1:0]== $past(D2_Latched[1:0]) ##1* ***(1.1)*** *R1 && R2 && D1[1:0] == $past(D1[1:0]) && D2[1:0] == $past(D2[1:0]) && D1_Latched[1:0] [1:0] == $past(D1_Latched[1:0]) && D2_Latched[1:0] [1:0] == $past(D2_Latched[1:0]) ##1* ***(1.2)*** *D1_Latched[1:0] == D1[1:0] && D2_Latched[1:0] == D2[1:0]} -name Acquisizione*

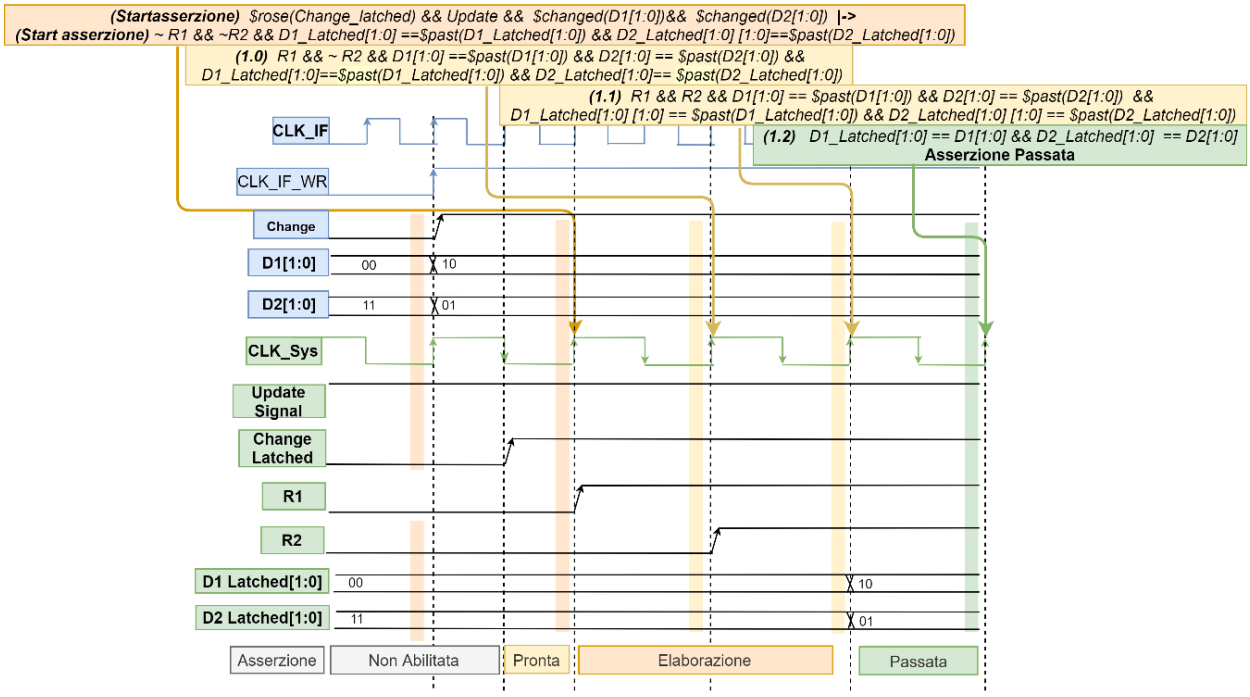

Figura 10-6: Diagramma di timing rappresentativo dell'asserzione del caso Dati DSP

Una volta identificato l'evento di partenza, si procede col verificare la sequenza di eventi che caratterizza l'asserzione. Con il simbolo "|->" si indica l'operatore "implicazione nello stesso ciclo della condizione di trigger". In questo caso, l'operazione di trigger è la serie di eventi di partenza dell'asserzione, ovvero la parte precedente a tale operatore: ***(Start asserzione)*** *$rose (Change_latched) && Update && $changed(D1[1:0]) && $changed(D2[1:0]).* Quindi, questo vuol dire che devono accadere nello stesso colpo di clock in cui vi è stato l'evento di partenza, la prima sequenza di eventi presente dopo il simbolo "|->" fino all'operatore successivo "##1". Nello stesso colpo di clock è necessario che sia il FF "R1" che il FF "R2" debbano essere al valore logico "0", ed al tempo stesso i

datI “D1_Latched[1:0]” e “D2_Latched[1:0]” devono assumere nel colpo di clock attuale il valore assunto nel colpo di clock precedente, condizione espressa attraverso l’operatore $past(“..”). Al colpo di clock successivo, indicato con l’operatore “##1”, deve verificarsi la commutazione al valore logico “1” del FF “R1”, mentre il “R2” deve rimanere fermo al valore logico “0”. La parte fondamentale dell’asserzione è la seguente: *“D1[1:0] == $past(D1[1:0])” e “D2[1:0] == $past(D2[1:0])”.* In questo modo, si vuole verificare che il valore assunto dai dati in questione D1 e D2 al colpo di clock attuale è uguale al valore assunto al colpo di clock precedente e dunque questo vuol dire che viene controllato che il dato non deve aver cambiato il suo valore da quando ha effettuato la sua prima commutazione. Questo ragionamento deve essere valido anche per i dati “D1_Latched[1:0]” e “D2_Latched[1:0]” che non devono aver ancora acquisito il nuovo valore dei dati “D1[1:0]” e “D2[1:0]”. Al colpo di clock successivo, deve avvenire la commutazione al valore logico “1” del FF “R2” e non devono ancora aver cambiato il loro valore i dati “D1[1:0]”,“D2[1:0]” e “D1_Latched[1:0]” e “D2_Latched[1:0]”. Infine, guardando all’ultimo step dell’asserzione, deve avvenire il corretto campionamento da parte dei registri “D1_Latched[1:0]”, “D2_Latched[1:0]” dei nuovi dati “D1[1:0]” e “D2[1:0]”, i quali sono rimasti stabili per tutta la procedura di acquisizione.

- **Risultato ottenuto**:

L’asserzione appena descritta viene riportata come “Passata” e quindi esente da controesempi. Attraverso tale asserzione è stato possibile validare il funzionamento della procedura di acquisizione e validare che la violazione identificata non è un problema CDC, perché i registri “D1_latched[1:0]” e “D2_latched[1:0]” procedono con il campionamento del dato “D1[1:0]” e “D2[1:0]” quando questi sono stabili e i dati “D1[1:0]” e “D2[1:0]” non hanno effettuato alcuna commutazione durante la procedura di acquisizione, ovvero non hanno effettuato alcuna commutazione per tutto il tempo in cui il segnale “Change” è rimasto asserito al valore logico “1”.

  - È stato quindi validata la procedura di acquisizione dei dati ed è stato quindi validato che non è presente un problema CDC nella violazione riportata, in quanto non è possibile avere una violazione dei tempi di setup/hold dei registri “D1_Latched[1:0]” e “D2_Latched[1:0]” al momento del campionamento dei dati “D1[1:0]” e “D2[1:0]”.

Nel momento in cui l’asserzione viene identificata come “Passata”, si ha la conferma che la serie di eventi descritti nell’asserzione avviene in maniera corretta e che quindi non esistono casi tali per cui questa serie di eventi non segue il comportamento aspettato. Dunque, non viene identificato un controesempio. Quest’ultimo è riportato nel momento in cui un’asserzione è validata come “fallita”, ovvero esiste una condizione tale per cui non si verifica la serie di eventi descritti nell’asserzione stessa. Dunque, in questo caso non viene presentato un controesempio che descrive il motivo per il quale non è avvenuta la serie di eventi descritta dall’asserzione.

- **Cover definita e risultato**:

Viene adesso presa in considerazione una Cover molto simile all’asserzione appena riportata e viene affiancata a quest’ultima con lo scopo di avere un’ulteriore validazione della robustezza del protocollo di acquisizione adottato.

```
cover { @(posedge Clock di sistema) $rose(Change_latched) && Update_signal &&
$changed(D1[1:0]) && $changed(D2[1:0]) ##0 ~ R1 && ~ R2 && D1[1:0] != $past(D1[1:0])
&& D2[1:0] != $past(D2[1:0]) ##1 R1 && ~ R2 && D1[1:0] != $past(D1[1:0]) && D2[1:0] !=
```

$past(D2[1:0]) ##1 R1 && R2 && D1[1:0] != $past(D1[1:0]) && D2[1:0] != $past(D2[1:0]) ##1 D1_latched [1:0] == D1[1:0] && D2_latched [1:0] == D2[1:0] } -name Cover_latched_NTF

La parte caratterizzante di tale cover è la seguente: “D1[1:0] != $past(D1[1:0]) && D2[1:0] != $past(D2[1:0])”. Essenzialmente, si sta provando a forzare il design, attraverso l’utilizzo della cover, ad avere ad ogni colpo di clock di sistema un valore sempre diverso dei dati “D1[1:0]”, “D2[1:0]” da quando è avvenuto l’evento di partenza dell’acquisizione. Tale Cover ha come risultato “Fallita” e questo risultato è in realtà positivo, perché si è riusciti ad avere la certezza che non può esserci in alcun modo una possibilità di effettuare un cambio di tale dato ad ogni colpo di clock da quando è stata scatenata la procedura di acquisizione. Quindi, questa cover ci porta ad avere un ulteriore sicurezza della condizione di cui abbiamo bisogno per poter identificare la violazione CDC come falsa negativa.

- Grazie quindi alla verifica formale della cover appena riportata ed al risultato ottenuto, ovvero “Fallita”, è possibile garantire che non può avvenire in alcun modo una commutazione continua dei dati “D1[1:0]”, “D2[1:0]” durante la procedura di acquisizione. Dunque, si ottiene un’ulteriore conferma della robustezza del protocollo di acquisizione e che non vi è alcuna possibilità di avere una violazione dei tempi di setup/hold dei registri “D1_latched[1:0]” e “D2_latched [1:0]”.

10.2.3 Verifica formale del waiver effettuato riguardante Reset Configurations

Nella seguente sezione si procede con la verifica formale del waiver inerente al caso CDC “Reset configurations” spiegato nella sezione 6.1.4. Al fine di avere una migliore comprensione dei procedimenti attuati, viene riportata la struttura di sincronizzazione in Figura 10-7.

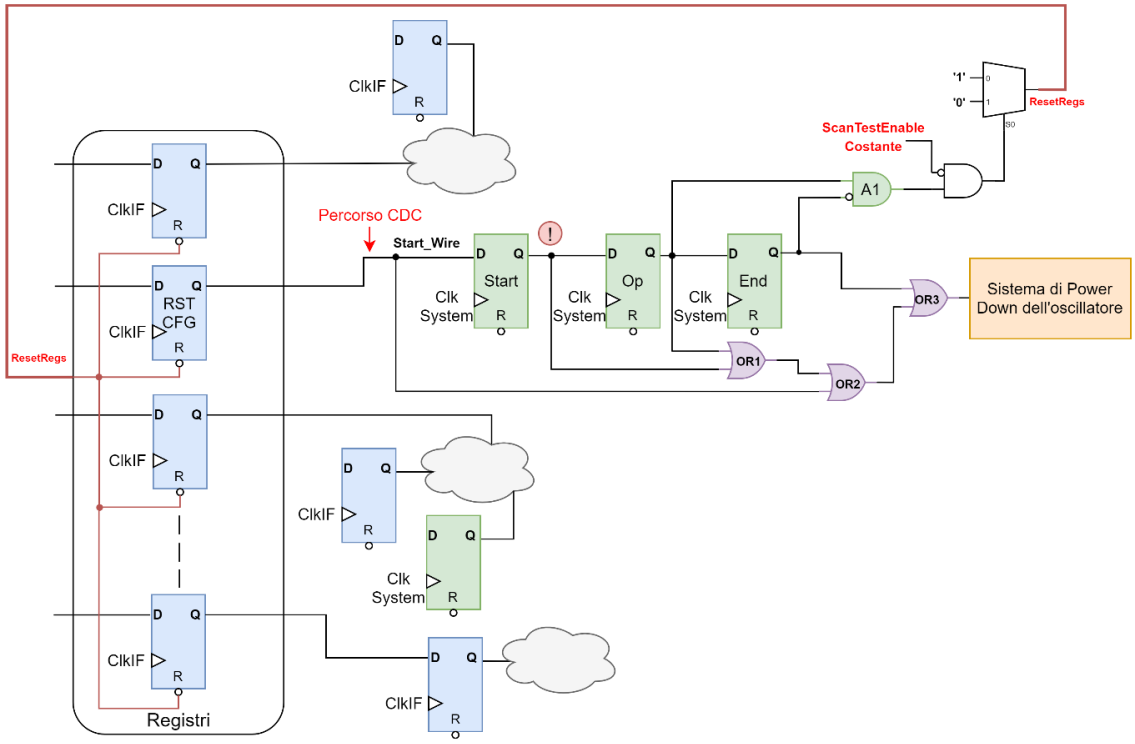

Figura 10-7: Struttura della violazione Reset Configurations

- **Descrizione del comportamento che si vuole verificare:**

Come già anticipato, si procede con il validare il corretto funzionamento della procedura "Reset Configurations" e con il verificare che sia rispettata la condizione in cui il FF "RST_CFG" deve rimanere alto per il tempo necessario al corretto campionamento da parte del FF "Start", appartenente al dominio di destinazione. Viene anche validato che il FF "RST_CFG" viene auto-resettato da parte del dominio di destinazione. Viene riportato il codice SVA al fine di validare tale caso. Si procede con la descrizione dei passaggi e con la spiegazione del loro significato.

- **Descrizione dei passaggi dell'asserzioni:**

Prima asserzione: Verifica della partenza della procedura del "reset configurations" effettuando la scrittura nel FF "RST_CFG" e controllando la sua propagazione verso l'ingresso del FF "Start". Con tale asserzione stiamo asserendo che ad ogni fronte di discesa del clock di interfaccia, "@(negedge Clock_Interface)", se il FF "RST_CFG" è al valore logico "1" allora nello stesso ciclo, l'operatore "|->", il dato in ingresso del FF "Start" definito "Start_wire" deve essere anche lui ad "1". Viene rappresentata tale asserzione nel diagramma di timing riportato in Figura 10-8.

1) *Assert {@(negedge Clock_Interface)* RST_CFG *|-> **(1)** Start_Wire } -name Avvio*

Seconda asserzione: Attraverso tale asserzione viene verificata la prosecuzione della procedura del "reset configurations" nel dominio di clock di sistema, ovvero quello di destinazione. L'asserzione verifica che nel momento in cui avviene una transizione 0-1 del bit "Start_wire" sul fronte di salita del clock di sistema, ci si aspetta che nello stesso colpo di clock il FF "Start" sia al valore logico "0". Successivamente a questo primo evento, l'asserzione non è terminata, ma si procede col verificare cosa avviene al colpo di clock successivo indicato con l'operatore "##1". Quindi, nel secondo colpo di clock si vuole verificare che "Start wire" deve essere al valore logico "1" e al tempo stesso il FF "Start" deve essere commutato al valore logico "1". Riprendendo la Figura 10-7, è possibile notare che al prossimo colpo di clock deve essere resettato il FF "RST_CFG" e che quindi l'uscita "Start Wire" deve essere riportata al valore logico "0". Ecco quindi descritto cosa avviene al terzo colpo di clock, ovvero al secondo "##1". Infine, si procede col verificare che al colpo di clock successivo la procedura sia terminata e che quindi il FF "Start" campioni il valore logico "0".

2) *Assert {@(posedge Clock_di_Sistema) $rose(Start_Wire) |-> **(2.0)** ~Start ##1 **(2.1)** Start_Wire && Start ##1 **(2.2)** ~ Start_Wire && Start ##1 **(2.3)** ~Start_Wire && ~ Start } -name CatenaReset*

Con questa asserzione viene anche validata la falsa negatività della violazione CDC. Infatti, è possibile notare che viene verificato che il bit "Start_Wire" rimane al valore logico stabile "1" per il tempo necessario al corretto campionamento da parte del FF "Start", ovvero il FF "Start" ha la possibilità di campionare al secondo colpo di clock di sistema nuovamente il valore del bit "Start_Wire", preveniente dal clock di interfaccia. Viene quindi validata la spiegazione riportata nella sezione 6.1.4 e viene riportato il diagramma di timing per illustrare l'asserzione nella Figura 10-8.

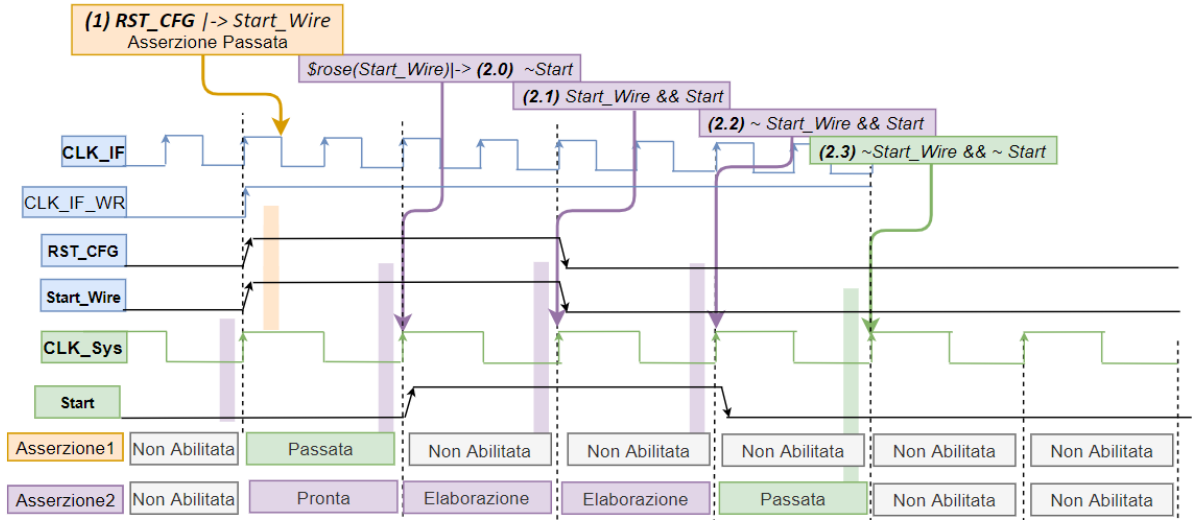

Figura 10-8: Prima e seconda asserzione per violazione reset configurations

Terza asserzione: Viene riportata tale asserzione al fine di verificare tutti i passaggi di tutti i registri presenti all'interno della procedura di reset in questione. Si può anche notare il collegamento della seguente asserzione con la precedente, in quanto si prende in considerazione il momento in cui vi è la transizione 0-1 del FF "Start". È anche possibile notare come viene controllata in questa asserzione che effettivamente sia stato resettato il FF "RST_CFG" nello stesso colpo di clock in cui l'uscita del mux "ResetRegs" è commutata al valore logico "0".

3) *Assert { @(posedge clock sistema) $rose(Start) |-> ~Op && ~End && ResetRegs && RST_CFG ##1 Start && Op && ~End && ~ResetRegs && ~Reg RST_CFG ##1 ~Start && Op && End && ResetRegs && ~ RST_CFG ##1 ~Start && ~Op && End && ResetRegs && ~Reg RST_CFG ##1 ~Start && ~Op && ~End && ResetRegs && ~ RST_CFG } -name ProceduraDiReset_TuttiGliStep*

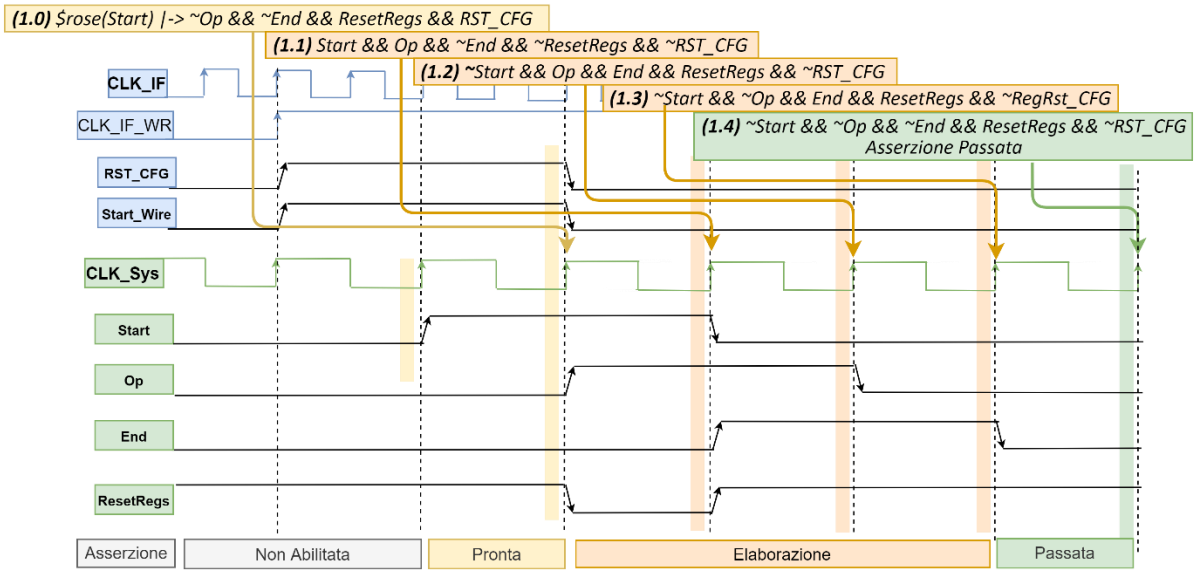

Figura 10-9: Terza asserzione per violazione reset configurations

Quarta asserzione: L'ultima asserzione viene effettuata al fine di verificare che tutti i registri di configurazione vengano correttamente resettati al loro valore di reset nel momento in cui avviene la commutazione allo "0" logico dell'uscita del mux "ResetRegs". Non vengono riportati tutti gli effettivi

registri di configurazione resettati, essendo questi molteplici, ma si prende in considerazione solo una parte dell'asserzione per poter illustrare il suo obiettivo. In questo caso si è preso in considerazione l'evento in cui il bit "Op" e "End" sono rispettivamente ad "1" e "0". Infatti, facendo riferimento alla Figura 6-12, è possibile notare che solo quando è rispettata questa condizione viene rilasciato l'impulso di reset.

4) *Assert {@(posedge clock sistema) Op && ~End |-> ~Reg1 && ~Reg2[5 downto 0] && ~Reg3 && ~Reg4 && Interrupt_CfgReg[7 downto 0] && ……..} -name ResetRegConfigCompletato*

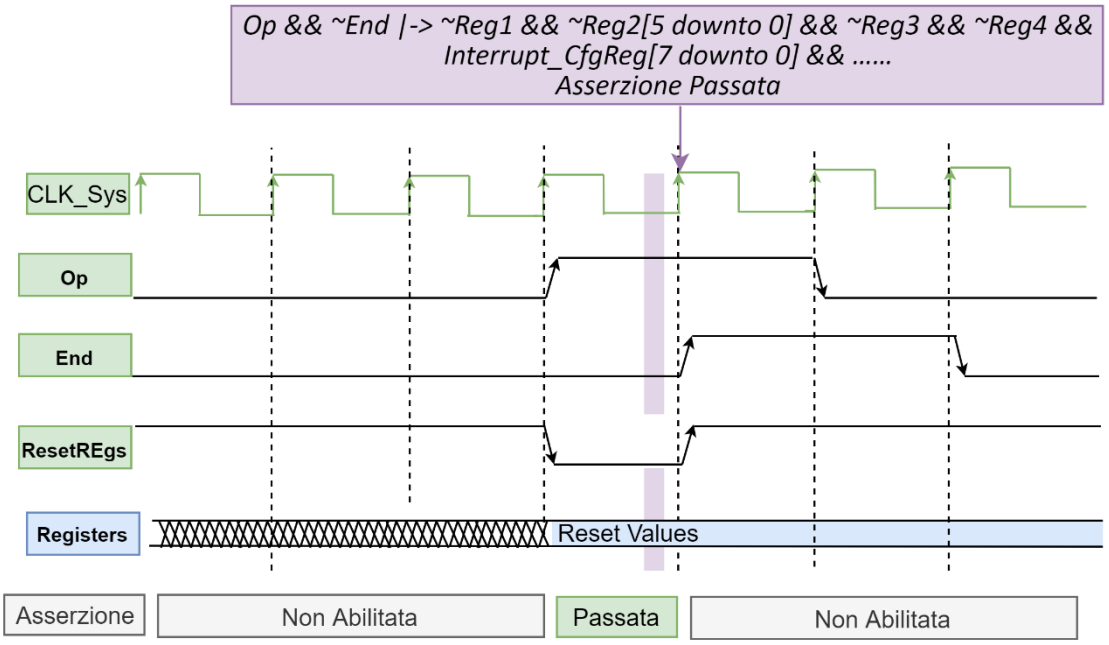

Figura 10-10: Quarta asserzione per violazione reset configurations

10.3 Conclusioni analisi formale

È stato quindi raggiunto l'obiettivo di validazione formale dei waivers effettuati, ottenendo dunque la conferma che sia rispettato da parte del design il comportamento aspettato e desiderato per tali protocolli di sincronizzazione. Quindi, con queste validazioni con approccio formale è possibile garantire il corretto comportamento delle funzionalità del design legate alle strutture incontrate. È dunque possibile procedere con l'aggiungere, al flusso utilizzato in questa sezione, il flusso di iniezione di metastabilità con approccio formale e in simulazione dinamica, con lo scopo di avere una maggiore conferma del corretto comportamento delle funzionalità del dispositivo.

11 Metastability Injection

Si procede adesso con l'introdurre l'iniezione della metastabilità all'interno del flusso progettato, col fine di validare funzionalmente il design anche in condizioni di metastabilità ed in presenza di glitch.

La verifica formale ed anche la verifica basata sulla simulazione dinamica "standard" non prendono in considerazione gli effetti della metastabilità durante la verifica funzionale del DUT e durante la validazione delle proprietà e asserzioni che descrivono le funzionalità del design. Sorge la necessità di validare il comportamento del design anche in presenza della metastabilità, in quanto eventuali cambiamenti del comportamento a causa dei suoi effetti non possono essere considerati durante la fase di verifica "standard".

Si distinguono principalmente due tipologie di modelli di iniezione:

1) L'iniezione della metastabilità attraverso l'approccio formale e dunque attraverso l'utilizzo di engine formali;

2) L'iniezione della metastabilità in simulazione dinamica durante la verifica funzionale;

- **Flusso eseguito per l'iniezione della metastabilità**

Viene adesso riportato e descritto il flusso da seguire per poter effettuare l'iniezione della metastabilità e poter verificare l'effetto funzionale di una possibile metastabilità. Il flusso è schematizzato nella Figura 11-1.

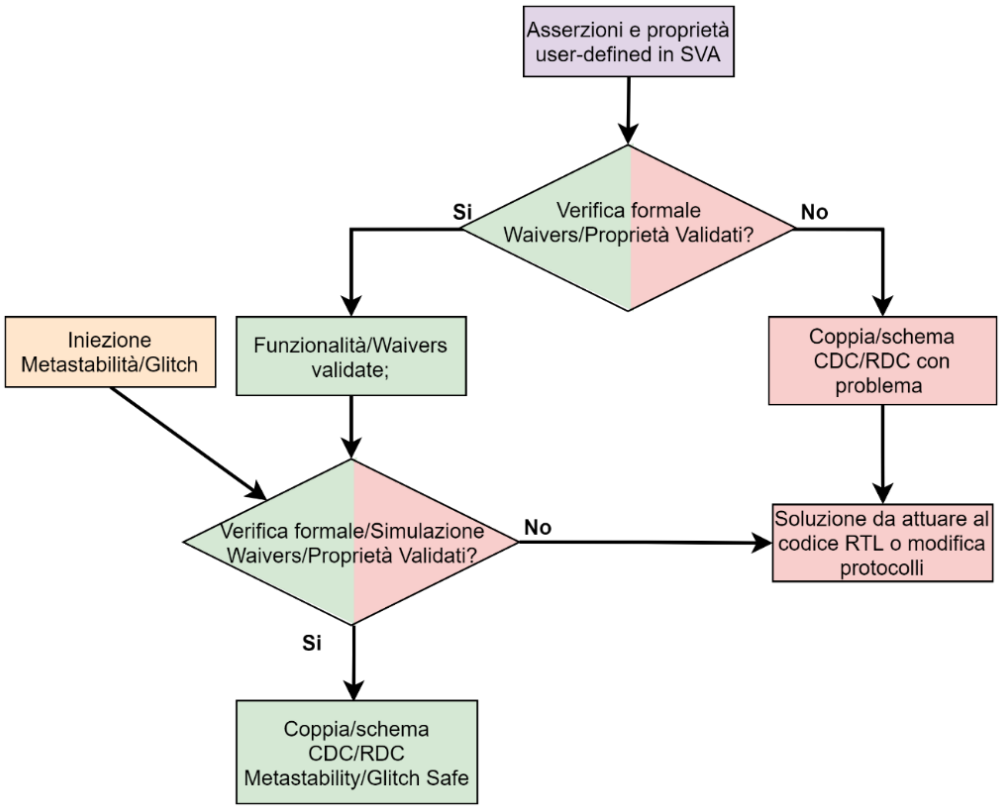

Figura 11-1: Flusso iniezione metastabilità

Come è possibile notare, dopo aver eseguito il flusso CDC strutturale e descritto le proprietà, assunzioni e asserzioni atte alla validazione dei protocolli di sincronizzazione adottati nel design, si procede con l'effettuare la validazione di quest'ultimi attraverso la verifica formale per poter assicurare che le funzionalità del design funzionino correttamente e che rispettino il comportamento aspettato. Dopo questo step, si procede quindi con l'aggiungere al flusso l'iniezione della metastabilità al fine di avere un ulteriore validazione che le proprietà e asserzioni descritte rispettino ancora il comportamento aspettato anche in presenza della metastabilità e in presenza di glitch.

In tale flusso CDC è anche necessario sottolineare che la verifica formale e la simulazione RTL lavorano assieme per raggiungere l'obiettivo di realizzare una sempre più efficiente forma di validazione delle funzionalità di un dispositivo come anche raggiungere l'obiettivo di realizzare un flusso CDC-Clean RTL Signoff. Una delle principali forme di necessità di collaborazione tra la verifica formale e la simulazione risiede nella possibilità di iniettare glitch combinatori nel design sotto analisi. L'iniezione di glitch consente di raggiungere l'obiettivo di validare che i percorsi CDC e RDC presenti all'interno del design siano "glitch-free" e tale obiettivo può essere raggiunto grazie all'iniezione in simulazione RTL mentre non è possibile raggiungerlo attraverso la verifica formale.

11.1 Iniezione della metastabilità con approccio formale

Essenzialmente, attraverso l'utilizzo di engine formali è possibile modellare gli effetti della metastabilità nel cono di logica presente in ogni percorso CDC che viene identificato nelle proprietà e asserzioni user-defined.

Per poter descrivere come avviene la modellazione della metastabilità attraverso l'engine formale, viene preso in considerazione un comune caso di violazione CDC e viene riportato il comportamento adottato in presenza dell'iniezione formale, nella Figura 11-2:

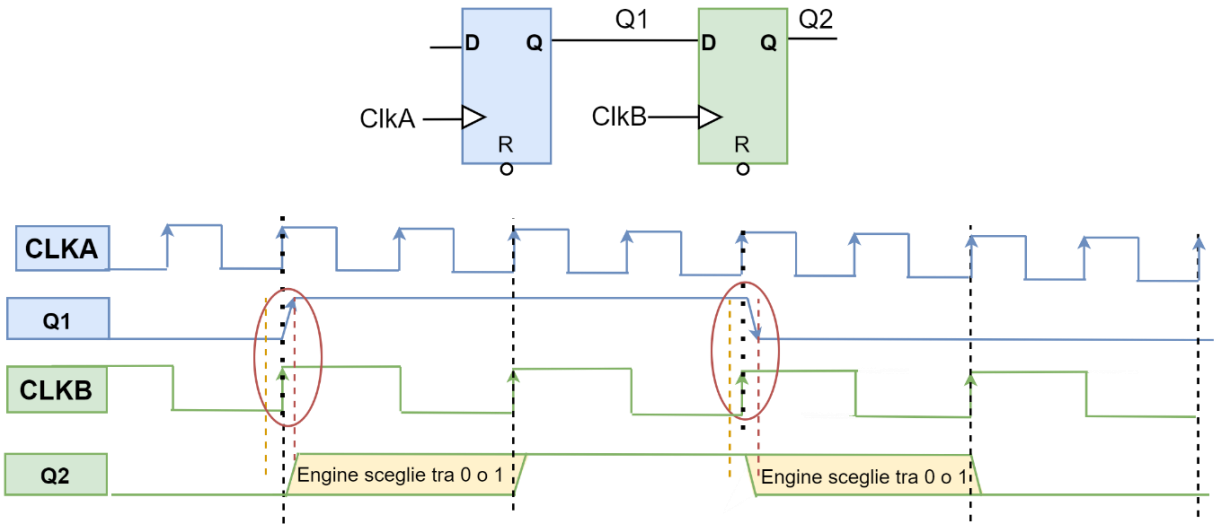

Figura 11-2: Iniezione della metastabilità con engine formale

Nel momento in cui viene applicata l'iniezione della metastabilità nel caso in questione, dove abbiamo un percorso CDC identificato dal passaggio tra il dominio di clock A e il dominio di clock B, il modello iniettato consente di inserire un colpo di clock di incertezza nel FlipFlop del dominio di destinazione.

Dunque, l'engine formale ha essenzialmente la possibilità di far sì che il FlipFlop di ricezione campioni il dato precedente oppure che campioni il dato nuovo. Questo avviene nel momento in cui il dato Q1 effettua una commutazione in prossimità del fronte attivo del clock di destinazione. Quindi, l'engine formale agisce sull'uscita Q2 decidendo liberamente se riportare in uscita uno "0" logico, dunque il dato precedente per il caso in questione, o "1" logico, dunque il dato nuovo per il caso in questione.

L'engine formale procede con il modellare bit per bit l'effetto della metastabilità. Nel momento in cui si prende in considerazione un solo dominio di trasmissione, da cui si ottengono diversi percorsi CDC verso multipli domini di destinazione asincroni, l'effetto della metastabilità viene modellata separatamente per ognuno dei percorsi CDC e dunque non viene applicato un solo effetto ma sempre una modellazione bit per bit.

Per poter meglio comprendere come può influire la modellazione della metastabilità con approccio formale e come può risaltare problemi CDC, si prende in considerazione l'esempio riportato nella Figura 11-3 .

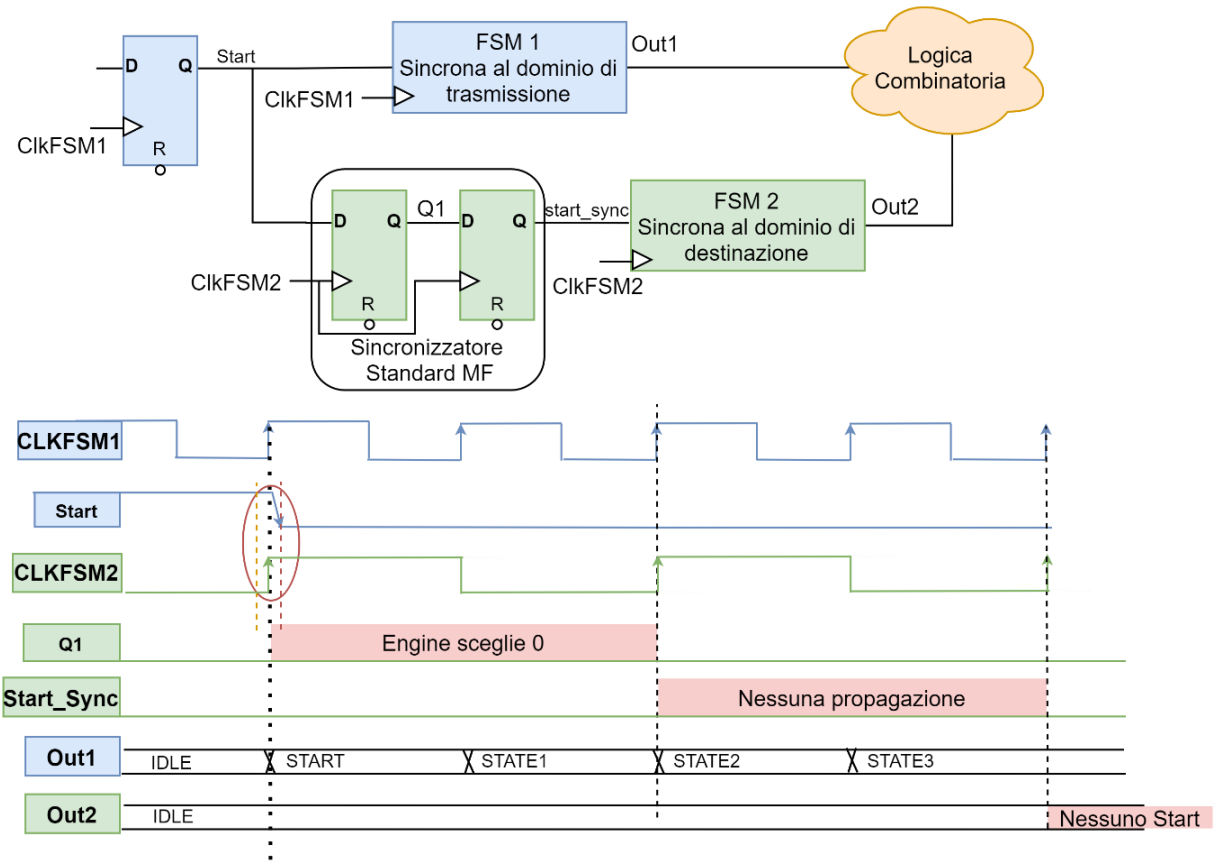

Figura 11-3: Caso in cui l'iniezione della metastabilità risalta un problema CDC

Nel caso riportato sono presenti due FSM legate da un unico segnale di Start ed appartenenti a due domini di clock differenti, rispettivamente al dominio di clock A e al dominio di clock B. Il segnale di enable è sincrono al dominio di clock della FSM "1" mentre deve essere sincronizzato per poter abilitare la FSM "2", essendo quest'ultima appartenente al dominio di clock B.

Dal punto di vista strutturale non vengono riportati errori, in quanto è presente una struttura di sincronizzazione standard Multi-Flop, in questo caso caratterizzata da due FF ed implementata per poter sincronizzare il segnale di start. Attraverso l'iniezione della metastabilità viene però riportata una violazione CDC. Come è possibile notare, il segnale di start effettua una commutazione in

prossimità del fronte attivo del clock di destinazione e dunque l'engine formale procede con l'effettuare la scelta tra l'avere il campionamento del nuovo segnale di start o procedere con il campionamento del valore precedente del segnale di start. Nel caso specifico viene iniettata la metastabilità al fine di non avere il campionamento del nuovo valore del segnale di start e questo si riflette nel non abilitare la FSM 2. Dunque, la FSM 1 ha ovviamente iniziato con il proseguire con la sua operatività mentre la FSM 2 non parte dopo i due colpi di clock aspettati. La violazione, quindi, non è basata sull'assenza di uno schema di sincronizzazione ma è basata su un errore del protocollo adottato, che in questo caso, in presenza di metastabilità, porta a non rispettare il comportamento aspettato.

Nella spiegazione appena effettuata non è stata però citata l'iniezione di glitch in percorsi CDC tramite l'approccio formale, questo perché la validazione del corretto funzionamento dei protocolli adottati anche in presenza di glitch avviene grazie alla loro iniezione in simulazione dinamica.

11.2 Metastability injection in simulazione

La metastability injection simulation è una simulazione dinamica dove sono prese in considerazione le commutazioni di dati e segnali tenendo conto delle eventuali violazioni dei tempi di setup/hold dei FF.

Per poter procedere con tale simulazione dinamica è necessario avere a disposizione il codice RTL del design ed i testbench e testcase che validano il corretto comportamento delle funzionalità del DUT. I clock e i reset presenti nel design sono le uniche informazioni CDC che devono essere dichiarate per eseguire tale iniezione di metastabilità e glitch. L'obiettivo che si vuole raggiungere attraverso tale flusso di iniezione in simulazione dinamica è quello di prendere in considerazione le simulazioni dinamiche esistenti e verificare queste siano ancora valide anche in presenza della metastabilità e glitch.

La principale caratteristica su cui bisogna porre l'attenzione è incentrata sul concetto di eseguire l'iniezione della metastabilità su tutti le coppie CDC presenti all'interno del design sotto analisi e non solo sui percorsi in cui sono presenti sincronizzatori standard. Sono dunque prese in considerazione tutte le coppie CDC a prescindere dalle informazioni descritte precedentemente nel corso dell'analisi. Quindi, segnali non sincronizzati, statici o costanti e tutte le altre informazioni strutturali CDC non sono considerate, ma ci si attiene esclusivamente ai comportamenti ottenuti dalla simulazione, non avendo in alcun modo una sua eventuale corruzione.

Durante la simulazione viene iniettata la metastabilità nell'uscita "Q" dei FF di destinazione esclusivamente nel momento in cui vi è una commutazione dei dati/segnali che portano ad avere una violazione dei tempi di setup e dei tempi di hold.

Dunque, al fine di "modellare" la metastabilità, nel momento in cui avviene la violazione dei due tempi viene impostato un valore casuale. Quest'ultimo viene scelto tra il valore precedente e il valore attuale ed anche effettuando una eventuale combinazione dei due. Dunque, nel percorso CDC in questione viene modellata un'incertezza pari ad un colpo di clock e nel momento in cui avviene effettivamente un errore funzionale dovuto all'incertezza iniettata, si conferma di avere una vera e propria violazione CDC. Per poter meglio comprendere come viene modellato l'effetto della metastabilità in simulazione dinamica, si riporta l'esempio delle violazioni dei tempi setup e di hold con e senza iniezione della metastabilità in Figura 11-4.

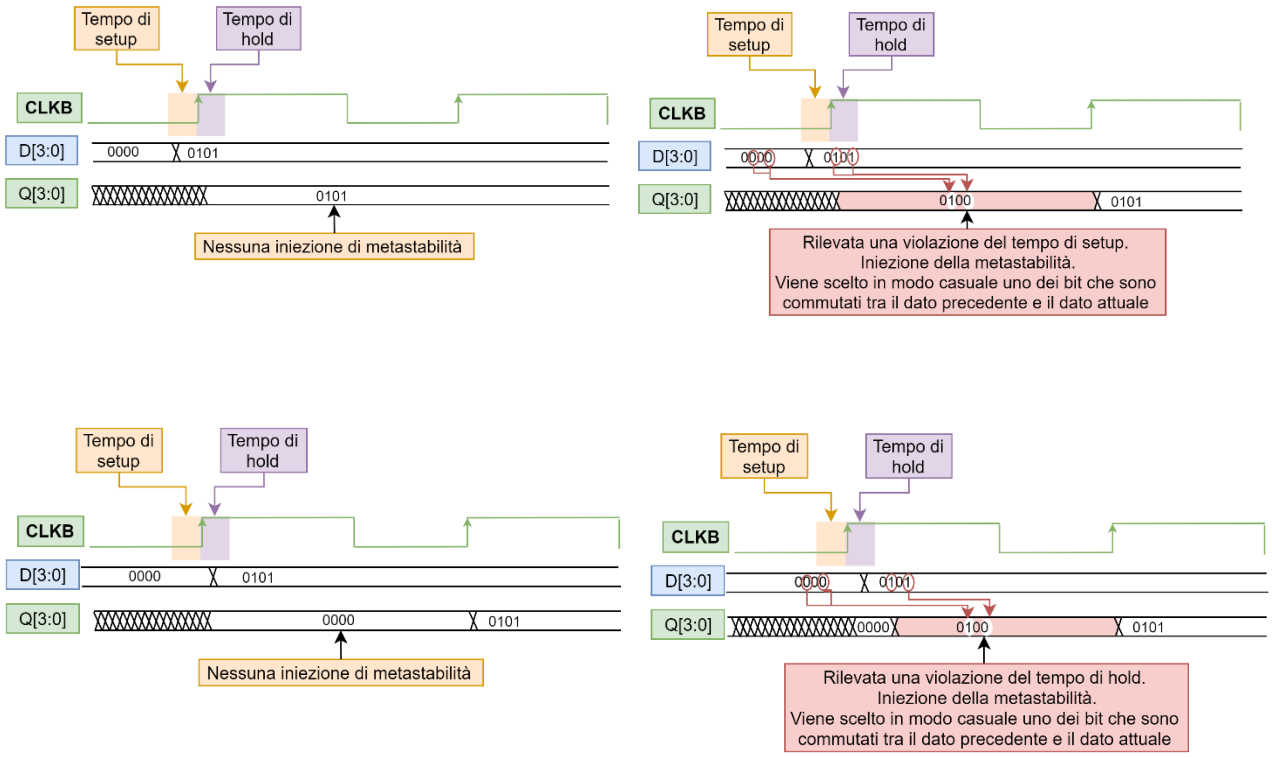

Figura 11-4: Iniezione della metastabilità in simulazione

Viene preso in considerazione nella parte superiore a sinistra un caso di violazione del tempo di setup da parte del dato D[3:0] senza considerare l'iniezione della metastabilità. In questo caso avviene l'acquisizione del dato e non viene considerata la sua violazione. Nella parte superiore a destra, invece, si prende in considerazione l'iniezione della metastabilità in quanto violato il tempo di setup e viene riportata sull'uscita Q[3:0] un dato errato, risultato dalla combinazione dei bit che sono cambiati tra il dato precedente e il dato nuovo. Essenzialmente, può essere scelto un valore casuale tra i seguenti: 0000, 0001, 0100, 0101. Dunque, viene generato un valore casuale prendendo in considerazione esclusivamente i bit che sono cambiati tra il dato precedente e il dato attuale.

Nella parte inferiore viene invece preso lo stesso discorso, ma in questo caso viene portato un esempio di violazione del tempo di hold. Ovviamente, è necessario definire manualmente una "Time Window", ovvero una finestra temporale con cui si definisce l'intervallo di tempo costituito dai tempi di setup e di hold.

Bisogna prestare particolare attenzione alla definizione della finestra temporale rappresentante i tempi di setup e di hold. Finestre temporali troppo "larghe" possono portare a continue violazioni dei tempi di setup e di hold "false", dovuta ad una troppo pessimistica rappresentazione dei tempi in questione. Una finestra temporale troppo stretta può invece portare a mascherare eventuali errori CDC effettivi. Vi è quindi la necessità di effettuare una ottimizzazione della finestra temporale. Vi è la possibilità di gestire più finestre temporali e non averne solamente un'unica per tutti i FF presenti nel DUT.

Ovviamente, durante l'iniezione della metastabilità non sono presi in considerazione segnali statici o costanti che non possono effettuare commutazioni durante la fase funzionale del dispositivo e non possono dunque portare a violazioni dei tempi di setup e di hold. Non sono considerati perché da simulazione risultano essere statici o costanti e non perché sono stati descritti come tali dalle informazioni strutturali CDC effettuate nel corso dell'analisi.

- **Iniezione di glitch combinatori**

Come citato precedentemente, l'iniezione della metastabilità con approccio formale non prende in considerazione l'eventuale formazione di glitch dovuti alla presenza di logica combinatoria lungo i percorsi CDC. Tale mancanza viene colmata con l'iniezione della metastabilità con l'approccio in simulazione dinamica, dove è possibile introdurre glitch ed analizzare il comportamento dei protocolli adottati in funzione di tali impulsi spuri. La presenza di logica combinatoria nei percorsi CDC non è consigliata, questo perché nasce la possibilità di generare impulsi spuri che possono essere campionati nei domini di destinazione ed è dunque necessario avere una conferma che i percorsi CDC in questione possano essere considerati "glitch-free".

Nella simulazione dinamica RTL "0-delay", ovvero zero ritardi, non è presa in considerazione che la commutazione contemporanea di segnali di ingresso di porte logiche possa portare alla formazione non volontaria di glitch. Dunque, potenziali violazioni legate ad impulsi spuri possono essere mascherate durante tali simulazioni RTL e potrebbero essere risaltate durante stadi di produzione successivi, come ad esempio durante la simulazione post-layout. Dunque, nella simulazione RTL zero delay non è riportata la possibilità di avere un campionamento di impulsi spuri nei domini di destinazione come anche non viene riportata la possibilità che tali impulsi spuri possono portare ad una violazione dei tempi di setup e di hold dei FF appartenenti ai domini di clock di destinazione.

Per poter ridurre i rischi legati alla possibile generazione di tali impulsi spuri ed al loro campionamento, viene introdotta nella simulazione dinamica un'ulteriore possibilità di iniezione pessimistica, ovvero la "Combinational Glitch Injection". Attraverso tale iniezione viene attuata una generazione randomica degli ingressi delle porte logiche presenti nei percorsi delle coppie CDC, al fine di creare l'opportunità di generare un impulso spurio. Dunque, il tool procede con il generare casualmente una combinazione dei segnali di ingresso al percorso CDC con l'obiettivo di ottenere una generazione ed un campionamento dei glitch.

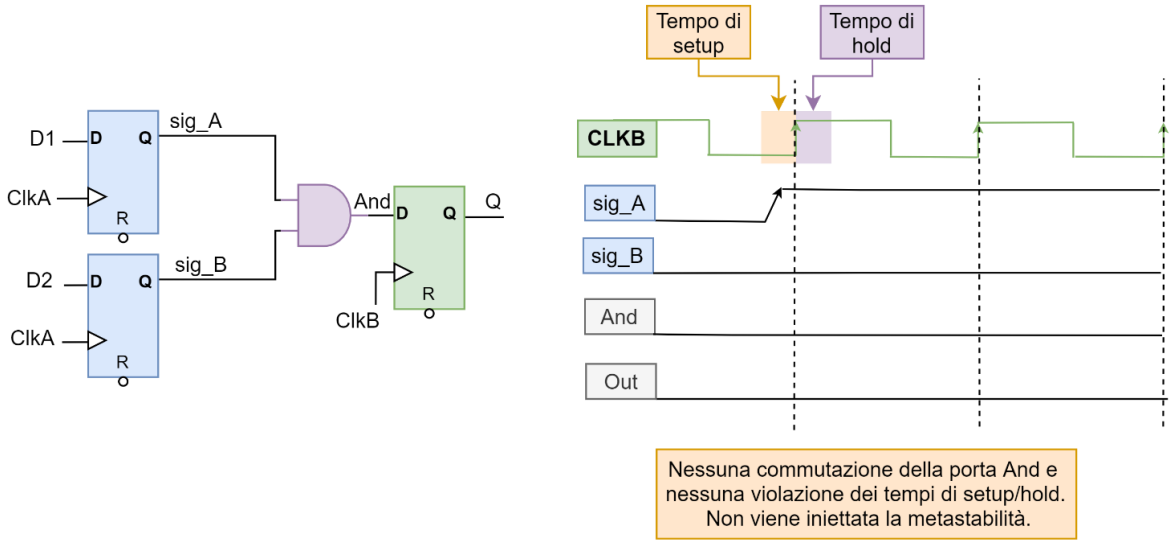

Figura 11-5: Nessuna iniezione di glitch o metastabilità

Con l'iniezione della metastabilità in simulazione dinamica, diventa quindi possibile rilevare la formazione di glitch nei percorsi CDC come anche diventa possibile modellare il loro comportamento. Viene riportato un esempio di iniezione di glitch nel percorso CDC in Figura 11-6. Nel caso in questione, è presente una porta logica AND i cui ingressi provengono dal dominio di clock A e la sua uscita si propaga verso il dominio di clock B. Come è possibile notare nel diagramma di timing della Figura 11-5, la transizione del segnale di ingresso "Sig_A" non porta ad avere nessuna commutazione del segnale di uscita della porta AND ed in questo caso non avviene nessun tipo di modellazione di glitch come, ovviamente, non viene riportata nessuna iniezione della metastabilità.

Invece, nel diagramma di timing presente a sinistra nella Figura 11-6, viene riportata una transizione dei segnali di ingresso "Sig_A" e "Sig_B" tale da avere una commutazione del segnale di uscita della porta AND in prossimità del fronte attivo del clock di destinazione B, portando quindi ad avere una violazione del tempo di setup del FF di ricezione. In questo caso, non viene modellata la presenza di un glitch, in quanto non vi è una transizione dei segnali tale da avere una sua formazione. Viene invece iniettata la metastabilità sull'uscita Q del FF di ricezione, perché vi è stata una violazione del tempo di setup, e dunque viene selezionato in maniera casuale il dato precedente o il dato attuale presente in ingresso del FF.

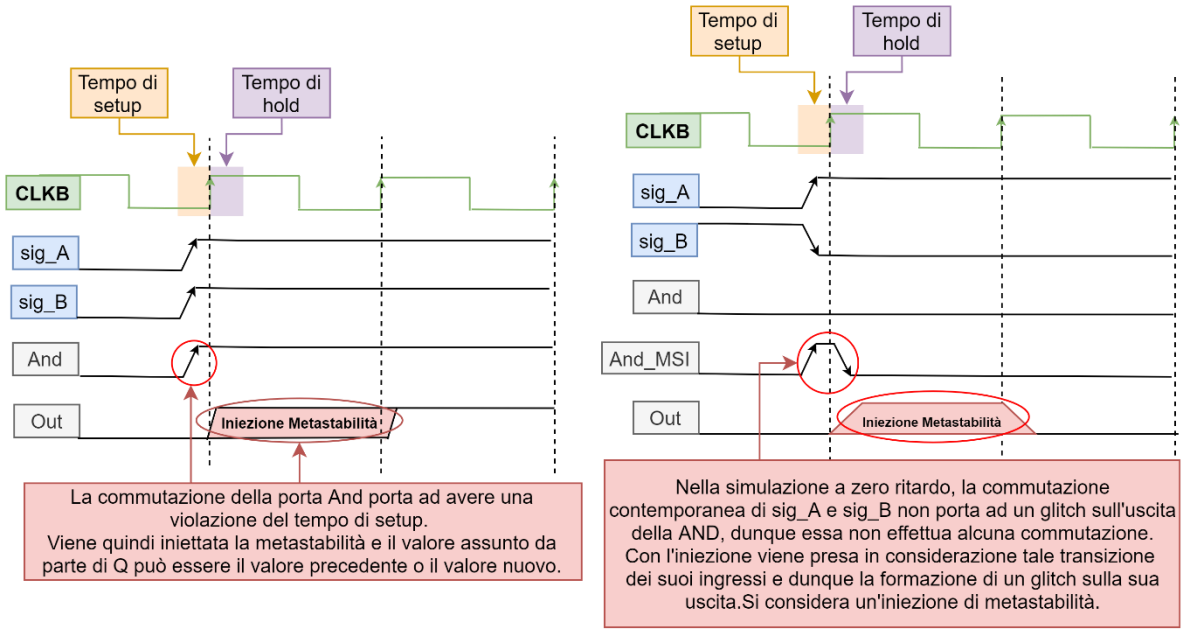

Figura 11-6: A sinistra semplice iniezione della metastabilità per violazione del tempo di setup; A destra iniezione della metastabilità e di glitch a causa della commutazione contemporanea dei segnali sig_A e sig_B nel tempo di setup;

Infine, nella parte a destra della Figura 11-6, viene riportato un caso tale per cui la transizione effettuata dai segnali in ingresso alla porta And possono portare ad avere la formazione di un glitch. In questo caso, la simulazione dinamica a zero ritardo non riporta tale glitch e non segnala quindi alcun problema CDC. Invece, attraverso l'iniezione della metastabilità viene iniettata la presenza di tale glitch e viene dunque presa in considerazione l'opportunità di iniettare la metastabilità nell'uscita Q del FF di ricezione, questo perché il glitch porta ad avere una violazione del tempo di setup.

12 Risultati ottenuti dall' iniezione della metastabilità

Dopo aver quindi introdotto come viene affrontata l'iniezione della metastabilità, si procede con l'integrarla nel flusso CDC-RDC al fine di validare che i protocolli e le funzionalità del dispositivo siano ancora rispettati in funzione della metastabilità e glitch.

12.1 Iniezione della metastabilità con approccio formale

12.1.1 Iniezione formale – Caso Reset Configurations

Viene adesso riportato il caso "Reset Configurations" introdotto nella sezione 6.1.4 e validato formalmente nella sezione 10.2.3.

La prima asserzione, validata formalmente ed illustrata nel dettaglio nella sezione 10.2.3, è la seguente e viene riportato il relativo diagramma di timing validato in Figura 12-1.

1) *Assert {@(posedge Clock_di_Sistema) $rose(Start_Wire) |-> **(2.0)** ~Start ##1 **(2.1)** Start_Wire && Start ##1 **(2.2)** ~ Start_Wire && Start ##1 **(2.3)** ~Start_Wire && ~ Start } -name CatenaReset. (Riportato in viola nel diagramma di timing sottostante).*

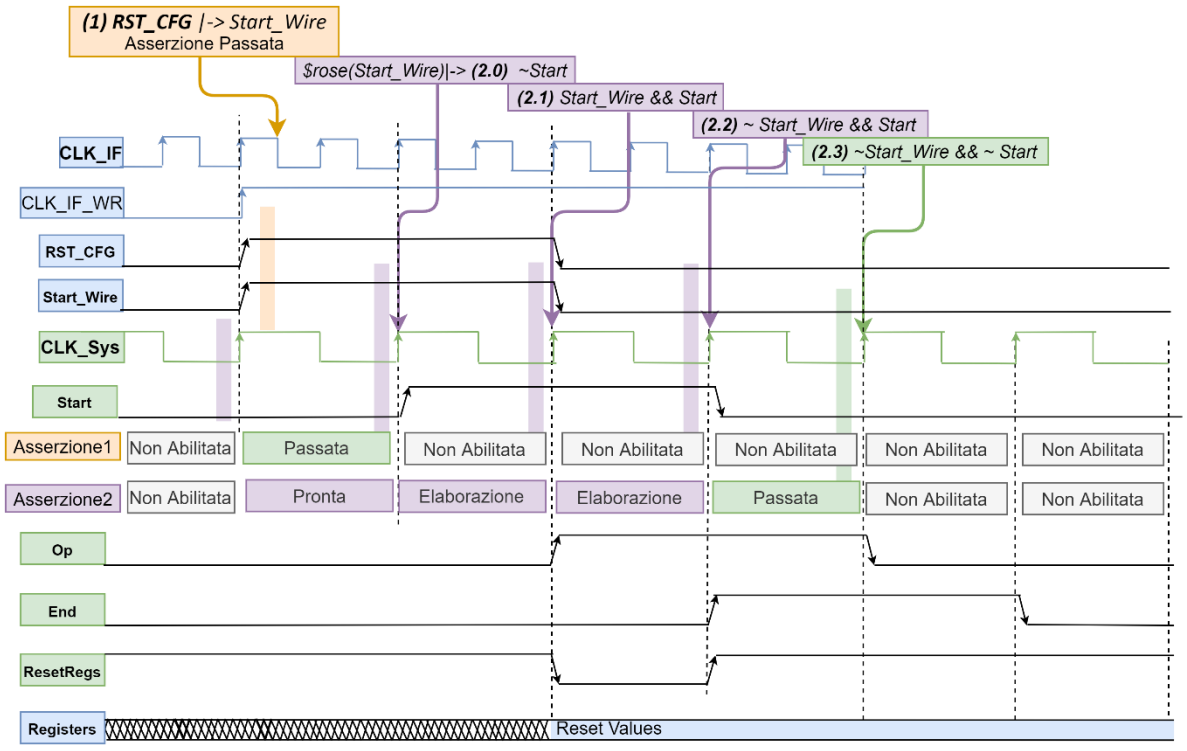

Figura 12-1: Diagramma di timing senza iniezione della metastabilità in formale

Tale asserzione validata formalmente non è invece validata attraverso l'iniezione della metastabilità, questo è dovuto per via della scrittura dell'asserzione che descrive colpo di clock per colpo di clock il

comportamento. Viene riportato nella Figura 12-2 il motivo per il quale tale asserzione fallisce in funzione della metastabilità.

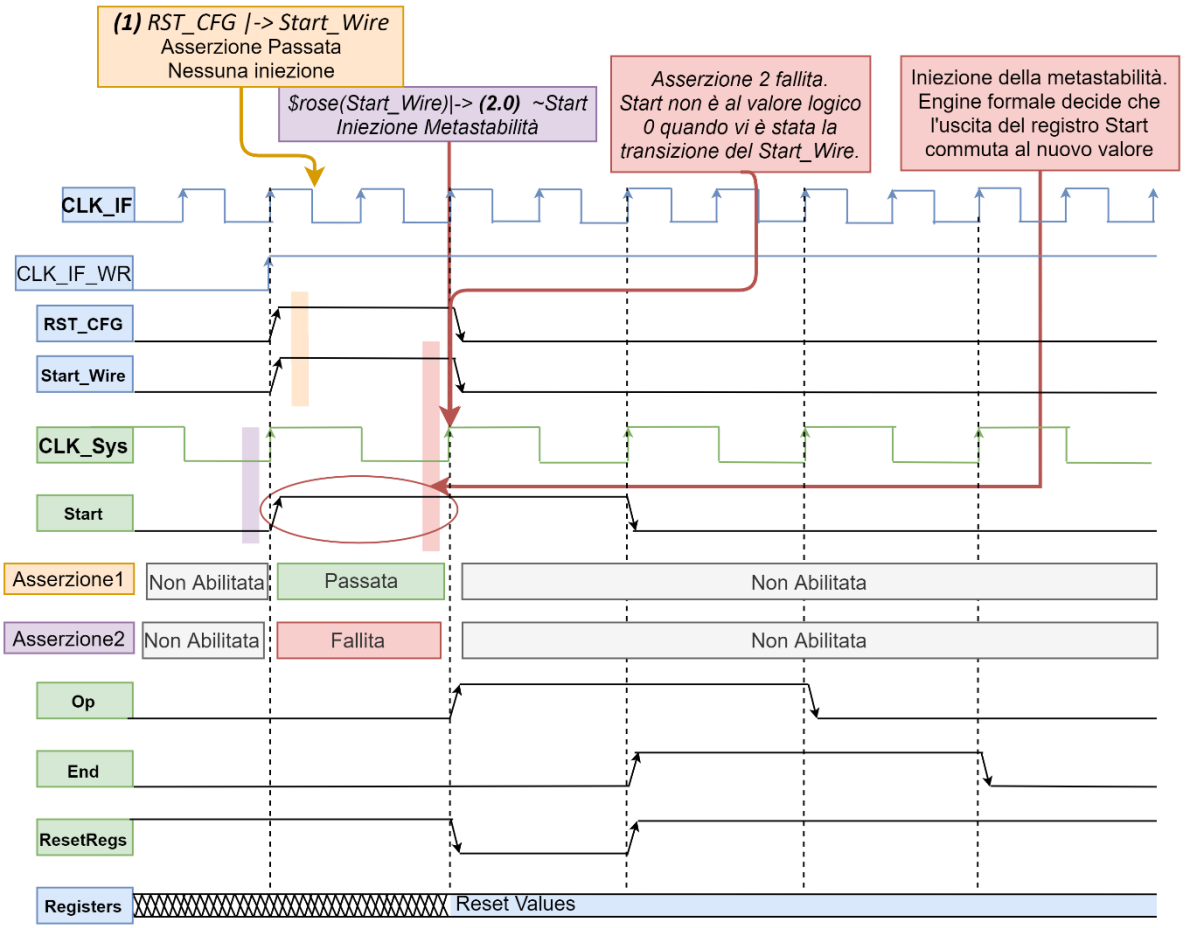

Figura 12-2: Diagramma di timing con iniezione della metastabilità in formale

Come è possibile notare nella Figura 12-2, essendo presente una violazione del tempo di setup del FF Start appartenente al dominio di clock di sistema, l'engine formale ha deciso che il FF Start è riuscito ad avere il campionamento corretto del nuovo dato portando dunque ad avere un fallimento dell'asserzione riportata. L'asserzione fallisce in quanto quest'ultima descrive che nel momento in cui avviene una transizione verso il valore logico "1" del segnale "Start wire", allora nello stesso ciclo il valore assunto dall'uscita del segnale "Start" deve essere il valore logico "0". Nel diagramma di timing riportato questo non avviene, in quanto nello stesso momento in cui avviene la transizione vi è stata anche la transizione del segnale "Start".

Ovviamente, tale fallimento con l'iniezione della metastabilità non si riflette nell'essere un problema, in quanto il protocollo viene anticipato semplicemente di un colpo di clock ma viene comunque rispettato ed il reset del FF RST_CFG e dei registri avviene correttamente, semplicemente anticipato di un colpo di clock ma comunque coerente al protocollo aspettato. Il protocollo è ancora valido anche in presenza della metastabilità e per non avere la segnalazione di un problema di metastabilità durante l'analisi formale è stato sufficiente riadattare l'asserzione scritta considerando che la procedura può partire all'interno dello stesso colpo di clock in cui è stato asserito il segnale di "RST CFG" o al colpo di clock successivo a tale evento e che dunque il segnale di Start può essere asserito nello stesso colpo di clock in cui è stato asserito il "RST_CFG" o può essere asserito al colpo di clock successivo.

Assert {@(posedge Clock_di_Sistema) $rose(Start_Wire) |-> #[0:1] Start #1 Op && ~End && ~ResetRegs && ~RST_CFG} -name MSIAware

La terminologia #[0:1] indica semplicemente la presenza di un delay, ovvero che l'evento deve avvenire dopo zero colpi di clock o dopo un colpo di clock.

12.2 Iniezione della metastabilità in simulazione

12.2.1 Glitch Injection in simulazione– PowerDown Memoria Volatile

Nella sezione seguente viene illustrato il risultato ottenuto dall'iniezione della metastabilità e glitch in simulazione riguardante il caso "PowerDown_Mem_Volatile" discusso nella sezione 6.2.1. Viene riportato nella Figura 12-3 lo schema della struttura in analisi presa in considerazione per la simulazione RTL senza e con l'iniezione della metastabilità e glitch.

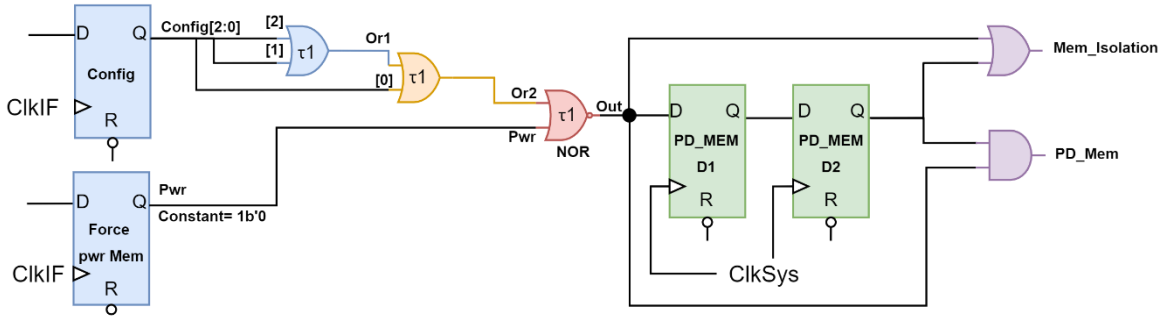

Figura 12-3: Struttura di powerdown della memoria volatile

Come già spiegato, tale struttura è caratterizzata dalla presenza di una logica combinatoria più o meno complessa nel percorso CDC in analisi, percorso caratterizzato dai registri di trasmissione "Config[2:0]", "Force pwr Mem" e il FF di ricezione "PD_MEM D1". Tale logica può portare ad un problema di insorgenza di glitch e vi è dunque la possibilità di avere il campionamento di tale impulso spurio nel dominio di ricezione. Il campionamento non voluto di tale glitch porta all'isolamento non volontario della memoria volatile per un intero colpo di clock di sistema, portando dunque ad un errore funzionale.

Come spiegato nella sezione 11.2, il primo passo per poter effettuare l'iniezione di metastabilità e glitch in simulazione è quello di prendere in considerazione un testcase e relativa simulazione che validino il corretto comportamento delle funzionalità del dispositivo e quindi una simulazione che rispecchi e validi il comportamento aspettato. Dunque, il primo passo è analizzare il comportamento della simulazione RTL "Not metastability aware", ovvero la simulazione RTL a cui non è stata ancora applicata l'iniezione della metastabilità ed effettuare successivamente il confronto con la simulazione "Metastability Aware".

- **Simulazione RTL standard “Not metastability aware”**

Viene riportata nella Figura 12-4 la simulazione RTL che illustra il funzionamento della struttura in questione. Come è possibile notare, i segnali di configurazione “Config[2:0]” effettuano diverse commutazioni consecutivamente come anche avviene un cambiamento simultaneo di più bit dei tre “Config[2:0]” in questione. Quindi, tale simulazione riporta la condizione necessaria affinchè via sia una possibile generazione di un glitch e possibile errore funzionale. È necessario sottolineare che tale simulazione RTL non è ancora soggetta all’iniezione della metastabilità come non è ancora soggetta all’iniezione di glitch.

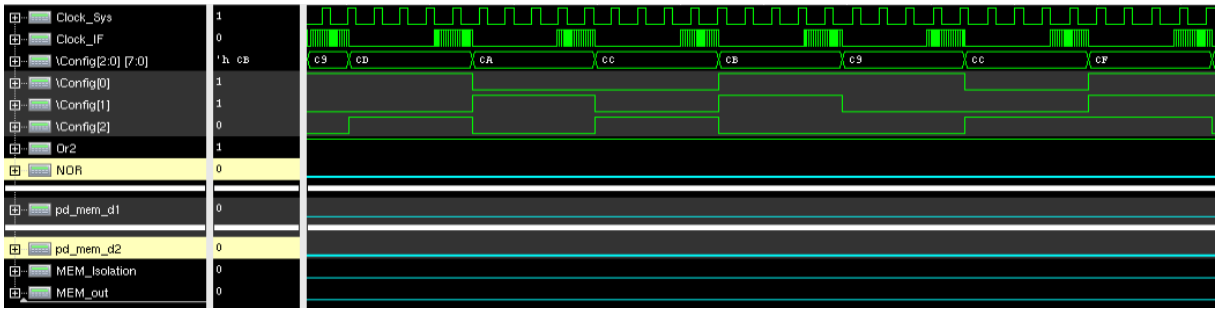

Figura 12-4: Simulazione not Metastability aware no glitch injection

È possibile notare che ad ogni transizione dei tre bit di configurazione il segnale di uscita “or2” rimane fisso al valore logico “1” e l’uscita della porta NOR rimane invece fissato al valore logico “0”. Dunque, il comportamento è quello desiderato ed è quello aspettato dalla simulazione RTL zero-delay, ovvero non vi è alcuna commutazione dei segnali di uscita PD_MEM_out e MEM_isolation ad ogni commutazione diversa da “000” dei tre bit di configurazione “Config[2:0]”. Tale simulazione riporta quindi il comportamento voluto ed il comportamento aspettato. Dunque, adesso è necessario effettuare il confronto di tale simulazione “not metastability aware” con quella “metastability aware” al fine di controllare se tale percorso risulta essere un percorso “glitch-free”.

- **Simulazione RTL “Metastability aware”**

Analizzato il risultato ottenuto ed aspettato dalla simulazione RTL “standard”, viene adesso introdotta l’iniezione della metastabilità e l’iniezione di glitch in simulazione col fine di visualizzare un eventuale campionamento di un glitch. Dunque, si procede con il verificare se viene ottenuto un comportamento errato per via della presenza della logica combinatoria nel percorso CDC, ovvero con il validare se può emergere un errore come quello illustrato nella sezione 6.2.1.

  - **Time window – tempo di setup & hold**

    Per poter procedere con tale simulazione “metastability aware” è però necessario definire la finestra temporale con il quale vengono identificati i tempi di setup e i tempi di hold dei registri coinvolti nei problemi CDC, finestra temporale introdotta nella sezione 11.2 e definita “time window”. Al fine di massimizzare il numero di opportunità in cui è possibile avere una violazione dei tempi di setup/hold dei FF e dunque massimizzare gli eventi di iniezione di metastabilità, è stata selezionata una finestra temporale per i tempi di setup/hold pari al 40% del clock di sistema, ovvero il clock lento. Ovviamente, tale finestra temporale può essere considerata pessimistica ma al tempo stesso efficace per poter per l’appunto massimizzare il numero di eventi in cui è possibile iniettare la metastabilità e verificarne i suoi effetti.

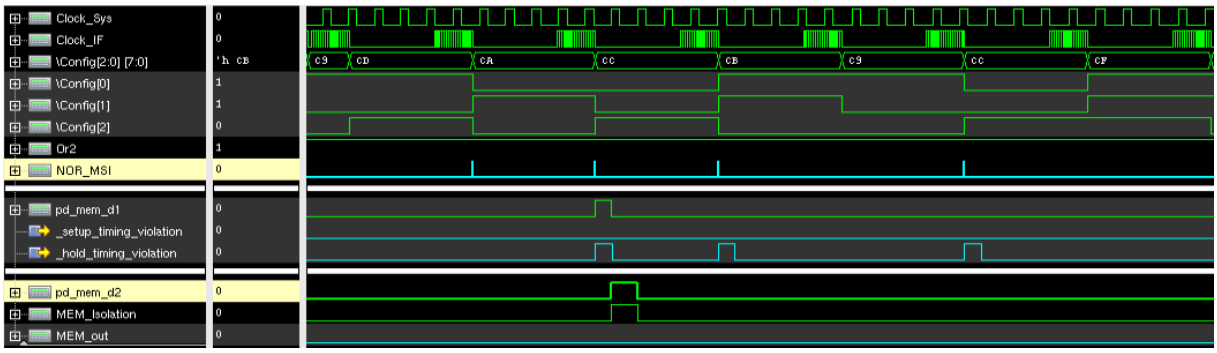

Figura 12-5: Simulazione Metastability aware and glitch Injection

Dunque, nella Figura 12-5 viene riportata la simulazione RTL in cui è stata introdotta l'iniezione della metastabilità ed anche l'iniezione di glitch.

Come è possibile notare, la prima differenza che può essere notata tra la simulazione metastability aware e la simulazione standard risiede nel segnale "NOR_MSI". Nel momento in cui avviene la scrittura "CA","CC","CB" dei bit costituenti "Config[2:0][7:0]", vi è la possibilità di ottenere la generazione ed il campionamento di un glitch e tale informazione viene riportata attraverso il segnale "NOR_MSI" che riporta in questi casi un impulso molto stretto. L'impulso generato ha una durata pari ad 1fs.

Focalizzandosi sulla prima scrittura "CC", avviene la commutazione contemporanea del bit "Config[1]" e del bit "Config[2]", i quali effettuano rispettivamente una transizione da 1->0 e una transizione da 0->1 in maniera simultanea, dunque transizioni molto simili a quelle descritte nella sezione 6.2.1. In questo caso avviene una violazione del tempo di hold del FF "PD_mem_d1" a causa dell'impulso spurio generato da tale cambiamento simultaneo dei due bit ed è quindi possibile notare che avviene una commutazione non voluta del q del FF, generando quindi l'errore funzionale aspettato. Al colpo di clock successivo avviene il campionamento dell'uscita del FF "PD_mem d1" da parte del FF "PD_mem_d2" e di conseguenza vi è la commutazione del segnale di uscita della porta logica OR "MEM_isolation". Tale segnale di uscita procede con l'isolamento della memoria volatile per un colpo di clock e dunque porta ad avere un errore funzionale.

- **Conclusioni**

Dunque, come è possibile notare dal risultato ottenuto dalla simulazione metastability aware appena illustrata, l'iniezione della metastabilità e dei glitch all'interno del flusso CDC ha confermato la possibilità di avere un campionamento di un glitch nel percorso CDC identificato e dunque ha confermato la possibilità di avere un errore funzionale nel momento in cui non viene rispettata una delle soluzioni illustrate precedentemente nella sezione 6.2.1.

La prima soluzione illustrata consiste nell'effettuare i passaggi tra le modalità di funzionamento della memoria volatile passando sempre attraverso la configurazione "000" dei bit costituenti "Config[2:0]", così evitando la possibilità di insorgenza di glitch. Soluzione riportata nella simulazione in Figura 12-6.

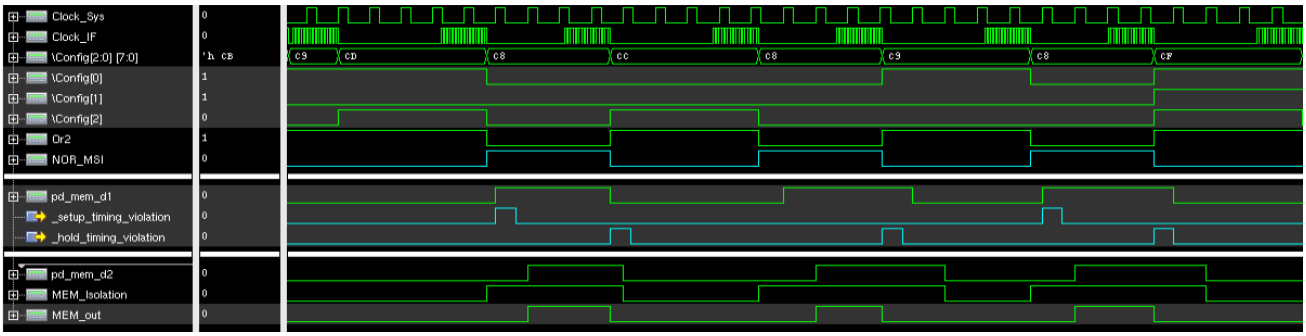

Figura 12-6: Simulazione con metastabilità e glitch passando dalla configurazione "000"

La seconda soluzione consiste nell'effettuare per prima la scrittura del FF "Force pwr Mem" e successivamente effettuare il cambio di modalità di funzionamento della memoria per poi effettuare nuovamente la scrittura di tale FF. In questo modo si assicura di non avere una generazione di un glitch, questo perché viene fissato al valore logico "0" il segnale di uscita della porta NOR quando il FF "Force pwr Mem" è scritto al valore logico "1".

12.2.2 Metastability Injection in simulazione – TestMode

Nella sezione seguente viene invece illustrato il risultato ottenuto dall'iniezione della metastabilità e glitch in simulazione riguardante il caso "TestMode", discusso nella sezione 6.1.1. Viene riportato nella Figura 12-7 lo schema della struttura presa in considerazione per la simulazione RTL con e senza l'iniezione della metastabilità, al fine di avere una migliore comprensione delle simulazioni successivamente riportate.

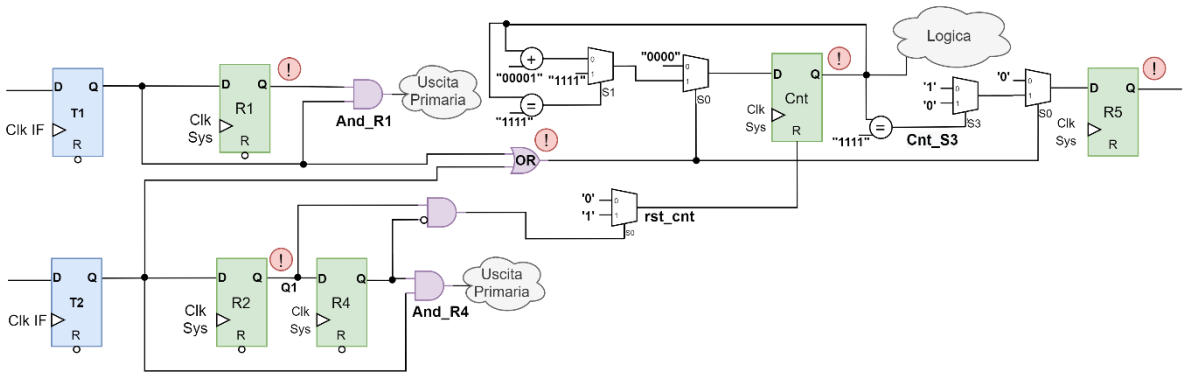

Figura 12-7: Struttura TestMode

- **Simulazione RTL standard "Not metastability aware" focalizzata sul comportamento Cnt**

Viene adesso riportata, nella Figura 12-8, la simulazione RTL not metastability aware che si focalizza sul comportamento ottenuto da parte dell'uscita del contatore "Cnt" presente nella struttura. Come aspettato, il contatore procede con l'incrementare il proprio valore senza problemi anche nel momento in cui avviene la commutazione dei segnali TestMode1(T1) ed TestMode2(T2) dai valori "01" ad "10". Dunque, è possibile apprezzare il comportamento desiderato e di riferimento per poter confrontare il risultato ottenuto dalla simulazione RTL not metastability aware.

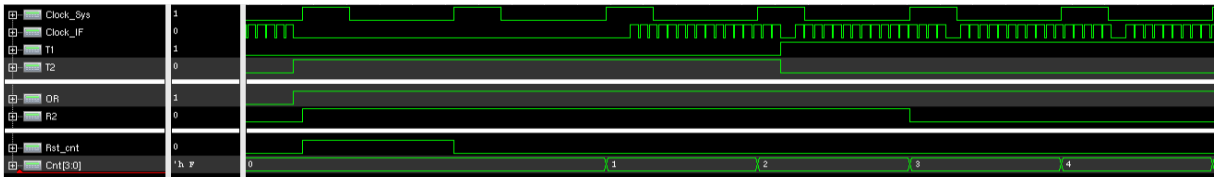

Figura 12-8: Simulazione not metastability aware Contatore

- **Simulazione RTL standard “metastability aware” focalizzata sul comportamento Cnt**

Nella Figura 12-9 viene riportata la simulazione RTL metastability aware incentrata sul comportamento ottenuto dall’uscita del contatore Cnt in funzione dell’iniezione della metastabilità e dei glitch.

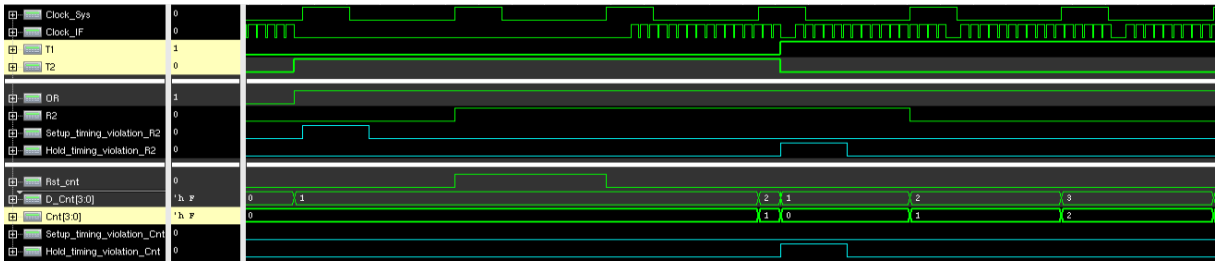

Figura 12-9: Simulazione metastability aware glitch contatore

  - **Time window – tempo di setup & hold**
    Anche in questo caso è stata selezionata una finestra temporale pari al 40% del clock di sistema, sempre con il fine di massimizzare gli eventi di violazione dei tempi di setup/hold.

È possibile notare sin da subito che il contatore non riporta lo stesso comportamento rappresentato nella simulazione not metastability aware, questo perché la presenza del glitch in prossimità del fronte di salita del clock di sistema porta ad avere una commutazione del selettore del mux in ingresso al FF Cnt e ad una conseguente violazione del tempo di hold del FF Cnt. Il risultato ottenuto è essenzialmente che l’uscita del FF Cnt assume il valore “0000”, valore presente all’ingresso “0” del mux. Dunque, il contatore non assume il valore corretto “0001” ma riassume nuovamente il valore “0000” a causa del glitch, perdendo dunque lo stato corretto in cui doveva trovarsi. Se tale glitch dovesse presentarsi quasi al termine della operazione di incremento porterebbe ad un grave errore funzionale perché viene perso completamente lo stato corretto del contatore. Ed anche possibile notare una violazione del tempo di setup del FF R2, il quale procede dunque con il commutare al valore logico 1 al colpo di clock di sistema successivo.

- **Simulazione RTL standard “Not metastability aware” focalizzata sul comportamento R5**

Nella Figura 12-10 viene riportata la simulazione RTL not metastability aware incentrata sul comportamento di R5. È possibile notare che all’interno della simulazione non vengono riportati problemi di violazione di setup ed hold dei registri coinvolti nei percorsi CDC come anche non sono riportati problemi dovuti ad una eventuale generazione di glitch nella commutazione di TestMode1(T1) e TestMode2(T2) dai valori “01” ad “10”, come invece spiegato nella sezione 6.1.1.

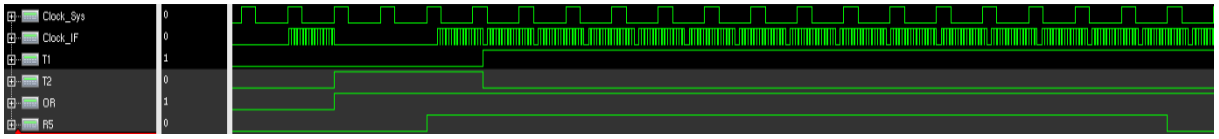

Figura 12-10: Simulazione not metastability aware R5

- **Simulazione RTL standard “metastability aware” focalizzata sul comportamento R5**

Invece, nella Figura 12-11, è possibile notare che si ha un comportamento differente da quello riportato nella simulazione RTL not metastability aware. Infatti, focalizzandosi sul comportamento ottenuto dal FF R5, è possibile notare che in prossimità della commutazione di TestMode1(T1) ed TestMode2(T2) dai valori “01” ad “10” avviene una commutazione non aspettata dell’uscita del FF R5. Dunque, a causa del glitch generato da tale commutazione dei segnali TestMode1(T1) ed

TestMode2(T2), avviene una violazione del tempo di hold del FF R5 che cambia il suo stato logico da “1” a “0” per un colpo di clock, quando il comportamento aspettato è che non vi deve essere tale transizione durante la fase operativa.

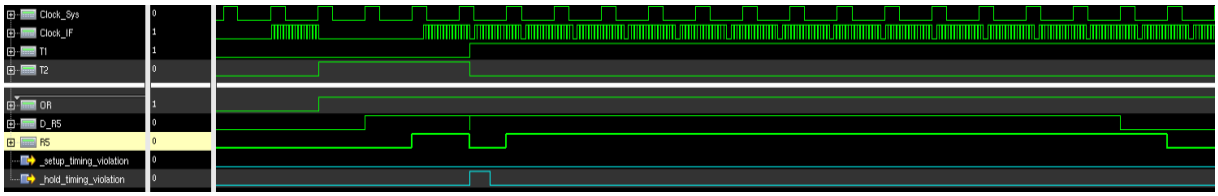

Figura 12-11: simulazione metastability aware Glitch R5

- **Simulazione RTL standard “not metastability aware” focalizzata sull’errore di convergenza**

Nella Figure 12-12 viene riportata la simulazione not metastability aware della struttura di TestMode in cui sono stati implementati i sincronizzatori standard multi-Flop per la sincronizzazione dei segnali TestMode1(T1) ed TestMode2(T2), struttura riportata nella Figura 6-4. Come è possibile notare da essa, non sono presenti errori all’uscita del FF Cnt come anche all’uscita del FF R5 in seguito alla commutazione di T1 e T2 dal valore 01 al valore 10. Infatti, è possibile notare che avviene un corretto campionamento da parte degli elementi di sincronizzazione dei nuovi valori assunti dai segnali T1 e T2. Dunque, nella simulazione not metastability aware non è possibile apprezzare un possibile errore di convergenza dovuto alla presenza della porta logica OR i cui ingressi sono le uscite dei due sincronizzatori.

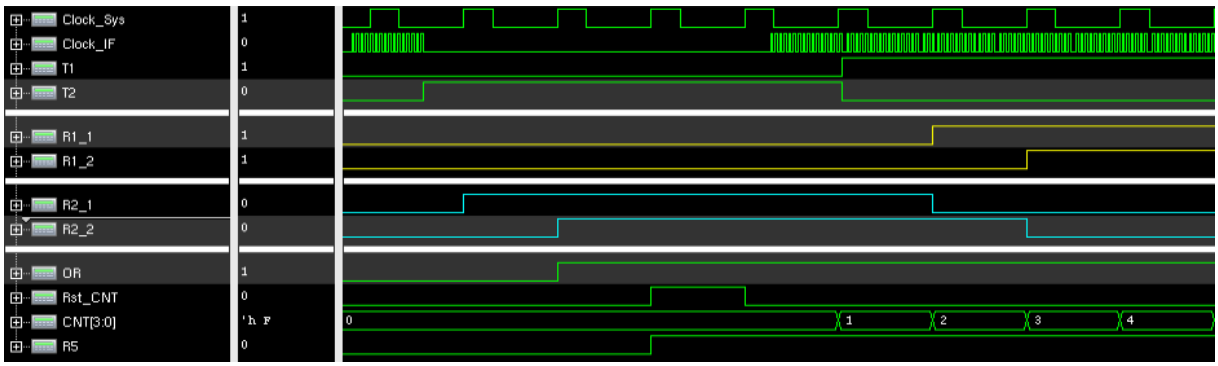

Figure 12-12: Simulazione not metastability aware incentrata sull'errore di convergenza

- **Simulazione RTL standard “metastability aware” focalizzata sull’errore di convergenza**

Nella Figura 12-13 viene invece riportato il comportamento assunto dalla struttura di TestMode, modificata con l’introduzione dei due sincronizzatori standard multi-Flop, in caso di errore di convergenza. Come è possibile notare nella simulazione, avviene la commutazione contemporanea di TestMode1(T1) ed TestMode2(T2) e l’uscita del primo FF del multi-Flop implementato per la sincronizzazione del segnale T2 procede con il commutare al nuovo valore logico 0. Il primo FF del multi-Flop implementato per la sincronizzazione del segnale T1, “R1-1”, invece prosegue con il fornire in uscita il valore logico precedente 0, portando dunque alla conclusione di avere all’uscita della porta logica “OR” uno 0 logico al colpo di clock successivo per un intero colpo di clock. Quindi, questo si riflette in un errore all’uscita del contatore, che non prosegue con il fornire in uscita il valore 3 ma procede con il commutare al valore 0, ed anche l’uscita del FF R5 procede con il commutare al valore logico 0 per un intero colpo di clock. È dunque possibile apprezzare un errore di convergenza dovuto alla presenza di una logica combinatoria a valle degli elementi di sincronizzazione e a causa di una commutazione contemporanea dei due

segnali che devono essere sincronizzati dai sincronizzatori multi-Flop. Tale errore viene risolto prendendo in considerazione la procedura progettata ed illustrata nella sezione 6.1.1.

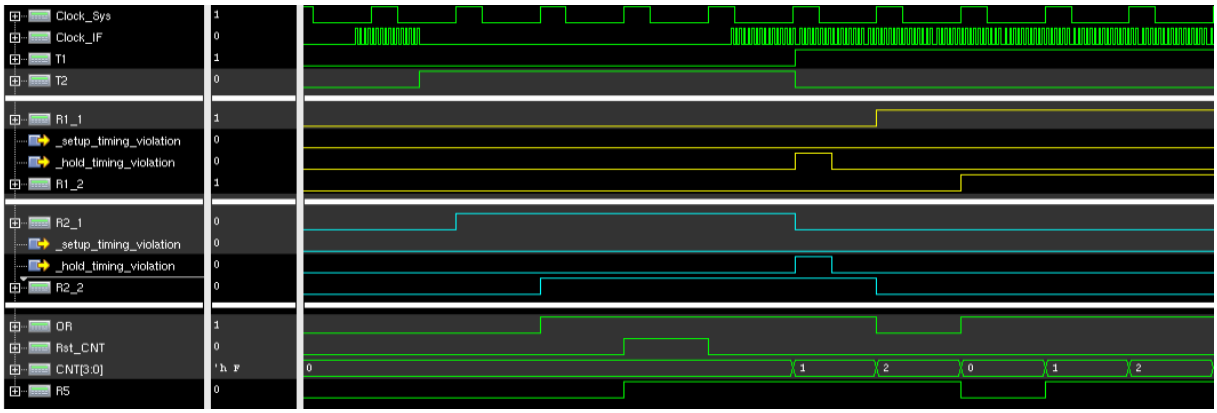

Figura 12-13: Errore di convergenza

- **Conclusioni**

Dunque, è stato possibile apprezzare la serie di errori che possono emergere a causa dell'iniezione di metastabilità e glitch all'interno della struttura. Adesso si procede con il verificare se la procedura progettata, illustrata nella sezione 6.1.1, risolve le problematiche CDC che possono emergere a causa di una possibile nascita di errore di convergenza. Viene effettuato un riepilogo di quale deve essere la procedura da seguire:

Considerando come punto di partenza i segnali TestMode1(T1) e TestMode2(T2) costanti al valore logico "00", essi devono rispettare la seguente serie di passaggi:

1) Il segnale TestMode1(T1) e il segnale TestMode2(T2) devono essere rispettivamente impostati ai valori logici "0" ed "1", successivamente è necessario attendere un tempo pari ad almeno diciotto colpi di clock di sistema prima di effettuare la prossima operazione;

2) Il segnale TestMode1(T1) e il segnale TestMode2(T2) devono essere rispettivamente impostati ai valori logici "1" ed "1", questo con lo scopo di non avere una commutazione simultanea dei due bit di TestMode. Può essenzialmente essere vista come una configurazione tale da non avere una commutazione contemporanea dei due segnali ai valori logici opposti;

3) Il segnale TestMode1(T1) e il segnale TestMode2(T2) devono essere rispettivamente impostati ai valori logici "1" ed "0", successivamente è necessario attendere un timing specifico per poter procedere con la lettura dei risultati ottenuti dalla struttura di TestMode;

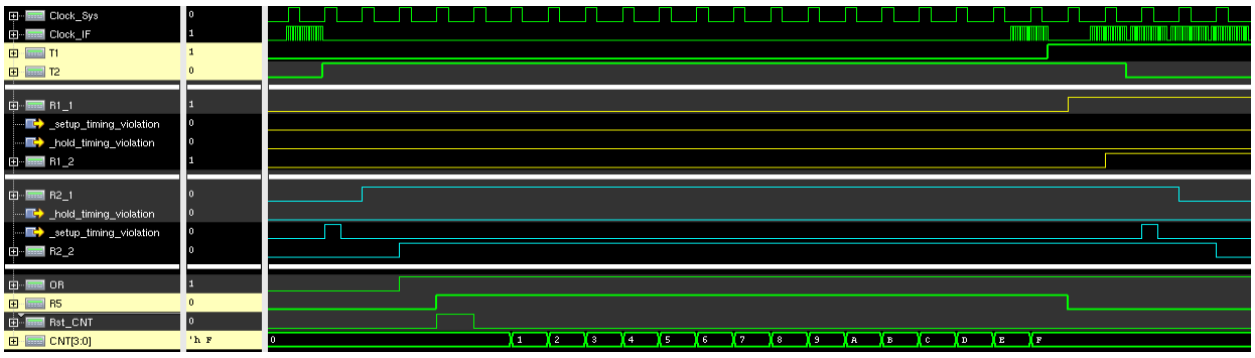

Figura 12-14: Simulazione con procedura da attuare

Nella Figura 12-14 è possibile notare che invece non emergono gli errori descritti precedentemente. Adottando tale procedura si assicura di non avere un errore sull'uscita del contatore come anche sull'uscita del FF R5. Dunque, è possibile assicurare che l'adozione di tale procedura di scrittura dei registri TestMode1(T1) e TestMode2(T2) esclude la possibilità di problematiche CDC inerenti alla possibile nascita di un errore di convergenza dovuto alla presenza della porta logica "OR" a valle dei due sincronizzatore implementati.

12.2.3 Metastability Injection in simulazione– Reset Configurations

Nella sezione seguente viene invece illustrato il risultato ottenuto dall'iniezione della metastabilità e glitch in simulazione riguardante il caso "Reset Configurations", discusso nella sezione 6.1.4. Viene riportato nella Figura 12-15 lo schema della struttura presa in considerazione per la simulazione RTL con e senza l'iniezione della metastabilità.

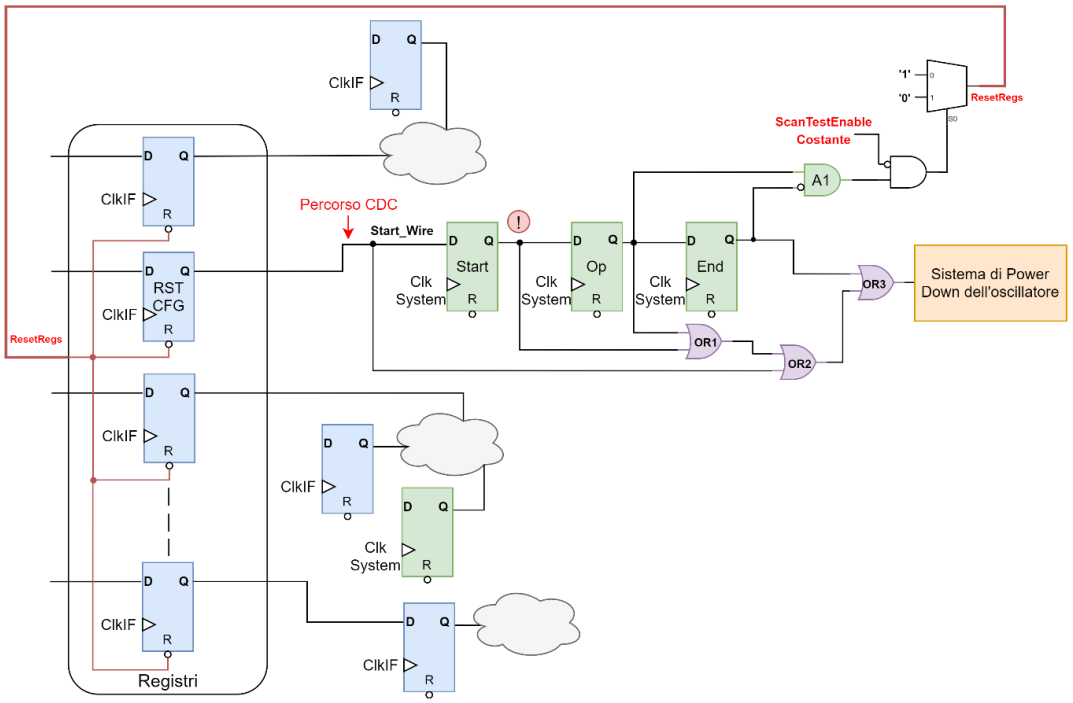

Figura 12-15: Struttura reset configurations

Come già precedentemente spiegato, tale struttura è caratterizzata da un problema CDC dovuto alla presenza di una logica combinatoria collegata all'uscita del FF Start. Tale FF costituisce il primo elemento della catena di sincronizzazione della struttura standard multi-flop e la presenza di logica combinatoria all'interno della catena di sincronizzazione è sconsigliata per la possibilità di avere una propagazione della metastabilità attraverso tale logica. Nel caso in questione non vi è un problema CDC nel momento in cui il segnale di uscita del FF RST_CFG viene asserito e mantenuto stabile per un tempo pari ad almeno due colpi di clock di sistema, ovvero per un tempo necessario per poter assicurare il campionamento da parte del FF Start e dunque assicurare la stabilità del segnale di uscita della porta logica OR. La violazione CDC riportata non risulta essere un reale problema CDC grazie al protocollo adottato. Grazie ad esso, il FF RST_CFG viene resettato da parte del clock di sistema, assieme agli altri registri di configurazione, nel momento in cui è stata assicurata la corretta esecuzione della procedura di reset configurations. Dunque, protocollo che può essere visto come un "closed-loop".

Si procede quindi con il primo passo per poter effettuare l'iniezione di metastabilità e glitch in simulazione, ovvero quello di prendere in considerazione un testcase e la relativa simulazione che valida il corretto comportamento delle funzionalità del dispositivo e che dunque rispecchia il comportamento aspettato.

- **Simulazione RTL standard "Not metastability aware"**

Viene riportata nella Figura 12-16 la simulazione RTL in assenza di metastabilità che illustra il funzionamento della struttura in questione. Come è possibile notare, una volta effettuata la scrittura del FF RST_CFG(RST) avviene anche il campionamento da parte del FF Start, appartenente al dominio di clock di sistema, senza alcun tipo di violazione. Ovviamente, una volta effettuata la scrittura del RST_CFG(RST) avviene anche la commutazione all' "1" logico del segnale di uscita della OR3 chiamato in questo caso PowerDown_Osc. Dunque, una volta avvenuto il campionamento da parte del FF Start, al colpo di clock successivo avviene la commutazione del segnale di uscita del secondo FF Op ed anche l'asserzione del segnale di reset dei registri di configurazione "ResetRegs". Dunque, una volta avvenuta la commutazione al valore logico "0" del segnale di ResetRegs è possibile notare che avviene il reset di una serie di registri appartenenti al sistema, che commutano al loro valore di reset, come anche avviene il reset del segnale RST_CFG(RST). È possibile anche notare di come, per tutta la procedura, il segnale PowerDown_Osc sia rimasto asserito al valore logico "1".

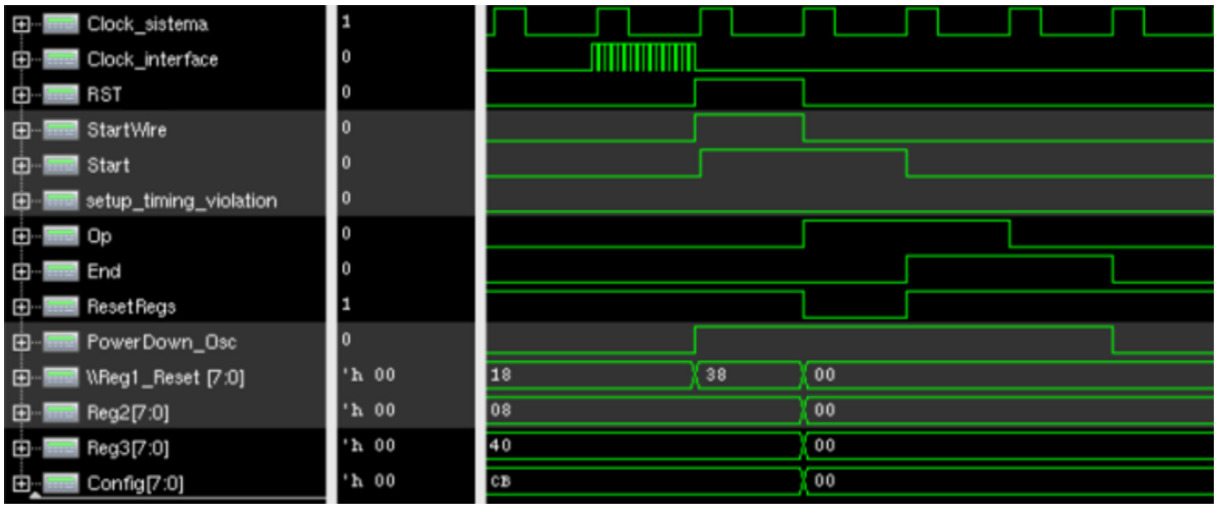

Figura 12-16: Simulazione not metastability aware

- **Simulazione RTL "Metastability aware"**

Analizzato il risultato ottenuto ed aspettato dalla simulazione RTL "standard", si procede adesso con introdurre l'iniezione della metastabilità.

  - **Time window – tempo di setup & hold**
    Anche in questo caso è stata selezionata una finestra temporale pari al 40% del clock di sistema, sempre con il fine di massimizzare gli eventi di violazione dei tempi di setup/hold.

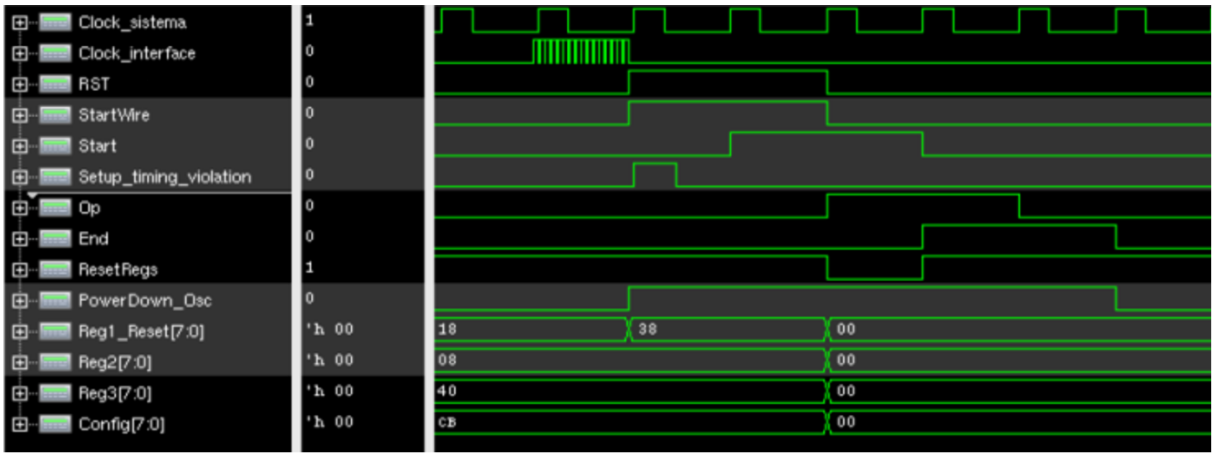

Figura 12-17 Simulazione metastability aware

Come è possibile notare nella Figura 12-17, effettuata la scrittura del FF RST_CFG(RST) avviene una violazione del tempo di setup del FF Start. In questo caso, il FF Start cambia il suo valore di uscita nel colpo di clock di sistema successivo a differenza della simulazione RTL “not metastability aware”. Dunque, la principale differenza tra le due simulazioni RTL risiede semplicemente in una latenza aggiuntiva di un colpo di clock dovuta all’iniezione della metastabilità. La procedura segue comunque il comportamento aspettato validando dunque la sua robustezza anche in funzione della metastabilità. Il segnale PowerDown_Osc segue lo stesso comportamento della simulazione RTL not metastability aware e rimane asserito alto per tutta la procedura anche in presenza dell’iniezione della metastabilità.

13 Conclusioni

Il lavoro di tesi è stato quindi incentrato sul progettare un flusso standardizzato e consistente di "CDC Clean Sign-off RTL", un flusso da integrare nei moderni tape-out per i futuri design asincroni col fine di validare o migliorare la loro robustezza a fronte dei problemi di Clock-Domain-Crossing e Reset-Domain-Crossing. Dunque, il flusso progettato si pone il traguardo di raggiungere ed assicurare un procedimento standardizzato di verifica strutturale e funzionale RTL con cui ottenere design asincroni sempre più "Metastability Safe" e "Combinational CDC glitch free".

Il mio obiettivo è stato quindi quello di realizzare un flusso di validazione CDC-RDC integrabile nello stadio di verifica RTL di un flusso di progettazione di design. La sua integrazione consente di ottenere sin dai primi stadi di progettazione, più precisamente sin dalle prime versioni del codice RTL, la conoscenza dei problemi di asincronia presenti all'interno del proprio design e di ottenere sin da subito coscienza dei limiti CDC ed RDC presenti. Grazie a tale flusso è possibile risaltare i problemi CDC e RDC in uno stadio di progettazione in cui è ancora possibile, in maniera efficiente, attuare modifiche RTL e verificarne la loro efficacia, riducendo al minimo il numero di errori CDC-RDC che possono essere rilevati durante la post-layout verification. È quindi possibile identificare e risolvere tutti i problemi CDC-RDC del DUT ancora prima di iniziare gli stadi di produzione di Sintesi e Place&Route con relative verifiche, stadi onerosi dal punto di vista del tempo richiesto di esecuzione e della loro elaborazione.

Per poter realizzare ed attuare tale flusso di analisi CDC-RDC, ho dovuto per prima effettuare uno studio dei problemi CDC ed RDC che possono essere presenti all'interno di un design e le principali condizioni e cause che possono scatenare tali problemi. Una volta ottenuta coscienza dei problemi CDC e RDC tipici, ho proceduto con lo studiare lo stato dell'arte attuale degli schemi di sincronizzazione implementabili all'interno di un design al fine di limitare la propagazione della metastabilità ed affrontare i problemi di asincronia presenti nella comunicazione tra molteplici domini di clock.

Il flusso è adottabile su design RTL più o meno complessi ma richiede comunque un certo livello di conoscenza del design e delle sue funzionalità, questo per poter massimizzare l'efficacia dell'analisi CDC-RDC e massimizzare l'esclusione di violazioni che risultano non essere problemi CDC-RDC reali all'interno del contesto di design in cui sono posti. Per poter raggiungere tale obiettivo, ho studiato il design su cui è stata effettuata l'analisi CDC-RDC attraverso il codice RTL ed attraverso le specifiche e datasheet del dispositivo. Lo studio del design è stato necessario sin dai primi step dell'analisi per poter avere una corretta impostazione dell'ambiente CDC-RDC come anche una corretta impostazione del design. Tale conoscenza del design è necessaria non solo per poter effettuare una verifica strutturale completa e la più accurata possibile, ma è anche necessaria per poter effettuare una verifica funzionale corretta dei protocolli presenti per la trasmissione di segnali e dati tra i domini di clock asincroni. Ovviamente, protocolli e soluzioni di design non sono presenti nella documentazione del design ed è stato quindi necessario effettuare un'analisi approfondita di ogni violazione CDC-RDC, protocollo relativo e contesto di logica del design per poter procedere con una eventuale esclusione o conferma della violazione.

Per poter procedere con la verifica funzionale del design e dei suoi relativi protocolli, ho effettuato uno studio sulla verifica formale per poter raggiungere l'obiettivo di scrivere assunzioni, asserzioni e cover con cui validare i protocolli e funzionalità presenti nei problemi CDC-RDC riscontrati. Ho quindi creato e scritto assunzioni, asserzioni e cover per ogni violazione riscontrata, come quelle illustrate nel lavoro di tesi, affinando sempre di più la loro efficacia e migliorando l'informazione portata da esse. Ovviamente, sono state effettuate sempre più correzioni ed aggiunte di assunzioni e cover per poter

escludere la segnalazione di falsi errori ed avere sempre più definite condizioni di contorno dei protocolli e delle loro funzionalità presenti.

Dopo aver quindi studiato e creato assunzioni, asserzioni e cover con cui è stato possibile validare il corretto funzionamento dei protocolli di acquisizione attraverso l'ausilio degli engine formali, ho proseguito con il raggiungere l'obiettivo di massimizzare la validazione della robustezza dei protocolli di sincronizzazione a fronte della metastabilità e glitch. Tale obiettivo è stato raggiunto aggiungendo al flusso CDC-RDC l'iniezione della metastabilità con approccio formale e l'iniezione della metastabilità e glitch in simulazione RTL. Ho quindi applicato l'iniezione della metastabilità attraverso la verifica formale sulle asserzioni su cui è stato validato il corretto comportamento dei protocolli e quindi verificato che tali protocolli siano ancora validi anche in funzione delle "latenze" introdotte a causa dell'iniezione della metastabilità. Lo stesso discorso è stato applicato per l'iniezione della metastabilità e glitch in simulazione, attraverso la quale è stato possibile avere una maggiore conferma dei problemi CDC-RDC che possono sorgere a causa della generazione di glitch e dei loro effetti sulla logica e funzionalità. Dunque, avere una maggiore coscienza, non teorica ma applicata, degli effetti che si possono avere all'interno del design e una validazione che i protocolli e configurazioni di segnali nei percorsi CDC e RDC sono effettivamente robusti a fronte della metastabilità e glitch. Le violazioni CDC e RDC riscontrate a livello strutturale e l'impatto che esse possono avere sul design a livello teorico, si sono rilevate essere reali problematiche CDC-RDC ed è stato possibile visionare il loro impatto sul comportamento del design attraverso la verifica formale ed attraverso la simulazione dinamica, con la quale è stato possibile visionare cosa potesse comportare una propagazione di un glitch.

Le problematiche CDC-RDC, una volta identificate e validato il loro effettivo impatto sulle funzionalità, sono state riportate e affrontate con il team di design e di verifica con il fine di validare se quest'ultime sono state effettivamente risolte nel corso dei nuovi aggiornamenti del codice RTL o se risolte con determinate configurazioni di segnali descritte nei datasheet aggiornati o se richiedevano l'implementazione di soluzioni da me progettate.

Infine, tra le diverse violazioni identificate, è stata rilevata una struttura del design che ha riportato diverse violazioni CDC su cui è stato necessario trovare delle soluzioni per poter irrobustire tale struttura, la quale altrimenti risulta essere un possibile problema CDC all'interno del design. Per questa problematica identificata ho dovuto progettare diverse soluzioni per poterla risolvere: una modifica RTL, una codifica ad hoc da applicare ai segnali coinvolti ed una procedura di gestione e di configurazione dei segnali coinvolti nella struttura in questione.

Il risultato dell'applicazione di tale flusso ha quindi riportato l'esito positivo di rilevare problematiche CDC-RDC, sin dal codice RTL, non precedentemente riscontrate dai team di verifica e di design, come anche confermare la validità di quei protocolli di sincronizzazione implementati per affrontare problematiche CDC-RDC conosciute a fronte dell'iniezione della metastabilità e di glitch.

14 Bibliografia

[1] A Meade Kathleen, Sharon Rosenberg, A Practical Guide to Adopting the Universal Verification Methodology (UVM) Second Edition.

[2] Erik Seligman, Tom Schubert, and M V Achutha Kiran Kumar. 2015. Formal Verification: An Essential Toolkit for Modern VLSI Design. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[3] Pong P.CHU, “RTL Hardware Design Using VHDL:Coding For Efficiency, Portability, and Scalability”. 2006.

[4] J. Sparsø. Principles of asynchronous circuit design: A system perspective. Kluwer Academic Publishers (Boston/Dordrecht/London), 2001.