

Contents

1 Guida agli esami	2
1.1 Mapping	2
1.1.1 Attivare una data sezione di memoria	2
1.2 Chip Select	2
1.2.1 Come funziona la lettura da un chip select	3
1.3 Lettura e scrittura dalle porte	3
1.3.1 Scrivere i dati in input letti da INPUT_PORT a un dato indirizzo	3
1.3.2 Scrivere un dato letto da un un idirizzo in OUTPUT_PORT	4
1.3.3 Leggere un dato da INPUT_PORT e scriverlo in OUTPUT_PORT	5
1.3.4 Trasferimenti da diverse porte di Input	7

1 Guida agli esami

1.1 Mapping

1.1.1 Attivare una data sezione di memoria

La consegna ci dice che è presente un'ulteriore memoria EPROM, denominata EPROM_OPT, da 1 GB mappata a partire da 0x40000000 in accordo a quanto indicato in seguito. All'avvio, EPROM_OPT dovrà essere disabilitata. Se non abilitata, EPROM_OPT dovrà abilitarsi in seguito alla lettura consecutiva di quattro byte divisibili per 8 (zero escluso) da INPUT_PORT mentre, quando abilitata, EPROM_OPT dovrà disattivarsi in seguito alla lettura consecutiva da INPUT_PORT di quattro byte divisibili per 8 (zero escluso). Il procedimento di abilitazione e disabilitazione di EPROM_OPT dovrà avvenire continuamente mediante opportune reti logiche e senza alcun ausilio software.

I chip select di EPROM_OPT saranno:

$CS_EPROM_OPT_0 = BA31 \cdot BA30 \cdot BE0 \cdot ACTIVE$

$CS_EPROM_OPT_1 = BA31 \cdot BA30 \cdot BE1 \cdot ACTIVE$

$CS_EPROM_OPT_2 = BA31 \cdot BA30 \cdot BE2 \cdot ACTIVE$

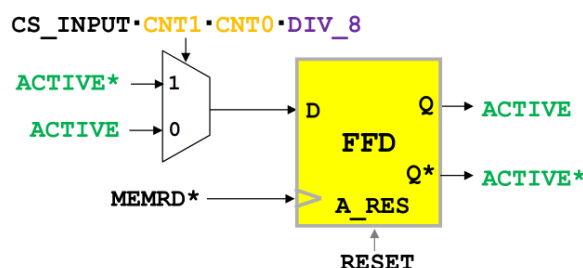
$CS_EPROM_OPT_3 = BA31 \cdot BA30 \cdot BE3 \cdot ACTIVE$

Mentre le reti logiche necessarie sono le seguenti:

$$DIV_8 = BD2 \cdot BD1 \cdot BD0 \cdot (BD7 + BD6 + BD5 + BD4 + BD3)$$



Un FFD, opportunamente inizializzato all'avvio, consente di generare il segnale **ACTIVE** che condiziona la presenza di **EPROM_OPT**:

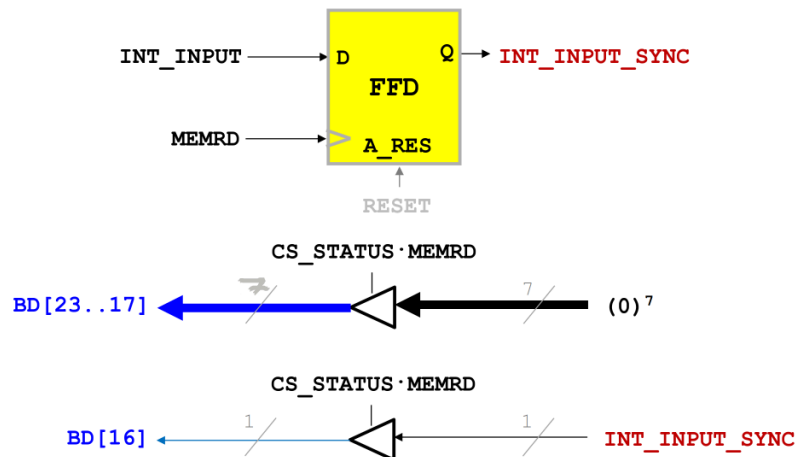


1.2 Chip Select

1.2.1 Come funziona la lettura da un chip select

Abbiamo

- un chip select CS_STATUS mappato a 0x80000002h
- BD[15..0] già utilizzato
- necessità di leggere un segnale INT_INPUT_SYNC



Quando CS_STATUS e MEMRD sono asseriti, il driver 3-state permette il passaggio dei dati. Quando viene effettuata una lettura all'indirizzo di CS_STATUS

- i bit BD[23..17] vengono impostati a 0
- il bit BD[16] viene impostato al valore corrente di INT_INPUT_SYNC

Ora prendiamo come riferimento questo schema:

Indirizzo: 0x80000002 (ultimi due bit: 10)
Word: BD[7..0] | BD[15..8] | BD[23..16] | BD[31..24]
Byte #: Byte 0 | Byte 1 | Byte 2 | Byte 3
Indirizzo: 00 01 10 11

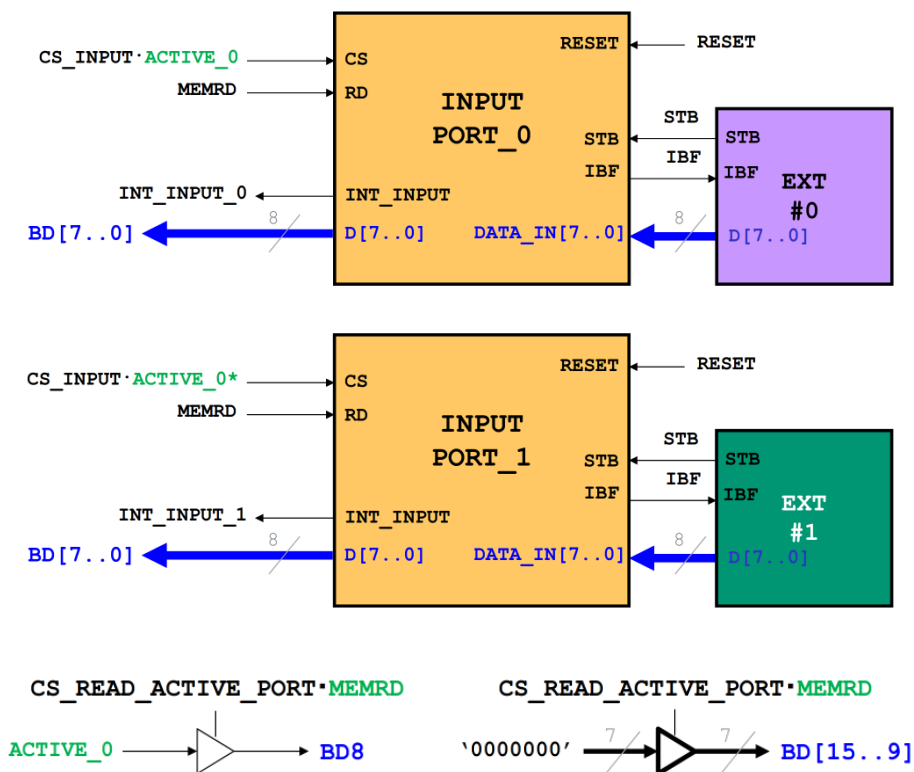
se si esegue una LBU andando a leggere su CS_STATUS (0x80000002h) si leggono i bit BD[23..16] che ha come bit meno significativo il valore di INT_INPUT_SYNC, mentre gli altri bit a 0.

1.3 Lettura e scrittura dalle porte

1.3.1 Scrivere i dati in input letti da INPUT_PORT a un dato indirizzo

Riporto l'esempio dell'esame del 21/12/2023 in cui viene chiesto che **il dato signed letto da INPUT_0 dovrà essere scritto a FFFFFFF0h**, mentre **i dati unsigned letti da INPUT_1 dovranno essere scritti a 80000000h**.

In questo caso le porte trasferiscono sullo stesso bus dati perché non trasferiscono mai in contemporanea.



Codice del DLX:

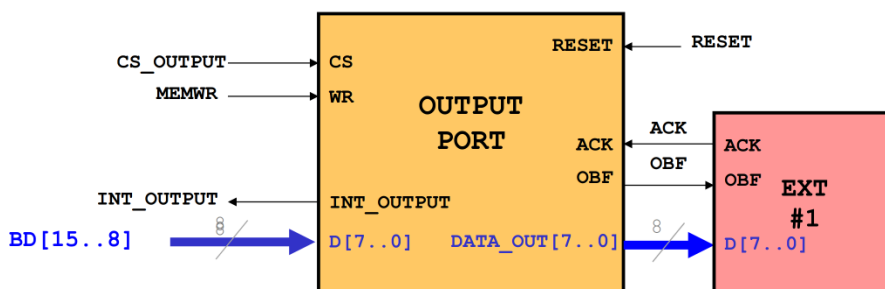
```

00000000: LHI R20,0x6000      ; R20 = 60000000h
00000004: LBU R21,0x0001(R20) ; legge il valore di ACTIVE_0
00000008: BEQZ R21,PORT_1     ; se R21=0 è attiva INPUT_PORT_1
;-----
;Questo è il codice per leggere da PORT_0
0000000C: SUBI R22,R0,0x0010  ; R22 = FFFFFFF0h, quindi carico in un registro
l'indirizzo in cui bisogna scrivere il dato
00000010: LB R21,0x0000(R20)   ; legge byte signed da INPUT_PORT_0
00000014: SB R21,0x0000(R22)   ; scrive il byte letto a FFFFFFF0h
00000018: RFE
;-----
;Questo è il codice per leggere da PORT_1
PORT_1:
0000001C: LHI R22,0x8000      ; R22 = 80000000h
00000020: LBU R21,0x0000(R20) ; legge byte unsigned da INPUT_PORT_1
00000024: SB R21,0x0000(R22) ; scrive il byte letto a 80000000h
00000028: RFE

```

1.3.2 Scrivere un dato letto da un un idirizzo in OUTPUT_PORT

La consegna dice che in OUTPUT_PORT dovrà essere scritto il byte letto all'indirizzo 0xFF000020

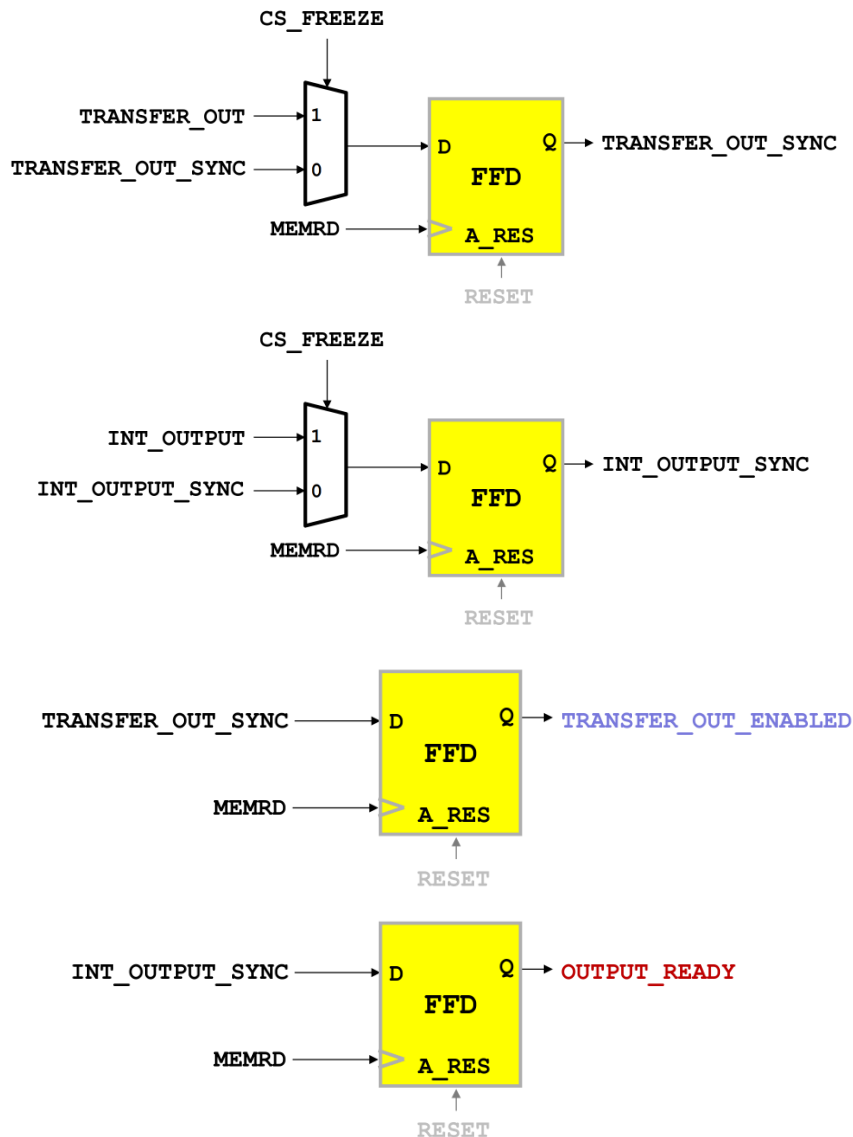


Codice DLX dell'interrupt handler

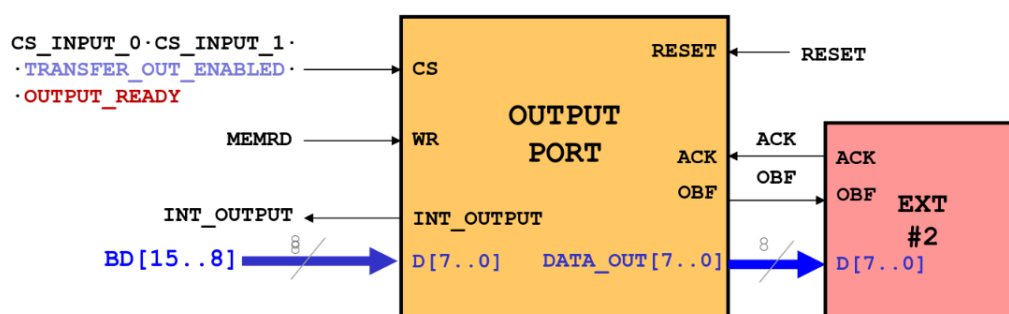
```

00000000: LHI R20,0x8000      ; R20 = 80000000h
00000004: LHI R24,0xFF00      ; R24 = FF000000h
00000008: LBU R21,0x0002(R20) ; legge il valore di INT_INPUT_SYNC
0000000C: BEQZ R21,OUTPUT     ; se R21=0, gestisce OUTPUT_PORT
00000010: LBU R21,0x0000(R20) ; legge byte da INPUT_PORT
00000014: SB R21,0x0000(R24) ; scrive il byte letto a FF000000h

```

Utilizzando due flip-flop in cascata (come mostrato nello schema con i due FFD), si garantisce che il segnale campionato sia stabile prima di essere utilizzato dal resto del sistema. Il **primo FFD cattura il segnale**, mentre il **secondo assicura che eventuali oscillazioni o glitch siano stati eliminati**.



N.B. CS_FREEZE è un segnale indispensabile in questi casi:

- Impedisce che variazioni transitorie dei segnali TRANSFER_OUT e INT_OUTPUT possano propagarsi nel sistema quando non desiderato
- Quando CS_FREEZE è attivo (1):
 - Il multiplexer seleziona l'ingresso 1 (TRANSFER_OUT o INT_OUTPUT)
 - Permette il campionamento di un nuovo valore
- Quando CS_FREEZE è inattivo (0):
 - Il multiplexer seleziona l'ingresso 0 (TRANSFER_OUT_SYNC o INT_OUTPUT_SYNC)
 - Mantiene il valore precedentemente campionato

Questo verrà utilizzato nel codice del DLX Interrupt prima di una lettura: si fa una lettura fittizia (*dummy read*) all'indirizzo in cui è mappato CS_FREEZE, cosicché le variazioni dei segnali

TRANSFER_OUT o INT_OUTPUT vengano propagate correttamente sulle uscite dei FFD, cioè sui segnali TRANSFER_OUT_ENABLED e OUTPUT_READY.

Codice DLX dell'interrupt handler:

```
00000000 LHI R25,0x4000      ; R25=40000000h
00000004 LBU R26,0x0002(R25)  ; CS_FREEZE (dummy read)
00000008 LHI R27,0xB000      ; R27=B0000000h
0000000C LHU R26,0x0000(R25)  ; legge 16 bit da porte in input
00000010 SH R26,0x0000(R27)   ; scrive 16 bit a B0000000h
00000014 RFE
```

Come si evince dal codice il trasferimento dei dati da INPUT_PORT_1 a OUTPUT_PORT non avviene via software, ma solo con l'ausilio di reti combinatorie.

1.3.4 Trasferimenti da diverse porte di Input

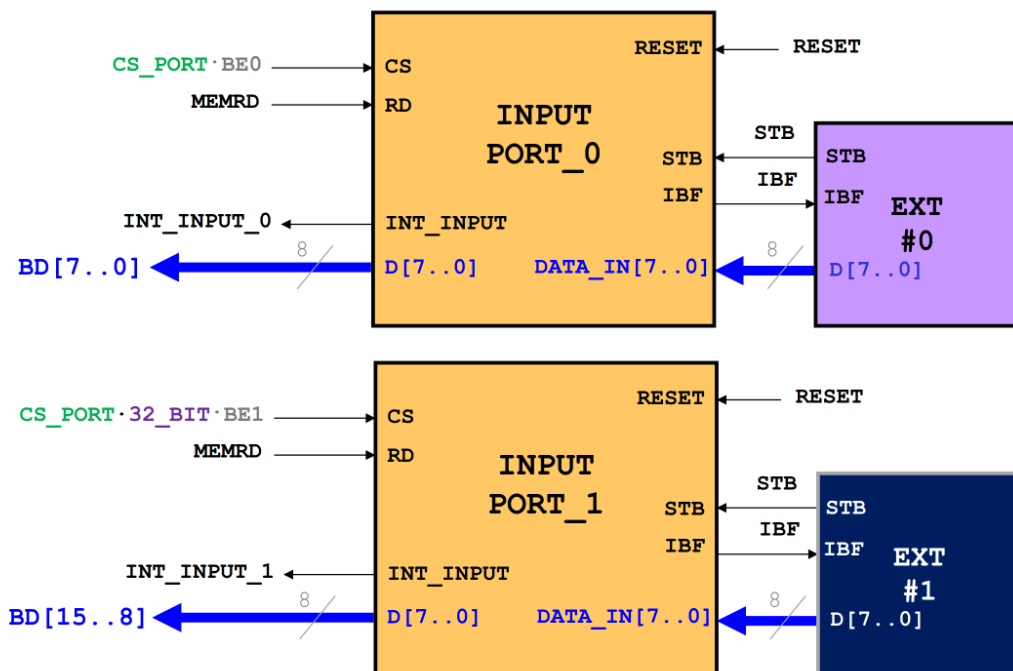
Riferimento all'esame del 17/01/2023.

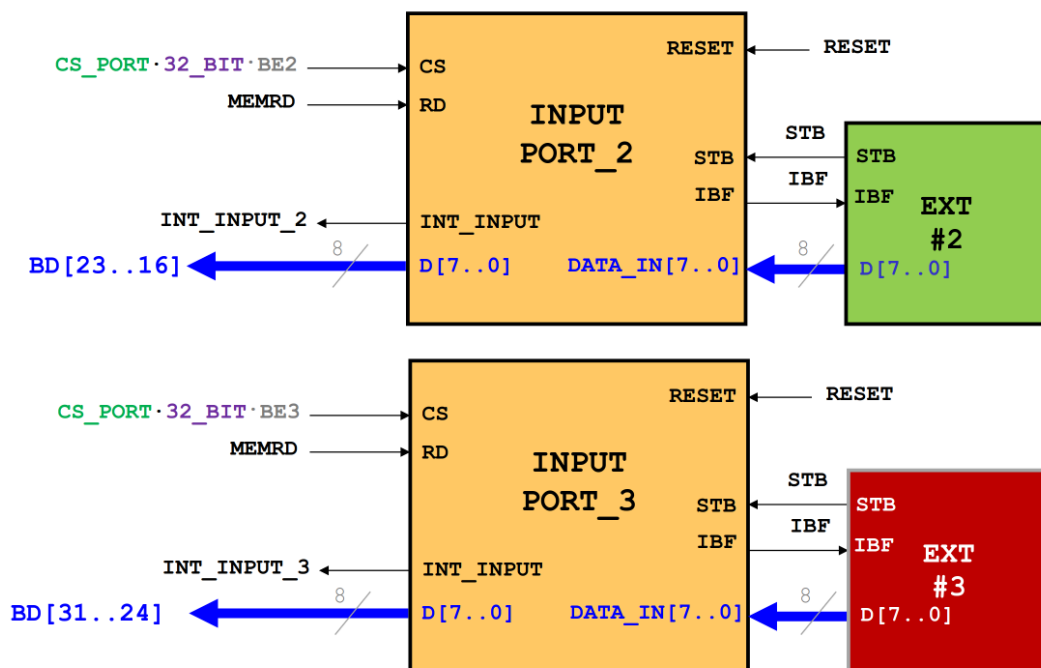
Progettare un sistema, basato su un processore DLX nel quale sono presenti quattro porte in input (denominate INPUT_PORT_0, INPUT_PORT_1, INPUT_PORT_2 e INPUT_PORT_3) e una porta in output (denominata OUTPUT_PORT).

Sin dall'avvio, e mediante l'ausilio di opportune reti logiche: **ogni 6 trasferimenti di un byte di tipo *signed* da INPUT_PORT_0, dovrà essere eseguito un unico trasferimento (a 32 bit) dalle 4 porte in input** e così via (i.e., 6 trasferimenti da INPUT_PORT_0, un unico trasferimento a 32 bit dalle quattro porte, 6 trasferimenti da INPUT_PORT_0, eccetera). Inoltre, il byte letto da INPUT_0 (indipendentemente dal fatto che si stia leggendo da una singola o dalle quattro porte in input) dovrà essere contemporaneamente inviato, nel caso questo sia possibile, anche a OUTPUT_PORT. Quanto letto dalla/e porta/e in input dovrà essere scritto, come word, all'indirizzo 0xF0000008.

I Chip Select delle porte saranno:

```
CS_PORT 60000000h
CS_INPUT_PORT_0 60000000h (CS_PORT + 0)
CS_INPUT_PORT_1 60000001h (CS_PORT + 1)
CS_INPUT_PORT_2 60000002h (CS_PORT + 2)
CS_INPUT_PORT_3 60000003h (CS_PORT + 3)
```





Un contatore modulo 8 consente di tenere traccia dei trasferimenti dalla/dalle porta/e in input in accordo a quanto indicato nel testo del problema. In particolare, il segnale 32_BIT asserito indica che il trasferimento dovrà essere effettuato contemporaneamente dalle 4 porte in input. Tale segnale, ottenuto elaborando l'uscita del contatore, risulta:

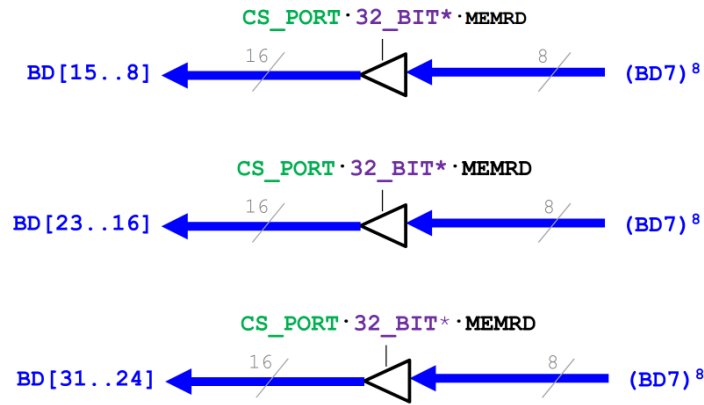
$$32_BIT = CNT2 \cdot CNT1$$



Il segnale 32_BIT è utilizzato anche per condizionare la richiesta di interrupt inviata al DLX nel modo seguente:

$$INT_DLX = INT_INPUT_0 \cdot 32_BIT^* + INT_INPUT_0 \cdot INT_INPUT_1 \cdot INT_INPUT_2 \cdot INT_INPUT_3 \cdot 32_BIT$$

Per velocizzare l'esecuzione dell'interrupt handler, si evita di verificare quale tipo di trasferimento è abilitato. Pertanto, non si legge il segnale 32_BIT che indica se il trasferimento deve avvenire unicamente da INPUT_PORT_0 o contemporaneamente dalle quattro porte in input. A tal fine, con l'ausilio della rete seguente, il codice dell'interrupt handler eseguirà sempre una lettura di una word anche quando è necessario trasferire solo da INPUT_PORT_0 (i.e., quando 32_BIT=0) evitando così la lettura via software di 32_BIT e una consecutiva istruzione di branch.

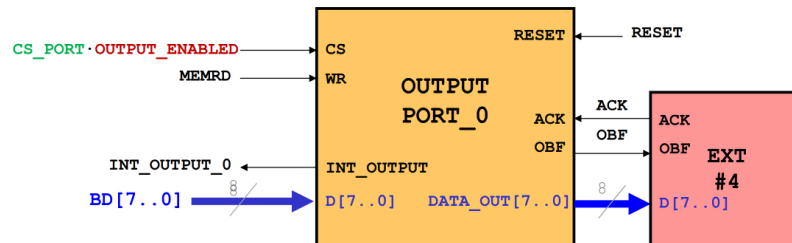


N.B. Siccome il byte da leggere da INPUT_PORT_0 è di tipo *signed*, non possiamo semplicemente mettere i bit BD[31..8] a 0, ma bisogna seguire la regola dell'etensione del segno:

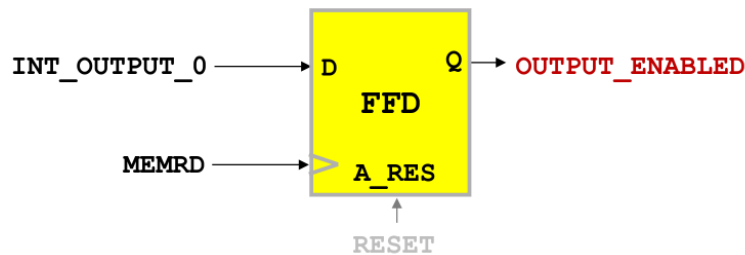
- se il numero è positivo, si pongono a 0 i bit rimanenti
- se il numero è negativo, si pongono a 1 i bit rimanenti

ed è per questo che colleghiamo a BD[15..8], BD[23..16], e BD[31..24] il bit (BD7)^8 (che indica il segno del numero), e non semplicemente degli 0.

Infine, nel sistema è anche presente una porta in output attraverso la quale trasferire, quando possibile, il dato letto da INPUT_PORT_0 contemporaneamente all'esecuzione di questa operazione.



Il segnale OUTPUT_ENABLED, utilizzato per condizionare il chip-select di OUTPUT_PORT_0, è ottenuto campionando sul fronte di salita di MEMRD il segnale INT_OUTPUT_0 come segue:



Codice DLX dell'interrupt handler:

```
00000000: LHI R20,0x6000      ; R20 = 60000000h
00000004: LW R21,0x0000(R20)  ; legge in R21 una word a 60000000h
00000008: LHI R22,0xF000      ; R22 = F0000000h
0000000C: SW R21,0x0008(R22)  ; scrive R21 a F0000008h
00000010: RFE
```