Sviluppo di SoC su FPGA con *profiling* dell'applicazione

Esercitazione 10 di Sistemi dedicati

Docente: Giuseppe Scollo

Università di Catania
Dipartimento di Matematica e Informatica
Corso di Laurea Magistrale in Informatica, AA 2018-19

Indice

argomenti dell'esercitazione

in questa esercitazione si trattano:

- integrazione di componenti hardware per lo sviluppo di SoC mediante Qsys
- strumenti per il *profiling* di applicazioni software
- progetto di un sistema Nios II dotato di un *performance counter*
- uso dell'API del *performance counter* in un esempio ben noto: calcolo del delay su una sequenza di traiettorie di Collatz, con input di utente della lunghezza della sequenza
- integrazione HW/SW mediante generazione del BSP nel Monitor Program, compilazione, caricamento su FPGA, debugging ed esecuzione
- esperienza di laboratorio:
  progetto e realizzazione di un sistema HW/SW con struttura e caratteristiche simili a quelle dell'esempio mostrato in questa esercitazione, adoperando gli stessi strumenti di sviluppo e *profiling*, ma per un'applicazione diversa

integrazione di sistema nello sviluppo di SoC

lo sviluppo di un SoC con applicazioni è una tipica attività di codesign HW/SW

consiste di progetto e sviluppo di componenti di entrambi i tipi e la loro *integrazione* a formare un sistema

lo strumento di Quartus usato in questa esercitazione per l'integrazione di componenti hardware nello sviluppo di SoC è *Qsys*

si consiglia di consultare l'introduzione a Qsys e di eseguire sulla DE1-SoC l'esempio ivi mostrato, per acquisire familiarità con l'uso dello strumento

un esempio un po' più complesso è oggetto di questa esercitazione:

- sviluppo di un SoC simile a quello indicato ma dotato di un componente per il *profiling* accurato di applicazioni software (v. appresso)
- sviluppo di una realizzazione software del calcolo del delay di traiettorie di Collatz, e di un'applicazione software per la misura del suo tempo di esecuzione

strumenti per il *profiling* di un'applicazione software

profiling di un programma: misura del tempo speso in diverse parti del programma, per identificarne quelle critiche per la velocità di esecuzione

utile nel codesign HW/SW per valutare quali parti di un programma siano meritevoli di eventuale accelerazione hardware, e stimare quindi il guadagno di prestazione conseguibile

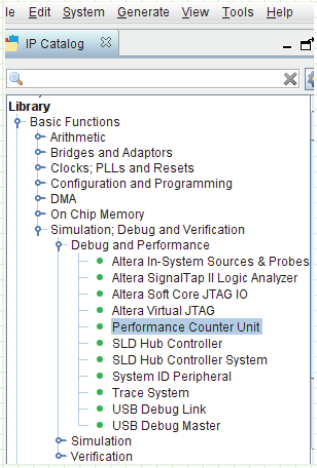
tre strumenti considerati nel documento (datato) *Profiling Nios II Systems*:

GNU gprof: misura software, alto sovraccarico di risorse software, alta distorsione della misura

Interval Timer: misura hardware, minimo sovraccarico di risorse, distorsione limitata

Performance Counter Unit: misura hardware, significativo sovraccarico di risorse hardware, distorsione minima, limite (7) sul numero di sezioni misurabili

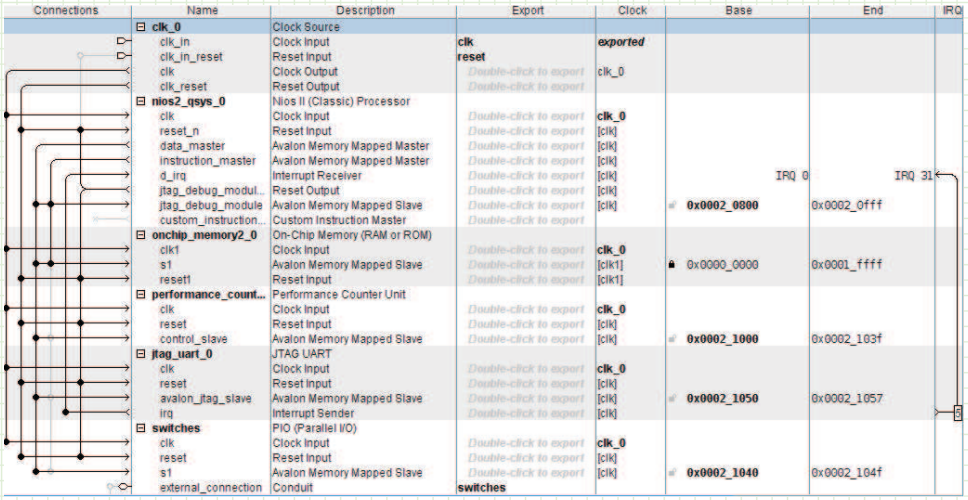
in questa esercitazione usiamo il terzo metodo, che dà la precisione migliore, l'uso più semplice nel programma, e poiché il detto limite non è un problema per l'applicazione in gioco



costruzione di un sistema Nios II con *performance counter*

la figura mostra lo schema Qsys del sistema Nios II con Performance Counter Unit

lo schema è simile a quello dell'esempio nell'introduzione a Qsys, eccetto per l'assenza della PIO LEDs e per la presenza del componente per il *profiling*



un esempio semplice e ben noto

la funzione C in figura è una realizzazione software del calcolo del delay di una traiettoria di Collatz di dato inizio

le direttive di preprocessing, tranne la quarta, fanno riferimento alla piattaforma hardware precedentemente costruita con Qsys, v. appresso

delay_collatz_timing.c

```c
#include "altera_avalon_performance_counter.h"
#include "system.h"
#define switches (volatile unsigned int *) SWITCHES_BASE // from "system.h"
#define N_FACTOR (unsigned int const) 8           // *switches scale factor
#define pca (void *) PERFORMANCE_COUNTER_0_BASE          // from "system.h"

unsigned int delay_collatz(unsigned int x0) {
  int d = 0;
  int x = x0;
  int hx;
  while (x > 1) {
    d++;
    hx = x >> 1;
    if ((x % 2) > 0) {
      d++;
      x += hx + 1;
    }
    else
      x = hx;
  }
  return d;
}
```


uso del *performance counter* nell'applicazione software

a differenza delle precedenti esperienze di laboratorio relative a realizzazioni hardware della funzione considerata, qui l'input di utente determina la lunghezza della sequenza di traiettorie da generare nel programma principale, ovvero il numero di invocazioni della funzione

all'input dagli switch si applica un fattore di scala per avere una durata ragionevole del test

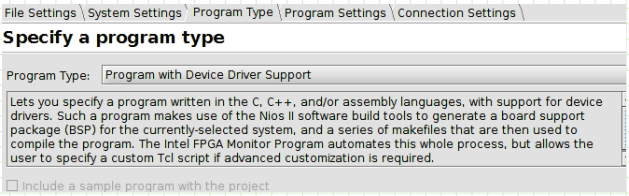
delay_collatz_timing.c

```c
int main() {
  unsigned int i, n, taps, c, x, d;
  n = *switches << N_FACTOR;
  alt_u32 cpu_freq = alt_get_cpu_freq();
  unsigned int seed = 0x1234;
  c = seed;
  PERF_RESET(pca);
  PERF_START_MEASURING(pca);
  for (i = 0; i < n; i++) {
  PERF_BEGIN(pca, 1);
    x = c;
    taps = (c & 0x1) ^ ((c & 0x4) >> 2) ^ ((c & 0x8) >> 3) ^ ((c & 0x20) >> 5);
    c = (taps << 15) | (c >> 1);
  PERF_END(pca, 1);
  PERF_BEGIN(pca, 2);
    d = delay_collatz(x);
  PERF_END(pca, 2);
  }
  PERF_STOP_MEASURING(pca);
  perf_print_formatted_report(
    pca,                            // peripheral's HW base address
    cpu_freq,                       // CPU frequency (Hz)
    2,                              // how many sections to print
    "traject_start",                // display-names of sections
    "delay_collatz");
  return 0;
}
```

generazione del BSP e integrazione HW/SW

le direttive di preprocessing mostrate prima abilitano l'uso dell'API del *performance counter* e di altri simboli (SWITCHES_BASE in questo caso) definiti nell'interfaccia software del sistema costruito con Qsys

l'interfaccia è fornita dal BSP, la cui costruzione è qui automatizzata dal *Monitor Program*, in seguito alla scelta del tipo di programma Program with Device Driver Support



ulteriori aspetti del BSP (e.g. opzioni di compilazione, di linking ecc.) possono essere specificati fornendo uno script Tcl custom

in particolare, mentre il livello di ottimizzazione di default stabilito dal Monitor Program è -O1, un livello diverso, e.g. -O3, può aversi creando uno script (con estensione .tcl) di una sola riga:

```
set_setting hal.make.bsp_cflags_optimization -O3
```

e indicandone il percorso nella input box BSP settings Tcl script (optional) delle impostazioni Program Settings

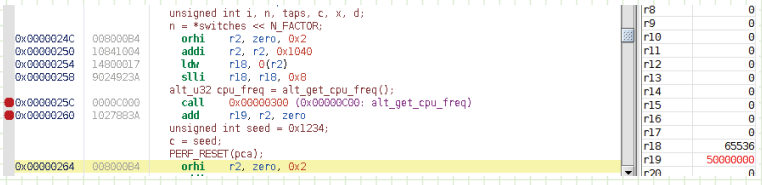
debugging ed esecuzione

il debugging nel Monitor Program può essere condotto anche al livello del sorgente C (visualizzazione dei valori delle variabili)

per questo è necessaria la compilazione con livello di ottimizzazione -O0

rimane comunque accessibile il disassembly del programma, in cui porre breakpoint ed esaminare lo stato dell'esecuzione in punti critici per la verifica della sua correttezza

per esempio, con i breakpoint in figura si può verificare la correttezza del calcolo del n. di iterazioni e della frequenza di clock, risp. in r18 e r19



dopo la rimozione di tutti i breakpoint, reset del sistema e rilancio dell'esecuzione, il modulo di profiling genera il *performance report* mostrato in figura

```
Terminal
JTAG UART link established using cable "DE-SoC [2-1.2]", device 2, instance 0x00
--Performance Counter Report--
Total Time: 7.52118 seconds  (376058945 clock-cycles)
+---------------+-----+----------+-------------+-----------+
| Section       |  %  | Time (sec)|  Time (clocks)|Occurrences|
+---------------+-----+----------+-------------+-----------+
|traject_start  | 2.53|    0.19005|      9502720|      65536|
+---------------+-----+----------+-------------+-----------+
|delay_collatz  | 96.3|    7.24331|    362165262|      65536|
+---------------+-----+----------+-------------+-----------+
```

esperienza di laboratorio

si propone di progettare e realizzare un sistema HW/SW con struttura e caratteristiche simili a quelle dell'esempio mostrato in questa esercitazione, adoperando gli stessi strumenti di sviluppo e *profiling*, ma per un'applicazione diversa; precisamente, il lavoro consiste di:

- costruzione di un sistema Nios II su FPGA, dotato di un componente per il *profiling* dell'applicazione
- sviluppo software di un programma per il test di coprimalità su una sequenza di una coppia di numeri, usando una funzione di calcolo del GCD
- integrazione HW/SW del sistema e dell'applicazione, con: *profiling* di quest'ultima mediante l'API del componente di cui sopra, generazione del BSP, compilazione, caricamento ed esecuzione su FPGA, con eventuale debugging se necessario

riferimenti

letture raccomandate:

Introduction to the Qsys System Integration Tool - *For Quartus Prime 16.1*, Intel Corp. - FPGA University Program, November 2016

letture per ulteriori approfondimenti:

Profiling Nios II Systems, AN-391-3.0, Altera Corp., July 2011

materiali utili per l'esperienza di laboratorio proposta:

Performance Counter Unit Core, Ch. 31 in: *Embedded Peripherals IP User Guide*, Intel Corp., UG-01085 | 2017.11.06

Intel FPGA Monitor Program Tutorial for Nios II - *For Quartus Prime 16.1*, Intel® FPGA University Program (November 2016)

file sorgenti per l'esperienza di laboratorio (nell'area riservata)