Realizzazione su FPGA di un coprocessore multicore mappato in memoria

Esercitazione 12 di Sistemi dedicati

Docente: Giuseppe Scollo

Università di Catania
Dipartimento di Matematica e Informatica
Corso di Laurea Magistrale in Informatica, AA 2017-18

Indice

argomenti dell'esercitazione

in questa esercitazione si trattano:

- realizzazione su FPGA della prima idea di progetto proposta nella lezione 12
  - progetto del coprocessore multicore
  - interfaccia hardware del coprocessore
  - mappa dei registri del coprocessore
  - sistema Nios II con coprocessore e Performance Counter
  - driver software
- test e misura delle prestazioni con il Monitor Program
  - test senza lettura del registro di stato
  - test con lettura del registro di stato

flusso di lavoro del progetto

fasi principali di sviluppo:

- descrizione in VHDL del coprocessore multicore
- descrizione in VHDL del coprocessore multicore con interfaccia Avalon MM
- costruzione Qsys di un sistema Nios II con coprocessore e performance counter, mapping del sistema su FPGA e compilazione
- stesura del driver software e di script TCL per la sua generazione in HAL
- stesura dell'applicazione software per test e misura della prestazione, in due versioni:
  sequenziale: esecuzione senza lettura del registro di stato del coprocessore
  status-tested: esecuzione con lettura del registro di stato del coprocessore
- compilazione ed esecuzione dell'applicazione mediante Monitor Program, per due varianti di ciascuna versione: una con valore di default del livello di ottimizzazione, l'altra con livello O3
- salvataggio dei performance report e archiviazione del progetto

progetto del coprocessore multicore

produzione della descrizione VHDL del coprocessore multicore in due passi:

- coprocessore 1-core con input a 64 bit
- coprocessore 2^n-core con output di stato a 2^n bit

i rispettivi sorgenti delay_collatz.vhd e multicore_delay_collatz.vhd sono disponibili nella cartella vhdl dell'archivio allegato, nonché nella cartella VHDL/code/e12 dell'area riservata di laboratorio

il coprocessore 1-core è ottenuto in modo analogo a quello della precedente esercitazione, con ovvia modifica del sorgente Gezel e analoga correzione dell'output VHDL del traduttore fdlvhd, come documentato nella cartella fdlvhd_patch

il coprocessore è dotato dell'input core_select a *n* bit che codifica il core a cui smistare l'operazione di I/O, mentre gli output done dei core individuali sono esposti quale stato globale in una porta parallela di output a 2^n bit

le porte dati x0 e delay dei core individuali sono disposte su segnali interni paralleli, rispettivamente da $64*2^n$ bit e $16*2^n$ bit, da cui la decodifica di core_select seleziona la parte di interesse all'operazione di I/O

le cartelle vuote delay_collatz, mc_delay_collatz e mc_interface sono intese ospitare progetti di compilazione e simulazione dei suddetti sorgenti e di quello descritto appresso; cartelle omonime in tests forniscono rispettivi file di input per le simulazioni

interfaccia hardware del coprocessore

un'istanza del componente coprocessore multicore è incorporata nell'interfaccia Avalon memory-mapped descritta dal sorgente multicore_delay_collatz_avalon_interface.vhd e accede ai seguenti segnali del bus Avalon:

clock, resetn, read, write, chipselect, address, waitrequest, writedata, readdata

- il segnale address ha larghezza *n*+2 bit e codifica l'indirizzo di registro del coprocessore (v. prossima pagina)
- i segnali writedata, readdata hanno larghezza 32 bit ciascuno, per il trasferimento di un dato con il processore Nios II in un solo ciclo

l'acquisizione dell'input a 64 bit per il coprocessore avviene dunque in due cicli di bus, perciò l'interfaccia deve memorizzare il dato ricevuto nel primo ciclo per poi concatenarlo a quello ricevuto nel secondo ciclo; ne consegue la classica struttura a due processi della descrizione:

- uno per l'aggiornamento di un registro con il dato del primo ciclo,
- l'altro per la rete combinatoria

d'altra parte, l'output a 32 bit del dato a 16 bit prodotto da un core del coprocessore ne richiede l'estensione con zeri, realizzata dall'interfaccia

la consultazione del sorgente multicore_delay_collatz_interface.vhd mostra le relazioni tra i segnali di I/O del componente di calcolo e i segnali all'interfaccia Avalon

mappa dei registri del coprocessore

la costruzione con Qsys del sistema Nios II con il componente coprocessore, simile a quella dell'esercitazione precedente, assegna al coprocessore un indirizzo base e, a partire da questo, un'area di memoria per i suoi registri di I/O

- la memoria è indirizzabile a byte e gli indirizzi dei registri di I/O sono allineati a parole da 4 byte, tuttavia il modello di programmazione dei driver software di un componente sul bus Avalon prevede per default che il registro sia individuato da un indice di parola, detto *register offset*, che coincide con l'input address della sua interfaccia Avalon

la seguente mappa dei registri indica anche i segnali del componente coprocessore individuati dai corrispondenti offset di registro, indicizzati dal valore di core_select in parentesi, dove $k = 2^n$ è il numero di core paralleli, e con la *legenda*:

- **ro**: register offset
- **ao**: memory address offset (rispetto all'indirizzo base)

ro	**segnale**	**ao**	**ro**	**segnale**	**ao**
0	x0(0)[31..0]	0	2k	delay(0)	8k
1	x0(0)[63..32]	4		...	
	...		3k-1	delay(k-1)	12k-4
2(k-1)	x0(k-1)[31..0]	8(k-1)	3k	status	12k
2k-1	x0(k-1)[63..32]	8k-4			

sistema Nios II con coprocessore e Performance Counter

le successive fasi di sviluppo sono simili a quelle dell'esercitazione precedente:

- costruzione del componente Qsys coprocessore
- costruzione del sistema Nios II con coprocessore e Performance Counter
- mapping su FPGA e compilazione

la costruzione Qsys del sistema Nios II è più celere se effettuata come modifica del sistema Qsys dell'esercitazione precedente, dal quale si rimuove il componente delay_collatz_avalon_interface e gli si aggiunge un'istanza del nuovo componente multicore_delay_collatz_avalon_interface

- *avvertenza*: fare attenzione a salvare il sistema modificato nella directory del progetto corrente, non in quella del progetto del sistema da modificare

driver software

gli script TCL per la generazione del driver software nel BSP del progetto, forniti nella cartella codesign/ip/multicore_delay_collatz_avalon_interface dell'archivio allegato, sono simili a quelli dell'esercitazione precedente

i sorgenti C del driver software, forniti nella cartella HAL allo stesso percorso, differiscono da quelli dell'esercitazione precedente nei seguenti aspetti:

- definizione della costante MDC_N_CORES = 32, il numero di core nel coprocessore
- per il lancio del calcolo di un core su una traiettoria di dato inizio si fornisce la funzione mdc_start che effettua due operazioni di scrittura sul bus (per la lunghezza di parola doppia del dato d'inizio), invece della macro che ne effettua una sola nel caso precedente
- in aggiunta alla funzione delay, di lettura del risultato calcolato da un dato core, si fornisce la funzione status, di lettura del registro di stato del coprocessore

programmi di test e misura delle prestazioni

i programmi di test e misura delle prestazioni forniti nelle cartelle codesign/amp* dell'archivio allegato calcolano il delay per 2M punti d'inizio a partire da X_BASE = 1128784494896128

N.B.: X_BASE+14 è il class record della classe 1746

in entrambe le versioni del test, il programma assegna al core j il calcolo del delay per i punti di inizio nella classe di congruenza j mod MDC_N_CORES, dunque per 2M/32 = 64K traiettorie (in media nella seconda versione); la differenza fra le versioni in codesign/amp_s* e quelle in codesign/amp_t* è la seguente:

- nel primo caso, detto *sequenziale*, si eseguono 64K iterazioni del ciclo di lettura di 32 risultati e rilancio dei core, senza lettura del registro di stato (dunque il processore resta in attesa quando richiede la lettura di un risultato non ancora disponibile)
- nel secondo caso, detto *status-tested*, il processore effettua la lettura del registro di stato e ne elabora il contenuto bit dopo bit, richiedendo la lettura dei risultati solo ai core che hanno completato il calcolo

i parametri di creazione dei progetti nel Monitor Program sono indicati nel file allegato MonitorNotes.txt

risultati delle misure di prestazioni

la compilazione, caricamento sulla FPGA ed esecuzione del programma sequential_multicore_delay_collatz_timing, nei due progetti codesign/amp_s e codesign/amp_s_o3 produce i Performance Counter Report in figura

come nella precedente esercitazione, la più marcata riduzione del tempo di esecuzione delle operazioni di lettura del delay nella seconda variante può spiegarsi con l'*inlining* della funzione nella compilazione O3

```
Terminal
--Performance Counter Report--
Total Time: 12.4375 seconds  (621876238 clock-cycles)
+-------------+-----+-----------+--------------+-----------+
| Section     |  %  | Time (sec)|  Time (clocks)|Occurrences|
+-------------+-----+-----------+--------------+-----------+
|outer_loop   |  100|   12.43742|     621871162|          1|
+-------------+-----+-----------+--------------+-----------+
|read_delays  | 37.4|    4.65568|     232783872|    2097152|
+-------------+-----+-----------+--------------+-----------+
```

```
Terminal
--Performance Counter Report--
Total Time: 11.8503 seconds  (592516551 clock-cycles)
+-------------+-----+-----------+--------------+-----------+
| Section     |  %  | Time (sec)|  Time (clocks)|Occurrences|
+-------------+-----+-----------+--------------+-----------+
|outer_loop   |  100|   11.85022|     592511041|          1|
+-------------+-----+-----------+--------------+-----------+
|read_delays  | 31.9|    3.77487|     188743680|    2097152|
+-------------+-----+-----------+--------------+-----------+
```

seguono infine i Performance Counter Report dell'esecuzione del programma statustest_multicore_delay_collatz_timing, nei due progetti codesign/amp_t e codesign/amp_t_o3

```
Terminal
--Performance Counter Report--
Total Time: 12.7676 seconds  (638378241 clock-cycles)
+-------------+-----+-----------+--------------+-----------+
| Section     |  %  | Time (sec)|  Time (clocks)|Occurrences|
+-------------+-----+-----------+--------------+-----------+
|outer_loop   |  100|   12.76748|     638373915|          1|
+-------------+-----+-----------+--------------+-----------+
|read_delays  | 37.2|    4.74522|     237260953|    2163959|
+-------------+-----+-----------+--------------+-----------+
```

```
Terminal
--Performance Counter Report--
Total Time: 11.8715 seconds  (593573152 clock-cycles)
+-------------+-----+-----------+--------------+-----------+
| Section     |  %  | Time (sec)|  Time (clocks)|Occurrences|
+-------------+-----+-----------+--------------+-----------+
|outer_loop   |  100|   11.87138|     593568840|          1|
+-------------+-----+-----------+--------------+-----------+
|read_delays  | 32.4|    3.84634|     192317173|    2164575|
+-------------+-----+-----------+--------------+-----------+
```


riferimenti

materiali utili per l'esperienza di laboratorio proposta:

- archivio con file sorgenti per la riproduzione del progetto
- documenti Intel Corp. indicati nell'esercitazione precedente
- Zwolinski, Ch. 4, Sect. 4.2.3, 4.5.2