Microprogrammazione: architetture e controllo

Lezione 06 di Sistemi dedicati

Docente: Giuseppe Scollo

Università di Catania
Dipartimento di Matematica e Informatica
Corso di Laurea Magistrale in Informatica, AA 2019-20

Copyleft © 2020 Giuseppe Scollo

Indice

Copyleft © 2020 Giuseppe Scollo

argomenti della lezione

di che si tratta:

- limitazioni delle FSM
- microprogrammazione: origini, interpreti microprogrammati
- controllo microprogrammato: architettura, vantaggi
- codifica delle microistruzioni
- datapath microprogrammato
- stesura di microprogrammi
- esempi:
  - architettura microprogrammata
  - codifica delle microstruzioni
  - microprogramma per il calcolo del GCD

Copyleft © 2020 Giuseppe Scollo

limitazioni delle FSM

i modelli FSM catturano bene gli aspetti di flusso del controllo e decisionali degli algoritmi, tuttavia mancano di *gerarchia*; ciò è fonte di limitazioni severe quando si tratta di sistemi di controllo complessi

esplosione degli stati

: la dimensione dello spazio degli stati di una *FSM prodotto* è il prodotto delle dimensioni degli spazi degli stati delle FSM componenti; peggio ancora, se queste hanno condizioni di transizione indipendenti, la numerosità delle condizioni nella FSM prodotto cresce esponenzialmente

gestione delle eccezioni

: le eccezioni sono condizioni che richiedono transizione a uno stato di gestione delle eccezioni qualunque sia lo stato corrente della macchina; l'aggiunta di transizioni per la gestione delle eccezioni a un dato diagramma di transizioni spesso offusca il corso principale del controllo

flessibilità a tempo di esecuzione

: il flusso del controllo cablato definito da una FSM può essere modificato solo rimpiazzando l'implementazione della FSM con un'altra; la motivazione per una maggiore flessibilità del modello è collegata a quella per una maggiore flessibilità dell'hardware

Copyleft © 2020 Giuseppe Scollo

l'idea della microprogrammazione

un controllo più flessibile si realizza con la sua *microprogrammazione*

la schedule fissa della FSM è rimpiazzata da quella, variabile, determinata da un *microprogramma*, costituito da *microistruzioni*, ciascuna delle quali è tradotta in segnali di controllo del datapath

la prima idea di microprogrammazione fu proposta da Maurice Wilkes, nel 1951, ma trovò ampia applicazione a partire dagli anni '60, per divenire dominante nel decennio successivo con la diffusione delle architetture CISC (*Complex Instruction-Set Computer*)

- in questa idea, il microprogramma è un *interprete*, residente in una piccola memoria di controllo
- il microinterprete preleva dalla memoria principale istruzioni macchina (qui viste come istruzioni di livello più alto) ed esegue ciascuna di esse mediante un *microroutine*, cioè una sequenza di *microistruzioni* di livello più basso

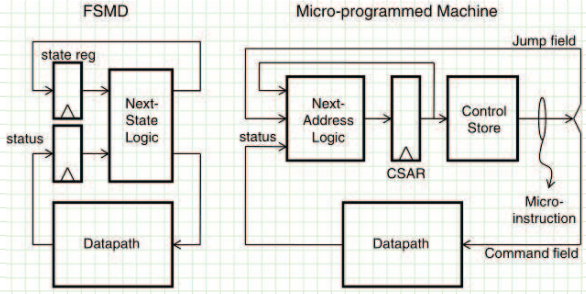
  la microarchitettura di un processore CISC acquista così l'aspetto di un "processore nel processore", con un microprogramma fisso che durante ogni ciclo macchina preleva un'istruzione ed esegue la microroutine determinata dalla decodifica del codice operativo dell'istruzione prelevata

Copyleft © 2020 Giuseppe Scollo

architettura del controllo microprogrammato

a partire dagli anni '80 le architetture RISC (*Reduced Instruction-Set Computer*) hanno conteso il campo a quelle CISC, per divenire quindi dominanti

la microprogrammazione è tuttora una tecnica molto utile a dare più flessibilità al progetto hardware



Schaumont, Figure 6.3 - In contrast to FSM-based control, microprogramming uses a flexible control scheme

CSAR (*Control Store Address Register*): l'analogo del Program Counter convenzionale

ciclo di clock determinato dal cammino critico nell'architettura microprogrammata

Copyleft © 2020 Giuseppe Scollo

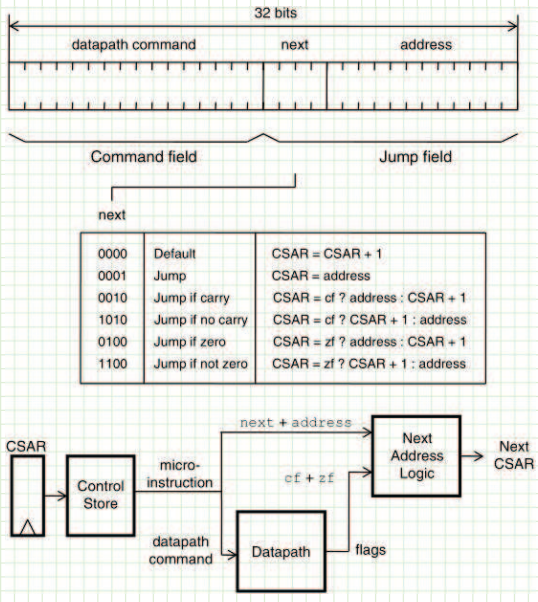
vantaggi del controllo microprogrammato

il controllo microprogrammato risolve i problemi delle FSM:

- scala facilmente con la complessità, per esempio con CSAR da 12 bit si può indirizzare una memoria di controllo da 4K istruzioni, mentre una FSM con 4K stati sarebbe estremamente complessa
- con piccole aggiunte all'architettura in figura 6.3 si può realizzare un controllo gerarchico, per esempio, aggiungendo un registro o una pila per il salvataggio e ripristino di CSAR, si possono definire microistruzioni di chiamata e rientro a/da microsottoprogramma
- le eccezioni possono essere gestite facilmente dalla logica di indirizzo, che alimenterebbe CSAR con l'indirizzo predefinito di una microroutine di gestione delle eccezioni
- è evidente il vantaggio di flessibilità hardware, per modificare il controllo del datapath basta riscrivere la memoria di controllo

Copyleft © 2020 Giuseppe Scollo

codifica di microistruzioni: campo indirizzo



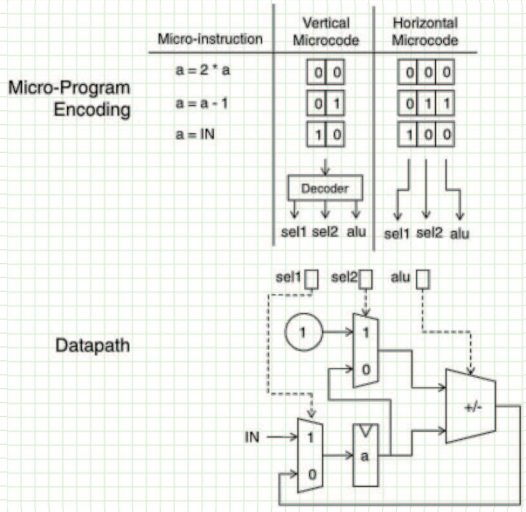
Schaumont, Figure 6.4 - Sample format for a 32-bit micro-instruction word

la definizione di formato e codifica delle microistruzioni è guidata da considerazioni di progetto; ecco un esempio di codifica

- ipotesi: micro-istruzioni da 32 bit, metà per il comando al datapath, l'altra metà per l'indirizzo della prossima micro-istruzione; cominciamo da quest'ultimo
- 12 bit per il campo indirizzo → memoria di controllo per max. 4K microistruzioni
- 4 bit per il campo next: seleziona la modalità di calcolo del prossimo indirizzo da caricare in CSAR, v. tabella in figura

Copyleft © 2020 Giuseppe Scollo

il formato in figura 6.4 non è ottimale, poiché il campo indirizzo è usato solo per istruzioni di salto: può avere altri usi con altre istruzioni

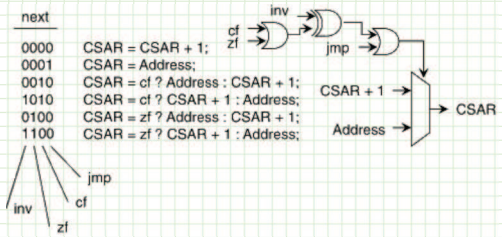


Schaumont, Figure 6.5 - Example of vertical versus horizontal micro-programming

un altro risparmio di spazio o tempo si ha con l'alternativa per il campo comando:

- *codifica orizzontale*: un bit distinto per ogni bit di controllo del datapath
- *codifica verticale*: la più breve codifica del controllo del datapath

spesso si ha una soluzione mista, *e.g.* la codifica nel campo next:



Schaumont, Figure 6.6 - CSAR encoding

Copyleft © 2020 Giuseppe Scollo

il datapath di una macchina microprogrammata consiste di tre elementi:

- unità funzionali di calcolo
- unità di memoria (banco di registri)
- gruppi di linee (bus) di comunicazione

a ciascuno di questi elementi possono corrispondere alcuni bit di controllo nella codifica della microistruzione, per esempio:

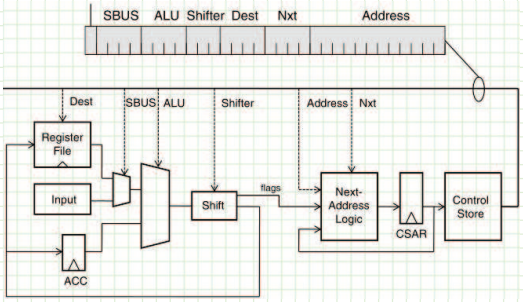
- unità di calcolo multifunzione: bit di selezione di funzione
- unità di memoria: bit di indirizzo e bit di comando lettura/scrittura
- bus di comunicazione: bit di controllo sorgente/destinazione

inoltre il datapath può generare flag di stato per il controllore microprogrammato

Copyleft © 2020 Giuseppe Scollo

ecco un esempio di controllo microprogrammato di un datapath che include: una ALU con registro a scorrimento, un banco di otto registri, un registro accumulatore e una porta di input

codifica mista orizzontale/verticale: globalmente orizzontale, poiché ogni unità nel datapath usa una porzione distinta della parola di controllo, codifica verticale dei segnali di controllo di ciascuna unità nella sua porzione



Schaumont, Figure 6.7 - A micro-programmed datapath

lo shifter genera anche bit di esito usati dal controllore microprogrammato per salti condizionati

campi della parola di controllo:

- Nxt, Address: usati dal controllore microprogrammato, gli altri campi sono usati dal datapath
- ALU: si possono codificare fino a 16 operazioni
- SBUS: selezione di un operando sorgente per la ALU, fra gli otto registri del banco e la porta di input, l'altro operando sorgente è l'accumulatore
- Dest: selezione della destinazione per l'operazione di ALU e shifter, fra i registri del banco e l'accumulatore
- Shifter: selezione della funzione di scorrimento, fino a otto funzioni

il datapath preleva ed esegue una microistruzione ogni ciclo di clock

Copyleft © 2020 Giuseppe Scollo

la tabella 6.1 presenta un esempio di codifica di microistruzioni per l'architettura vista (prima parte):

Field	Width	Encoding			
SBUS	4	Selects the operand that will drive the S-Bus			
		0000	R0	0101	R5
		0001	R1	0110	R6
		0010	R2	0111	R7
		0011	R3	1000	Input
		0100	R4	1001	Address/Constant
ALU	4	Selects the operation performed by the ALU			
		0000	ACC	0110	ACC | S-Bus
		0001	S-Bus	0111	not S-Bus
		0010	ACC + S-Bus	1000	S-Bus + 1
		0011	ACC – S-Bus	1001	ACC + 1
		0100	S-Bus – ACC	1010	0
		0101	ACC & S-Bus	1011	1

Copyleft © 2020 Giuseppe Scollo

esempio di codifica delle microistruzioni (2)

tabella 6.1 (seconda parte):

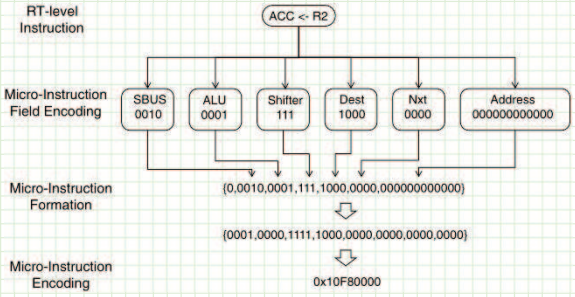
Field	Width	Encoding			
Shifter	3	Selects the function of the programmable shifter			
		000	logical SHL(ALU)	100	arith SHL(ALU)
		001	logical SHR(ALU)	101	arith SHR(ALU)
		010	rotate left ALU	111	ALU
		011	rotate right ALU		
Dest	4	Selects the target that will store S-Bus			
		0000	R0	0101	R5
		0001	R1	0110	R6
		0010	R2	0111	R7
		0011	R3	1000	ACC
		0100	R4	1111	unconnected
Nxt	4	Selects next-value for CSAR			
		0000	CSAR + 1	1010	cf ? CSAR + 1 : Address
		0001	Address	0100	zf ? Address : CSAR + 1
		0010	cf ? Address : CSAR + 1	1100	zf ? CSAR + 1 : Address

Copyleft © 2020 Giuseppe Scollo

stesura di microprogrammi

con la codifica definita in tabella 6.1, una microistruzione si forma selezionando una funzione per ciascuno dei moduli nel datapath e un indirizzo per il campo Address (con un opportuno valore di indifferenza per esso qualora Nxt sia nullo)

a mo' di esempio, vediamo come una istruzione RTL, quale ACC ← R2, si traduce in una microistruzione



Schaumont, Figure 6.8 - Forming micro-instructions from register-transfer instructions

- l'operando sorgente è in R2: SBUS = 0010
- la ALU passa l'input S-Bus all'output: ALU = 0001
- lo shifter inoltra l'output dell'ALU senza modifiche: Shifter = 111
- l'output dello shifter aggiorna l'accumulatore: Dest = 1000
- CSAR ha l'incremento di default: Nxt = 0000 e Address ha un valore d'indifferenza, e.g. nullo

Copyleft © 2020 Giuseppe Scollo

esempio di microprogramma per il calcolo del GCD

strutture di controllo complesse, quali cicli e strutture if-then-else, possono esprimersi come combinazione (o sequenza) di istruzioni RTL

- a titolo di esempio, sviluppiamo un microprogramma che calcola il GCD di due numeri in input usando l'algoritmo di Euclide

il microprogramma è scritto in una notazione RTL simbolica immediatamente traducibile in microistruzioni in modo simile all'esempio precedente

	Command Field	||	**Jump Field**
	IN → R0			
	IN → ACC			
Lcheck:	(R0 – ACC)	||	JUMP_IF_Z Ldone
	(R0 – ACC) << 1	||	JUMP_IF_C Lsmall
	(R0 – ACC) → R0	||	JUMP Lcheck
Lsmall:	ACC – R0 → ACC	||	JUMP Lcheck
Ldone:				JUMP Ldone

Schaumont, Listing 6.1 - Micro-program to evaluate a GCD

Copyleft © 2020 Giuseppe Scollo

riferimenti

letture raccomandate:

- Schaumont, Ch. 6, Sect. 6.1-6.4

per ulteriore consultazione:

- Schaumont, Ch. 6, Sect. 6.6-6.8

Copyleft © 2020 Giuseppe Scollo