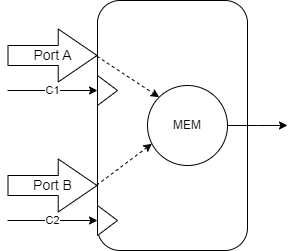
# Memorie Dual Port

Le memorie “dual port” sono dispositivi che possono essere pilotati (e letti) contemporaneamente da due porte di interfaccia distinte, ma che condividono il medesimo dato salvato. Visto che l’operazione di lettura non riveste particolari difficoltà in questo esercizio ci concentreremo nella realizzazione di una singola cella di memoria che però possa essere controllata attraverso due distinte porte di entrata, ciascuna sincronizzata sul proprio segnale di clock



Si vuole pertanto progettare una classe di Flip Flop che siano in grado di sincronizzare il loro funzionamento su due eventi di clock separati, portati al sistema su due segnali di sincronismo distinti (c1 e c2).

Nello specifico supponiamo che entrambi i sincronismi avvengano sul fronte di **salita** di uno qualsiasi dei due segnali c1 e c2. (Ovviamente si potrebbero prendere in considerazione sistemi diversi che si sincronizzino sui fronti di discesa oppure siano di tipo DDR o ancora siano una combinazione di questi dove ad esempio una porta è di tipo DDR e l’altra è sincronizzata sul solo fronte positivo)

Si inizierà quindi progettando il circuito più semplice possibile, sprovvisto di qualsivoglia segnale di controllo e che si limiti solamente a cambiare di stato quando viene rilevato un fronte di salita su uno qualsiasi dei due segnali di clock (Flip Flop T senza controllo) Successivamente verrà aggiunto un segnale di controllo atto a inibire la commutazione sull’uno o sull’altro dei due clock (Flip Flop T con segnali controllo) ed eventualmente dotato di un segnale di Reset asincrono atto a forzarlo in una condizione nota. Da ultimo si vedrà come impiegare questi segnali di controllo appena aggiunti per estendere il funzionamento del Flip-Flop modificandone l’interfaccia di ingresso ed alterandone il funzionamento per trasformarlo in una cella di memoria di tipo JK oppure D, o ancora aggiungendovi ulteriori segnali di controllo sincroni, ciascuno facente riferimento alla propria interfaccia, scandita dal proprio evento di sincronismo.

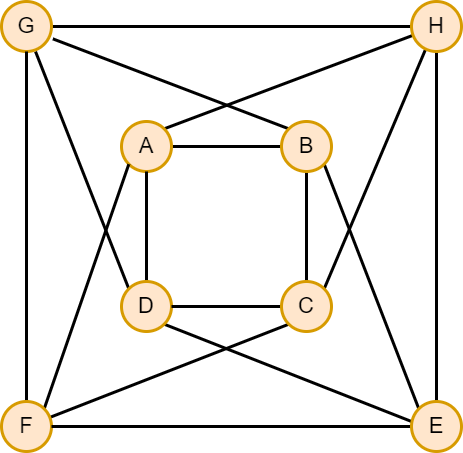
Flip Flop T – senza controllo:

Si descriva attraverso la tavola di Huffman il dispositivo in oggetto: esso sarà dotato di due segnali di ingresso (c1 e c2) che rappresentano i segnali di sincronismo e di una sola uscita Z. Il dispositivo alternerà il valore dell’uscita tra i valori 0 e 1 ogni qualvolta rilevi in ingresso un fronte di salita su uno oppure sull’altro dei segnali c1 o c2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| st\c1c2 | 00 | 01 | 11 | 10 | z |
| A | A | F | - | H | 0 |
| B | A | B | G | - | 0 |
| C | - | B | C | D | 0 |
| D | A | - | G | D | 0 |
| E | E | B | - | D | 1 |
| F | E | F | C | - | 1 |
| G | - | F | G | H | 1 |
| H | E | - | C | H | 1 |

La macchina è stata concepita per avere 4 stati stabili per ciascuna condizione dell’uscita, ciascuno che viene raggiunto con una opportuna combinazione dei sincronismi (c1 e c2). Partendo ora da un qualsiasi stato stabile quando viene rilevato un fronte di salita o di discesa su uno qualsiasi dei due clock la macchina passa ad uno stato diverso: Se viene rilevato un fronte di salita allora passerà allo stato che presenta uscita opposta alla propria ma se viceversa viene rilevato un fronte di discesa, allora passerà allo stato corrispondente alla nuova combinazione dei sincronismi ma con uscita identica alla propria. Ad esempio ipotizziamo la macchina si trovi in B (quindi con i segnali in ingresso c1 e c2 rispettivamente a 0 e a 1) se all’ingresso viene presentato un fronte di salita su c1 (ovvero c1-c2 passa a 1-1) la macchina si sposta su G ed inverte l’uscita, mentre se venisse rilevato un fronte di discesa su c2 (ovvero c1-c2 passa a 0-0) la macchina cambia stato, spostandosi in A che ha un’uscita identica a quella dello stato B di partenza.

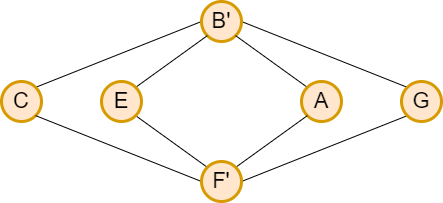
Effettuare una codifica degli stati per questa macchina che evitino corse critiche e che al contempo minimizzi il numero delle variabili di stato sarebbe cosa piuttosto ostica. Si può infatti notare dalla mappa delle transizioni come ciascun stato richiederebbe l’adiacenza di almeno 4 altri stati:



Però dalla macchina sopra esposta si può notare che 8 stati sono tutto sommato ridondanti e si può operare una semplificazione: infatti gli stati B e D risultano essere compatibili e possono essere riuniti in uno stato B’, così come lo sono F ed H che possono essere riuniti in F’.

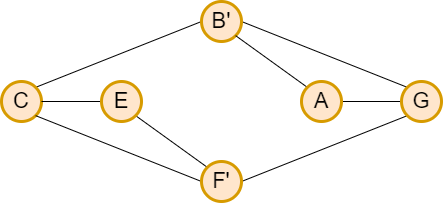
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| st\AB | 00 | 01 | 11 | 10 | z |
| A | A | F' | - | F’ | 0 |
| B' | A | B' | G | B' | 0 |
| C | - | B' | C | B’ | 0 |
| E | E | B' | - | B’ | 1 |
| F' | E | F' | C | F' | 1 |
| G | - | F' | G | F’ | 1 |

Questa situazione risulta migliore della precedente, ma purtroppo gli stati B’ ed F’ richiederebbero, per evitare corse critiche, nella fase di codifica, ciascuno l’adiacenza di 4 stati:



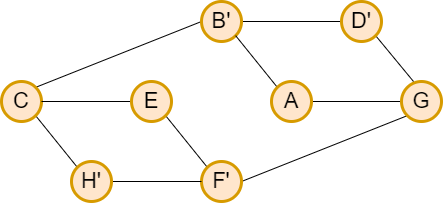
Per eliminare qualche “adiacenza” si può però ricorrere a transizioni multiple: ad esempio sostituendo la transizione A🡪F’ ad esempio con A🡪G🡪F’ e la transizione E🡪B’ con E🡪C🡪B’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| st\AB | 00 | 01 | 11 | 10 | z |
| A | A | G | - | G | 0 |
| B' | A | B' | G | B' | 0 |
| C | - | B' | C | B’ | 0 |
| E | E | C | - | C | 1 |
| F' | E | F' | C | F' | 1 |
| G | - | F' | G | F’ | 1 |



Il diagramma delle transizioni richiede ora che B’ e F’ abbiano solo tre stati adiacenti. Si è venuto però a creare un nuovo problema: quello delle due “maglie” triangolari AB’G e ECF’ che in questa forma rendono impossibile una codifica che mantenga adiacenti gli uni agli altri triplette di stati, ma a questo problema si può ovviare facilmente passando attraverso degli stati intermedi e convertendo ad esempio la transizione B’🡪 G in una transizione multipla attraverso un ulteriore stato B’🡪D’🡪G, così come la transizione F’🡪 C può essere modificata in F’🡪H’🡪C. L’uscita del sistema quando transita attraverso questi nuovi stati di transizione è stata per comodità lasciata indeterminata (d.c.).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| st\AB | 00 | 01 | 11 | 10 | z |
| A | A | G | - | G | 0 |
| B' | A | B' | D' | B' | 0 |
| C | - | B' | C | B’ | 0 |
| D' | - | - | G | - | - |
| E | E | C | - | C | 1 |
| F' | E | F' | H' | F' | 1 |
| G | - | F' | G | F’ | 1 |
| H' | - | - | C | - | - |



La macchina così concepita consente di essere realizzata adottando una codifica con solo tre variabili di stato pur in grado di garantire distanze unitarie tra stati interessati da una transizione, quale ad esempio la codifica qui sotto riportata.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| y1\y2y3 | 00 | 01 | 11 | 10 |
| 0 | A | B' | D' | G |
| 1 | E | C | H' | F' |

Adottando tale codifica la tavola di flusso della macchina risulta pertanto essere:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| st\'000B | 00 | 01 | 11 | 10 | z |
| A-000 | 000 | 010 | - | 010 | 0 |
| B-001 | 000 | 001 | 011 | 001 | 0 |
| C-101 | - | 001 | 101 | 001 | 0 |
| D-011 | - | - | 010 | - | - |
| E-100 | 100 | 101 | - | 101 | 1 |
| F-110 | 100 | 110 | 111 | 110 | 1 |
| G-010 | - | 110 | 010 | 110 | 1 |
| H-111 | - | - | 101 | - | - |

Per semplificare le successive mappe di Karnaugh risulta essere più opportuno riordinare la mappa secondo l’asse verticale

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | st\'000B | 00 | 01 | 11 | 10 | z |
| A | 000 | 000 | 010 | - | 010 | 0 |
| B | 001 | 000 | 001 | 011 | 001 | 0 |
| D | 011 | - | - | 010 | - | - |
| G | 010 | - | 110 | 010 | 110 | 1 |
| E | 100 | 100 | 101 | - | 101 | 1 |
| C | 101 | - | 001 | 101 | 001 | 0 |
| H | 111 | - | - | 101 | - | - |
| F | 110 | 100 | 110 | 111 | 110 | 1 |

Si può pertanto utilizzare la suddetta tavola di flusso per trovare le funzioni che definiscono sia l’uscita che le variabili di stato:

Per l’uscita:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| y1\y2y3 | 00 | 01 | 11 | 10 |
| 0 | 0 | 0 | - | 1 |
| 1 | 1 | 0 | - | 1 |

Per quanto riguarda la variabile Y1   
(I min-term evidenziati in grigio sono condivisi da più implicanti)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| y1y2y3\c1c2 | 00 | 01 | 11 | 10 |
| 000 | 0 | 0 | - | 0 |
| 001 | 0 | 0 | 0 | 0 |
| 011 | - | - | 0 | - |
| 010 | - | 1 | 0 | 1 |
| 100 | 1 | 1 | - | 1 |
| 101 | - | 0 | 1 | 0 |
| 111 | - | - | 1 | - |
| 110 | 1 | 1 | 1 | 1 |

Per quanto riguarda la variabile Y2

(I max-term evidenziati in grigio sono condivisi da più implicanti)

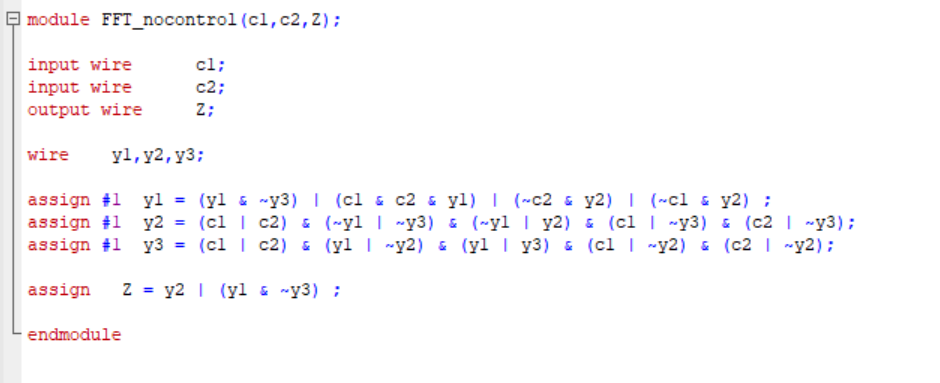
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| y1y2y3\c1c2 | 00 | 01 | 11 | 10 |
| 000 | 0 | 1 | - | 1 |
| 001 | 0 | 0 | 1 | 0 |
| 011 | - | - | 1 | - |
| 010 | - | 1 | 1 | 1 |
| 100 | 0 | 0 | - | 0 |
| 101 | - | 0 | 0 | 0 |
| 111 | - | - | 0 | - |
| 110 | 0 | 1 | 1 | 1 |

Per quanto riguarda la variabile Y3

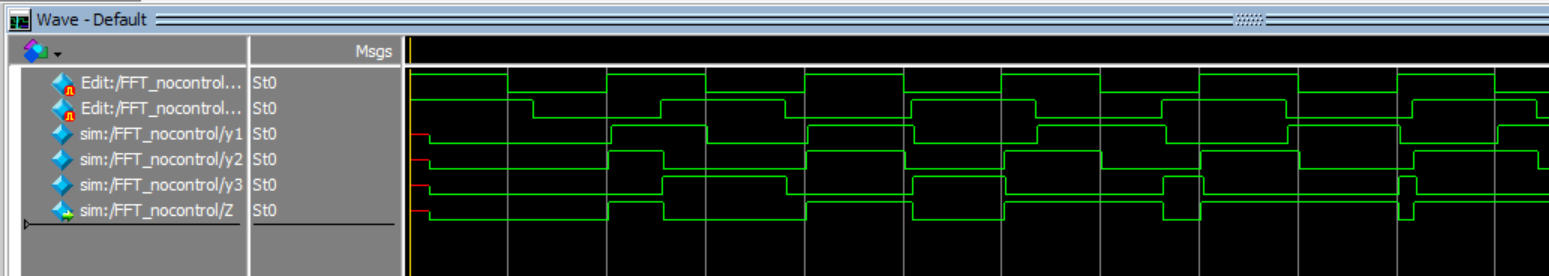
(I max-term evidenziati in grigio sono condivisi da più implicanti)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| y1y2y3\c1c2 | 00 | 01 | 11 | 10 |
| 000 | 0 | 0 | - | 0 |
| 001 | 0 | 1 | 1 | 1 |
| 011 | - | - | 0 | - |
| 010 | - | 0 | 0 | 0 |
| 100 | 0 | 1 | - | 1 |
| 101 | - | 1 | 1 | 1 |
| 111 | - | - | 1 | - |
| 110 | 0 | 0 | 1 | 0 |

Il circuito così progettato, descritto in Verilog HDL



Opportunamente simulato sembra funzionare correttamente.



Flip Flop T – con segnali di controllo:

Si noti che però per poter eseguire la simulazione nelle fasi iniziali si è dovuto provvedere a “forzare” le variabili di stato in una condizione iniziale attraverso il comando Modelsim “Force” ed a svincolarle (con il comando “NoForce”) qualche istante dopo (in particolare quando gli ingressi avevano assunto il valore 00.

Quindi la prima operazione da effettuare sarà quella di dotare il circuito di un segnale di Reset Asincrono in grado di forzare attraverso un comando esterno il dispositivo in una condizione di partenza desiderata (ad esempio lo stato 000).

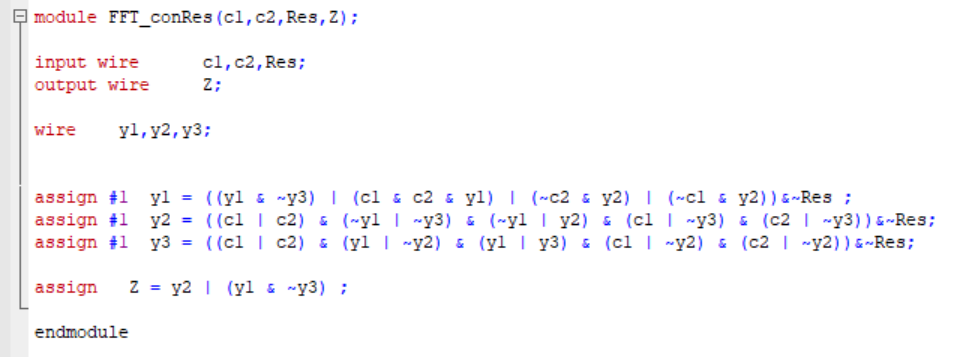
Si può erroneamente pensare che esso possa essere fatto semplicemente intercettando le tre linee che rappresentano le variabili di stato e forzandole a zero attraverso un’opportuna logica che interrogando un segnale di Reset (attivo alto) stabilisca se la variabile di stato debba passare inalterata oppure debba essere forzata a zero.

Tale logica combinatoria da applicare a ciascuna variabile di stato sarebbe estremamente semplice e potrebbe essere formalizzata nella seguente mappa di Karnaugh

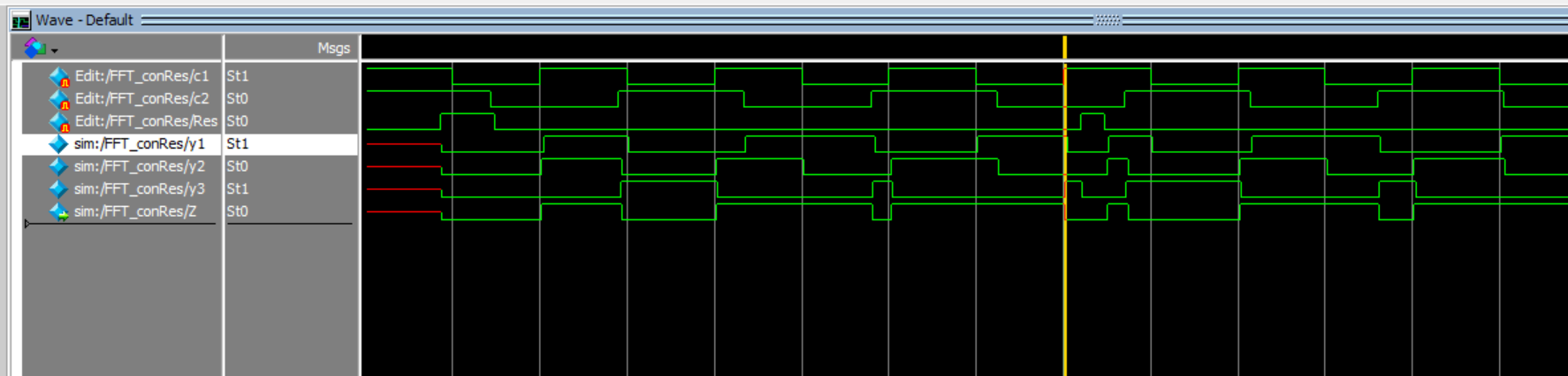
|  |  |  |
| --- | --- | --- |
| y|Res | 0 | 1 |
| 0 | 0 | 0 |
| 1 | 1 | 0 |

E quindi realizzata attraverso una semplice porta AND tra la variabile di stato *y* e *not( Res).*

Il circuito così realizzato, descritto dal seguente sorgente



Se però simuliamo il circuito notiamo che tale reset NON opera ESATTAMENTE come vorremmo.



Infatti sebbene il primo impulso di reset porti la macchina nello stato desiderato dalla quale poi parte propriamente, questo risulta essere solo un caso fortuito, infatti sebbene la logica così concepita forza a 0 tutte e tre le variabili di stato quando Res=1, cosa accadrebbe se tale segnale di reset dovesse ritornare a zero mentre i segnali su c1 e c2 sono diversi tra di loro (ad esempio 01 o 10)? Talla Tavola di flusso si nota una rapida evoluzione che porterebbe la macchina momentaneamente in 010 per poi transitare immediatamente in 110 in cui corrisponde l’uscita pari a 1 (come da simulazione), ovvero l’uscita si attiverebbe NON in corrispondenza di un fronte di clock ma in questo caso al solo estinguersi del segnale di reset.

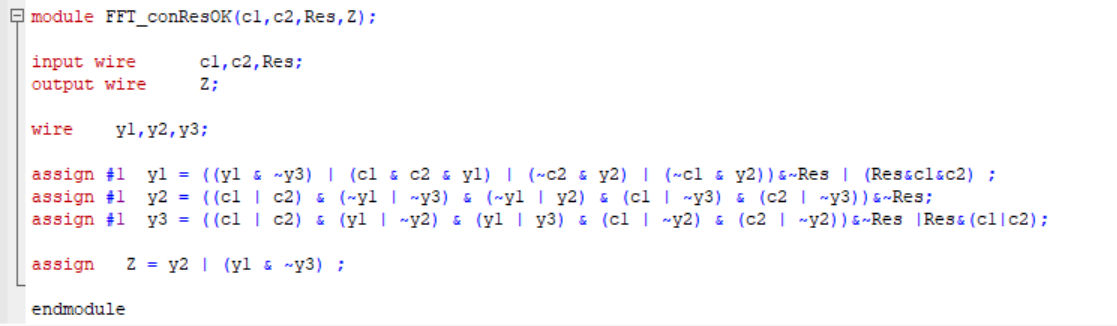
Per evitare questo malfunzionamento la tecnica con la quale operare il reset deve essere più raffinata ovvero l’idea è quella di forzare il sistema negli stati A, B o C (gli stati stabili ad uscita 0) in base ai valori presenti in ingresso quando il reset è attivo.

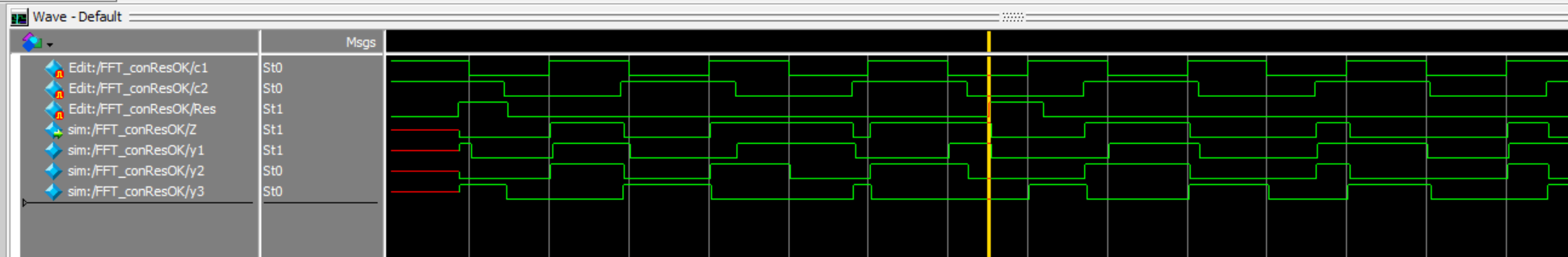
Dalla codifica adottata pertanto, quando il reset è attivo le tre variabili di stato dovranno venir forzate rispettivamente i valori:

|  |  |  |
| --- | --- | --- |
| c1c2 | y1y2y3 | Stato |
| 00 | 000 | A |
| 01 | 001 | B |
| 11 | 101 | C |
| 10 | 001 | B |

Mentre dovranno rimanere immutate quando il reset è basso. Questo può pertanto essere formalizzato nelle seguenti equazioni:

Che implementate nel sorgente Verilog ed opportunamente simulate dimostrano la correttezza della realizzazione:





Al momento però il dispositivo NON possiede alcun segnale atto a controllarne in qualche forma l’evoluzione. La casistica dei segnali che vi si potrebbero aggiungere è piuttosto ampia: Oltre al segnale di Reset Asincrono già considerato vi si potrebbero aggiungere ulteriori segnali sia asincroni (quali ad esempio un segnale di Set) sia sincroni, ovvero il cui effetto si faccia sentire solo in corrispondenza degli istanti di sincronismo. Segnali senz’altro utili sono peraltro i segnali di Enable, che stabiliscono quando attivare il funzionamento del circuito e quando no. In questo caso specifico è utile peraltro introdurre due segnali di abilitazione distinti ciascuno per abilitare o disabilitare le commutazioni indotte da ogni specifico segnale di clock ovvero suddividere i segnali di controllo in due porte indipendenti.

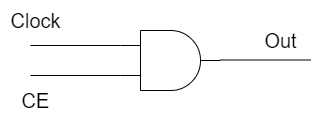
Flip Flop T – Segnali di Enable:

Per aggiungere i segnali di abilitazione, tipicamente contrassegnati con “CE” quale acronimo di “clock enable”, si aprono diverse strade progettuali: la prima, più complessa potrebbe essere di riprogettare il dispositivo partendo dalla macchina a stati ed aggiungendovi i segnali di controllo desiderati, ma si può anche pensare di integrarli a quanto finora realizzato semplicemente integrando il dispositivo con ulteriori blocchi funzionali atti ad aggiungere le specifiche desiderate (similmente a quanto è stato fatto quando è stato aggiunto il segnale di reset).

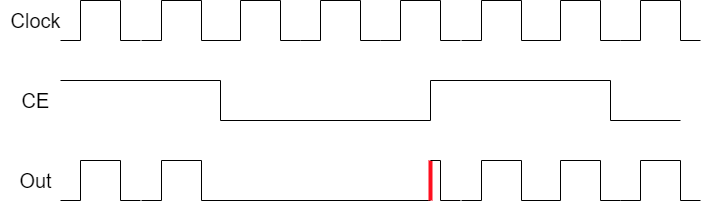
In questo caso specifico la soluzione che si potrebbe percorrere è pertanto quella di “intercettare” i segnali di clock in ingresso e di attivarli o meno in base al un’opportuna logica basata sull’informazione presente su CE, ovvero se CE=1 il clock “passa inalterato”, altrimenti esso viene “bloccato”. Questa descrizione anche se intuitivamente condivisibile è però molto approssimativa e si presta a diverse interpretazioni che rischiano di portare a risultati errati. Vi sono infatti due punti che devono essere ben chiariti ovvero

1. Come si deve agire quando il segnale CE fosse riattivato.
2. Cosa si intende “esattamente” per “bloccare” il clock.

Prendiamo ad esempio una soluzione banale, ovvero quella di usare una semplice porta AND controllata da CE per attivare/disattivare il segnale di clock.

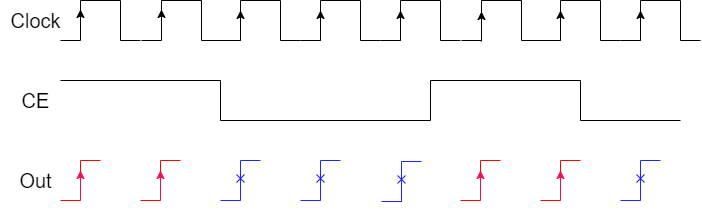


Così facendo ci ritroveremmo nella situazione sotto rappresentata: Sebbene sia assolutamente evidente che quando CE=0 il segnale di clock risulta stabile a livello basso e quindi non può produrre alcuna commutazione sul circuito a valle, il problema si pone quando CE dovesse ritornare attivo: essendo questo un segnale completamente asincrono rispetto il clock, può accadere benissimo che esso si riattivi nel periodo in cui il clock è alto, venendo così a generare un fronte (evidenziato in rosso) e con esso una commutazione nel dispositivo che segue e che nel segnale originale NON era presente.



Quindi ciò che è richiesto non è tanto quello di operare una “finestratura” sul segnale di clock quanto piuttosto sui fronti positivi di quest’ultimo (perché sono questi gli eventi di sincronismo che pilotano le commutazioni nel dispositivo a valle) discriminando i fronti che arrivano quando CE è attivo da quelli che arrivano invece quando CE è a livello basso e generare quindi in uscita un segnale che presenti i medesimi eventi di sincronismo del clock di ingresso ma opportunamente “filtrati” attraverso CE.

L’idea di ciò che si vuole ottenere in linea di massima riportata nella figura:

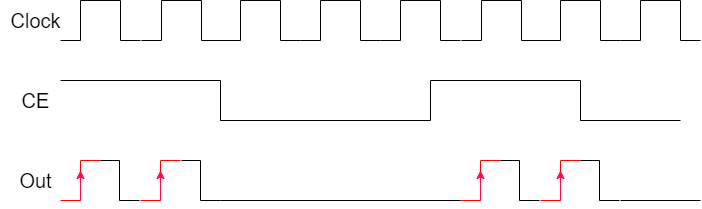


Ovviamente al momento il segnale d’uscita è appena abbozzato e si deve trovare una strategia per completarlo, ovvero per definire quando riabbassare il segnale d’uscita in modo che esso sia pronto a formare un nuovo fronte positivo quando le condizioni lo richiederanno.

In particolare si possono elaborare due strategie abbastanza simili:

1. Abbassare il segnale d’uscita contemporaneamente a quando si abbassa il clock di ingresso.
2. Abbassare il segnale d’uscita se si abbassa o il clock in ingresso oppure il CE.

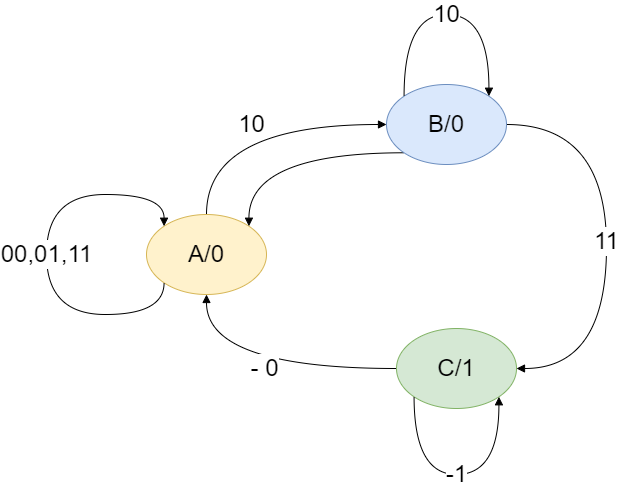
Delle due la più promettente a prima vista è la prima che consente di mantenere impulsi di clock sempre di durata congrua ed uguale a quelli in ingresso senza rischiare di vederela generazione di impulsi di durata troppo breve (dovuti all’assenza di sincronismo tra i segnali Clock e CE) che potrebbero poi non venir rilevati correttamente.



La realizzazione di un siffatto dispositivo richiede la realizzazione di una macchina sequenziale ad hoc che può essere descritta verbalmente con le seguenti affermazioni:

1. Il segnale d’uscita si porta allo stato alto quando CE=1 e si presenta un fronte sul segnale clk.  
   Ovvero si deve riconoscere la sequenza di segnali “CE,clk” passa da 10 a 11
2. Il segnale d’uscita si abbassa quando il segnale clk torna a zero.   
   Ovvero clk =0

Tale macchina può essere schematizzata nel seguente diagramma di Moore:



E quindi nella seguente tavola di Huffman

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| st\CE,clk | 00 | 01 | 11 | 10 | OUT |
| A | A | A | A | B | 0 |
| B | A | - | C | B | 0 |
| C | A | C | C | B | 1 |

Nella fase di codifica l’unica cautela da seguire è che la distanza tra lo stato B e lo stato C sia unitaria in quanto tutte le altre transizioni non sono critiche.

Scegliendo ad esempio A=00,B=01;C=11 si perviene alla seguente tavola di flusso.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| y1y2\CE,clk | 00 | 01 | 11 | 10 | OUT |
| 00 | 00 | 00 | 00 | 01 | 0 |
| 01 | 00 | - | 11 | 01 | 0 |
| 11 | 00 | 11 | 11 | 01 | 1 |
| 10 | - | - | - | - | - |

Che porta ed evidenziare le seguenti equazioni da adottare nel funzionamento del circuito:

Per quanto riguarda l’uscita:

Per quanto riguarda la variabile y1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| y1y2\CE,clk | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | - | 1 | 0 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | - | - | - | - |

Per quanto riguarda la variabile y2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| y1y2\CE,clk | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 1 |
| 01 | 0 | - | 1 | 1 |
| 11 | 0 | 1 | 1 | 1 |
| 10 | - | - | - | - |

Si noti che le equazioni mostrano la presenza di una retroazione SOLO sulla variabile Y2, mentre la variabile Y1 è in pratica impiegata solo ed esclusivamente per definire l’uscita. Ciò è dovuto alla scelta iniziale di descrivere la macchina attraverso il modello di Moore, se si fosse proceduto a descrivere la macchina attraverso modello di Mealey, si sarebbero trovati due soli stati, ma un’uscita che dipende anche dagli ingressi oltre che dalle variabili di stato.

Infatti se ripartissimo dalla macchina descritta sopra e provvedessimo a trasformarla in un modello equivalente di Mealey:

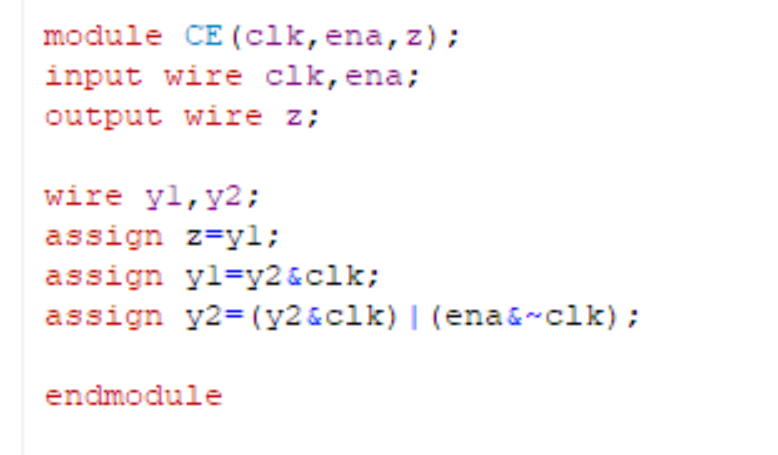
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| st\clk,CE | 00 | 01 | 11 | 10 |
| A | A/0 | A/0 | A/0 | B/0 |
| B | A/0 | - | C/1 | B/0 |
| C | A/0 | C/1 | C/1 | B/0 |

Si Potrebbe facilmente notare che gli stati B e C risultano compatibili e pertanto la macchina si potrebbe semplificare in:

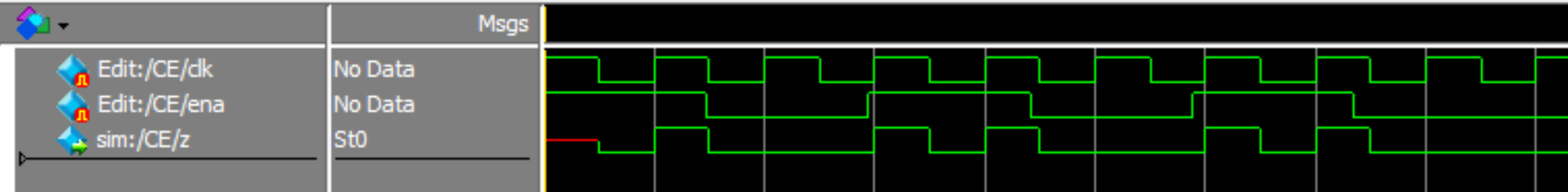
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| st\clk,CE | 00 | 01 | 11 | 10 |
| A | A/0 | A/0 | A/0 | B/0 |
| B | A/0 | B/1 | B/1 | B/0 |

Che opportunamente codificata porterebbe alle medesime equazioni di cui sopra ma riunendo le prime due in una sola :

Il sistema appena sviluppato può essere descritto in VerilogHDL:



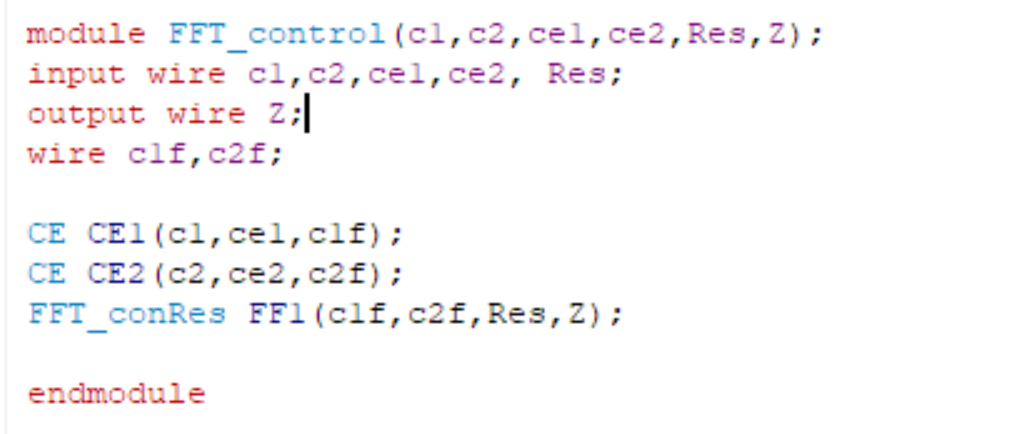
E quindi simulato:



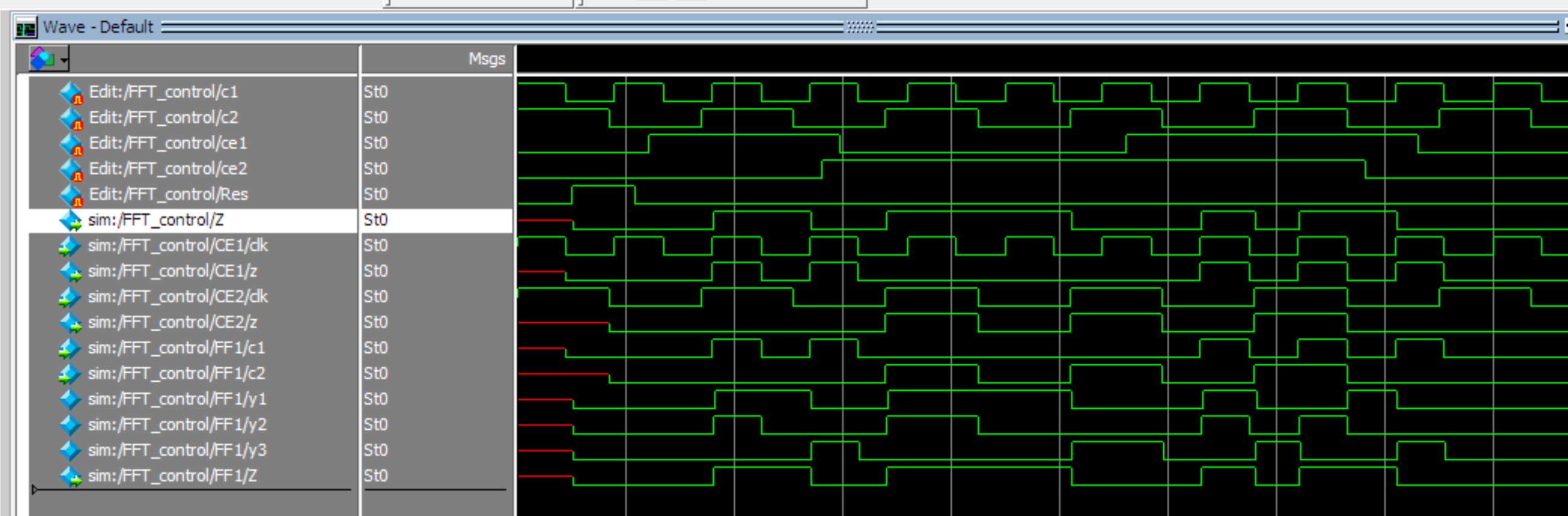
Dimostrando il funzionamento desiderato.

Attraverso l’aggiunta di questa semplice logica di controllo su entrambi i segnali di ingresso del sistema sviluppato in precedenza si può ottenere ora la possibilità di controllarne l’attivazione di entrambi i sui ingressi.

Il sistema completo, descritto in VerilogHDL, sfruttando una descrizione strutturale che connetta tra di loro i blocchi già realizzati:



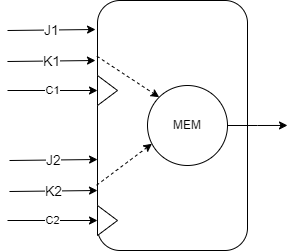
Può essere opportunamente simulata dimostrando il suo corretto funzionamento:



Flip Flop Generico – Segnali di Controllo:

Giunti a questo punto disponiamo di un dispositivo che si comporta come un FlipFlop di tipo T dual Port dotato di segnale di reset asincrono e due segnali di CE1 e CE2 che operano indipendentemente sui due segnali di clock c1 e c2 delle due porte. Grazie all’impiego di questi segnali si può ora integrare il dispositivo con i segnali di controllo che si ritengono più idonei per ciascuna porta.

Supponiamo di voler controllare il FF come se fosse un modello JK e questo per quanto riguarda ciascuna porta, ovvero ciascuna porta, durante il fronte positivo del proprio clock abbia la possibilità di attivarlo, disattivarlo, invertirne lo stato o lasciarlo immutato.



Poiché ogni porta opera indipendentemente dall’altra si può innanzitutto descrivere il funzionamento desiderato dal dispositivo visto nell’ottica di un circuito sincrono, in base ai suoi segnali di ingresso:

L’evoluzione del FF JK è espressa dalla seguente macchina sincrona:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stato\J,K | 00 | 01 | 11 | 10 |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |

Evidenziando in rosso i cambi di stato e lasciando in nero le condizioni in cui lo stato rimane inalterato:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stato\J,K | 00 | 01 | 11 | 10 |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |

Ora per “forzare” i cambi di stato bisognerà attivare il segnale di CE (in corrispondenza all’istante di sincronismo), mentre per non alterare lo stato il segnale CE verrà posto a livello basso.

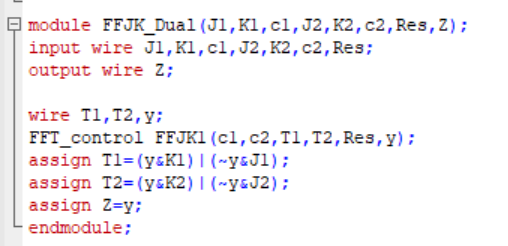
Pertanto la funzione da applicare al segnale CE è la seguente:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stato\J,K | 00 | 01 | 11 | 10 |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 |

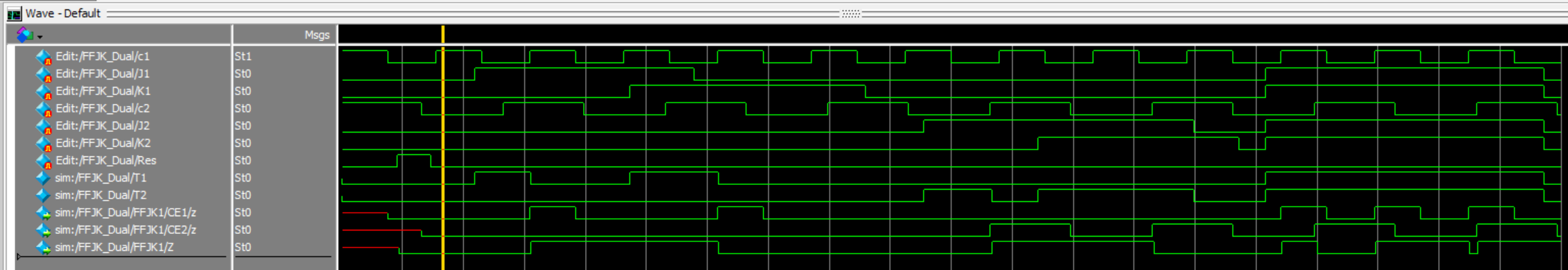
Che può essere formalizzata nella seguente equazione:

Questa relazione può ovviamente essere applicata ora tanto alla porta A che alla porta B del dispositivo, ottenendo quindi un Flip Flop dual port di tipo JK controllabile indipendentemente da due porte separate.

Sfruttando una descrizione strutturale che istanzi i blocchi già sviluppati in precedenza il circuito può venir descritto secondo il seguente codice:



Che opportunamente simulato produce i seguenti segnali da cui si deduce il corretto funzionameto



Si noti che dopo il reset asincrono il FF nella prima parte (con J2=0 e K2=0) viene controllato attraverso i segnali sulla porta 1 (J1 e K1) nella parte centrale (in cui J1=K1=0) il sistema è controllato attraverso i segnali sulla porta 2. Nella parte finale che vede J1=K1=J2=K2=1 il FF commuta ad ogni fronte positivo del clock sia su C1 che su c2.