Introduzione

Il ***Driver per SAA1064 v1.0*** pilota, seguendo il protocollo I2C, un dispositivo SAA1064 in modo da gestire 4 display 7 segmenti, sfruttando l'utilizzo di un dispositivo programmabile (FPGA). E' predisposto, utilizzando delle schede di espansione, di cui viene fornito solo il master, per gestire in modo diverso fino a 4 SAA1064 per un totale di 16 display.

Il sistema è costituito da due parti: una Hardware e una Software.
La parte Hardware è composta da due schede:
- "*Scheda Xess*" XSA50 Board su cui è montata l'FPGA Xilinx Spartan2 XC2S50
- "*Scheda di visualizzazione*" su cui è montato il chip Philips SAA1064 e quattro display.

La parte Software è composta da tutti i sorgenti necessari alla programmazione dell'FPGA e al corretto funzionamento del sistema.

Funzionamento

Il sistema è stato progettato e realizzato per leggere dei dati, codificarli, con l'ausilio dell'FPGA, e trasmetterli alla "scheda di visualizzazione" utilizzando il protocollo opportuno (I2C). L'FPGA ha l'ulteriore compito di aggiungere ai dati validi i codici necessari all'utilizzo del chip SAA1064 che gestirà l'illuminazione dei display. La lettura dei dati avviene attraverso un bus da 8 linee, a cui corrispondono i led dei display 7 segmenti, associato ad un impulso su una linea opportuna (Write) che ha lo scopo di decidere l'istante di prelievo.

Il ***Driver per SAA1064 v1.0*** può funzionare in quattro modalità differenti:
- Chip singolo con visualizzazione automatica;
- Chip singolo con visualizzazione manuale;
- Multi-chip con visualizzazione automatica;
- Multi-chip con visualizzazione manuale.

Modalità chip singolo con visualizzazione automatica.

In questa modalità il driver gestisce un solo chip (quello della scheda in dotazione). All'arrivo di un impulso (Write) vengono letti i dati in ingresso e trasmessi su uno dei 4 display. La scelta del display avviene in modo automatico da sinistra verso destra; il primo display scritto è quello all'estrema sinistra.

Modalità chip singolo con visualizzazione manuale.

In questa modalità il driver gestisce un solo chip (quello della scheda in dotazione). All'arrivo di un impulso (Write) sono letti i dati in ingresso e trasmessi su uno dei 4 display. La scelta del display avviene dall'esterno utilizzando opportune linee di indirizzamento (disp_adrs(1:0))

Multi-chip con visualizzazione automatica.

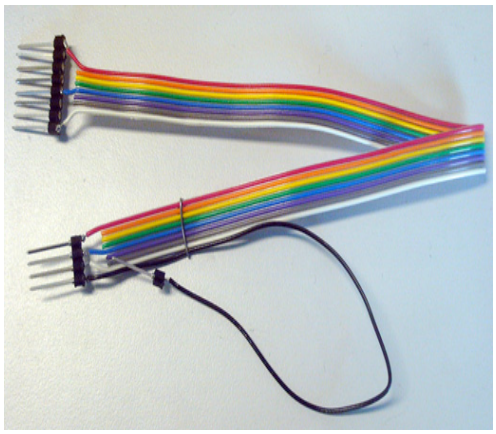
In questa modalità il driver può gestire, con l'ausilio delle schede d'espansione (non fornite), fino a 4 chip indipendenti, per un totale di 16 display. All'arrivo di un impulso (Write) i dati in ingresso sono letti e trasmessi sul display opportuno. La scelta del chip avviene dall'esterno, attraverso due linee dedicate (chip_add0, chip_add1), mentre quella del display avviene in modo automatico da sinistra verso destra **indipendentemente dalla scelta del chip** (Se l'ultimo display appena scritto è il n°3 il prossimo sarà il n°4 indipendentemente dal chip scelto); il primo display scritto è quello all'estrema sinistra.

Multi-chip con visualizzazione manuale.

In questa modalità il driver può gestire, con l'ausilio delle schede d'espansione (non fornite), fino a 4 chip indipendenti, per un totale di 16 display. All'arrivo di un impulso (Write) il dato in ingresso viene letto e trasmesso su sul display opportuno. Sia la scelta del chip che quella del display avviene dall'esterno utilizzando opportune linee di indirizzamento (disp_adrs(1:0) + chip_addX)

Utilizzo

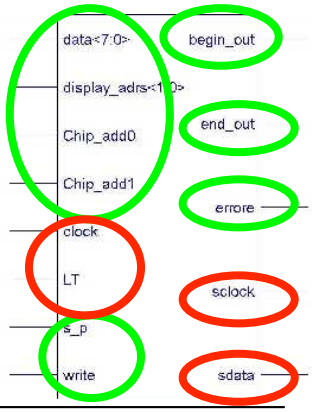
Preparazione hardware



Le due schede fornite lavorano in sinergia e devono essere opportunamente collegate per funzionare correttamente. Per il collegamento delle due schede si utilizzi il cavo a 4 poli fornito. Come da figura in appendice collegare, ad alimentazione scollegata, prima il terminale "A" sulla "Scheda di Visualizzazione", e successivamente il terminale "B" alla "Scheda Xess"; in entrambi in casi prestare attenzione a rispettare il "pin costraint".
Collegare alimentazione e verificare che sul display della "Scheda Xess" sia visualizzato il numero 7 procedere poi con il collegamento del cavo parallelo.

Preparazione software

La parte software può essere ridotta al blocco in figura. Come si può notare sono presenti due tipologie di linee, quelle impostabili a piacere dall'utente (cerchiate in verde), e quelle vincolate (cerchiate in rosso) la cui configurazione **deve seguire** le indicazioni fornite.



Linee configurabili:

Input:

- 8 linee Dati: rappresentano la configurazione del display 7 segmenti.

- 1 linea Write: all'arrivo di un impulso su questa linea corrisponde la lettura dei dati.
- 1 linea S_P: indica la modalità di scrittura dei display.
- 2 linee display_adrs(X): sono necessarie all'indirizzamento manuale dei display.
- 2 linee Chip_AddX: sono necessarie all'indirizzamento del Chip **(devono essere posti allo stato 0 se non vengono utilizzate schede di espansione)**

Output:

- 1 linea Error: segnala che la trasmissione dalla "scheda Xess" alla "Scheda di Visualizzazione" non è andata a buon fine.
- 1 linea Begin_out: segnala che la trasmissione ha avuto inizio
- 1 linea End_out: segnala che la trasmissione è stata completata.

Linee non configurabili:

Input:

- 1 linea di clock: CLOCK di sistema.
- 1 linea LT: ingresso per il Lamp-Test.[1]

Output:

- 1 Linea SClock: rappresenta la linea SCL del protocollo I2C,
- 1 linea SData: rappresenta la linea SDA del protocollo I2C.

Queste linee devono essere obbligatoriamente collegate rispettivamente alle "Pad d'ingresso" e alle "Pad di uscita" seguendo questi vincoli:

clock=pin(88);
LT=pin(93);
SClock=pin(77);
SData= pin(76).

N.B.: Per le linee SDA e SCL c'è un ulteriore vincolo, essendo pre-bufferizzate non è possibile connettere niente oltre alle pad d'uscita

Test del sistema

Per verificare preventivamente il corretto funzionamento dell'intero sistema, è stata prevista una fase di test strutturata in due fasi:

test a vuoto e test a carico.

Test a vuoto:

Il test a vuoto ha lo scopo di controllare se la "Scheda Xess" è in grado di rilevare errori nella comunicazione con SAA1064 (che possono essere: il chip non è presente, il chip non funziona, la linea si è interrotta...)

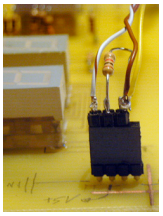
1. Prendere la "Scheda Xess" connettere alimentazione, connettere una resistenza da 10 Kohm tra il piedino 76 e VCC**, essere sicuri che oltre a questo collegamento ci sia solo la connessione con il PC**, e fare l'upload del file **TEST.bit** sull'FPGA. Questa parte del test è superata se sul display della "Scheda Xess" appare la lettera "***E***". Se così non fosse ripetere il **punto 1** verificando preventivamente che sul piedino 76 ci sia livello logico alto, nel caso di ulteriore fallimento del test cambiare "scheda Xess";

[1] Questa linea non è critica quanto le altre, ma si consiglia caldamente di seguire le indicazioni (viene collegata al pulsante della "scheda XESS".

2. Scollegare l'alimentazione dalla "Scheda Xess", e quest'ultima dal PC. Rimuovere la resistenza del punto 1 e connetterla tra il piedino 76 e massa, collegare la scheda al PC e all'alimentazione e fare l'upload del file **TEST.bit** In questo caso il display **deve essere spento**. Nel caso in cui non lo sia verificare i collegamenti ai capi della resistenza e il livello logico del piedino: deve essere a livello basso;
3. Senza scollegare l'alimentazione, **con un po' d'accortezza**, spostare il piedino della resistenza che era collegato a massa, porlo a VCC e successivamente premere il pulsante della "Scheda Xess". Deve apparire la lettera "***E***". Se così non è, ripetere dal punto 2 facendo attenzione ai collegamenti. Nel caso in cui i collegamenti fossero corretti e il test fallisse nuovamente cambiare "Scheda Xess".

Test a carico

Il test a carico ha lo scopo di verificare il funzionamento del chip SAA1064 e dei display ad esso collegati.



4. Dopo aver completato tutto il test a vuoto programmare la scheda per lavorare a 100 kHz (vedi manuale "Scheda Xess");
5. Collegare il pulsante di test (rosso) fornito come da figura:
   filo giallo al pin 23 dell'FPGA,
   connettore nero come da figura (filo marrone a VCC, bianco a GND)
6. Dopodiché scollegare la "Scheda Xess" dall'alimentazione e dal PC, collegarla prima alla "Scheda di visualizzazione", come indicato nel paragrafo "***Preparazione Hardware***", poi al PC e all'alimentazione; effettuare quindi l'upload del file **TEST.BIT**. Ad upload completato tutti i segmenti dei 4 display devono accendersi e successivamente spegnersi. Se non si dovesse vedere niente è possibile che i segmenti si siano spenti troppo velocemente, allora provare a tenere premuto il pulsante della "Scheda Xess": i display dovrebbero rimanere accesi per tutto il tempo in cui il pulsante è premuto, poi spegnersi. Se ciò non funzionasse e sul display è apparsa una "**E**" ripetere il punto 5 controllando le connessioni. Questo serve anche a testare specificatamente i display.
7. Ad ogni pressione del pulsante il display deve visualizzare, accendendo, o modificando un display per volta, i numeri esadecimali (da 0 a F)
8. In qualsiasi momento (a meno di non tenere premuto il pulsante) è possibile premere il pulsante sulla "Scheda Xess": questo farà accendere tutti i led del display finché il pulsante non viene rilasciato[2], in quel momento riapparirà la scritta precedente.

Modalità d'uso

Premessa

In qualsiasi configurazione descritta sopra, se si pone LT=0 vengono illuminati i segmenti di tutti i display della scheda selezionata; questi ultimi rimangono accesi almeno per un tempo pari a $T_{clk}/4096$ (comunque dopo la successiva commutazione di LT) dopodichè verrà visualizzata la scritta precedente

Modalità chip singolo con visualizzazione automatica.

Per utilizzare questa modalità è necessario dapprima impostare la maggior parte degli ingressi seguendo queste specifiche:

LT=1; P=0, Chip_add0=0, Chip_add1=0, Display_adrs(1:0)=00;

[2] Dopo aver rilasciato il pulsante i display potrebbero rimanere accesi, ma comunque per un tempo inferiore a 2 secondi

Da cui segue che le uniche linee effettivamente utilizzate sono: *Write* e *Data(7:0).*
Quando sulla linea Write è presente un livello logico alto il sistema acquisisce i dati sul bus e li scrive sul display opportuno, muovendosi come spiegato precedentemente; quindi è **necessario** che i dati presenti nelle linee *Data(7:0)* nell'istante in cui si ha *Write=1* **siano stabili.**

Modalità chip singolo con visualizzazione manuale.

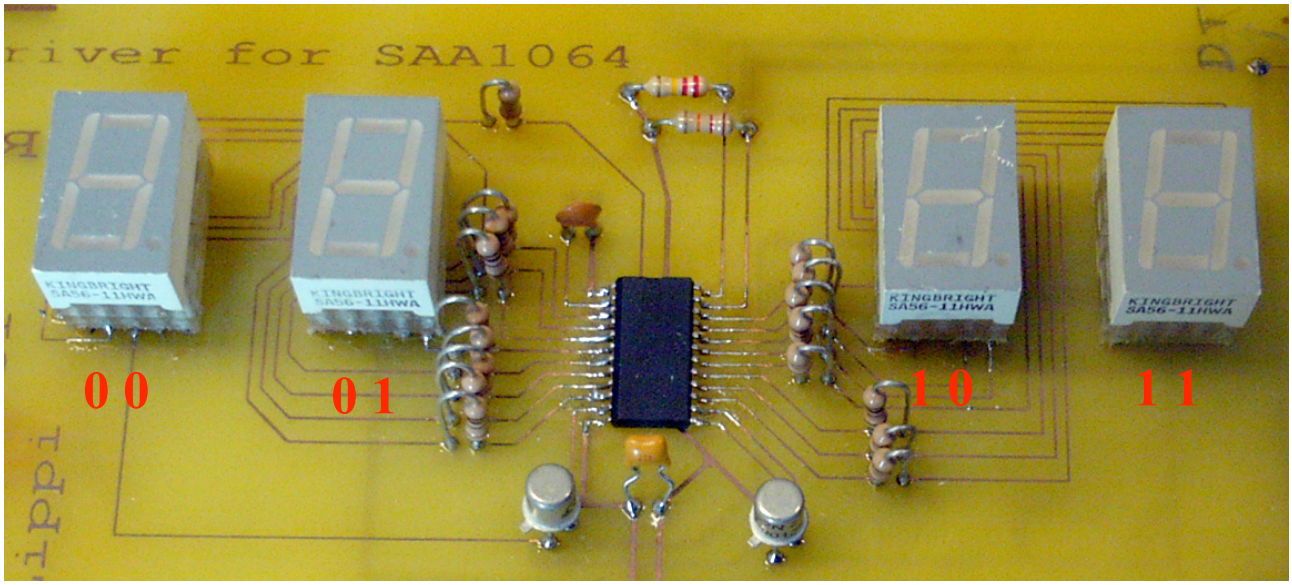
In questo caso è necessario avere a disposizione delle linee per scegliere il display su cui scrivere; infatti le impostazioni di *default* risultano:

LT=1, Chip_add0=0, Chip_add1=0, S_P=1

Da cui segue che le uniche linee effettivamente utilizzate sono:

Write, Data(7:0), Display_adrs(1:0)

Quando sulla linea *Write* è presente un livello logico alto il sistema acquisisce i dati sul bus e li scrive sul display selezionato da *Display_adrs (1:0).*
La selezione del display avviene con la seguente codifica



Anche in questo caso i dati presenti sulle linee *Data(7:0)* nell'istante in cui si ha W*rite=1* devono essere **stabili.**
NB: Quando si passa da *S_P=1 a S_P=0* il primo display scritto sarà sempre quello all'estrema sinistra

Multi-chip con visualizzazione automatica.

Per utilizzare questa modalità impostare le seguenti linee:

LT=1, S_P=0, Display_adrs(1:0)=00

Questa modalità permette di utilizzare più di una "scheda di visualizzazione" (fino a 4) in diversi modi; per selezionare la "scheda di visualizzazione" si sfruttano le linee *Chip_add0 e Chip_add1* [3].
Quando sulla linea *Write* è presente un livello logico alto il sistema acquisisce i dati sul bus e li scrive sul display della scheda scelta.

[3] Chip_add0=Chip_add1=0 identifica anche la scheda fornita; le schede di espansione hanno un jumper opportuno per impostare il loro indirizzo

NB: La variazione di una (o di entrambe) delle linee *Chip_addX* non comporta nessuna variazione nella selezione automatica del display.
Anche in questo caso dati presenti nelle linee *Data(7:0*nell'istante in cui si ha *Write=1* devono essere **stabili.**

Multi-chip con visualizzazione manuale.

Le uniche linee da impostare sono *S_P=1, LT=1*.
Questa modalità permette di utilizzare al massimo le potenzialità del sistema:
Quando sulla linea Write è presente un livello logico alto il sistema acquisisce i dati sul bus e li scrive sul display selezionato da *Display_adrs*. della scheda selezionata da *Chip_add0 e Chip_add1*[4].
NB: Quando si passa da *S_P=1* a *S_P=0* il primo display scritto sarà sempre quello all'estrema sinistra
NB: La variazione di una delle linee *Chip_addX* non comporta nessuna variazione nel "conteggio" dei display.
Anche in questo caso dati presenti nelle linee *Data(7:0)* nell'istante in cui si ha W*rite=1* devono essere **stabili.**

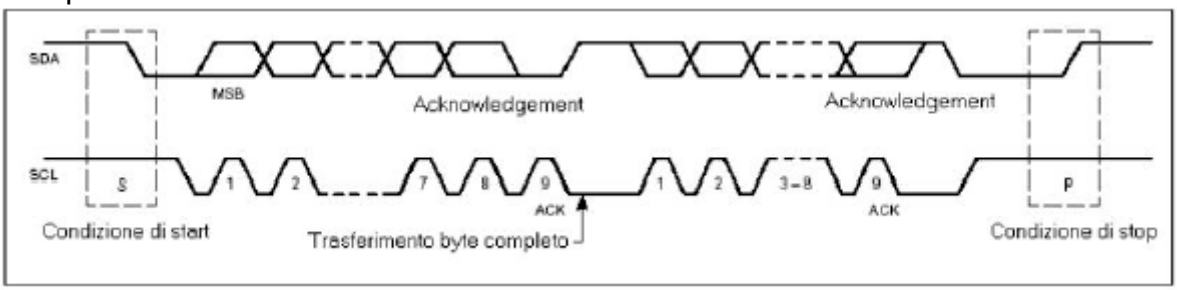
[4] Vedi 1

Informazioni per lo sviluppo:

Le informazioni di seguito riportate sono a uso esclusivo di personale competente. Si declina ogni responsabilità per malfunzionamenti dovuti a modifiche effettuate facendo uso di queste informazioni

Introduzione al protocollo I2C:

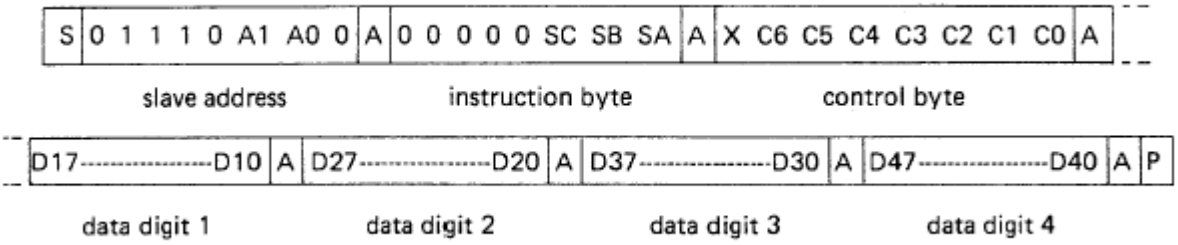
Il protocollo I2C prevede l'utilizzo di un bus formato da due linee bidirezionali. Le due linee, chiamate rispettivamente SCL e SDA, trasportano la tempistica di sincronizzazione (chiamata anche clock) e i dati. I segnali che transitano sulla linea hanno valore 1 o 0 e le tensioni che li rappresentano sono quelle d'alimentazione e di massa rispettivamente. Le due linee non sono lasciate ad un valore indefinito, ma vengono collegate all'alimentazione attraverso una resistenza di pull-up. In questo modo le due linee permangono ad un valore alto-debole, facilmente modificabile da un dispositivo.
Ogni dispositivo collegato a queste linee ´e dotato di un indirizzo univoco, di 7 o 10 bit e può agire sia da master che da slave, secondo le funzioni previste al suo interno. Il master si occupa di iniziare la trasmissione e di generare la tempistica del trasferimento, mentre lo slave è quello che riceve una richiesta.



La comunicazione inizia con la generazione di un segnale di start, immesso sulla linea dal master, dopo un controllo sull'occupazione del bus. La condizione di start consiste nel lasciare la linea SCL allo stato alto, mentre la linea SDA, subisce una transizione dallo stato 1 allo stato 0. Dopo la generazione del segnale di start inizia la trasmissione dei dati vera e propria. Il primo byte trasmesso ´e quello composto dall'indirizzo dello slave con l'aggiunta di un ulteriore valore che indica al ricevente quale è l'operazione a lui richiesta(lettura o scrittura). L'ordine di trasmissione dei bit è quello dal più significativo al meno significativo. Dopo la trasmissione d'ogni byte chi trasmette ha l'obbligo di lasciare la linea SDA allo stato alto, in modo da permettere a chi riceve dei dati, di darne conferma tramite il meccanismo dell'acknowledgement: esso consiste nell'abbassare la linea SDA in corrispondenza del nono impulso presente sulla linea SCL.
Se l'acknowledgement non venisse generato, chi ha iniziato la trasmissione può interrompere la comunicazione, utilizzando la condizione di stop. Il segnale di stop è sempre generato dal master e consiste nel far variare la linea SDA dallo stato basso a quello alto in corrispondenza del periodo alto della linea SCL.

Introduzione al dispositivo SAA1064:

Il dispositivo SAA1064 può pilotare fino a 4 display 7-segmenti con punto decimale ricevuti attraverso il protocollo I2C; l'indirizzo del dispositivo può essere programmato in quattro modi diversi (attraverso una opportuna tensione nell'opportuno piedino).
Il dispositivo possiede al suo interno 5 registri: il primo serve per la "programmazione" (byte di controllo) mentre gli altri contengono i dati effettivi.
Dopo aver selezionato il dispositivo (attraverso l'invio del Byte opportuno) viene inviato un dato che rappresenta sempre l'indirizzo del registro interno che sarà scritto per primo; nel caso in cui la trasmissione non venga chiusa il dispositivo scrive i dati successivi nel registro seguente (Vedi Figura).



Dove il bit C3[5] rappresenta il bit per il Lamp-Test: se C3=1 il dispositivo accende tutti i segmenti dei display e li spegne solo quando riceve C3=0.
Per il control byte è stata effettuata la seguente scelta: C0=C1=C2=C4=C5=C6=1, C3=$\overline{LT}$

Descrizione della trasmissione

Prima di presentare come avviene la trasmissione è doveroso definire due concetti:
Trasmissione: La trasmissione vista dell'esterno del sistema
Trasmissione I2C: Una singola trasmissione con protocollo I2C (da S a P)
Tenendo conto del fatto che è possibile indirizzare la scrittura sul singolo registro interno si effettua una trasmissione per ogni display. La trasmissione si compone di due Trasmissioni I2C contenenti solo 3 byte ciascuno:

- La prima, serve per configurare il dispositivo in una delle seguenti modalità:
  modalità scrittura (C3=0)
  modalità Lamp-Test(C3=1);

In entrambi i casi INSTRUCTION BYTE contiene il CONTROL BYTE

- La seconda, serve per trasmettere il dato o per riconfigurare il dispositivo:
  Per inviare il dato l'INSTRUCTION BYTE contiene l'indirizzo del DATA-DIGIT1.
  Per riconfigurare il dispositivo l'INSTRUCTION BYTE contiene l'indirizzo del CONTROL BYTE, che contiene C3=0.

NB: All'accensione il sistema esegue una trasmissione modalità Lamp-Test

[5] Per dettagli più accurati si veda il Data-Sheet allegato

Descrizione funzionale dei blocchi

Global system (Vedi Appendice):

Il Global System è composto, principalmente dai seguenti blocchi:

I2C_block
sp_display_selector
input_data_manager
fdc (FF tipo D con Reset asincrono)
ftc (FF tipo T con Reset asincrono)

- Il blocco ***I2C_block*** all'arrivo di un impulso sulla linea di *Write* legge i dati del bus *DATA(7:0)* e li trasmette, utilizzando le specifiche del protocollo I2C, attraverso le linee SDA e SCL. Nel caso in cui la trasmissione vada a buon fine viene alzata la linea *byte_tx_ok*, in caso contrario viene alzata la linea *byte_tx_ko*.
  Le linee *begin_tx* e *end_tx* sono utilizzate per monitorare la trasmissione:
  - La linea begin_tx viene alzata quando, la trasmissione dei dati effettivi ha inizio[6], e rimane alta fintanto che la trasmissione dei dati è completata;
  - La linea *end_tx* viene alzata quando una trasmissione viene completata, e rimane alta fintanto che non ne inizia un'altra

  La linea di *reset* ha il compito di resettare il blocco (condizione di partenza).
- Il blocco ***sp_display_selector*** si occupa di gestire la scrittura dei display, più precisamente indica al blocco successivo (***input_data _manager***) il display che dovrà essere scritto.
- Il blocco ***input_data_manager*** si occupa di preparare, in modo opportuno, i dati da trasmettere[7], di attivare il blocco I2C e di gestire i possibili errori di trasmissione rilevati da quest'ultimo.
- Il compito del **FDC** è di portare, all'arrivo del *clock*[8], l'uscita a livello logico alto e mantenerla fintanto che non arriva un impulso di *reset*[9].
- Il compito dei due **FTC** è di portare alta l'uscita ogni due impulsi d'ingresso[10], in quanto una *trasmissione* è composta da due *trasmissioni I2C*

SP_Display_Selector (Vedi Appendice)

Si occupa di gestire la scrittura dei display: indica, in *uscita(1:0),* il codice del display che dovrà essere scritto.

- L'ingresso *Counter* ha un duplice scopo:
- Nella transizione alto-basso, se la linea *Sel=0* (selezione automatica), incrementata il valore del contatore (CB2RE);
- Nella transizione basso-alto, se *write=1*, trasmette in uscita (attraverso la coppia di FF tipo D al centro del disegno), il codice corrispondente al display da scrivere che viene prelevato all'uscita dei due MUX.
- I due MUX hanno lo scopo di scegliere il codice del display:
- Se *Sel=0* l'indirizzo viene prelevato dall'uscita del contatore.
- Se *Sel=1* l'indirizzo viene prelevato dell'esterno: dalle linee *adrs_load(1:0)*.
- Il FF tipo D avente l'ingresso a livello logico alto ha lo scopo di resettare, all'accensione, il contatore, indipendentemente dal valore di *Sel*, il sistema si porta a regime dopo una transizione basso-alto di counter.

[6] Vedi protocollo I2C
[7] Vedi protocollo I2C e caratteristiche SAA1064
[8] Trasmissione non riuscita: Vedi I2C_block
[9] Trasmissione riuscita:Vedi I2C_Block
[10] Vedi I2C_block

Input_Data_Manager (Vedi Appendice)

- ***byte_block***: prepara i dati da trasmettere tenendo conto di:
  LT: Lamp-Test
  Data_in(7:0): contiene il numero da scrivere (codifica 7 segmenti)
  wr_adrs(1:0), codice che indica il display da scrivere
  device_adrs(1:0), codice del dispositivo su cui scrivere
  rd_adrs(2:0), esegue la scelta del byte da trasmettere
  e all'arrivo del *Write* li scive sul bus *data_out(7:0)*
- ***write_manager***: genera il segnale di *Write* necessario per abilitare altri blocchi (interni o esterni)
- ***adrs_lt_manager***: si occupa di generare:
  gli "indirizzi" necessari al ***byte_block*** per preparare i dati da inviare;
  se richiesto, i segnali di Lamp-Test *(LT, LT1, LT2)*

Adrs_lt_manager (Vedi Appendice)

Il funzionamento è molto semplice:
All'accensione si è nella seguente configurazione (con *reset=0, ext_lt=0*):
LT=1, LT1=1, LT2=0, rd_adrs(2:0)=000, ***wait_block*** NON ATTIVO (start)

Ad ogni colpo di clock *(clk)* il sistema evolve nel seguente modo:
*LT=1, LT1=1, LT2=0, rd_adrs(2:0)=000, **wait_block*** NON ATTIVO (start)
*LT=1, LT1=1, LT2=0, rd_adrs(2:0)=001, **wait_block*** NON ATTIVO
*LT=1, LT1=1, LT2=0, rd_adrs(2:0)=010, **wait_block*** NON ATTIVO
*LT=0, LT1=1, LT2=1, rd_adrs(2:0)=000, **wait_block*** ATTIVO

Quando wait_block è attivo significa che aspetta *2^(4*(2* wait1+wait0))* cicli di clock (clock di sistema) prima di porre *LT2=1*[11]; l'uscita rimane ferma fintanto che il blocco (***wait_block***) non viene resettato[12]
LT=0, LT1=1, LT2=1, rd_adrs(2:0)=001, ***wait_block*** ATTIVO
LT=0, LT1=1, LT2=1, rd_adrs(2:0)=010, ***wait_block*** ATTIVO
LT=0, LT1=0, LT2=0, rd_adrs(2:0)=000, ***wait_block*** NON ATTIVO (stand-by)
LT=0, LT1=0, LT2=0, rd_adrs(2:0)=001, ***wait_block*** NON ATTIVO
LT=0, LT1=0, LT2=0, rd_adrs(2:0)=010, ***wait_block*** NON ATTIVO
LT=0, LT1=0, LT2=0, rd_adrs(2:0)=100, ***wait_block*** NON ATTIVO
LT=0, LT1=0, LT2=0, rd_adrs(2:0)=101, ***wait_block*** NON ATTIVO
LT=0, LT1=0, LT2=0, rd_adrs(2:0)=110, ***wait_block*** NON ATTIVO
LT=0, LT1=0, LT2=0, rd_adrs(2:0)=000, ***wait_block*** NON ATTIVO (stand-by)…..

[11] In tutto questo periodo tutti i 7x4=28 led (del dispositivo selezionato) rimangono accesi
[12] Il sistema esterno è stato progettato in modo che lo stato successivo possa essere raggiunto solo se LT2=1, infatti è possibile prolungare il tempo di LT2 tenendo alto l'ingresso ext_LT.
Osservazione1: nel caso in cui accada che ext_LT=end_tx=1 il sistema si riporta nello stato "start" quindi si consiglia di tenere ext_LT=1 solo per il tempo strettamente necessario, al fine di non aspettare troppo lo spegnimento di questi
Osservazione2: Le linee wait0 e wait1 sono visibili all'interno del Global_system, ma non lo sono all'esterno in quanto in questa versione sono poste (per default), entrambe a 0

Write_manager (Vedi Appendice)

La linea *internal_wr* (scrittura nel ***byte_block***) viene attivata solo se:

- Non è in corso una trasmissione (*end_tx=1*) e non si è in fase di Lamp-Test (LT1=0) e viene dato dall'esterno l'impulso di write

La linea *write* viene attivata solo se:

- Inizia una fase di trasmissione dati (*internal_wr=1*)
- Inizia una fase di Lamp-Test (*LT=1*)
- E' ancora in corso la trasmissione dati (*rd_adrs(0) + rd_adrs(1) + rd_adrs(2)* end_tx_half=1*)
- E' ancora in fase di Lamp-Test (*rd_adrs(0) + rd_adrs(1) + LT2*end_tx_half=1*)

I2C_block (Vedi Appendice)

Si occupa di trasmettere i dati in ingresso utilizzando il protocollo I2C; è composto da 4 blocchi:

- ***ss_gen***: si occupa, quando viene attivato (linea Start o Stop), di commutare le linee *SDA* e *SCL* per generare lo Start o lo Stop, e di indicare in uscita quando la sequenza è stata completata (*Start_ok* o *Stop_ok*)
- ***bit_tx_block***: effettua la trasmissione del byte effettivo; a ogni bit trasmesso viene abilitata la linea *tx_ok*, mentre alla conclusione dell'intero byte viene abilitata la linea *byte_tx_ok*
- ***ack_det***: si occupa di rilevare l'impulso di *acknowledge* e di attivare una delle linee *byte_tx_ok*, *byte_tx_ko* in dipendenza della ricezione o meno dell'impulso.
- ***controller***: si occupa di coordinare i vari blocchi, attivandoli al momento opportuno, e aspettando da questi l'indicazione di "sequenza completata"

ss_gen (Vedi Appendice)

All'avvio i FF sono resettati e il blocco si trova nella condizione STOP, infatti *stop_ok=1,* all'arrivo del segnale di *Start*[13] viene abbassata la linea *stop_ok* ed esegue la "sequenza di Start" che termina ponendo *start_ok=1*[14].
Nel momento in cui arriva il segnale di *Stop*, viene effettuata la sequenza contraria: abbassa la linea *Start_ok*, esegue la sequenza di Stop (inversa a quella di Start) e alza la linea *Stop_ok*
NB: Le linee *nsdata* e *nsclock* rappresentano le linee *SDA* e *SCL*, ma negate

Bit_tx_block (Vedi Appendice)

Il Blocco è suddiviso in due parti:

- Coppia di FF che si occupa di trasmettere il singolo bit:

Quando i FF vengono attivati (*bit_tx=1*) viene eseguita una sequenza di due passi, nei quali *sclock (SCL)* parte da *sclock=0* ed effettua due commutazioni, mentre *tx_ok=1* alla seconda commutazione di *sclock*

- Un contatore che utilizza come *clock* la linea *bit_tx*:

Quando il contatore ha effettuato un ciclo completo viene attivata la linea *byte_tx_ok*

[13] In questa condizione l'ingrasso STOP è irrilevante
[14] In questa condizione l'ingrasso START è irrilevante

ack_det (Vedi Appendice)

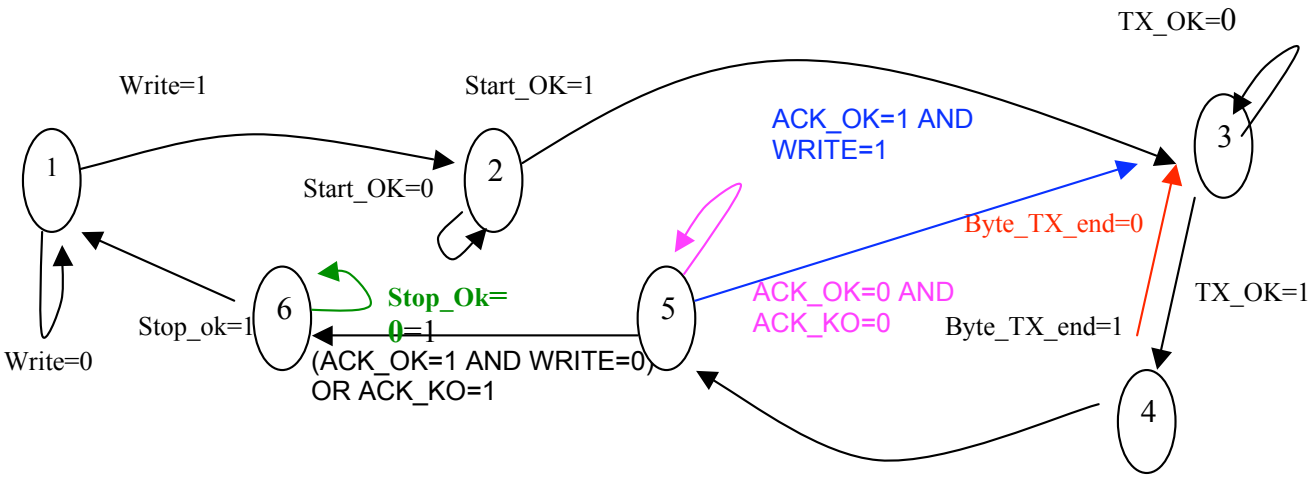
Quando attivato, attiva l'alta impedenza della linea *SDA*, legge ciò che è presente sulla linea stessa, disattiva l'alta impedenza della linea *SDA*, e alza una delle due linee *ack_ko* o *ack_ok* dipendentemente da ciò che è stato precedentemente letto

Controller (Vedi Appendice)

E' una macchina a stati che ha il compito di abilitare gli altri blocchi e aspettare che questi finiscano il lavorare.

Gli stati sono:

Stato 1: **Stand_by:** Start=0, Bit_TX=0, Mux_sel=0, R_count=0, Ack=0, Stop=0;
Stato 2: **Start:** Start=1, Bit_TX=0, Mux_sel=0, R_count=0, Ack=0, Stop=0;
Stato 3: **Bit_TX_State:** Start=0, Bit_TX=1, Mux_sel=1, R_count=1, Ack=0, Stop=0;
Stato 4: **Byte_TX_State:** Start=0, Bit_TX=0, Mux_sel=1, R_count=1, Ack=0, Stop=0;
Stato 5: **Ack_Test:** Start=0, Bit_TX=0, Mux_sel=0, R_count=0, Ack=1, Stop=0;
Stato 6: **Stop:** Start=0, Bit_TX=0, Mux_sel=0, R_count=0, Ack=0, Stop=1.



Descrizione parte Hardware

La parte Hardware è composta dalla “scheda di visualizzazione” (in dotazione) e dalla “scheda d’espansione” (non in dotazione). Per il progetto delle due schede si è fatto uso del software freeware allegato: FIDOCAD v0.96 comunque scaricabile[15] al seguente indirizzo: http://www.enetsystems.com/~lorenzo/fidocad_win.asp
Le schede, i cui PCB sono sia in appendice[16] che su file FIDOCAD (.fcd), sono composte dai seguenti componenti:

- “scheda di visualizzazione”:

R= 100 ohm
R1= 220 Kohm
R2= 12 Kohm
C1= 2.7 nF
C2= 100 nF
Q= transistor BJT NPN 2n2222 o BJT NPN BC107

- “scheda di espansione”

R=100 ohm
R1= 220 Kohm
R2= 12 Kohm
R3= 10 Kohm
C1= 2.7 nF
C2= 100 nF
Q= transistor BJT NPN 2n2222 o BJT NPN BC107
J= n°1 jumper che collega il centro con 1 dei 4 pin adiacenti (seleziona l’indirizzo del chip)

Limiti d’impiego

Si **garantisce** il corretto funzionamento **solo** nei seguenti casi:
→ si usa la scheda Xess XSA 50 Board con FPGA Xilinx Spartan2 XC2550
→ il clock di sistema è inferiore a F≤200 KHz
→ si rispettano le condizioni di utilizzo della scheda Xess e del chip SAA1064

[15] In data 23/12/2004
[16] Non in dimensione reale

Bibliografia

Protocollo I2C

- I2C Bus Specification, Philips 1995 (www.philips.com)
- Tesi di Laurea in Controlli Digitali e PLC C/o Universita' degli Studi del Sannio, Facolta' di Ingegneria, C.d.L. Ing informatica"Progetto e Simulazione di una centralina per reti di domotica" di Angelo Palladino a.a. 2002-2003

Implementazione

- "Simulazione di una comunicazione fra dispositivi che utilizzano il protocollo i2c" di Piccolo Fabio (http://imagets10.univ.trieste.it/didattica/tesine_elettro/i2c/i2c.pdf)

SAA1064 Data Sheet

- www.philips.com

FidoCad

- http://www.enetsystems.com/~lorenzo/