

Interfacce di microprocessore

Lezione 10 di Sistemi dedicati

Docente: Giuseppe Scollo

Università di Catania
Dipartimento di Matematica e Informatica
Corso di Laurea Magistrale in Informatica, AA 2016-17

Indice

1. Interfacce di microprocessore
2. argomenti della lezione
3. registro mappato in memoria
4. integrazione nella gerarchia di memoria
5. mailbox
6. coda FIFO
7. memoria condivisa
8. interfacce di coprocessore
9. interfacce di istruzioni custom
10. flusso di progetto di ASIP
11. esempio: interfaccia di istruzione custom Nios-II
12. banchi di registri per istruzioni custom Nios-II
13. riferimenti

di che si tratta:

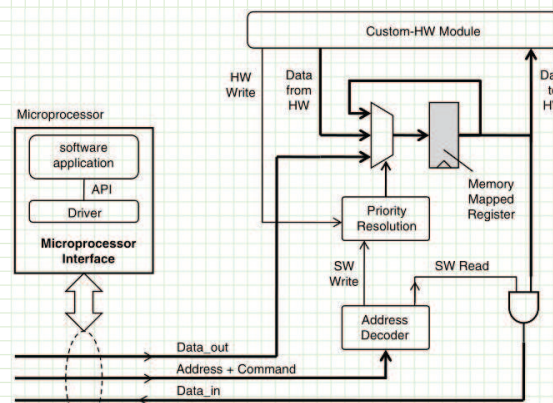
- interfacce mappate in memoria
 - registro mappato in memoria
 - mailbox
 - code FIFO
 - protocolli di handshake
 - memoria condivisa
- interfacce di coprocessore
- flusso di progettazione di ASIP
- interfacce di istruzioni custom

esempio: l'interfaccia di istruzioni custom Nios-II

registro mappato in memoria

le interfacce mappate in memoria sono il tipo più generale di interfaccia HW/SW
nei programmi il supporto è dato dall'uso di puntatori, e.g.:

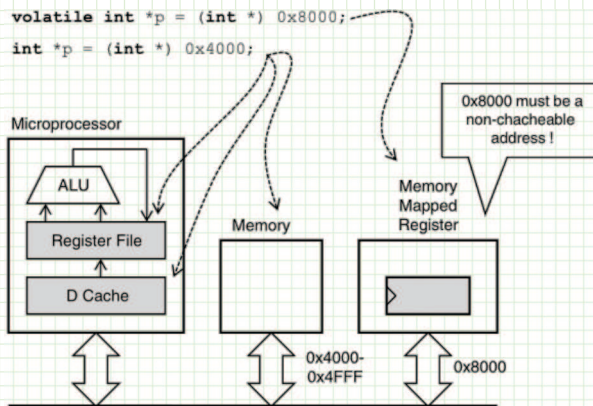
```
volatile unsigned int *MMRegister = (unsigned int *) 0x8000;
// write the value '0xFF' into the register
*MMRegister = 0xFF;
// read the register
int value = *MMRegister;
```



Schaumont, Figure 11.1 - A memory-mapped register

perché il puntatore deve essere volatile?

- in generale, il valore di un `int *` può trovarsi in tre tipi distinti di locazione nella gerarchia di memoria: memoria principale, memoria cache, registro del processore
- qualificare il puntatore come volatile fa sì che il compilatore eviti di collocare fra i registri del processore il registro mappato in memoria



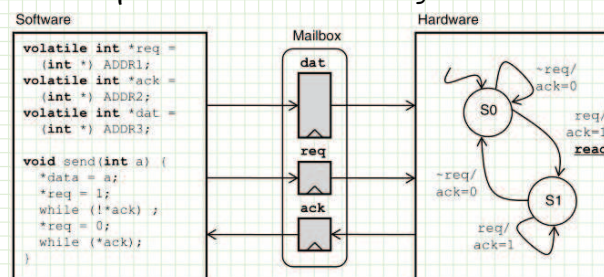
Schaumont, Figure 11.2 - Integrating a memory-mapped register in a memory hierarchy

tuttavia, definire un registro mappato in memoria con un puntatore volatile non impedisce che l'indirizzo di memoria sia mantenuto nella cache! due approcci per far fronte a questo problema:

- allocazione in area di memoria esclusa dalla cache, se il processore (e.g. Microblaze) ha una cache configurabile
- uso di apposite istruzioni del processore (e.g. Nios-II) che evitano la cache

mailbox

semplice estensione del registro mappato in memoria con un meccanismo di handshake, con cui le parti comunicanti si segnalano reciprocamente lo stato del registro



Schaumont, Figure 11.3 - A mailbox register between hardware and software

il protocollo mostrato in figura ha due punti di sincronizzazione, subito dopo che `req` e `ack` assumono lo stesso valore

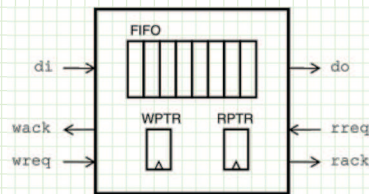
dovrebbe essere abbastanza semplice raddoppiare il throughput di questo protocollo... come?

i due svantaggi principali di questo protocollo:

- accoppiamento stretto, per l'alternanza delle operazioni di scrittura/lettura
- elevato sovraccarico di trasferimenti sul bus, per controllare lo stato della mailbox

coda FIFO

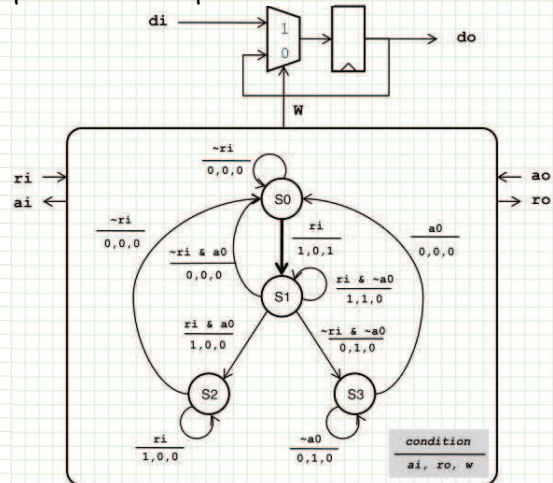
l'uso di una coda FIFO compensa squilibri temporanei fra il throughput di scrittura e quello di lettura



```
dp fifo(in di : ns(8);
      in wreq : ns(1);
      out wack : ns(1);
      out do : ns(8);
      in rreq : ns(1);
      out rack : ns(1)) {
  sig read, write : ns(1);
  reg rp, wp : ns(3);
  use dualport_mem(di, wp, write, // write port
                  do, rp, read); // read port
  always (
    read = (rreq & (wp != rp)) ? 1 : 0;
    write = (wreq & ((wp + 1) != rp)) ? 1 : 0;
    wp = write ? wp + 1 : wp;
    rp = read ? rp + 1 : rp;
    wack = write;
    rack = read;
  )
}
```

Schaumont, Figure 11.4 - A FIFO with handshakes on the read and write ports

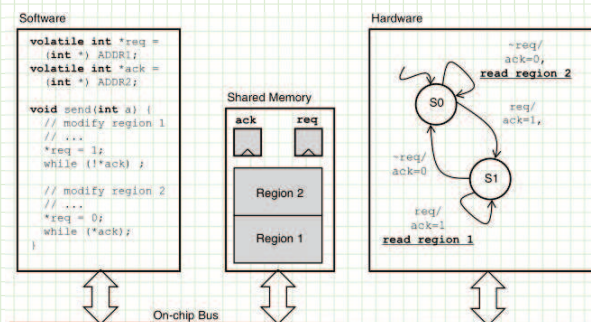
si può costruire una FIFO a stadi in cascata, dove ciascuno stadio ha l'input da slave e l'output da master



Schaumont, Figure 11.5 - A one-place FIFO with a slave input handshake and a master output handshake

memoria condivisa

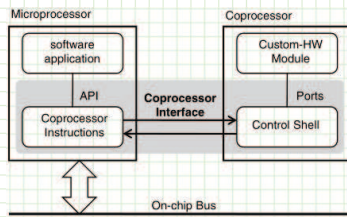
invece di controllare l'accesso a un solo registro, un handshake può essere usato anche per il controllo di accesso a una regione di memoria



Schaumont, Figure 11.6 - A double-buffered shared memory with a memory-mapped request/acknowledge handshake

in una fase del protocollo in figura, è permessa la modifica della regione 1 della memoria, mentre nell'altra fase è permessa la modifica della regione 2

questo schema realizza il pipelining delle operazioni di lettura/scrittura, il che, per applicazioni che alternano riorganizzazione dei dati a stadi di elaborazione, può produrre miglioramenti sostanziali della prestazione



Schaumont, Figure 11.7 - Coprocessor interface

interfacce di coprocessore

quando occorre un throughput di dati elevato tra il software e l'hardware custom, un'interfaccia dedicata supera le prestazioni di quelle mappate in memoria

un'interfaccia di coprocessore non usa il bus su chip bensì una porta dedicata sul processore, pilotata da istruzioni di coprocessore

sia l'insieme di istruzioni di coprocessore che la specifica interfaccia di coprocessore dipendono dal tipo di processore; non tutti i processori hanno un'interfaccia di coprocessore
la decisione di usare una specifica interfaccia di coprocessore lega il modulo hardware custom a un particolare processore, dunque limita la riusabilità del modulo ai sistemi che usano lo stesso processore

vantaggi principali di un'interfaccia di coprocessore rispetto al bus su chip:

- *throughput più alto*: perché non vincolato dalla larghezza del bus su chip, né dal meccanismo di trasferimento load/store
- *latenza fissa*: un bus di coprocessore è una connessione dedicata punto-punto con temporizzazione stabile e predicibile

interfacce di istruzioni custom

l'integrazione di hardware e software può essere considerevolmente accelerata come segue:

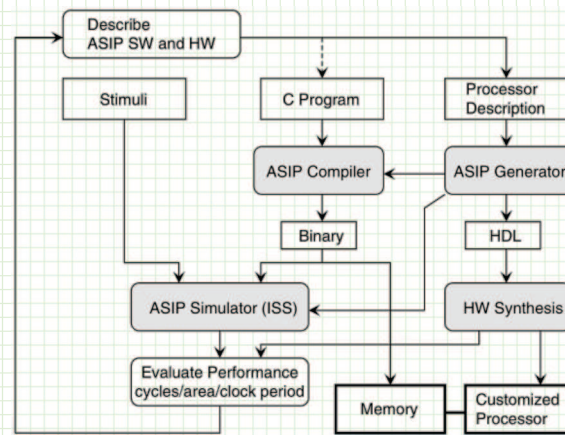
1. riservare una parte dei codici operativi di un microprocessore per nuove istruzioni
2. integrare i moduli hardware custom direttamente nella microarchitettura del microprocessore
3. controllare i moduli hardware custom usando nuove istruzioni derivate dai codici operativi riservati

il risultato di un tale progetto è un *Application-Specific Instruction-set Processor (ASIP)* mentre l'insieme di istruzioni di un coprocessore è parte del microprocessore, il progetto di quello di un ASIP è definito dall'applicazione

il progetto di ASIP automatizza gli aspetti più difficili del codesign HW/SW:

- il meccanismo di prelievo e smistamento delle istruzioni nel microprocessore assicura che hardware custom e software restino sincronizzati
- il progetto di ASIP procede in modo incrementale, il che evita di dover fare modifiche drastiche all'architettura di sistema nell'esplorazione di opzioni diverse

flusso di progetto di ASIP

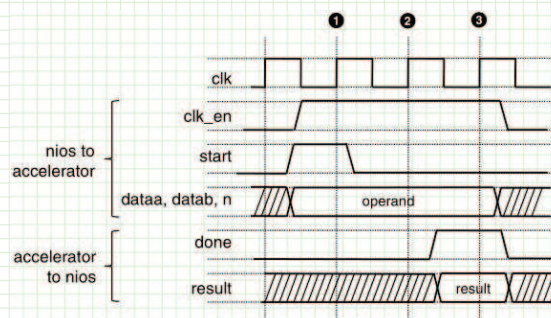


Schaumont, Figure 11.12 - ASIP design flow

il progetto di un ASIP sequenziale generalmente non produce prestazioni migliori del progetto di SOC basato su moduli hardware custom, tuttavia ha minor propensione all'errore
progressi significativi sono stati fatti nell'automazione del flusso di progetto di ASIP: per tutti i blocchi ombreggiati in figura sono reperibili strumenti commerciali

esempio: interfaccia di istruzione custom Nios-II

il processore softcore Nios-II ha un'interfaccia di coprocessore su cui si possono definire istruzioni custom e collegare moduli hardware



Schaumont, Figure 11.15 - Nios-II custom-instruction interface timing

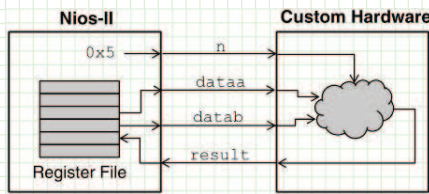
l'interfaccia supporta l'esecuzione a durata variabile di istruzioni custom mediante doppio handshake
l'input clk_en si usa per disabilitare l'hardware custom quando l'istruzione è inattiva

nel software si esegue l'istruzione custom mediante l'istruzione custom, e.g.:

custom 0x5, r2, r3, r5

assegna il valore 0x5 a n e associa le porte dataa, datab, result ai registri r2, r3, r5, nell'ordine;
uso di n: moltiplicazione di diverse istruzioni custom nel modulo hardware

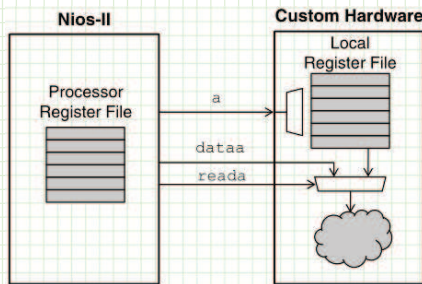
è supportato anche l'uso di un banco di registri locale nel modulo hardware custom



Schaumont, Figure 11.16a - Nios-II custom-instruction integration with processor register file

un'istruzione custom può avere operandi nei due banchi di registri: registri con prefisso r sono collocati nel processore, mentre registri con prefisso c sono collocati nel modulo hardware

è permesso l'uso di entrambi in una stessa istruzione, e.g. custom 0x5, c2, c3, r5



Schaumont, Figure 11.16b - Nios-II custom-instruction integration with local register file

la figura 11.16b illustra solo il caso del primo operando: il segnale di controllo reada seleziona il banco di registri, del processore o locale

- nel primo caso, l'operando è fornito attraverso la porta dataa, associata a un registro del processore
- nel secondo caso, l'input a seleziona il registro locale da usare quale operando

riferimenti

letture raccomandate:

Schaumont (2012) Cap. 11, Sez. 11.1.1-11.1.5, 11.2.0, 11.3.0-11.3.1, 11.3.3

per ulteriore consultazione:

Schaumont (2012) Cap. 11, Sez. 11.1.6, 11.2.1-11.2.2, 11.3.2, 11.3.4