Introduzione

Questo file è pensato come una grossa raccolta di Domanda - Risposta tratte dai Temi d'Esame di Sistemi di Elettronica Digitale della prof.ssa Alessandra Flammini.

Formato:

- Le domande vanno scritte come intestazione 3
- Le risposte vanno scritte come testo normale
- Riportare sempre qua sotto i dati del TE da cui avete tratto

Data	Domande Tratte	Chi l'ha fatto
2020-12-21 preapp	tutte	wade
Facsimile completo	tutte	wade
Facsimile 2 del 2021	tutte	wade
2022-02-16	tutte	wade
Facsimile 3 del 2021	tutte	wade
2022-01-18	tutte	wade
2021-03-29	Ci sono solo i problemi	wade
facsimile 2020-12-xx	tutte	wade
2021-12-23	Ci sono solo i problemi	wade

TE

Domande chiuse

- 1) Il FANOUT dinamico è pari a.... dove Cin è la capacità di ingresso del ricevitore, Cin0 è la capacità di ingresso del trasmettitore, Cl è la capacità di carico del trasmettitore, Cl0 è la capacità di carico massima per la quale valgono le caratteristiche dinamiche (es. Tphl, Tplh, ...) del trasmettitore

( )a Cl0/Cin
( )b Cin0/Cin
( )c Cl0/Cin0

- 2) Lo stadio di uscita three-state rispetto all'open collector...

( )a ha l'uscita che può assumere un numero maggiore di stati
( )b è più affidabile
( )c viene usato come traslatore di livello

- 3) Un contatore asincrono può ....

( )a necessitare di un segnale di reset Schmidt trigger
( )b soffrire del problema del disallineamento (skew) delle uscite
( )c necessitare di un segnale di reset open collector

- 4) Un oscillatore può essere realizzato mediante un dispositivo...

( )a con ingresso Schmidt trigger
( )b con uscita open collector
( )c con uscita 3-state

- 5) Una memoria ... perde i dati quando viene tolta alimentazione

( )a ROM
( )b DRAM
( )c EEPROM

- 6) In VHDL il mode BUFFER di un signal indica un segnale ...

( )a che ha elevato FANOUT
( )b che è un'uscita ma con la possibilità di essere letto
( )c che può essere configurato come ingresso o come uscita

- 7) In Arduino la funzione Tone( ) è relativa a ....

( )a Timing Processing Unit

( )b interfaccia seriale

( )c convertitore A/D

2) Lo stadio di uscita open-collector rispetto al three-state ...

( )a ha l'uscita che può assumere un numero maggiore di stati

( )b è più veloce

( )c viene usato come traslatore di livello

3) Un contatore sincrono può ....

( )a soffrire del problema della metastabilità

( )b soffrire del problema del disallineamento (skew) delle uscite

( )c necessitare di un segnale di reset open collector

5) Una memoria ... non perde i dati quando viene tolta alimentazione

( )a RAM

( )b DRAM

( )c EEPROM

6) In VHDL il mode BUFFER di un signal indica un segnale ...

( )a che ha elevato FANOUT

( )b che è un'uscita ma con la possibilità di essere letto

( )c che può essere configurato come ingresso o come uscita

7) In Arduino la funzione pulseIn( ) è relativa a ....

( )a Timing Processing Unit

( )b interfaccia seriale

( )c convertitore A/D

2) Nei dispositivi con stadio di ingresso Schmidt trigger...

( )a L'uscita è sempre ben posta anche se l'ingresso è in zona di incertezza

( )b Le uscite si possono collegare insieme

( )c Il FAN-OUT e l'immunità al rumore sono ideali

3) In una porta NOT CMOS con l'uscita a "1" conduce:

( )a il transistore PMOS

( )b il transistore NMOS

( )c entrambi i transistori NMOS e PMOS

4) In un contatore sincrono ...

( )a non può esserci Reset asincrono

( )b tutti i flip-flop hanno le linee di clock collegate insieme

( )c ci sono degli stati spuri delle uscite per cui serve uno stadio di sincronizzazione

5) La porta XOR

( )a porta sull'uscita selezionata la linea di ingresso attiva

( )b porta sull'uscita selezionata l'unica linea di ingresso

( )c è un negatore programmabile

6) La memoria DRAM

( )a mantiene i dati anche senza alimentazione

( )b può essere scritta solo dal costruttore

( )c perde i dati in mancanza di alimentazione

7) I dispositivi .... hanno la matrice di AND programmabile e la matrice di OR fissa

( )a PLA

( )b PLE

( )c GAL

1) Se l'ingresso di un dispositivo digitale è a zero ...

( )a l'uscita assorbe corrente

( )b l'ingresso assorbe corrente

( )c l'ingresso eroga corrente

2) Lo stadio di uscita open-collector rispetto al three-state ...

( )a è più affidabile

( )b è più veloce

( )c è usato nei bus dati

3) Un sommatore....

( )a soffre del problema della metastabilità

( )b ha un tempo di propagazione che dipende dal numero di bit della parola da confrontare

( )c è un dispositivo asincrono dove il clock dei flip-flop non è collegato insieme

4) Un flip-flop T...

( )a campiona l'ingresso sul fronte positivo del clock

( )b insegue l'ingresso per tutta la durata a uno del segnale di clock

( )c commuta l'uscita ad ogni fronte positivo del clock

5) Una memoria EPROM ...

( )a è scritta dal costruttore e non perde mai i dati

( )b perde i dati in assenza di rinfresco periodico

( )c si cancella solo mediante raggi ultravioletti

6) Se si vuole collegare un pulsante ad Arduino ....

( )a Si collega il pulsante tra massa e un pin di ingresso che si configura come INPUT_PULLUP: se si legge '1' il pulsante è premuto, se si legge '0' il pulsante non è premuto

( )b Si collega il pulsante tra massa e un pin di ingresso che si configura come INPUT_PULLUP: se si legge '0' il pulsante è premuto, se si legge '1' il pulsante non è premuto

( )c Si collega il pulsante tra tensione di alimentazione e un pin di ingresso che si configura come INPUT: se si legge '1' il pulsante è premuto, se si legge '0' il pulsante non è premuto

7) Nei microcontrollori si trovano difficilmente i convertitori D/A perché al loro posto si usano....

( )a uscite PWM

( )b uscite output compare

( )c GPIO

1)Con il simbolo Voh si intende....

( )a La minima tensione in uscita quando l'uscita e' a livello "1" ed entra una corrente Ioh

( )b La minima tensione in uscita quando l'uscita e' a livello "1" ed esce una corrente Ioh

( )c La minima tensione in uscita quando l'uscita e' a livello "1" e si opera a vuoto

2) Nei dispositivi CMOS con stadio di uscita open-drain manca il transistore PMOS di uscita

( )a Vero

( )b Falso

( )c Falso ma solo nei dispositivi ad arricchimento

3) In una porta NOT CMOS con l'uscita a "0" conduce:

( )a il transistore PMOS

( )b il transistore NMOS

( )c entrambi i transistori NMOS e PMOS

4) In un latch di tipo D...

( )a l'uscita Q è pari all'ingresso Enable/Clock in corrispondenza del fronte di salita di D

( )b l'uscita Q è pari all'ingresso D in corrispondenza del fronte di salita di Enable/Clock

( )c l'uscita Q insegue l'ingresso D per tutta la durata di Enable/Clock="1"

5) Il multiplexer

( )a porta sull'uscita selezionata la linea di ingresso attiva

( )b porta sull'uscita selezionata l'unica linea di ingresso

( )c porta sull'unica uscita la linea di ingresso selezionata

6) La rete R/2R

( )a viene utilizzata nei convertitori A/D ad approssimazioni successive

( )b viene utilizzata nei convertitori A/D di tipo sigma-delta

( )c viene utilizzata nei convertitori D/A

7) Nei dispositivi .... Non è possibile realizzare funzioni asincrone (con più linee di clock)

( )a FPGA

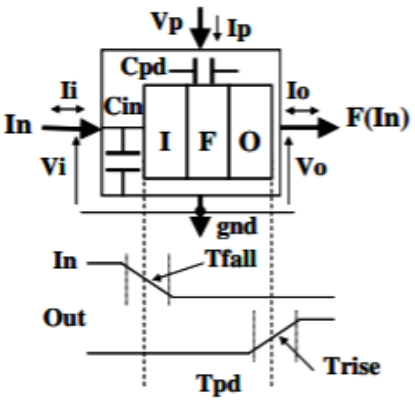
( )b ASIC

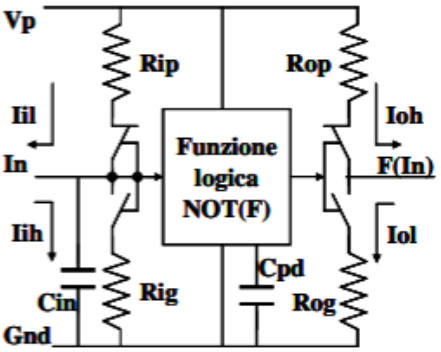
( )c GAL

Domande Aperte

- Dato il modello rappresentato in figura, definire l'immunità al rumore e il FAN-OUT statico e dinamico (1 punto). Definire la dissipazione di potenza dinamica e la sua dipendenza dalle capacità riportate nel modello. (1 punto)

Problema a)





L'**immunità al rumore** Vn è definita come il minimo tra l'immunità al rumore a zero Vn0 e l'immunità al rumore a uno Vn1, ossia il massimo rumore che è possibile sommare/sottrarre al segnale in ingresso mantenendo la certezza del corretto riconoscimento.

Vn = Min(Voh-Vih; Vil-Vol) e normalmente Vn>100mV

Il **FAN-OUT statico** indica il massimo numero di carichi pilotabili

FAN-OUT = min(Ioh/Iih; Iol/Iil) e normalmente FAN-OUT>>1

Le relazioni tra tensioni e correnti sono le seguenti:

La **dissipazione di potenza dinamica** è pari a Pd = Vp2*f*(Cl+Cpd), dove Vp è la tensione di alimentazione, f la frequenza del segnale in ingresso, Cpd è la capacità interna e Cl è la capacità di carico, pari alla somma delle capacità Cin dei dispositivi pilotati.

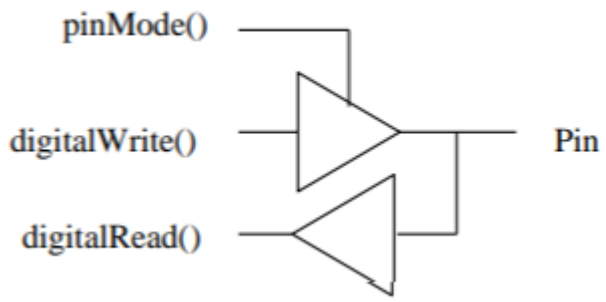
La **capacità interna** Cpd, oltre a descrivere l'effetto delle capacità interne al dispositivo, include l'effetto delle correnti di breve durata che si possono avere durante la commutazione e che vengono assimilate a processi di carica e scarica di capacità.

- Descrivere i buffer 3-state, anche mediante tabella della verità, indicando un vantaggio e uno svantaggio rispetto ai buffer open collector (1 punto). Si descriva come i three-state siano impiegati nei GPIO dei microcontrollori, anche con riferimento alle istruzioni Arduino pinMode(), digitalRead() e digitalWrite() (2 punti).

Un **buffer** è un componente con funzione logica Identità o NOT (buffer invertente) e tipicamente ha una funzione elettrica. I buffer 3-state e i buffer open collector sono utilizzati per la capacità di avere in uscita lo stato "z" di alta impedenza, che permette di avere più uscite connesse insieme. I buffer 3-state sono più veloci e versatili, mentre i buffer open-collector sono più affidabili in quanto più semplici e tolleranti al guasto in quanto, non disponendo dello stato "1" attivo, non possono generare conflitto.
Il buffer 3-state è il buffer che, sulla base degli ingressi EN e IN, può avere in uscita i tre stati logici: 1, 0, Z (alta impedenza).

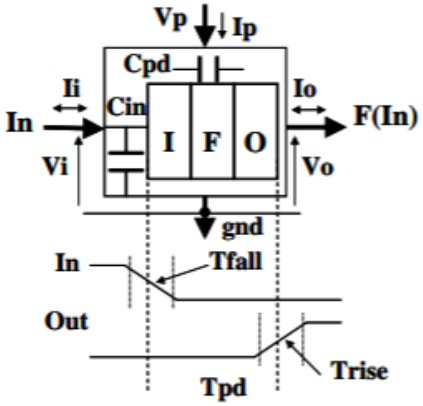
EN	IN	OUT
0	X	Z
1	0	0
1	1	1

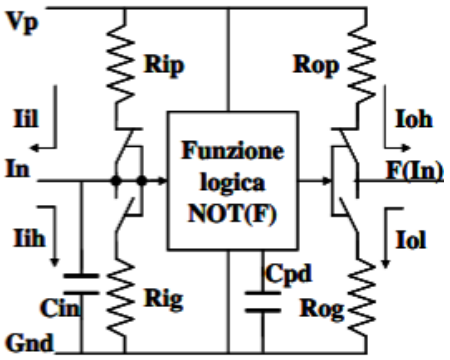
I **GPIO** (General Purpose Input Output) dei microcontrollori sono pin che possono essere configurati come ingresso, come uscita, come bit isolati o come bit facenti parte di una periferica. Con riferimento al disegno seguente, con l'istruzione pinMode() si va ad egire sulla linea EN di un three state: infatti se pinMode(<pin>, INPUT) allora EN=0 e l'accesso al pin avviene attraverso la digitalRead(), se invece pinMode(<pin>, OUTPUT) allora EN=1 e l'accesso al pin può avvenire sia con la digitalRead() che con la digital Write().



- Dato il modello rappresentato in figura, definire Tphl e Tplh e da quali termini è composto (1 punto). Se una porta NAND alimentata a 3V ha Tphl,typ=Tplh,typ=8ns con Cl0=50pF e Cin=5pF e Trise=Tfall=1ns+Cl*40ps/pF, si calcolino i tempi tipici a vuoto, con 10 carichi e con 200 carichi (1 punto).

Problema a)





I **tempi di propagazione** sono composti dalla somma di tre termini: un termine Ti che tiene conto dei ritardi dei circuiti di ingresso del dispositivo, un tempo Tf che tiene conto del ritardo di propagazione attraverso lo stadio funzionale, e un tempo To, tipicamente pari a 0,5 Trise o 0,5 Tfall, che tiene conto dei ritardi dello stadio di uscita.

In particolare si ha: Tphl = Ti + Tf + 0,5 Tfall e Tplh = Ti + Tf + 0,5 Trise

A vuoto Cl = 0 e quindi Tplh = Tphl = 8ns – 0,5(1ns+2ns) + 0,5(1ns) = 7ns

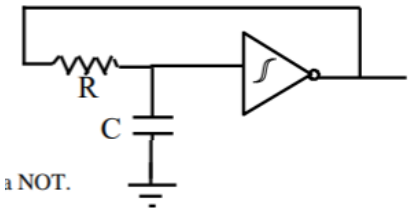
Con 10 carichi Cl = 50pF = Clo e quindi Tphl=Tplh=8ns

Con 200 carichi Cl = 1000pF e quindi Tplh = Tphl = 8ns – 0,5(1ns+2ns) + 0,5(1ns+40ns) = 27ns

Nel caso di 200 carichi il tempo di propagazione fa si che il segnale permanga nella regione di incertezza per un tempo superiore al tempo di reazione del dispositivo, no garantendone il corretto funzionamento.

- Descrivere gli oscillatori ad anello e gli oscillatori RC con Schmidt trigger e si valuti il periodo di oscillazione nel caso dei seguenti livelli: R=10k, C=100nF, Vol=0,1V, Voh=3,2V, Vtn=1,1V e Vtp=2,2V. (2 punti)

Gli oscillatori ad anello sono costituiti da una serie di M porte NOT con tempi di propagazione Tphl e Tplh, dove M è un numero dispari (M=2N+1) e il periodo di oscillazione T è pari a M(Tphl+Tplh). L'oscillatore RC con porta NOT Schmidt trigger si basano sul ritardo introdotto da una rete RC secondo quanto riportato in figura.

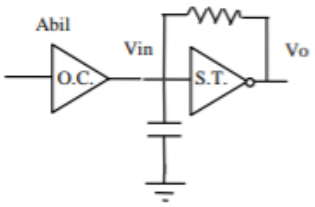


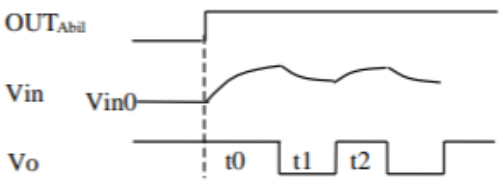
La carica e scarica del condensatore è governata dalla legge

$$Vin(t) = Vin(0) + (Vin(\infty)-Vin(0))(1-e^{-t/\tau}) \text{ dove } \tau=R\cdot C$$

E Vin(t) indica la tensione al punto di ingresso della porta NOT.

A parte il transitorio iniziale (t0), il periodo di oscillazione è dato da t1+t2, dove t1 indica il tempo in cui l'uscita della porta NOT è alta e il segnale Vin varia da Vtn a Vtp, mentre t2 indica il tempo in cui l'uscita della porta NOT è bassa e il segnale Vin varia da Vtp a Vtn

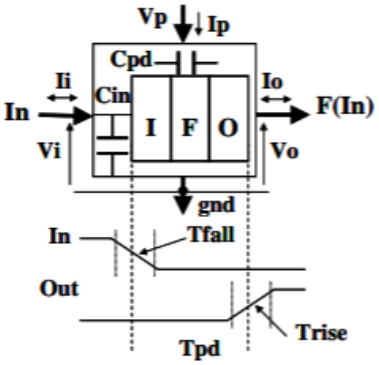


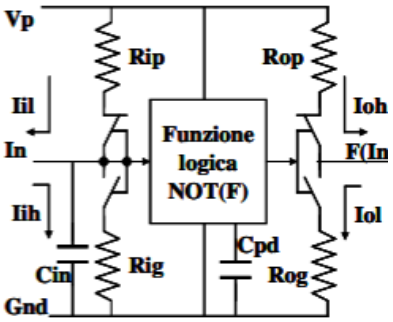


Sostituendo i valori indicati nel testo si ottiene τ=1ms, t1=0,74ms, t2=0,74ms e quindi T= 1,48ms

- Dato il modello rappresentato in figura, definire il problema della regione di incertezza, l'immunità al rumore e il FAN-OUT statico e dinamico (1 punto). Motivare perché non si può avere Rop=Rip=Rig=Rog. (1 punto)

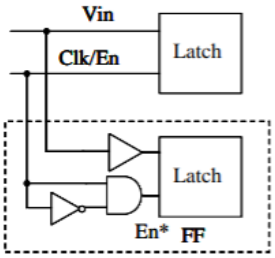
La **regione di incertezza** è data dai valori di tensione Vin applicati in ingresso tali che Vil<Vin<Vih;
in queste condizioni non c'è certezza che sia correttamente riconosciuto il livello logico e di conseguenza l'uscita è indeterminata e tale condizione permane fintanto che l'ingresso Vin è in regione di incertezza.

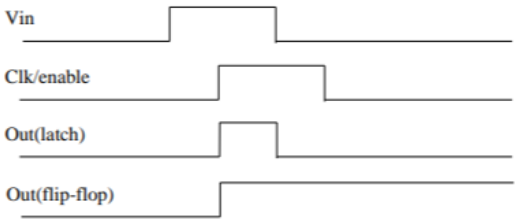




L'**immunità al rumore** Vn è definita come il minimo tra l'immunità al rumore a zero Vn0 e l'immunità al rumore a uno Vn1, ossia il massimo rumore che è possibile sommare/sottrarre al segnale in ingresso mantenendo la certezza del corretto riconoscimento.
Vn = Min(Voh-Vih; Vil-Vol) e normalmente Vn>100mV
Il **FAN-OUT statico** indica il massimo numero di carichi pilotabili
FAN-OUT = min(Ioh/Iih; Iol/Iil) e normalmente FAN-OUT>>1
Le relazioni tra tensioni e correnti sono le seguenti:
Iol = Vol/Rog Ioh = (Vp-Voh)/Rop Iil = (Vp-Vil)/Rip Iih = Vih/Rig
Ma per il FAN-OUT, si ha che Iol/Iil>>1 ossia (Rip/Rog)((Vol/(Vp-Vil))>>1 ma Vol/(Vp-Vil)<1 e quindi Rip/Rog>>1 ossia Rip>>Rog. Si perviene ad analoghe conclusioni, ossia Rig>>Rop, partendo da Ioh/Iih>>1

- Illustrare le principali differenze tra Latch e Flip-Flop anche mediante diagramma dei segnali. Definire in cosa consiste la metastabilità e quali sono le possibili cause d'innesco e cosa è il resolving time. (2 punti)





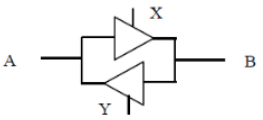
Il **Latch** è un elemento di memoria che riporta l'ingresso D in uscita Q per tutto il tempo in cui l'ingresso di abilitazione En rimane attivo (alto nel diagramma dei segnali), mentre il **Flip-Flop** riporta l'ingresso in uscita campionando in corrispondenza del fronte di salita del clock Ck.
La **metastabilità** è uno stato in cui l'uscita di un dispositivo sequenziale è impredicibile e cioè può assumere qualunque valore con qualsiasi dinamica. Si tratta di un fenomeno che tende a decadere in uno stato stabile ma non predicibile secondo una legge probabilistica che segue un andamento esponenziale e pertanto un'attesa di qualche centinaio di ns porta, nella maggior parte dei casi, la probabilità di sopravvivenza dello stato metastabile allo stesso ordine di grandezza della probabilità di guasto del componente.
Nei circuiti sequenziali attivati da un clock (flipflop) i segnali di ingresso devono essere stabili per un certo tempo prima che il clock venga applicato ("set-up time" Tset-up) e devono rimanere tali per un tempo dopo che la transizione è avvenuta ("hold time" Thold): se tali condizioni non venissero rispettate il flip-flop potrebbe entrare in uno stato metastabile. Inoltre l'impulso di clock, così come gli impulsi di set o di reset, devono avere una durata minima Tw perché possano essere riconosciuti ed eseguiti senza causare metastabilità.
Il **resolving time** è il tempo entro il quale la metastabilità decade verso uno stato stabile e la probabilità di sopravvivenza dello stato metastabile è confrontabile con la probabilità di guasto.

- Dato il modello rappresentato in figura, definire la capacità Cpd indicando di quali fenomeni tiene conto (1 punto). Definire la dissipazione di potenza statica e dinamica indicando le relative formule (1 punto)

La **capacità Cpd** è un modello che tiene conto degli effetti capacitivi interni al dispositivo, come ad esempio le capacità interne e le capacità di uscita, e delle correnti di breve durata che possono essere assimilate a trasferimenti di carica, come ad esempio le correnti che scorrono durante la commutazione dei transistors o le correnti di carica e scarica delle capacità parassite.

La **dissipazione di potenza statica** Ps=Vp*Ip è data dal prodotto tra la tensione di alimentazione Vp e la corrente Ip che scorre tra il dispositivo, l'alimentazione e la massa.
La **dissipazione di potenza dinamica** Pd=f*Vp2*(Cpd+Cl) dipende dalle correnti di carica e di scarica delle capacità di carico Cl e della capacità interna Cpd, dalla frequenza f delle commutazioni e dal quadrato della tensione di alimentazione Vp.

- Illustrare le differenze tra i dispositivi con uscita open-collector e dispositivi three-state (1 punto). Descrivere il transceiver in termini di segnali: A, B, Enable, Direction e come Enable e Direction sono legati ai segnali X e Y in figura (1 punto)

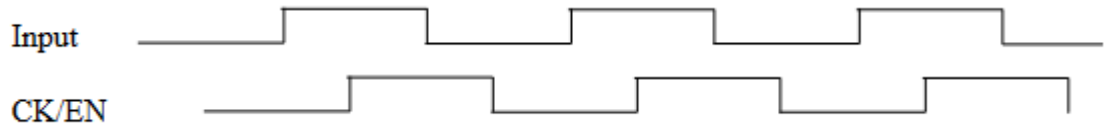


I dispositivi con uscita **open collector** possono commutare la propria uscita tra due stati: 0 e Z, ossia alta impedenza; per poter ricostruire l'1 hanno bisogno di resistenze esterne di pull-up. Avendo solo uno stato attivo a bassa impedenza non vi può essere conflitto nel caso di due o più uscite connesse insieme.
I dispositivi open-collector sono molto affidabili, ma lenti nella commutazione da 0 a 1. I dispositivi open-collector sono semplici, affidabili ed economici e permettono di pilotare tanta corrente.
I dispositivi con uscita **three-state** possono commutare la propria uscita tra 3 livelli: 0, 1 e Z. Rispetto ai dispositivi open-collector sono più veloci ma necessitano di un protocollo per evitare che due dispositivi con l'uscita collegata insieme possano essere in conflitto.
Il **transceiver** è un buffer bidirezionale costituito da due dispositivi three-state in antiparallelo e permette di regolare il traffico tra A e B. Se Enable è a 0, allora A e B sono isolati; se Enable è a 1, allora Direction determina se il flusso è da A a B oppure da B ad A.
X = Enable&Direction
Y = Enable&!Direction

- Dato il modello rappresentato in figura, descrivere le relazioni tra correnti e tensioni in ingresso e in uscita a livello 1 e a livello 0. (1 punto)
  Definire il FANOUT statico e il FANOUT dinamico e descrivere la capacità Cpd e quali effetti rappresenta. (1 punto)

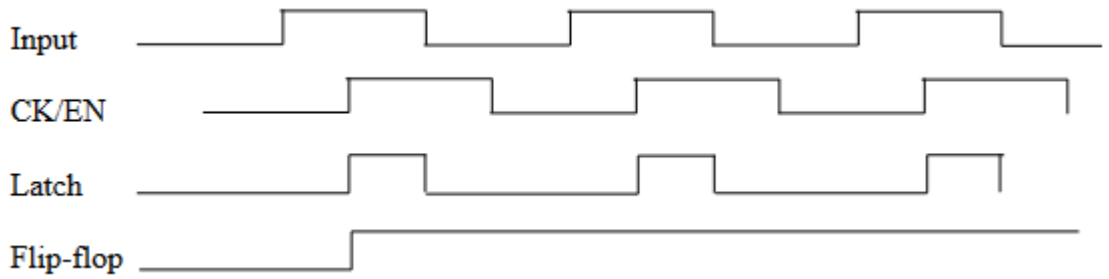
Iil = (Vp-Vil)/Rip
Iih = Vih/Rig
Ioh = (Vp-Voh)/Rop
Iol = Vol/Rog
Il FANOUT statico è = min (Ioh/Iih, Iol/Iil) e rappresenta la capacità di pilotaggio di dispositivi elettronici verso altri dispositivi elettronici, ossia il numero massimo di carichi. Se un dispositivo pilota un numero di carichi superiore al FANOUT non è garantito il corretto funzionamento. Il FANOUT dinamico è = min (Clo/Cl), dove Clo è la massima capacità di carico per la quale sono garantite le prestazioni dinamiche dichiarate; rappresenta il massimo numero di carichi che è possibile pilotare senza degrado delle prestazioni.
La capacità interna Cpd modellizza gli effetti di trasferimenti di carica e/o di capacità parassite e/o di correnti impulsive di breve durata che si hanno durante la commutazione. Include ad esempio l'effetto della capacità di gate degli stadi interni e la corrente che fluisce negli interruttori durante il transitorio di apertura e/o di chiusura.

- Si descrivano brevemente le funzionalità di latch e flip-flop, anche indicando graficamente la risposta ai seguenti segnali e motivando se vi sono condizioni di metastabilità



Un **latch** è l'elemento base di memoria che, se la linea di Enable è attiva, riporta in uscita il valore di ingresso, come un buffer; se invece la linea di Enable è inattiva, allora il dispositivo mantiene inalterato il valore dell'uscita.
Il **flip-flop** è invece un latch con un segnale di Enable estremamente stretto che in pratica campiona l'ingresso solo in corrispondenza di questa finestra molto stretta. La finestra, la cui durata è pari alla somma tra il tempo di hold (tempo in cui il dato deve essere stabile dopo l'evento di clock) e il tempo di set-up (tempo in cui il dato deve essere stabile prima dell'evento di clock), è normalmente progettata per agire in corrispondenza del fronte di salita del clock.

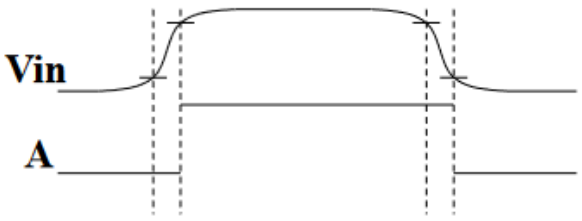
Si ha metastabilità in caso di mancato rispetto dei tempi di set-up di hold e se l'impulso di clock non rispetta un valore di minima larghezza, normalmente della durata della finestra di cui sopra.
Con riferimento al grafico in figura si ha:



- Descrivere come e perché la porta Schmidt Trigger rivolve il problema della zona di incertezza (2 punti)

Si ha zona di incertezza quando la tensione di ingresso Vin(t) è compresa tra Vil e Vih e il segnale non viene classificato con certezza con il suo valore booleano A(t), dove A(t) = 1 se Vin(t)>Vih e A(t)=0 se Vin(t)<Vil.
Il dispositivo Schmidt Trigger introduce nell'ingresso una sorta di elemento di memoria, che permette di valutare il valore di tensione all'istante t e il valore booleano A(t-τ) di corretto riconoscimento dell'istante t-τ immediatamente precedente e, nel caso di segnale d'ingresso che entra in zona di incertezza, si mantiene il valore correttamente riconosciuto un attimo prima.

Vin(t)	**A(t-τ)**	**A(t)**
< Vtn	X	"0"
Vtn<Vin<Vtp	"0"	"0"
Vtn<Vin<Vtp	"1"	"1"
> Vtp	X	"1"



- Dato il modello rappresentato in figura, definire il problema della regione di incertezza, l'immunità al rumore e il FAN-OUT statico e dinamico (1 punto). Motivare perché si deve avere Ri>Ro (1 punto)

Per regione di incertezza, immunità al rumore e FAN-OUT ci sono altre almeno 2 o 3 risposte. Anche la Ri>Ro eh però ecco

Le relazioni tra tensioni e correnti sono le seguenti:

Iol = Vol/Rog

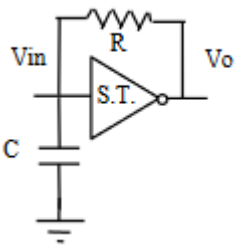
Ioh = (Vp-Voh)/Rop

Iil = (Vp-Vil)/Rip

Iih = Vih/Rig

Ma per il FAN-OUT, si ha che Iol/Iil>>1 ossia (Rip/Rog)((Vol/(Vp-Vil))>>1 ma Vol/(Vp-Vil)<1 e quindi Rip/Rog>>1 ossia Rip>>Rog. Si perviene ad analoghe conclusioni, ossia Rig>>Rop, partendo da Ioh/Iih>>1.

- Illustrare le differenze tra i dispositivi con ingresso Schmidt trigger e quelli con stadio di ingresso normale (1 punto) e calcolare Ton e Toff nell'oscillatore a Schmidt trigger basato sul dispositivo 74HC132 le cui caratteristiche sono: Vol = 0,1V Voh = 4,9V Vtp = 2,8V Vtn = 1,9V R = 10kOhm C = 100nF (1 punto)

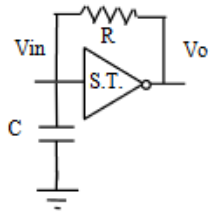


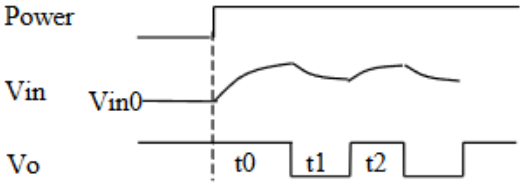
Il dispositivo con ingresso Schmidt trigger è stato progettato per risolvere il problema della zona di incertezza. La tabella seguente riporta il funzionamento di un dispositivo con ingresso normale e di un dispositivo con ingresso Schmidt trigger (per semplicità grafica si riporta con il simbolo Vin- il valore di Vil di un dispositivo con ingresso normale e il valore di Vtn di un dispositivo con ingresso Schmidt trigger (analogamente per Vin+, Vih e Vtp).

Vin	Input di dispositivo normale	Input di dispositivo Schmidt trigger
0<=Vin<=Vin-	"0"	"0"
Vin-<Vin=Vin+	? (ingresso non valido)	Memoria (ultimo ingresso valido)
Vin+<Vin<=Vcc	"1"	"1"

Si definisce Isteresi Vh=Vt+ - Vt-. Cambia anche l'immunità al rumore, nel senso che se l'uscita di una porta pilota è a "0", ossia a Vol, il dispositivo pilotato continuerà a vedere uno "0" anche in presenza di un rumore Vn0 = Vt+ - Vol. Il dispositivo con

ingresso Schmidt trigger è quindi sempre in grado di lavorare con un ingresso valido, tuttavia è più lento e consuma leggermente di più.
Il circuito ha il seguente comportamento:
- Quando l'ingresso Vin<Vin- allora l'uscita si porta a Voh, di conseguenza C inizia a caricarsi attraverso R fino a quando non raggiunge Vt+ (punto finale di t0)
- Non appena Vin raggiunge il valore di Vt+, allora l'uscita commuta da Voh a Vol, per cui la capacità C inizia a scaricarsi attraverso R, fino a quando non raggiunge Vt- (punto finale di t1)
- Non appena Vin raggiunge il valore di Vt-, allora l'uscita commuta da Vol a Voh, per cui la capacità C inizia a caricarsi attraverso R, fino a quando non raggiunge Vt+ (punto finale di t2)





La legge di carica e scarica di C è $Vin(t) = (Vfinale - Viniziale)(1 - e^{-t/RC}) + Viniziale$
Vin=Vtn t1 = RC*ln((Vtp-Vol)/(Vtn-Vol)) = Toff = 0,405 ms
Vin=Vtp t2 = RC*ln(Voh-Vtn)/(Voh-Vtp) = Ton = 0,357 ms

- Dato il modello rappresentato in figura, definire Tphl e Tplh e da quali termini è composto (1 punto). Se una porta NAND alimentata a 3V ha Tphl,typ=Tplh,typ=8ns con Cl0=50pF e Cin=5pF e Trise=Tfall=1ns+Cl*40ps/pF, si calcolino i tempi tipici a vuoto, con 10 carichi (1 punto) e con 200 carichi (1 punto).

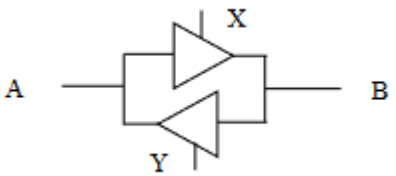
I tempi di propagazione sono composti dalla somma di tre termini: un termine Ti che tiene conto dei ritardi dei circuiti di ingresso del dispositivo, un tempo Tf che tiene conto del ritardo di propagazione attraverso lo stadio funzionale, e un tempo To, tipicamente pari a 0,5 Trise o 0,5 Tfall, che tiene conto dei ritardi dello stadio di uscita.
In particolare si ha: Tphl = Ti + Tf + 0,5 Tfall e Tplh = Ti + Tf + 0,5 Trise
A vuoto Cl = 0 e quindi Tplh = Tphl = 8ns – 0,5(1ns+2ns) + 0,5(1ns) = 7ns
Con 10 carichi Cl = 50pF = Clo e quindi Tphl=Tplh=8ns

Con 200 carichi Cl = 1000pF e quindi Tplh = Tphl = 8ns – 0,5(1ns+2ns) + 0,5(1ns+40ns) = 27ns
Nel caso di 200 carichi il tempo di propagazione fa si che il segnale permanga nella regione di incertezza per un tempo superiore al tempo di reazione del dispositivo, non garantendone il corretto funzionamento.

- Dato il modello dinamico riportato si definiscano i tempi di propagazione Tphl e Tplh, il tempo di salita e il tempo di discesa (1 punto)
  Si descriva come si stima la massima frequenza operativa Fmax e su quale componente del tempo di propagazione Tpd ha influenza Cin.

Il tempo di salita **Trise** è il tempo che passa da quando il segnale raggiunge il 10% dell'intera commutazione positiva a quando raggiunge il 90%
Il tempo di discesa **Tfall** è il tempo che passa da quando il segnale raggiunge il 10% dell'intera commutazione negativa a quando raggiunge il 90%
Tplh = tempo di propagazione da ingresso a uscita per una commutazione dell'uscita da basso a alto (Test Conditions: Cl=Clo, Vp=typ., T=20°C)
Tphl = tempo di propagazione da ingresso a uscita per una commutazione dell'uscita da alto a basso (Test Conditions: Cl=Clo, Vp=typ., T=20°C)
Tphl = Ti + Tf + To = Ti + Tf + Tfall/2
Tplh = Ti + Tf + To = Ti + Tf + Trise/2
Dove Ti è la componente del ritardo di propagazione che dipende dallo stadio I di ingresso e Tf la componente che dipende dallo stadio funzionale F. Tfall e Trise sono in genere funzioni lineari della capacità di carico Cl = ΣCin quindi la capacità d'ingresso Cin dei dispositivi pilotati influenza il tempo di uscita To del dispositivo pilota. La **massima frequenza operativa** del circuito **fmax ~ (Tphl+Tplh)^(-1)**

- Si illustri il principio di funzionamento di un transceiver. (1 punto)
  Si illustri il funzionamento delle linee !Select e Direction dei dispositivi reali transceiver rispetto ai segnali X e Y indicati in figura. (1 punto)
- Si illustri come sono configurate le linee di !Select e Direction di un GPIO Arduino in un digitalRead e in un digitalWrite. (1 punto)



Un transceiver è composto da due buffer three-state connessi in antiparallelo e viene utilizzato per disaccoppiare bus bidirezionali. Infatti se X=1 e Y=0 il flusso dei dati va da A verso B, mentre se X=0 e Y=1 il flusso dati viaggia nella direzione opposta. Se X=Y=0 A e B sono isolati ed e' opportuno fissare il punto di riposo di A e B in modo che le linee non rimangano in alta impedenza. La configurazione X=Y=1 non è ammessa.
Nei transceiver reali, per evitare la configurazione non ammessa X=Y=1, i transceiver vengono pilotati mediante le linee !Select (attiva bassa) e Direction (Dir=1 -> flusso da A verso B) che non coincidono con le linee X e Y

!Select	Direction	Operazione	X	Y	
0	0	Flusso da B a A	0	1	X = Direction&Select
0	1	Flusso da A a B	1	0	Y = ! Direction&Select
1	X	A e B isolati	0	0	

GPIO di Arduino sono costituiti da transceiver dove si ipotizzi la linea A interna al dispositivo e la linea B connessa al pin verso l'esterno del dispositivo. Al reset il transceiver è configurato con Select=Direction=0. Durante una digitalRead si ha Select=1 e Direction=0, mentre durante una digitalWrite si ha Select=1 e Direction=1

- Dato il modello rappresentato in figura, definire:
  - i livelli elettrici statici (Iol, Ioh, Vol, Voh, Iil, Iih, Vil, Vih) (1 punto),
  - le relazioni che ne legano i valori a Rip, Rig, Rop e Rog (1 punto),
  - il FAN-OUT statico e dinamico e la dissipazione di potenza statica e dinamica (1 punto).

Iol = massima corrente assorbita dall'uscita quando l'uscita è a "0"
Ioh = massima corrente erogata dall'uscita quando l'uscita è a "1"
Iil = massima corrente erogata dall'ingresso quando l'ingresso è a "0"
Ioh = massima corrente assorbita dall'ingresso quando l'ingresso è a "1"
Vol = massima tensione di uscita quando l'uscita è a "0" e assorbe una corrente pari a Iol
Voh = minima tensione di uscita quando l'uscita è a "1" ed eroga una corrente pari a Ioh

Vil = massima tensione di ingresso riconosciuta come “0”
Vih = minima tensione di ingresso riconosciuta come “1”

Vol = Iol*Rog
Voh = Vp – Ioh*Rop
Vih = Iih*Rig
Vil = Vp – Iil*Rip

FAN-OUT **statico** = min(Iol/Iil , Ioh/Iih)
FAN-OUT **dinamico** = Clo/Cin dove Clo è la capacità di carico del dispositivo pilota per la quale sono definiti i tempi di propagazione e Cin è la capacità di ingresso dei dispositivi pilotati.
La **dissipazione di potenza statica** è pari a Ps = Vp*Il, dove Vp è la tensione di alimentazione e Il è la corrente scambiata con l’alimentazione in assenza di commutazioni.
La **dissipazione di potenza dinamica** è pari a Pd = Vp2*f*(Cl+Cpd), dove Vp è la tensione di alimentazione, f la frequenza del segnale in ingresso, Cpd è la capacità interna e Cl è la capacità di carico, pari alla somma delle capacità Cin dei dispositivi pilotati. La capacità interna Cpd, oltre a descrivere l’effetto delle capacità interne al dispositivo, include l’effetto delle correnti di breve durata che si possono avere durante la commutazione e che vengono assimilate a processi di carica e scarica di capacità.

- Descrivere il funzionamento delle seguenti funzioni Arduino:
  - delay() e millis() (1 punto)
  - pulseIn() (1 punto)

delay() è la funzione che attende un numero di millisecondi che è argomento della funzione. Esiste anche la delayMicroseconds() che attende un numero di microsecondi che è argomento della funzione, per cui delay(4) blocca il programma per 4 ms analogamente a quanto farebbe la delayMicroseconds(4000).

millis() è la funzione che restituisce il numero di millisecondi trascorsi dal reset o dal Power ON; in pratica è la lettura di una variabile intera a 32 bit che si incrementa ogni millisecondo.
micros() è la funzione che, analogamente alla millis(), restituisce il numero di microsecondi trascorsi dal reset o dal Power ON; in pratica è la lettura di un contatore a 32 bit con clock a 1MHz che costituisce il timer di sistema.
pulseIn() è la funzione che restituisce la durata in microsecondi di un impulso di un segnale letto ad un certo pin configurato come INPUT o INPUT_PULLUP. La funzione può essere utilizzata in due modi:
- pulseIn(pin, value), dove pin è il pin del segnale di cui si vuole misurare la larghezza di impulso, e value è HIGH o LOW a seconda che si voglia misurare la durata di un impulso a 1 o a 0.
- pulseIn(pin, value, timeout), dove timeout è un tempo massimo, espresso in microsecondi, entro il quale la misura deve essere conclusa, altrimenti la misura si interrompe e la funzione restituisce 0.
La funzione pulseIn() attende che il segnale si porti a !value, quindi attende che si porti a value e subito registra il valore A del timer di sistema mediante una chiamata alla funzione micros() quindi attende che il segnale si porti a !value e subito registra il valore B del timer di sistema mediante una chiamata alla funzione micros() e restituisce il valore B-A

Esercizi

- 1) Si realizzi mediante linguaggio booleano per una GAL22V10 un contatore C in decremento modulo 10 (che conta da 9 a 0) con reset asincrono ARESET e load sincrono LOAD che carica il valore da 4 bit di ingresso L3-L0. (2 punti).
- 2) Si realizzi la funzione del punto precedente in linguaggio VHDL (1 punto), evidenziando quali segnali di ingresso potrebbero portare a metastabilità (1 punto)
- 3) Modificare il programma del punto 2 aggiungendo la seguente funzionalità: il contatore decrementa il valore se EN=1; altrimenti, se EN=0, mantiene il valore precedente. (1 punto)

Contatore

C3	C2	C1	C0	C3	C2	C1	C0
0	0	0	0	1	0	0	1
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1
0	0	1	1	0	0	1	0
0	1	0	0	0	0	1	1
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	1
0	1	1	1	0	1	1	0
1	0	0	0	0	1	1	1
1	0	0	1	1	0	0	0
1	0	1	0	0	0	0	0
1	0	1	1	0	0	0	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	0	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	0

C0 = !C3&!C0 + C3&!C2&!C1&!C0
C1 = !C3&C1&C0 + C3&!C2&!C1&!C0
C2 = !C3&C2&C1 + !C3&C2&C0 + C3&!C2&!C1&!C0
C3 = C3&!C2&!C1&C0

Il Contatore con LOAD diventa
C0 = !C3&!C0&!LOAD + C3&!C2&!C1&!C0&!LOAD + L0&LOAD
C1 = !C3&C1&C0&!LOAD + C3&!C2&!C1&!C0&!LOAD + L1&LOAD
C2 = !C3&C2&C1&!LOAD + !C3&C2&C0&!LOAD + C3&!C2&!C1&!C0&!LOAD + L2&LOAD
C3 = C3&!C2&!C1&C0&!LOAD + L3&LOAD

ck	Areset	L3	L2	L1	L0	LOAD	nc	nc	nc	nc	gnd
nc	C0	C1	C2	C3	nc	nc	nc	nc	nc	nc	Vdd

AR = Areset
C0.D = !C3&!C0&!LOAD + C3&!C2&!C1&!C0&!LOAD + L0&LOAD C0.oe = Vdd
C1.D = !C3&C1&C0&!LOAD + C3&!C2&!C1&!C0&!LOAD + L1&LOAD C1.oe = Vdd
C2.D = !C3&C2&C1&!LOAD + !C3&C2&C0&!LOAD + C3&!C2&!C1&!C0&!LOAD + L2&LOAD C2.oe = Vdd
C3.D = C3&!C2&!C1&C0&!LOAD + L3&LOAD C3.oe = Vdd

2) In linguaggio VHDL si avrebbe

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_unsigned.all;



ENTITY blocco IS
PORT (Areset, Ck, Load_unf: IN std_logic;
        LoadData_unf: IN std_logic_vector(3 DOWNTO 0);
        OutC: BUFFER std_logic_vector(3 DOWNTO 0));
END blocco;


ARCHITECTURE archblocco OF blocco IS
SIGNAL Load: std_logic; -- segnale di appoggio necessario per creare il filtro antimetastabilità
SIGNAL LoadData: std_logic_vector(3 DOWNTO 0); -- segnale di appoggio necessario per creare il filtro
antimetastabilità

BEGIN

PROCESS(Areset, Ck)
BEGIN
        IF Areset = '0' THEN
                OutC <= "0000";
        ELSIF rising_edge(Ck) THEN

                Load <= Load_unf;   -- filtro anti-metastabilità
                LoadData <= LoadData_unf;   -- filtro anti-metastabilità

                IF Load = '1' THEN
                        IF LoadData <= x"9" THEN
                                OutC <=LoadData;
                        ELSE
                                OutC <=x"0";
                        END IF
                ELSIF OutC =x"0" THEN
                        OutC <= x"9";
                ELSE
                        OutC <= OutC -1;
                END IF;
        END IF;
END PROCESS;

END archblocco;
```

Nota: si è tenuto conto della possibile metastabilità sui segnali Load_unf e LoadData_unf filtrando con un filtro di sincronizzazione anti-metastabilità nell'ipotesi che la frequenza di clock sia compatibile con il resolving time; infatti tali segnali sono asincroni rispetto al Ck e potrebbero commutare proprio in corrispondenza del rising-edge di Ck. Al contrario, i segnali Load e LoadData,, sebbene possano essere affetti da metastabilità Tphl(/Tplh) dopo il rising_edge di Ck, sono stabili al prossimo rising_edge(Ck), in quanto il resolving time è inferiore a Tck e Thold<Tphl(/Tplh)

3)

```
ENTITY blocco IS
PORT (Areset, Ck, EN_unf, Load_unf: IN std_logic;
        LoadData_unf: IN std_logic_vector(3 DOWNTO 0);
        OutC: BUFFER std_logic_vector(3 DOWNTO 0));
END blocco;


ARCHITECTURE archblocco OF blocco IS
SIGNAL Load: std_logic; -- segnale di appoggio necessario per creare il filtro antimetastabilità
SIGNAL LoadData: std_logic_vector(3 DOWNTO 0); -- segnale di appoggio necessario per creare il filtro
SIGNAL En: std_logic; -- segnale di appoggio per filtro anti metastabilità
BEGIN

PROCESS(Areset, Ck)




BEGIN
        IF Areset = '0' THEN
                OutC <= "0000";
        ELSIF rising_edge(Ck) THEN

                Load <= Load_unf;   -- filtro anti-metastabilità
                LoadData <= LoadData_unf;   -- filtro anti-metastabilità
                En <= En_unf; -- filtro anti-metastabilità

                IF Load = '1' THEN
                        IF LoadData <= x"9" THEN
                                OutC <=LoadData;
                        ELSE
                                OutC <=x"0";
                        END IF
                ELSIF En='1' THEN
                        IF OutC =x"0" THEN
                                OutC <= x"9";
                        ELSE
                                OutC <= OutC -1;
                        END IF;   -- memoria implicita
                END IF;
        END IF;
END PROCESS;

END archblocco;
```

Nota: analogamente a quanto fatto precedentemente per Load e LoadData, anche per il segnale asincrono En si è tenuto conto della possibile metastabilità filtrando con un filtro di sincronizzazione anti-metastabilità nell'ipotesi che la frequenza di clock sia compatibile con il resolving time.

- 1) Si realizzi mediante linguaggio booleano per una GAL22V10 un comparatore a 3 bit che confronti 3 bit A2 A1 A0 con l'uscita di un contatore a 3 bit C2 C1 C0 con reset asincrono ARES; il comparatore deve fornire l'uscita A=C (1 punto).
- 2) Si realizzi la funzionalità del punto precedente in linguaggio VHDL. (1 punto)
- 3) Si modifichi il programma del punto 2 in modo tale da abilitare il conteggio (il conteggio si ferma in caso di mancanza di abilitazione) attraverso un segnale esterno ABIL che è asincrono rispetto al clock. (1 punto)

1) Il contatore a 3 bit conta da 0 a 7, come indicato in tabella.

C2	C1	C0	C2	C1	C0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Il Comparatore invece, stante che X=Y sta a significare X&Y + !X&!Y

C2	A2	C1	A1	C0	A0	AeqC
0	1	X	X	X	X	0
1	0	X	X	X	X	0
(C2=A2)		0	1	X	X	0
(C2=A2)		1	0	X	X	0
(C2=A2)		(C1=A1)		0	1	0
(C2=A2)		(C1=A1)		1	0	0
(C2=A2)		(C1=A1)		(C0 = A0)		1

Per cui si ha:

ck	Ares	nc	nc	nc	nc	nc	nc	nc	nc	nc	gnd
nc	C0	C1	C2	nc	nc	AeqC	nc	nc	nc	nc	Vdd

C0.D =!C0& C1.D =!C1&C0& + C1&!C0& C2.D = C2&!C1+ C2&!C0 + !C2&C1&C0
AeqC = (A2=C2)&(A1=C1)&(A0=C0) = (A2&C2+!A2&!C2)&(A1&C1+!A1&!C1)&(A0&C0+!A0&!C0)
L'espressione di cui sopra consta di 8 minterm quindi è compatibile con il FAN-IN della porta OR di qualsiasi macrocella di uscita.
C0.oe = Vdd C1.oe = Vdd C2.oe = Vdd AeqC.oe = Vdd
AR = Ares

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY countercomp IS
        PORT ( Ares, Ck: IN std_logic;
               A: IN std_logic_vector(2 DOWNTO 0));
               C: BUFFER std_logic_vector(2 DOWNTO 0));
               AeqC: OUT std_logic;
END countercomp;

ARCHITECTURE archcountercomp OF countercomp IS
BEGIN
        PROCESS (Ck, Ares)
        BEGIN
               IF Ares = '1' then
                      C <= "000";
               END IF;
               ELSEIF rising_edge(Ck) then
                      IF C = "111" then
                             C <= "000";
                             ELSE C <= C+1;
                      END IF;
               END IF;
        END PROCESS;

        AeqC <= '1' WHEN (A=C) ELSE '0';

END archcountercomp;
```

```
ENTITY countercomp IS
        PORT ( Ares, Ck, Abil_unf: IN std_logic;
               A: IN std_logic_vector(2 DOWNTO 0));
               C: BUFFER std_logic_vector(2 DOWNTO 0));
               AeqC: OUT std_logic;
END countercomp;

ARCHITECTURE archcountercomp OF countercomp IS
SIGNAL Abil: std_logic;
BEGIN
        PROCESS (Ck, Ares)
        BEGIN
               IF Ares = '1' then
                      C <= "000";
               END IF;
               ELSEIF rising_edge(Ck) then
                      ABIL <= ABIL_unf;
                      IF ABIL = '1' then
                         IF C = "111" then
                              C <= "000";
                              ELSE C <= C+1;


                         END IF;
                      END IF;
               END IF;
        END PROCESS;

        AeqC <= '1' WHEN (A=C) ELSE '0';

END archcountercomp;
```

- 1) Si realizzi mediante linguaggio booleano per una GAL22V10 un contatore a 2 bit il cui valore C1, C0 viene confrontato con gli ingressi I1, I0 e genera il segnale IeqC che va a 1 quando (C1,C0)=(I1,I0). Il contatore ha reset asincrono Areset (2 punti)
- 2) Si descriva il problema della metastabilità indicando quali segnali del punto precedente possono generare metastabilità (1 punto).
- 3) Si realizzi la funzionalità del punto 1) in linguaggio VHDL (1 punto)

Contatore a 2 bit

C1	C0	C1	C0
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

Comparatore a 2 bit

C1	C0	I1	I0	IeqC
0	0	0	0	1
0	0	0	1	0
0	0	1	x	0
0	1	0	0	0
0	1	0	1	1
0	1	1	x	0
1	0	0	x	0
1	0	1	0	1
1	0	1	1	0
1	1	0	x	0
1	1	1	0	0
1	1	1	1	1

CK	Ares	I1	I0	nc	nc	nc	nc	nc	nc	nc	gnd
nc	C1	C0	IeqC	nc	nc	nc	nc	nc	nc	nc	Vdd

C0.D =!C0
C1.D = !C0&C1+C0&!C1
IeqC = !C1&!C0&!I1&!I0 + C1&!C0&I1&!I0 + !C1&C0&!I1&I0 + C1&C0&I1&I0
C0.oe = Vdd C1.oe = Vdd IeqC.oe = Vdd
AR = Ares

La **metastabilità** è uno stato in cui l'uscita di un dispositivo sequenziale è impredicibile e cioè può assumere qualunque valore con qualsiasi dinamica. Si tratta di un fenomeno che tende a decadere in uno stato stabile ma non predicibile secondo una legge probabilistica che segue un andamento esponenziale e pertanto un'attesa di qualche centinaio di ns porta, nella maggior parte dei casi, la probabilità di sopravvivenza dello stato metastabile allo stesso ordine di grandezza della probabilità di guasto del componente. Nei circuiti sequenziali attivati da un clock (flip-flop) i segnali di ingresso devono essere stabili per un certo tempo prima che il clock venga applicato ("set-up time" Tset-up) e devono rimanere tali per un tempo dopo che la transizione è avvenuta ("hold time" Thold): se tali condizioni non venissero rispettate il flip-flop potrebbe entrare in uno stato metastabile. Inoltre l'impulso di clock, così come gli impulsi di set o di reset, devono avere una durata minima Tw perché possano essere riconosciuti ed eseguiti senza causare metastabilità.
Il **resolving time** è il tempo entro il quale la metastabilità decade verso uno stato stabile e la probabilità di sopravvivenza dello stato metastabile è confrontabile con la probabilità di guasto. Nessuno dei segnali può dare origine a metastabilità.

In VHDL:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use ieee.std_logic_unsigned.all;

ENTITY counter IS
        PORT ( Ares, C: IN std_logic;
               I: IN std_logic_vector(1 DOWNTO 0);
               IeqC: OUT std_logic;
               C: BUFFER std_logic_vector(1 DOWNTO 0));
END counter;

ARCHITECTURE archcounter OF counter IS
Signal Abil: std_logic;
BEGIN
        PROCESS (Ck, Ares)
        BEGIN
               IF Ares = '1' then
                       C <= "00";
               ELSEIF rising_edge(clock) then
                       IF C >= "11" then
                              C <= "00";
                       ELSE C <= C+1;
                       END IF;
               END IF;
        END PROCESS;
        IeqC <= 1 WHEN I = C ELSE 0;

END archcounter;
```

- 1) Si realizzi mediante linguaggio booleano per una GAL22V10 un sommatore a 2 bit e un contatore a 3 bit resettabile in modo asincrono e con conteggio abilitabile attraverso il segnale En e con un'uscita OUT che si attiva quando il valore del contatore è zero (2 punti).
- 2) Si realizzi la funzione del punto precedente in linguaggio VHDL(1 punto), tenendo conto della possibile metastabilità sul segnala En. 1 punto)

Per il Contatore a 3 bit si ha:

C2	C1	C0	C2	C1	C0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

C0 = !C0&En
C1 = !C1&C0&En + C1&!C0&En
C2 = !C2&C1&C0&En + C2&!C1&En + C2&C0&En

Per l'uscita OUT che si attiva quando il contatore è zero, si ha

OUT = !C0&!C1&!C2

Purtroppo la GAL ha solo 10 uscite quindi si rinuncia alle uscite OUT0 e OUT1 in favore dell'uscita OUT01, dove
OUT01 = OUT0&OUT1 = C0&S0&C1&S1 + !C0&!S0&C1&S1 + C0&S0&!C1&!S1 + !C0&!S0&!C1&!S1
Il programma della GAL risulta quindi:

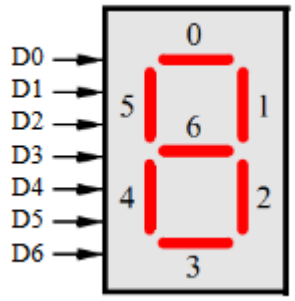
ck	Ares	A1	A0	B1	B0	En	nc	nc	nc	nc	gnd
nc	C0	C1	C2	Q0	Q1	Q2	nc	nc	nc	OUTx	Vdd

AR = Ares
C0.D = !C0&En — C0.oe = Vdd
C1.D = !C1&C0&En + C1&!C0&En — C1.oe = Vdd
C2.D = !C2&C1&C0&En + C2&!C1&En + C2&C0&En — C2.oe = Vdd
S0 = !B0&A0 + B0&!A0 — S0.oe = Vdd
S1 = !B1&!B0&A1 + B1&!B0&!A1 + !B1&B0&!A1&A0 + !B1&B0&A1&!A0 + B1&B0&!A1&!A0 + B1&B0&A1&A0 — S1.oe = Vdd
S2 = !B1&B0&A1&A0 + B1&!B0&A1 + B1&B0&A0 + B1&B0&A1 — S2.oe = Vdd
OUTx = !C0&!C1&!C2 — OUTx.oe = Vdd

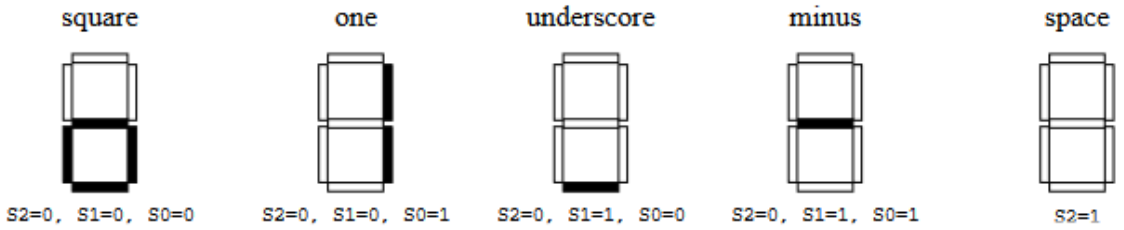
2) La differenza tra signal e variable si applica nei process, dove il valore dei signal sono aggiornati solo alla fine del process, mentre i valori delle variable sono aggiornati immediatamente

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_unsigned.all;


ENTITY blocco IS
PORT (Ares, Ck, En, En-unf: IN std_logic;
        A, B: IN std_logic_vector(1 DOWNTO 0);
        C, S: BUFFER std_logic_vector(2 DOWNTO 0);
        OUTx: OUT std_logic);
END blocco;

ARCHITECTURE archblocco OF blocco IS
BEGIN
PROCESS(Ares, Ck)
BEGIN
        IF Ares = '0' THEN
                C <= "000";
        ELSIF rising_edge(Ck) THEN
                En <= En_unf;   // filtro anti-metastabilità
                IF En = 1
                        IF C = “111” THEN C <= “000”
                                ELSE C <= C + 1;
                        END IF;
                END IF;
        END IF;
END PROCESS;
S <= A+B;
OUTx <= ‘1’ WHEN C=”000” ELSE ‘0’;

END archblocco;
```

Il filtro alla metastabilità funziona sempre in ipotesi che la frequenza di clock sia compatibile con il resolving time.



- 1) Un display a sette segmenti è un dispositivo composto da 7 led di forma allungata, ognuno dei quali pilotato da un opportuno segnale digitale, disposti in modo tale da poter rappresentare determinati caratteri. Di seguito è mostrato uno schema di un led a sette segmenti con la disposizione dei led ed i relativi ingressi di controllo (es. D0 è connesso al segmento 0).
  Si realizzi mediante linguaggio booleano per una GAL22V10 un decoder per il display a 7 segmenti di cui sopra che permette la visualizzazione di cinque diversi caratteri, mostrati di seguito, a seconda del valore impostato sui tre ingressi di selezione S2, S1, S0. Si consideri un display a catodo comune (accensione del led scrivendo '1' sul relativo segmento).
- 2)Si realizzi il decoder del punto precedente in linguaggio VHDL. (2 punti)
- 3) Si modifichi il programma del punto precedente in modo tale da rendere il decoder adatto a funzionare anche con display ad anodo comune (accensione del led con il relativo ingresso a '0'), mediante un ingresso aggiuntivo T di selezione tipologia display (T='0'anodo comune; T='1'catodo comune)



1) Solo 5 dei 7 led del display sono coinvolti nella rappresentazione dei simboli richiesti, quindi sarà necessario ricavare le equazioni booleane solo per D1, D2, D3, D4 e D6. Gli altri ingressi del led a 7 segmenti dovranno essere mantenuti al valore logico '0'. La tabella della verità e le equazioni booleane che descrivono il decoder richiesto sono riportati di seguito.

S2	S1	S0	D1	D2	D3	D4	D6
0	0	0	0	1	1	1	1
0	0	1	1	1	0	0	0
0	1	0	0	0	1	0	0
0	1	1	0	0	0	0	1
1	X	X	0	0	0	0	0

Il programma della GAL sarebbe:

```
nc      S2      S1      S0      nc      nc      nc      nc      nc      nc      nc      gnd
nc      D0      D1      D2      D3      D4      D5      D6      nc      nc      nc      Vdd

D0 = gnd
D1 = !S2&!S1&S0
D2 = !S2&!S1&!S0 + !S2&!S1&S0
D3 = !S2&!S1&!S0 + !S2&S1&!S0
D4 = !S2&!S1&!S0
D5 = gnd
D6 = !S2&!S1&!S0 + !S2&S1&S0

D0.oe = Vdd
D1.oe = Vdd
D2.oe = Vdd
D3.oe = Vdd
D4.oe = Vdd
D5.oe = Vdd
D6.oe = Vdd
```

2) In linguaggio VHDL, i vettori consentono di gestire in modo ottimale gruppi omogenei di bit, come ad esempio la parola composta dai bit D0 – D7 in ingresso al display a sette segmenti. Un'implementazione in VHDL del decoder in oggetto è illustrata di seguito.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY dec7seg IS
        PORT ( S: IN std_logic_vector(2 DOWNTO 0);
               D: OUT std_logic_vector(0 TO 6));
END dec7seg;

ARCHITECTURE archdec7seg OF dec7seg IS

CONSTANT square: std_logic_vector(0 TO 6):="0011101";
```

```
CONSTANT one: std_logic_vector(0 TO 6):="0110000";
CONSTANT underscore: std_logic_vector(0 TO 6):="0001000";
CONSTANT minus: std_logic_vector(0 TO 6):="0000001";
CONSTANT space: std_logic_vector(0 TO 6):="0000000";

BEGIN

        PROCESS (S)
        BEGIN

                CASE S IS
                        WHEN "000" => D <= square;
                        WHEN "001" => D <= one;
                        WHEN "010" => D <= underscore;
                        WHEN "011" => D <= minus;
                        WHEN OTHERS => D <= space;
                END CASE;

        END PROCESS;

END archdec7seg;
```

3) non lo trascrivo

- 1) Si realizzi mediante linguaggio booleano per una GAL22V10 un contatore di secondi modulo 10 con segnale di CK esterno a 1 Hz, Reset asincrono ARES, segnale di ABIL che abilita il conteggio (ABIL=1) o lo ferma (ABIL=0) (1 punto)
- 2) Si descriva il problema della metastabilità indicando quali segnali del punto precedente possono generare metastabilità (2 punti).
- 3) Si realizzi la funzionalità del punto 1) in linguaggio VHDL includendo i filtri per la metastabilità (1 punto)

Uscite attuali				Uscite future			
C3	C2	C1	C0	C3	C2	C1	C0
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
1	0	1	X	0	0	0	0
1	1	X	X	0	0	0	0

C0 = !C3&!C0 + C3&!C2&!C1&!C0
C1 = !C3&!C1&C0 + !C3&C1&!C0
C2 = !C3&!C2&C1&C0 + !C3&C2·!C1 + !C3&C2·!C0
C3 = !C3&C2&C1&C0 + C3·!C2&!C1&!C0

Si noti che Abil potrebbe andare in metastabilità, dato che potrebbe commutare in corrispondenza del clock CK, tuttavia, grazie al resolving time, il contatore risulterebbe immune in quanto si inserisce un flip-flop di filtro tra Abil_nf e Abil (l'eventuale metastabilità su Abil si manifesterebbe dopo un tempo di propagazione a partire dal clock e quindi si risolverebbe in uno stato stabile prima dell'arrivo del successivo fronte di clock).

CK	Ares	Abil_nf	nc	nc	nc	nc	nc	nc	nc	nc	gnd
nc	C0	C1	C2	C3	Abil	nc	nc	nc	nc	nc	Vdd

Abil.D = Abil_nf
C0.D = !C3&!C0&Abil + C3&!C2&!C1&!C0&Abil + C0
C1.D = !C3&!C1&C0&Abil + !C3&C1&!C0&Abil + C1
C2.D = !C3&!C2&C1&C0&Abil + !C3&C2·!C1&Abil + !C3&C2·!C0&Abil + C2
C3.D = !C3&C2&C1&C0&Abil + C3·!C2&!C1&!C0&Abil + C3
C0.oe = Vdd C1.oe = Vdd C2.oe = Vdd C3.oe = Vdd Abil.oe = Vdd
AR = Ares

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use ieee.std_logic_unsigned.all;

ENTITY counter IS
        PORT ( Ares, Ck, Abil_nf: IN std_logic;
               C: BUFFER std_logic_vector(3 DOWNTO 0));
END counter;

ARCHITECTURE archcounter OF counter IS
Signal Abil: std_logic;
BEGIN
        PROCESS (Ck, Ares)
        BEGIN
               IF Ares = '1' then
                      C <= "0000";
                      Abil <= '0';
               ELSEIF rising_edge(clock) then
                      Abil <= Abil_nf;
                      IF Abil = '1' then
                             IF C >= "1001" then
                                    C <= "0000";
                             ELSE C <= C+1;
                             END IF;
                      END IF;
               END IF;
        END PROCESS;

END archcounter;
```

- Si realizzi mediante linguaggio booleano per una GAL22V10 un contatore a una cifra decimale con Reset asincrono ARES attivo basso (2 punti). Si realizzi la funzionalità del punto precedente in linguaggio VHDL, includendo i filtri anti-metastabilità ove necessario. (1 punto)

Il funzionamento di un contatore decimale, da 0 a 9 ("C3 C2 C1 C0") è il seguente:

Uscite attuali				Uscite future			
C3	C2	C1	C0	C3′	C2′	C1′	C0′
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
1	0	1	X	0	0	0	0
1	1	X	X	0	0	0	0

Si noti che il segnale di reset asincronon non ha effetti in termini di metastabilità.

CK	ARES	nc	nc	nc	nc	nc	nc	nc	nc	nc	gnd
nc	C0	C1	C2	C3	nc	nc	nc	nc	nc	nc	Vdd

C0.D = !C3·!C0 + C3·!C2·!C1·!C0
C1.D = !C3·C1·!C0 + !C3·!C1·C0
C2.D = !C3·C2·!C1 + !C3·C2·!C0 + !C3·!C2·C1·C0
C3.D = !C3·C2·C1·C0 + C3·!C2·!C1·!C0

C0.oe = Vdd C1.oe = Vdd C2.oe = Vdd C3.oe = Vdd
AR = Ares

In linguaggio VHDL, sempre considerando che non si pone il problema della metastabilità in quanto Ares è asincrono, si avrebbe il seguente programma.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use ieee.std_logic_unsigned.all;

ENTITY counter IS
	PORT ( Ares, Ck: IN std_logic;
		C: BUFFER std_logic_vector(3 DOWNTO 0));
END counter;

ARCHITECTURE archcounter OF counter IS
BEGIN
	PROCESS (Ck, Ares)
	BEGIN
		IF Ares = '1' then
			C <= "0000";
		ELSEIF rising_edge(clock) then
			IF C = "1001" then
				C <= "0000";
			ELSE C <= C+1;
			END IF;
		END IF;
	END PROCESS;

END archcounter;
```

- 1) Si realizzi mediante linguaggio booleano per una GAL22V10 un contatore Up e Down a 3 bit C2 C1 C0 con reset asincrono ARES e Direzione DIR (DIR=1 il contatore incrementa, DIR=0 il contatore decrementa) (1 punto)
- 2) Si indichi quali dei segnali al punto 1 potrebbero dare origine a metastabilità e progettare il filtro (1 punto).
- 3) Si realizzi la funzionalità dei punti precedenti in linguaggio VHDL (1 punto).

1) Il contatore a 3 bit conta da 0 a 7, come indicato in tabella.

Stato Attuale			Stato prossimo (DIR=1)			Stato prossimo (DIR=0)		
C2	C1	C0	C2	C1	C0	C2	C1	C0
0	0	0	0	0	1	1	1	1
0	0	1	0	1	0	0	0	0
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	0	1	0
1	0	0	1	0	1	0	1	1
1	0	1	1	1	0	1	0	0
1	1	0	1	1	1	1	0	1
1	1	1	0	0	0	1	1	0

Per cui si ha:

CK	ARES	DIR	nc	nc	nc	nc	nc	nc	nc	nc	gnd
nc	C0	C1	C2	DIRF	nc	nc	nc	nc	nc	nc	Vdd

```
DIRF.D = DIR    -- filtro contro la metastabilità; il segnale DIR potrebbe commutare in corrispondenza del clock
provocando metastabilità per violazione di Thold o Tset-up a livello di segnale DIRF. Il segnale DIRF è tuttavia
sincronizzato e quindi, se il periodo di clock è superiore del resolving time, non può generare metastabilità a livello di
contatore
C0.D =!C0
C1.D =!C1&C0&DIRF + C1&!C0&DIRF + C1&C0&!DIRF + !C1&!C0&!DIRF
C2.D = C2&!C1&DIRF + C2&!C0&DIRF + !C2&C1&C0&DIRF + C2&C1&!DIRF + C2&C0&!DIRF
+ !C2&!C1&!C0&!DIRF
```

L'espressione di cui sopra consta di 8 minterm quindi è compatibile con il FAN-IN della porta OR di qualsiasi macrocella di uscita. Si noti che, nell'ipotesi che i segnali CK e il segnale ARES rispettino il vincolo di minima larghezza di impulso, nessun altro segnale oltre a DIR può generare metastabilità.

In VHDL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;

entity my_counter is
	Port ( ARES, CK, DIR : in STD_LOGIC;
		C : out STD_LOGIC_VECTOR(2 downto 0));
end counter;

architecture archcounter of counter is
signal DIRF : STD_LOGIC;
signal CONT : UNSIGNED(2 downto 0);
begin
	process (CK, ARES)
	begin
		if ARES = '1' then
			CONT <= "000";
		elsif rising_edge(CK) then
			DIRF <= DIR;
			if DIRF = '1' then
				if CONT = "111" then CONT <= "000";
				else CONT <= CONT + 1;
				end if;
			elsif DIRF = '0' then
				if CONT = "000" then CONT <= "111";
				else CONT <= CONT - 1;
				end if;
			end if;
		end if;
	end process;

	C <= std_logic_vector(CONT);
end archcounter;
```