Esami svolti di Calcolatori Elettronici M

Kevin Michael Frick

6 gennaio 2021

Nota: Nell'anno accademico in cui questo documento è stato redatto, un modulo del corso è stato tenuto dal professor Bartolini, il quale ha strutturato il proprio modulo in maniera leggermente diversa dal professor Cinotti. Questo si è riflettuto nella formulazione delle prove d'esame, nelle quali non è più richiesto di disegnare nel dettaglio gli interfacciamenti delle periferiche come in passato, cosa che portava essenzialmente a dover ricopiare gli schemi presenti nelle slide. Lo stesso vale per la costruzione dei descrittori, anch'essa non più richiesta. Questo formulazione del corso e della prova d'esame si riflette nelle soluzioni dettagliate di seguito.

1 2018-12-20

1.1 Specifiche

Il sistema dispone di:

- una porta seriale UART di I/O S1 gestita in DMA con velocità di 19200b/s.
- un PIC;
- 1 MiB di EPROM mappata agli indirizzi alti;
- 1 MiB di RAM mappata agli indirizzi bassi;
- 8 KiB di cache a 2 vie e linee da 32B, gestita con politica write-around.

Un segmento dati SD mappato a `0x0000 3800` contiene due vettori di 64B ciascuno denominati A e B.

Il sistema esegue i seguenti passi:

```
A[0] <- 0;
A[i] <- A[i - 1] + 3, i = 1..63;
Per sempre:
  Trasmissione di A su S_1 in DMA;
  Ricezione di B da S_1 in DMA;
  A <- A + B
```

1.2 Mappa della memoria

- Il vettore A è mappato (come da consegna) da `0x0000 3800` a `0x0000 383F`.
- Il vettore B è mappato (come da consegna) da `0x0000 3840` a `0x0000 387F`.
- La UART è mappata (per scelta progettuale) da `0x4000` a `0x4007`.
- Il DMAC è mappato (per scelta progettuale) da `0x5000` a `0x500F`.
- I latch HL e HH che consentono al DMAC di emettere 32 bit di indirizzo sono mappati (per scelta progettuale) a `0x6000` e `0x7000`.
- Il PIC è mappato (per scelta progettuale) da `0x8000` a `0x8001`.
- La RAM è mappata da `0x0` a `0x000F FFFF`.
- La EPROM è mappata da `0xFFF0 0000` a `0xFFFF FFFF`.

1.3 Interfacciamento UART

Si considerino tutti i segnali come attivi alti, per chiarezza. Il pin chip select della porta seriale è collegato a `CS = !HOLDA * CS_S_1 + HOLDA * DACK0`, con `CS_S_1 = BA14`.

I pin di indirizzamento della porta seriale sono collegati a `Ai = !HOLDA * BAi + HOLDA * 0` in modo da permettere di indirizzare gli 8 registri interni quando la CPU è master (`HOLDA = 0`) e di leggere il registro contenente il dato quando il DMAC è master (`HOLDA = 1`).

Il segnale `SRQ` è collegato al pin `DRQ0` del DMAC.

1.4 DMAC

Il segnale `DRQ0` è collegato al pin `SRQ` di richiesta di servizio della UART.

Il DMA è programmato come segue prima degli invii di A:

1. Maschera il canale 0
2. Reset FF interni
3. `BAR0 = 0x3800`
4. `BCR0 = 0x003F`
5. `MR = 01 0 0 10 00 = 0x48` (Single mode, autoincr, no autoinit, read, ch00)
6. `HH = 0x00`
7. `HL = 0x00`
8. Smaschera il canale 0

```
mov al 0x04 ; agisci sul bit mask del canale 00, settalo a 1 (1b ## 00b)
outb 0x500A ; Write Single Mask Register Bit (1010b)

mov al 0x01
outb 0x500B ; Clear Byte Pointer FlipFlop (1100b)

mov al 0x00 ; BAR0 = 0x3800
outb 0x5000 ; BAR0[7..0]
mov al 0x38
outb 0x5000 ; BAR0[15..8]

mov al 0x3F ; BCR0 = 0x003F
outb 0x5001 ; BCR0[7..0]
mov al 0x00
outb 0x5001 ; BCR0[15..8]

mov al 0x48
outb 0x500B ; MR0

mov al 0x00
outb 0x6000 ; HH

mov al 0x00
outb 0x7000 ; HL

mov al 0x00 ; unmask ch00
outb 0x500A
```

E come segue prima delle ricezioni di B:

1. Maschera il canale 0
2. Reset FF interni
3. `BAR = 0x3840`

4. `BCR = 0x003F`
5. `MR = 01 0 0 01 00 = 0x44` (Single mode, autoincr, no autoinit, write, ch00)
6. `HH = 0x00`
7. `HL = 0x00`
8. Smaschera il canale 0

```
mov al 0x04 ; agisci sul bit mask del canale 00, settalo a 1 (1b ## 00b)
outb 0x500A ; Write Single Mask Register Bit (1010b)

mov al 0x01
outb 0x500B ; Clear Byte Pointer FlipFlop (1100b)

mov al 0x40 ; BAR0 = 0x3840
outb 0x5000 ; BAR0[7..0]
mov al 0x38
outb 0x5000 ; BAR0[15..8]

mov al 0x3F ; BCR0 = 0x003F
outb 0x5001 ; BCR0[7..0]
mov al 0x00
outb 0x5001 ; BCR0[15..8]

mov al 0x44
outb 0x500B; MR0

mov al 0x00
outb 0x6000 ; HH

mov al 0x00
outb 0x7000 ; HL

mov al 0x00 ; unmask ch00
outb 0x500A
```

1.5 Cache

1.5.1 Indici

La cache ha dimensione 8KiB, linee da 32 byte e 2 vie. Questo significa che vi sono $S = \frac{8192}{32 \cdot 2} = 128$ set. Si hanno quindi 7 bit di set ID, 5 bit di offset, e i restanti 20 bit di tag.

Il vettore A è mappato da `0x3800 = 0x03 ## 0x40 ## 0x00` a `0x383F = 0x03 ## 0x41 ## 0x1F`.

A occuperà quindi i set `0x40, 0x41` e gli offset `0x00-0x1F` e avrà tag `0x03`.

Il vettore B è mappato da `0x3840 = 0x03 ## 0x42 ## 0x00` a `0x387F = 0x03 ## 0x43 ## 0x1F`.

B occuperà quindi i set `0x42, 0x43` e gli offset `0x00-0x1F` e avrà tag `0x03`.

I due vettori hanno lo stesso tag e occupano gli stessi intervalli di offset, ma non collidono perché occpano set diversi.

I vettori possono quindi stare contemporaneamente in cache, occupando 4 linee.

1.5.2 Dinamica della cache

1. La scrittura di `A[0]` provoca **una miss** in scrittura.
2. Al momento della lettura di `A[0]` si ha **una miss** in lettura.

   In risposta, i valori di `A[31..0]` vengono caricati in cache, **passando dallo stato** I **allo stato** E.

   Dopo la scrittura di `A[1]`, la linea **passa in stato** M.

Si ha **una miss** in lettura al momento della lettura di `A[32]`, **portando i valori di** `A[63..32]` **in cache con stato** E. Come prima, dopo la scrittura di `A[32]`, la linea che contiene `A[63..32]` **passa in stato** M.

I passi 1 e 2 coinvolgono 64 scritture, 63 letture e 2 miss.

3. Al momento della trasmissione dei valori di `A[i]` le linee di cache che contengono `A[63..32]` e `A[31..0]` **passano dallo stato** M **allo stato** S.

   In questa fase si hanno **due cicli di implicit WB**, uno per ogni metà del vettore.

   Questo passo non richiede accessi alla cache da parte della CPU.

4. La ricezione dei valori di `B` in DMA non altera gli stati di alcun vettore.

   Queesto passo non richiede accessi alla cache da parte della CPU.

5. Si hanno **due** miss in lettura al momento della lettura dei valori di `B[0]` e `B[32]`, che fanno passare rispettivamente `B[31..0]` e `B[63..32]` in cache, **portandoli dallo stato** I **allo stato** E.

   Come prima, dopo la scrittura di `A[0]`, la linea che contiene `A[31..0]` **passa in stato** M. Lo stesso accade alla linea che contiene `A[63..32]` dopo la scrittura di `A[32]`.

   Questa fase quindi richiede 64 letture dei valori di A, 64 letture dei valori di B, e 64 scritture dei valori di A, e provoca due miss.

6. La seconda esecuzione del passo 3 si svolge in maniera analoga alla prima.

7. La seconda esecuzione del passo 4 porta lo stato delle linee di cache nelle quali è memorizzato il vettore `B` **a passare da** E **a** I.

8. La seconda esecuzione del passo 5 si svolge in maniera analoga alla prima.

I passi 1 e 2 richiedono quindi 127 accessi alla cache complessivi, mentre i passi 3, 4 e 5 ne richiedono 192. Eseguire una volta i passi 1 e 2 e ripetere due volte i passi 3, 4, 5 richiede quindi 511 accessi in memoria.

Si hanno quindi 5 miss complessive su 511 accessi, per una **miss rate** totale pari a $MR = 0.97\%$.

2 2017-02-07

2.1 Specifiche

Il sistema dispone di:

- una porta seriale di input `S_IN` gestita in DMA con velocità di 9600 b/s;
- una porta seriale di output `S_OUT` gestita in DMA con velocità di 9600 b/s;
- un PIC;
- 4 MiB di EPROM mappata agli indirizzi alti;
- 1 MiB di RAM mappata agli indirizzi bassi
- 8 KiB di cache a 2 vie con linee da 32B gestita in write-around.

Un segmento dati SD mappato all'indirizzo `0x0002 1602` contiene tre vettori A, B, C da 12B ciascuno.

Il sistema esegue i seguenti passi:

```
Per sempre:
  Ricezione di A in DMA da S_IN;
  A <- A + 5;
  Ricezione di B in DMA da S_IN;
  C <- A + B;
  Trasmissione di C in DMA su S_OUT;
```

2.2 Mappa della memoria

- il vettore A è mappato (come da consegna) da `0x0002 1602` a `0x0002 160D`;
- il vettore B è mappato (come da consegna) da `0x0002 160E` a `0x0002 1619`;

- il vettore C è mappato (come da consegna) da `0x0002 161A` a `0x0002 1625`;
- il DMAC è mappato (per scelta progettuale) da `0x3000` a `0x300F`;
- `S_IN` è mappata (per scelta progettuale) da `0x4000` a `0x4003`;
- `S_OUT` è mappata (per scelta progettuale) da `0x5000` a `0x5003`;
- il PIC è mappato (per scelta progettuale) da `0x6000` a `0x6001`;
- I latch HL e HH che consentono al DMAC di emettere 32 bit di indirizzo sono mappati (per scelta progettuale) a `0x7000` e `0x8000`.
- la RAM è mappata (come da consegna) da `0x0000 0000` a `0x000F FFFF`;
- la EPROM è mappata (come da consegna) da `0xFFC0 0000` a `0xFFFF FFFF`.

2.3 Interfacciamento UART

Si considerino tutti i segnali come attivi alti, per chiarezza.

Il pin chip select di `S_IN` è collegato a `CS = !HOLDA * CS_S_IN + HOLDA * DACK0`.

Il pin chip select di `S_OUT` è collegato a `CS = !HOLDA * CS_S_OUT + HOLDA * DACK1`.

I pin di indirizzo `A[1..0]` di entrambe le porte sono collegati a `Ai = !HOLDA * BAi + HOLDA * 0` in modo da permettere alla CPU, quando essa è master, di abilitare o disabilitare la ricezione o la trasmissione quando è master e al DMAC, quando esso è invece master, di leggere o scrivere il singolo byte.

Non è necessario interfacciare anche il pin `A2` dato che ogni porta ha tre possibili indirizzi: abilitazione dell'I/O, disabilitazione dell'I/O, lettura/scrittura del singolo byte.

Il segnale `SRQ` di `S_IN` è collegato al pin `DREQ0` del DMAC. Il segnale `SRQ` di `S_OUT` è collegato al pin `DREQ1` del DMAC.

2.4 DMAC

Il DMAC è programmato come segue prima delle ricezioni di `A`:

1. Maschera il canale 0
2. Reset FF interni
3. `BAR0 = 0x1602`
4. `BCR0 = 0x000B`
5. `MR = 01 0 0 01 00` (Single mode, autoincr, no autoinit, write, ch00)
6. `HH = 0x00`
7. `HL = 0x02`
8. Smaschera il canale 0

Il DMAC è programmato come segue prima delle ricezioni di `B`:

1. Maschera il canale 0
2. Reset FF interni
3. `BAR0 = 0x160E`
4. `BCR0 = 0x000B`
5. `MR = 01 0 0 01 00` (Single mode, autoincr, no autoinit, write, ch00)
6. `HH = 0x00`
7. `HL = 0x02`
8. Smaschera il canale 0

Il DMAC è programmato come segue prima degli invii di `C`:

1. Maschera il canale 1
2. Reset FF interni
3. `BAR0 = 0x160A`
4. `BCR0 = 0x000B`
5. `MR = 01 0 0 10 01` (Single mode, autoincr, no autoinit, read, ch01)
6. `HH = 0x00`
7. `HL = 0x02`
8. Smaschera il canale 1

2.5 Cache

2.5.1 Indici

La cache ha dimensione 8 KiB, linee da 32B e 2 vie. Questo significa che vi sono $S = \frac{8K}{32 \cdot 2} = 128$ set. Si hanno quindi 7 bit di set ID, 5 bit di offset, e i restanti 20 bit vengono usati per i tag.

Il vettore A è mappato da `0x0002 1602` a `0x0002 160D`, indirizzi che in base 2 equivalgono a:

```
   0    2    1    6    0    2
0000 0010 0001|0110000|0 0010
          0x21     0x30  0x02

   0    2    1    6    0    D
0000 0010 0001|0110000|0 1101
          0x21    0x30   0x0D
```

il vettore A occuperà quindi gli offset `0x02-0x0D` del set `0x30` con tag `0x21`.

Il vettore B è mappato da `0x0002 160E` a `0x0002 1619`, indirizzi che in base 2 equivalgono a:

```
   0    2    1    6    0    E
0000 0010 0001|0110000|0 1110
          0x21     0x30  0x0E


   0    2    1    6    1    9
0000 0010 0001|0110000|1 1001
          0x21    0x30   0x19
```

il vettore B occuperà quindi gli offset `0x0E-0x19` del set `0x30` con tag `0x21`.

Il vettore C è mappato da `0x0002 161A` a `0x0002 1625`, indirizzi che in base 2 equivalgono a:

```
   0    2    1    6    1    A
0000 0010 0001|0110000|1 0110
          0x21    0x30   0x1A

   0    2    1    6    2    5
0000 0010 0001|0110011|0 0101
          0x21    0x31   0x05
```

il vettore C occuperà quindi gli offset `0x1A-0x1F` del set `0x30` e gli offset `0x00-0x05` del set `0x31` con tag `0x21`.

I tre vettori non presentano collisioni e possono stare contemporaneamente in cache, occupando due linee.

2.5.2 Dinamica della cache

1. La ricezione di `A` in DMA non richiede accessi alla memoria da parte della CPU.

   Il DMAC opera su un byte alla volta, quindi questa fase implice **12 cicli di bus esterni**.

2. La lettura di `A[0]` provoca **una miss** in lettura.

   In risposta, la CPU porta in cache la porzione di memoria che contiene `A[11..0]`, `B[11..0]` e `C[5..0]`. La linea di cache corrispondente a questa porzione di memoria **passa dallo stato** I **allo stato** E.

   Al momento della scrittura di `A[0]` la stessa linea **passa dallo stato** E **allo stato** M.

   Questo passo richiede 12 letture dalla memoria e 12 scritture.

   Si ha **un ciclo burst di bus esterno** per riempire la linea di cache.

3. La lettura di `B` in DMA porta la linea di cache contenente il vettore **dallo stato** M **allo stato** I.

   Questo passo non richiede accessi alla memoria da parte della CPU.

   Il DMAC opera su un byte alla volta, quindi questa fase implice **12 cicli di bus esterni**.

   Si ha **un ciclo di WB** prima del trasferimento del vettore, che assicura che le modifiche apportate al vettore `A` siano salvate in memoria prima che la cache venga invalidata.

4. La lettura di `A[0]` provoca **una miss** in lettura e la CPU porta in cache la porzione di memoria che contiene `A[11..0]`, `B[11..0]` e `C[5..0]`. La linea di cache corrispondente a questa porzione di memoria **passa dallo stato** I **allo stato** E.

   La scrittura di `C[0]` **porta la linea di cache in stato** M.

   La scrittura di `C[11..6]` provoca **sei miss** in scrittura.

   Qesto passo richiede 24 letture dalla memoria e 12 scritture.

   Si ha **un ciclo burst di bus esterno** per riempire la linea di cache e **sei cicli di bus esterni** per scrivere gli ultimi 6 bit di `C`

5. La scrittura di `C[0]` in DMA fa passare la linea di cache corrispondente **dallo stato** M **allo stato** S.

   Questo passo non richiede accessi alla memoria da parte della CPU.

   Questo passo provoca **un ciclo di implicit WB**, che corrisponde a **un ciclo burst di bus esterno**, e **12 cicli di bus esterni** richiesti dal DMAC.

Si hanno quindi 8 miss su 60 accessi, per una miss rate totale pari a $MR = 13.3\%$.

3 2017-01-10

3.1 Specifiche

Il sistema dispone di:

- una porta seriale di I/O `S_IN_OUT` gestita in DMA;
- un PIC;
- 2 MiB di EPROM mappata agli indirizzi alti;
- 1 MiB di RAM mappata agli indirizzi alti;
- 8KiB di cache dati a 2 vie e linee da 32 byte gestita in write-around.

Un segmento dati SD mappato a `0x0002 0800` contiene tre vettori `A, B, C` da 20B ciascuno.

Il sistema esegue i seguenti passi:

```
Per sempre:
  Ricezione di A in DMA da S_IN_OUT
  A = NOT A
  Ricezione di B in DMA da S_IN_OUT
  C = A XOR B
  Trasmissione di C in DMA su S_IN_OUT
```

3.2 Mappa della memoria

- `S_IN_OUT` è mappata (per scelta progettuale) da `0x3000` a `0x3003`;
- il PIC è mappato (per scelta progettuale) da `0x4000` a `0x4001`;
- la EPROM è mappata (come da consegna) da `0xFFE0 0000` a `0xFFFF FFFF`;
- la RAM è mappata (come da consegna) da `0x0` a `0x000F FFFF`;
- il vettore A è mappato (come da consegna) da `0x0002 0800` a `0x0002 0813`;
- il vettore B è mappato (come da consegna) da `0x0002 0814` a `0x0002 0827`;
- il vettore C è mappato (come da consegna) da `0x0002 0828` a `ox0002 083B`;
- i latch HL e HH che consentono al DMAC di emettere 32 bit di indirizzo sono mappati (per scelta progettuale) a `0x5000` e `0x6000`.
- il DMAC è mappato (per scelta progettuale) da `0x7000` a `0x700F`;

3.3 Interfacciamento UART

Si considerino tutti i segnali come attivi alti, per chiarezza.

Il pin CS della UART è collegato a `CS = !HOLDA * CS_S_IN_OUT + HOLDA * DACK0`.

I pin `Ai` della UART sono collegati a `Ai = HOLDA * 0 + !HOLDA * BAi` in modo da permettere alla CPU di indirizzare i registri interni quando essa è master e al DMAC, quando esso invece è master, di leggere o scrivere un byte in ingresso o in uscita.

3.4 DMAC

Il DMAC è programmato come segue prima di ogni ricezione di `A`:

1. Maschera il canale 0
2. Reset FF interni
3. `BAR0 = 0x0800`
4. `BCR0 = 0x0013`
5. `MR = 01 0 0 01 00` (Single mode, autoincr, no autoinit, write, ch00)
6. `HH = 0x00`
7. `HL = 0x02`
8. Smaschera il canale 0

Il DMAC è programmato come segue prima di ogni ricezione di `B`:

1. Maschera il canale 0
2. Reset FF interni
3. `BAR0 = 0x0814`
4. `BCR0 = 0x0013`
5. `MR = 01 0 0 01 00` (Single mode, autoincr, no autoinit, write, ch00)
6. `HH = 0x00`
7. `HL = 0x02`
8. Smaschera il canale 0

Il DMAC è programmato come segue prima di ogni scrittura di `C`:

1. Maschera il canale 0
2. Reset FF interni
3. `BAR0 = 0x0828`

4. `BCR0 = 0x0013`
5. `MR = 01 0 0 10 00` (Single mode, autoincr, no autoinit, read, ch00)
6. `HH = 0x00`
7. `HL = 0x02`
8. Smaschera il canale 0

3.5 Cache

3.5.1 Indici

La cache ha dimensione 8 KiB, linee da 32B e 2 vie. Questo significa che vi sono $S = \frac{8K}{32 \cdot 2} = 128$ set. Si hanno quindi 7 bit di set ID, 5 bit di offset, e i restanti 20 bit vengono usati per i tag.

Il vettore A è mappato da `0x0002 0800` a `0x0002 0813`, indirizzi che in base 2 corrispondono a

```
   2    0    8    0    0
0010 0000|1000000|0 0000
 0x20    0x40   0x00

   2    0    8    1    3
0010 0000|1000000|1 0011
 0x20    0x40   0x13
```

Il vettore A occuperà quindi gli offset `0x00-0x13` del set `0x40` con tag `0x20`.

Il vettore B è mappato da `0x0002 0814` a `0x0002 0827`, indirizzi che in base 2 corrispondono a

```
   2    0    8    1    4
0010 0000|1000000|1 0100
     0x20    0x40   0x14

   2    0    8    2    7
0010 0000|1000001|0 0111
     0x20    0x41   0x07
```

Il vettore B occuperà quindi gli offset `0x14-0x1F` del set `0x40` e gli offset `0x00-0x07` del set `0x41` con tag `0x20`.

Il vettore C è mappato da `0x0002 0828` a `ox0002 083B`, indirizzi che in base 2 corrispondono a

```
   2    0    8    2    8
0010 0000|1000001|0 0100
     0x20    0x41   0x08

   2    0    8    3    B
0010 0000|1000001|1 1100
     0x20    0x41   0x1B
```

Il vettore C occuperà quindi gli offset `0x08-0x1B` del set `0x81` con tag `0x20`.

I tre vettori non presentano collisioni e possono stare contemporaneamente in cache, occupando due linee.

3.5.2 Dinamica della cache

1. La ricezione di `A` in DMA non richiede accessi alla memoria da parte della CPU e non altera lo stato della cache.

   Il DMAC lavora su un byte alla volta, quindi la ricezione di `A` richiede **20 cicli di bus esterni**.

2. La lettura di `A[0]` provoca **una miss in lettura**.

   In risposta, la CPU carica in cache `A[19..0]` e `B[11..0]`, portando la corrispondente linea di cache **dallo stato** I **allo stato** E mediante **un ciclo burst di bus esterno**.

   La scrittura di `A[0]` porta la corrispondente linea di cache **dallo stato** E **allo stato** M.

3. La ricezione di `B` porta la linea di cache contenente `B[11..0]` **dallo stato** M **allo stato** I, provocando inoltre **un ciclo di WB** per scrivere in memoria i nuovi valori di `A` prima di invalidare la cache.

   Il DMAC lavora su un byte alla volta, quindi la ricezione di `B` richiede **20 cicli di bus esterni**.

4. La lettura di `A[0]` provoca **una miss in lettura**.

   In risposta, la CPU carica in cache `A[19..0]` e `B[11..0]`, portando la corrispondente linea di cache **dallo stato** I **allo stato** E mediante **un ciclo burst di bus esterno**.

   La scrittura di `C[11..0]` provoca **12 miss in scrittura**.

   La lettura di `B[12]` provoca **una miss in lettura**.

   In risposta, la CPU carica in cache `B[19..12]` e `C[19..0]` mediante **un ciclo burst di bus esterno**.

   La scrittura di `C[12]` porta la corrispondente linea di cache **dallo stato** E **allo stato** M.

5. La trasmissione di `C` in DMA porta la linea di cache contenente il vettore **dallo stato** M **allo stato** S.

   La richiesta di trasmissione provoca **un ciclo di WB** che si traduce in **un ciclo burst di bus esterno**.

   Il DMAC trasmette un byte alla volta, per cui la trasmissione richiede **20 cicli di bus esterni**.

Si hanno 15 miss e 100 accessi alla memoria, per una miss rate complessiva pari a $MR = 15\%$.

4 2018-09-11

4.1 Specifiche

Il sistema dispone di:

- due porte seriali di I/O `S_1, S_3` gestite in DMA;
- una porta seriale di I/O `S_2` gestita ad interrupt;
- un PIC;
- 4 MiB di EPROM mappata agli indirizzi alti;
- 1 MiB di RAM mappata agli indirizzi bassi;
- 8 KiB di cache a 2 vie e linee da 32B, gestita in write-around.

Un segmento dati SD mappato a `0x0004 0000` contiene tre vettori `A, B, C` da 24B.

Il sistema esegue i seguenti passi:

```
Per sempre:
  Ricezione di A da S_1 in DMA;
  Ricezione di B da S_2 a interrupt;
  C = A OR B;
  Trasmissione di C su S_3 in DMA;
```

4.2 Mappa della memoria

- `S_1` è mappata (per scelta progettuale) da `0x5000` a `0x5007`;
- `S_2` è mappata (per scelta progettuale) da `0x6000` a `0x6007`;
- `S_3` è mappata (per scelta progettuale) da `0x7000` a `0x7007`;
- il PIC è mappato (per scelta progettuale) da `0x8000` a `0x8001`;
- i latch HL e HH che consentono al DMAC di emettere 32 bit di indirizzo sono mappati (per scelta progettuale) a `0x9000` e `0xA000`.
- il DMAC è mappato (per scelta progettuale) da `0xB000` a `0xB00F`;
- il vettore A è mappato (come da consegna) da `0x0004 0000` a `0x0004 0017`;
- il vettore B è mappato (come da consegna) da `0x0004 0018` a `0x0004 002F`;
- il vettore C è mappato (come da consegna) da `0x0004 0030` a `0x0004 0047`;

- la RAM è mappata (come da consegna) da `0x0000 0000` a `0x000F FFFF`;
- la EPROM è mappata (come da consegna) da `0xFFC0 0000` a `0xFFFF FFFF`.

4.3 Interfacciamento UART

Si considerino tutti i segnali come attivi alti, per chiarezza.

Il pin CS di `S_1` è collegato a `CS = !HOLDA * CS_S_1 + HOLDA * DACK0`.

Il pin CS di `S_3` è collegato a `CS = !HOLDA * CS_S_3 + HOLDA * DACK1`.

Il pin CS di `S_2` è collegato a `CS = CS_S_2`.

Il pin `SRQ` di `S_1` è collegato al pin `DRQ0` del DMAC.

Il pin `SRQ` di `S_3` è collegato al pin `DRQ1` del DMAC.

Il pin `SRQ` di `S_2` è collegato al pin `IR0` del PIC.

4.4 Interfacciamento PIC

Si considerino tutti i segnali, compresi i `bank enable`, come attivi alti, per chiarezza.

Il pin `INT` del PIC è collegato al pin `INT` della CPU.

Il pin `INTA` del PIC è collegato al pin `INTA` della CPU.

Il pin `CS` del PIC è collegato a `CS_PIC = BA15`.

Il pin `SRQ` di `S_3` è collegato al pin `IR0` del PIC.

I pin `WR, RD` del PIC sono collegati ai pin `IOWR, IORD` della CPU.

I pin `D[7..0]` del PIC sono collegati ai pin `IOB[7..0]` del bus di I/O.

Il pin `A0` del PIC è collegato a `BA0 == BE0`.

4.5 DMAC

Il DMAC è programmato come segue prima delle ricezioni di `A`:

1. Maschera il canale 0
2. Reset FF interni
3. `BAR0 = 0x0000`
4. `BCR0 = 0x0017`
5. `MR = 01 0 0 01 00` (Single mode, autoincr, no autoinit, write, ch00)
6. `HH = 0x00`
7. `HL = 0x04`
8. Smaschera il canale 0

Il DMAC è programmato come segue prima degli invii di `C`:

1. Maschera il canale 1
2. Reset FF interni
3. `BAR0 = 0x0030`
4. `BCR0 = 0x0017`
5. `MR = 01 0 0 10 01` (Single mode, autoincr, no autoinit, read, ch01)
6. `HH = 0x00`
7. `HL = 0x04`
8. Smaschera il canale 1

4.6 Cache

4.6.1 Indici

La cache ha dimensione 8 KiB, linee da 32B e 2 vie. Questo significa che vi sono $S = \frac{8K}{32 \cdot 2} = 128$ set. Si hanno quindi 7 bit di set ID, 5 bit di offset, e i restanti 20 bit vengono usati per i tag.

Il vettore A è mappato da `0x0004 0000` a `0x0004 0017`, indici che in base 2 valgono

```
   4    0    0    0   0
0100 0000|0000000|00000
     0x40    0x00  0x00

   4    0    0    1   7
0100 0000|0000000|10111
     0x40    0x00  0x17
```

A occuperà quindi gli offset `0x00-0x17` del set `0x00` con tag `0x40`.

Il vettore B è mappato da `0x0004 0018` a `0x0004 002F`, indici che in base 2 valgono

```
   4    0    0    1   8
0100 0000|0000000|11000
     0x40    0x00  0x18

   4    0    0    2   F
0100 0000|0000001|01111
     0x40    0x01  0x0F
```

B occuperà quindi gli offset `0x18-0x1F` del set `0x00` e gli offset `0x00-0x0F` del set `0x01` con tag `0x040`.

Il vettore C è mappato da `0x0004 0030` a `0x0004 0047`, indici che in base 2 valgono

```
   4    0    0    3   0
0100 0000|0000001|10000
     0x40    0x01  0x10

   4    0    0    4   7
0100 0000|0000010|00111
     0x40    0x02  0x07
```

C occuperà quindi gli offset `0x10-0x1F` del set `0x01` e gli offset `0x00-0x07` del set `0x02` con tag `0x40`.

4.6.2 Dinamica della cache

1. La ricezione di `A` in DMA non richiede accessi alla memoria da parte della CPU e non altera lo stato della cache.

   Il DMAC lavora su un byte alla volta, quindi la ricezione di A richiede **24 cicli di bus esterni**.

2. La ricezione di `B` a interrupt provoca **24 miss** in scrittura e richiede **24 cicli di bus esterni**.

3. La lettura di `A[0]` provoca **una miss** in lettura. In risposta, la CPU carica in cache `A[23..0]` e `B[7...0]`, portando la linea di cache contenente quei vettori **dallo stato** I **allo stato** E mediante **un ciclo burst di bus esterno**.

   La scrittura di `C[7..0]` provoca **8 miss in scrittura** e implica **8 cicli di bus esterni**.

   La lettura di `B[8]` provoca **una miss in lettura**. In risposta, la CPU carica in cache `B[23..8]` e `C[15..0]`, portando la linea di cache con tenente quei vettori **dallo stato** I **allo stato** E mediante **un ciclo burst di bus esterno**.

   La scrittura di `C[8]` porta la linea di cache nella quale è contenuto **dallo stato** E **allo stato** M.

   La scrittura di `C[23..16]` provoca **8 miss in scrittura** e implica **8 cicli di bus esterni**.

- Nel caso l'operazione vettoriale parta da `C[23]`, questo passo si svolge in maniera diversa. La lettura di `A[23]` provoca **una miss** in lettura. In risposta, la CPU carica in cache `A[23..0]` e `B[7...0]`, portando la linea di cache contenente quei vettori **dallo stato** I **allo stato** E mediante **un ciclo burst di bus esterno**.

  La lettura di `B[23]` provoca **una miss** in lettura. In risposta, la CPU carica in cache `B[23..8]` e `C[15...0]`, portando la linea di cache contenente quei vettori **dallo stato** I **allo stato** E mediante **un ciclo burst di bus esterno**.

  La scrittura di `C[23..16]` provoca **8 miss in scrittura** e implica **8 cicli di bus esterni**.

  La scrittura di `C[15]` porta la linea di cache in cui è contenuto **dallo stato** E **allo stato** M.

4. La trasmissione di `C` in DMA porta la linea di cache contenente `B[23..8]` e `C[15..0]` **dallo stato** M **allo stato** S e causa **un ciclo di WB** che implica **un ciclo burst di bus esterno**.

   Il DMAC lavora su un byte alla volta, quindi la trasmissione di C richiede **24 cicli di bus esterni**.

Si hanno quindi, nel caso di indice crescente, 40 miss in scrittura e 2 miss in lettura su un totale di 96 accessi in memoria, per una miss rate pari a $MR = 43.75\%$.

Nel caso di indice decrescente, invece, si hanno 32 miss in scrittura e due miss in lettura su un totale di 96 accessi in memoria, per una miss rate pari a $MR \approx 0.33\%$.

5 2018-07-18

5.1 Specifiche

Il sistema dispone di:

- due porte seriali `S_1, S_2` gestite a interrupt;
- una porta seriale `S_3` gestita in DMA;
- un PIC;
- 4 MiB di EPROM mappata agli indirizzi alti;
- 512 KiB di RAM mappata agli indirizzi bassi;
- 8 KiB di cache a 2 vie e linee da 32B gestita in write-around.

Un segmento dati SD mappato a `0x0003 0000` contiene tre vettori A, B, C da 24 byte ciascuno.

Il sistema esegue i seguenti passi:

```
Per sempre:
  Ricezione di A da S_1 a interrupt
  Ricezione di B da S_2 a interrupt
  C = A XOR B
  Trasmissione di C su S_3 in DMA
```

5.2 Mappa della memoria

- `S_1` è mappata (per scelta progettuale) da `0x4000` a `0x4007`;
- `S_2` è mappata (per scelta progettuale) da `0x5000` a `0x5007`;
- `S_3` è mappata (per scelta progettuale) da `0x6000` a `0x6007`;
- il DMAC è mappato (per scelta progettuale) da `0x7000` a `0x700F`;
- il PIC è mappato (per scelta progettuale) da `0x8000` a `0x8001`;
- I latch HL e HH che consentono al DMAC di emettere 32 bit di indirizzo sono mappati (per scelta progettuale) a `0x9000` e `0xA000`.
- il DMAC è mappato (per scelta progettuale) da `0xB000` a `0xB00F`;
- il vettore A è mappato (come da consegna) da `0x0003 0000` a `0x0003 0017`;
- il vettore B è mappato (come da consegna) da `0x0003 0018` a `0x0003 002F`;

- il vettore C è mappato (come da consegna) da `0x0003 0030` a `0x0003 0047`;

5.3 Interfacciamento UART

Si considerino tutti i segnali come attivi alti, per chiarezza.

Il pin CS di `S_1` è collegato a `CS = CS_S_1`.

Il pin CS di `S_2` è collegato a `CS = CS_S_2`.

Il pin CS di `S_3` è collegato a `CS = !HOLDA * CS_S_3 + HOLDA * DACK1`.

Il pin `SRQ` di `S_3` è collegato al pin `DRQ0` del DMAC.

Il pin `SRQ` di `S_1` è collegato al pin `IR0` del PIC.

Il pin `SRQ` di `S_2` è collegato al pin `IR1` del PIC.

5.4 Interfacciamento PIC

Si considerino tutti i segnali, compresi i `bank enable`, come attivi alti, per chiarezza.

Il pin `INT` del PIC è collegato al pin `INT` della CPU.

Il pin `INTA` del PIC è collegato al pin `INTA` della CPU.

Il pin `CS` del PIC è collegato a `CS_PIC = BA15`.

Il pin `SRQ` di `S_1` è collegato al pin `IR0` del PIC.

Il pin `SRQ` di `S_2` è collegato al pin `IR1` del PIC.

I pin `WR, RD` del PIC sono collegati ai pin `IOWR, IORD` della CPU.

I pin `D[7..0]` del PIC sono collegati ai pin `IOB[7..0]` del bus di I/O.

Il pin `A0` del PIC è collegato a `BA0 == BE0`.

5.5 DMAC

Il DMAC è programmato come segue prima degli invii di `C`:

1. Maschera il canale 0
2. Reset FF interni
3. `BAR0 = 0x0030`
4. `BCR0 = 0x0017`
5. `MR = 01 0 0 10 00` (Single mode, autoincr, no autoinit, read, ch00)
6. `HH = 0x00`
7. `HL = 0x03`
8. Smaschera il canale 0

5.6 Cache

5.6.1 Indici

La cache ha dimensione 8 KiB, linee da 32B e 2 vie. Questo significa che vi sono $S = \frac{8K}{32 \cdot 2} = 128$ set. Si hanno quindi 7 bit di set ID, 5 bit di offset, e i restanti 20 bit vengono usati per i tag.

Il vettore A è mappato da `0x0003 0000` a `0x0003 0017`, indici che in base 2 valgono

```
   3    0    0    0   0
0011 0000|0000000|00000
     0x30    0x00  0x00
```

```
   3    0    0    1   7
0011 0000|0000000|10111
     0x30    0x00  0x17
```

A occuperà quindi gli offset `0x00-0x17` del set `0x00` con tag `0x30`.

Il vettore B è mappato da `0x0003 0018` a `0x0003 002F`, indici che in base 2 valgono

```
   3    0    0    1   8
0011 0000|0000000|11000
     0x30    0x00  0x18

   3    0    0    2   F
0011 0000|0000001|01111
     0x30    0x01  0x0F
```

B occuperà quindi gli offset `0x18-0x1F` del set `0x00` e gli offset `0x00-0x0F` del set `0x01` con tag `0x040`.

Il vettore C è mappato da `0x0003 0030` a `0x0003 0047`, indici che in base 2 valgono

```
   3    0    0    3   0
0011 0000|0000001|10000
     0x30    0x01  0x10

   3    0    0    4   7
0011 0000|0000010|00111
     0x30    0x02  0x07
```

C occuperà quindi gli offset `0x10-0x1F` del set `0x01` e gli offset `0x00-0x07` del set `0x02` con tag `0x30`.

5.6.2 Dinamica della cache

1. La ricezione di `A` a interrupt provoca **24 miss** in scrittura e richiede **24 cicli di bus esterni**.

   La ricezione di `B` a interrupt provoca **24 miss** in scrittura e richiede **24 cicli di bus esterni**.

2. La lettura di `A[0]` provoca **una miss** in lettura. In risposta, la CPU carica in cache `A[23..0]` e `B[7..0]`, portando la linea di cache corrispondente **dallo stato** I **allo stato** E mediante **un ciclo burst di bus esterno**.

   La scrittura di `C[7..0]` provoca **8 miss** in scrittura e richiede **8 cicli di bus esterni**.

   La lettura di `B[8]` provoca **una miss** in lettura. In risposta, la CPU carica in cache `B[23..8]` e `C[15..0]`, portando la linea di cache corrispondente **dallo stato** I **allo stato** E mediante **un ciclo burst di bus esterno**.

   La scrittura di `C[8]` porta la linea di cache corrispondente **dallo stato** E **allo stato** M.

   La scrittura di `C[23..16]` provoca **8 miss in scrittura** e richiede **8 cicli di bus esterni**.

   - Se il calcolo di C avviene iniziando da `C[23]`, questo passo avviene in modo diverso.

     La lettura di `A[23]` provoca **una miss** in lettura. In risposta, la CPU carica in cache `A[23..0]` e `B[7..0]`, portando la linea di cache corrispondent e **dallo stato** I **allo stato** E mediante **un ciclo burst di bus esterno**.

     La lettura di `B[23]` provoca **una miss** in lettura. In risposta, la CPU carica in cache `B[23..8]` e `C[15..0]`, portando la linea di cache corrispondente **dallo stato** I **allo stato** E mediante **un ciclo burst di bus esterno**.

     La scrittura di `C[15]` porta la linea di cache corrispondente **dallo stato** E **allo stato** M.

     La scrittura di `C[23..16]` provoca **8 miss in scrittura** e richiede **8 cicli di bus esterni**.

3. La trasmissione di C in DMA porta la linea di cache contenente `B[23..8], C[15..0]` **dallo stato** M **allo stato** S, richiede **24 cicli di bus esterni** e implica **un ciclo di implicit WB**.

Si hanno 64 miss in scrittura e 2 miss in lettura su 120 accessi alla memoria, per una miss rate complessiva pari a $MR = 55\%$.

Nel caso in cui il calcolo di C avvenga in ordine inverso, si hanno 56 miss in scrittura e 2 miss in lettura su 120 accessi alla memoria, per una miss rate complessiva pari a $MR \approx 46.7\%$.

Disclaimer: Questo documento può contenere errori e imprecisioni che potrebbero danneggiare sistemi informatici, terminare relazioni e rapporti di lavoro, liberare le vesciche dei gatti sulla moquette e causare un conflitto termonucleare globale. Procedere con cautela.