Esempi di FSMD in Gezel e in VHDL

Esercitazione 06 di Sistemi dedicati

Docente: Giuseppe Scollo

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

Indice

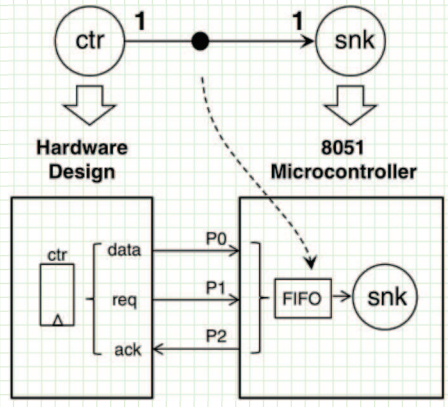
in questa esercitazione si trattano:

- codesign e cosimulazione di modelli FSMD in Gezel
- metodi di realizzazione hardware di modelli FSMD
- esempio di progetto e realizzazione hardware di FSMD
  - progetto di una datapath di calcolo di mediana
  - sequenzializzazione mediante una FSMD
- descrizione di FSM in VHDL
- esperienza di laboratorio:
  - codesign FSMD/C e cosimulazione di un testbench del calcolo del delay di traiettorie di Collatz

problema: partizionamento HW/SW della realizzazione di un sistema dataflow

un *protocollo di comunicazione* è da realizzare all'interfaccia HW/SW

ecco un esempio molto semplice:



Schaumont, Figure 3.17 – Hybrid hardware/software implementation of a dataflow graph

- protocollo di *handshake* per sincronizzare la comunicazione di token
- la descrizione hardware include l'attore `ctr`, il lato trasmittente del protocollo e modelli del microcontrollore e delle sue interfacce hardware sul bus
- il modello software include l'attore `snk`, una coda FIFO e il lato ricevente del protocollo
- l'hardware è descritto da un modello FSMD
  - in questo caso però alla FSM si dà solo il ruolo di osservare gli eventi di comunicazione di token e mostrarne il valore e la tempistica mediante una direttiva di cosimulazione

esempio di codesign FSMD/C in Gezel

```
dp send_token( out dout : ns(8);
               out req : ns(1);
               in ack : ns(1)) {
  reg ctr : ns(8);
  reg rack : ns(1);
  reg rreq : ns(1);
  always {
    rack = ack;
    rreq = rack ? 0 : 1;
    ctr = (rack & rreq) ? ctr + 1 : ctr;
    dout = ctr;
    req = rreq;
  }
  sfg transfer {
    $display($cycle, " token ", ctr);
  }
  sfg idle {}
}

fsm ctl_send_token(send_token) {
  initial s0;
  @s0 if (rreq & rack) then (transfer) -> s0;
                       else (idle) -> s0;
}
```

```
ipblock my8051 {
iptype "i8051system";
    ipparm "exec=df.ihx";
    ipparm "verbose=1";
    ipparm "period=1"; }
ipblock my8051_data(in data : ns(8)) {
    iptype "i8051systemsink";
    ipparm "core=my8051";
    ipparm "port=P0"; }
ipblock my8051_req(in data : ns(8)) {
    iptype "i8051systemsink";
    ipparm "core=my8051";
    ipparm "port=P1"; }
ipblock my8051_ack(out data : ns(8)) {
    iptype "i8051systemsource";
    ipparm "core=my8051";
    ipparm "port=P2"; }

dp sys {
  sig data, req, ack : ns(8);
  use my8051;
  use my8051_data(data);
  use my8051_req (req);
  use my8051_ack (ack);
  use send_token (data, req, ack);
}

system S { sys; }
```

```
#include <8051.h>
#include "fifo.c"

void collect(fifo_t *F) {
if (P1) { // if hardware has data
  put_fifo(F, P0); // then accept it
  P2 = 1; // indicate data was taken
  while (P1 == 1); // wait until hardware ack
    P2 = 0; // and reset
  }
}

unsigned acc;
void snk(fifo_t *F) {
  if (fifo_size(F) >= 1)
    acc += get_fifo(F);
}

void main() {
  fifo_t F1;
  init_fifo(&F1);
  put_fifo(&F1, 0); // initial token
  acc = 0;
  while (1) {
    collect(&F1);
    snk(&F1);
  }
}
```

Schaumont, Listing 3.5 - GEZEL hardware description of data flow example of Fig. 3.17

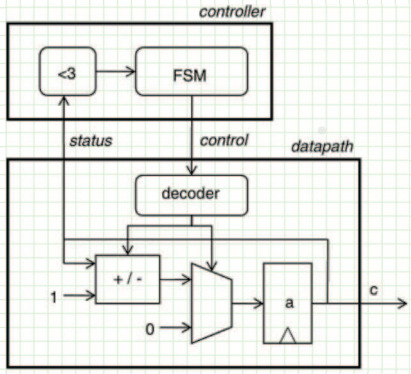
realizzazione hardware di modelli FSMD

realizzazione hardware di espressioni del datapath: stesse regole di base che in blocchi always, tuttavia:

la struttura hardware del datapath dipende anche dalla schedule di istruzioni della FSM...

perché?

vediamo l'esempio di contatore bidirezionale dalla lezione precedente:



Schaumont, Figure 5.8 - Implementation of the up-down counter FSMD

- il modello FSM implica che istruzioni diverse siano eseguite in cicli di clock diversi
- istruzioni mai concorrenti possono condividere hardware di operatori
  l'addizionatore/sottrattore in questo caso
- una decodifica locale converte la codifica delle istruzioni in segnali di controllo del datapath, per abilitare il percorso di esecuzione appropriato
- una decodifica locale nel controllore estrae informazione di stato del datapath per le condizioni nella FSM

esempio di progetto di FSMD: calcolo di mediana

specifica della funzione: calcolo della mediana in una lista di cinque numeri

problema: progettare un filtro che elabori una *stream* di dati, con un nuovo output per ogni nuovo input

filtri simili hanno applicazione nell'elaborazione delle immagini, per la riduzione del rumore

per un algoritmo rapido, senza ordinamento della lista:

in una lista di $2n+1$ elementi distinti, la mediana ha n elementi minori

```
dp median(in a1, a2, a3, a4, a5 : ns(32); out q1 : ns(32)) {
   sig z1, z2, z3, z4, z5, z6, z7, z8, z9, z10 : ns(3);
   sig s1, s2, s3, s4, s5 : ns(1);
   always {
      z1 = (a1 < a2);
      z2 = (a1 < a3);
      z3 = (a1 < a4);
      z4 = (a1 < a5);
      z5 = (a2 < a3);
      z6 = (a2 < a4);
      z7 = (a2 < a5);
      z8 = (a3 < a4);
      z9 = (a3 < a5);
      z10 = (a4 < a5);
      s1 = ((     z1 +      z2 +      z3 +      z4) == 2);
      s2 = (((1-z1) +      z5 +      z6 +      z7) == 2);
      s3 = (((1-z2) + (1-z5) +      z8 +      z9) == 2);
      s4 = (((1-z3) + (1-z6) + (1-z8) +     z10) == 2);
      q1 = s1 ? a1 : s2 ? a2 : s3 ? a3 : s4 ? a4 : a5;
   }
}
```

Schaumont, Listing 5.19 - GEZEL Datapath of a median calculation of five numbers

l'algoritmo esposto da questo datapath funziona anche quando alcuni elementi sono uguali

- la descrizione è quasi identica a quella di una funzione C per lo stesso algoritmo
- ma con una sostanziale differenza pratica:

  l'esecuzione degli assegnamenti in C è sequenziale, quella nel datapath è parallela

  dove la produzione di ogni output richiede un solo ciclo di clock!

a condizione che i dati in ingresso siano tutti simultaneamente disponibili ...

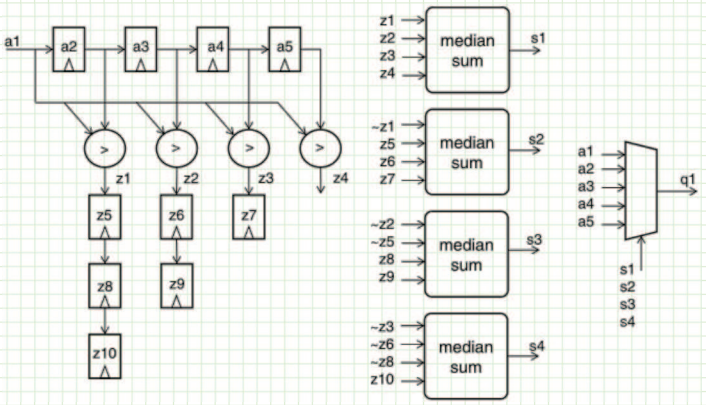
una FSMD può fare risparmiare molto spazio, come stiamo per vedere

datapath di un filtro di calcolo di mediana

l'algoritmo esposto impegna 192 linee di I/O e 10 comparatori

un filtro in streaming può ridurre l'I/O a 64 linee e impiegare solo 4 comparatori

a tal fine l'hardware usa registri per i quattro dati precedenti l'ultimo dalla stream di input e a ogni iterazione con un nuovo dato in input riusa i risultati memorizzati di sei confronti dalle tre iterazioni precedenti



Schaumont, Figure 5.9 - Median-calculation datapath for a stream of values

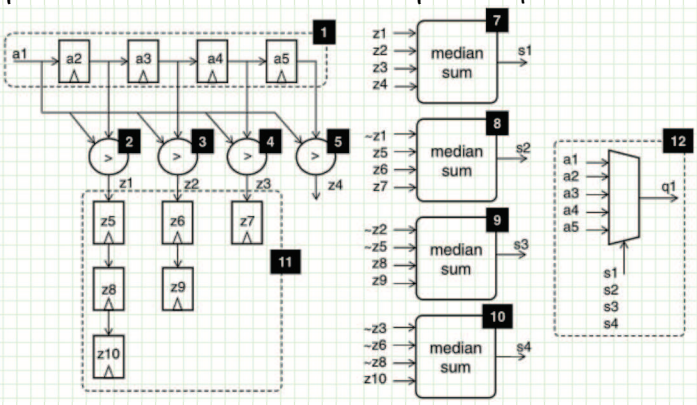
```
dp median(in a1 : ns(32); out q1 : ns(32)) {
   reg a2, a3, a4, a5 : ns(32);
   sig z1, z2, z3, z4;
   reg z5, z6, z7, z8, z9, z10 : ns(3);
   sig s1, s2, s3, s4, s5 : ns(1);
   always {
      a2 = a1;
      a3 = a2;
      a4 = a3;
      a5 = a4;
      z1 = (a1 < a2);
      z2 = (a1 < a3);
      z3 = (a1 < a4);
      z4 = (a1 < a5);
      z5 = z1;
      z6 = z2;
      z7 = z3;
      z8 = z5;
      z9 = z6;
      z10 = z8;
      s1 = ((     z1 +      z2 +      z3 +      z4) == 2);
      s2 = (((1-z1) +      z5 +      z6 +      z7) == 2);
      s3 = (((1-z2) + (1-z5) +      z8 +      z9) == 2);
      s4 = (((1-z3) + (1-z6) + (1-z8) +     z10) == 2);
      q1 = s1 ? a1 : s2 ? a2 : s3 ? a3 : s4 ? a4 : a5;
   }
}
```

sequenzializzazione mediante una FSMD

il filtro in fig. 5.9 accetta un nuovo input e produce un nuovo output a ogni ciclo di clock

si può ridurre ancora il numero dei componenti di calcolo distribuendo la computazione su più cicli di clock e imponendo una schedule sequenziale all'esecuzione delle operazioni, sì da riusare un solo comparatore e un solo modulo per il calcolo di s1, s2, s3, s4, a costo di aggiungere dei multiplatori e registri per i segnali interni

la figura illustra la schedule, la FSMD che realizza questa idea è in Schaumont Sez. 5.5.4, dove è anche presentata la FSMD di un testbench riprodotta qui accanto

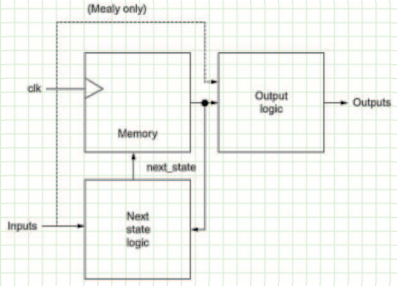


Schaumont, Figure 5.10 - Sequential schedule median-calculation datapath for a stream of values

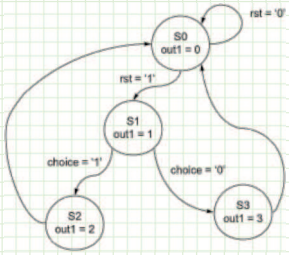
```
dp t_median {
   sig istr, ostr : ns(1);
   sig a1_in, q1 : ns(32);
   use median(istr, a1_in, ostr, q1);
   reg r : ns(1);
   reg c : ns(16);
   always { r = ostr; }
   sfg init { c = 0x1234; }
   sfg sendin { a1_in = c;
                c = (c[0] ^ c[2] ^ c[3] ^ c[5]) # c[15:1];
                istr = 1; }
   sfg noin { a1_in = 0;
              istr = 0; }
}
fsm ctl_t_median(t_median) {
   initial s0;
   state s1, s2;
   @s0 (init, noin) -> s1;
   @s1 (sendin) -> s2;
   @s2 if (r) then (noin) -> s1;
            else (noin) -> s2;
}
```


descrizione di FSM in VHDL

struttura generale hardware di una FSM ed esempio di descrizione VHDL di una FSM di Moore:



Wilson, Figure 22.1 - Hardware state machine structure



Wilson, Figure 22.2 - State transition diagram

la descrizione in VHDL dell'esempio è tratta dal cap. 22 del testo di Wilson, con numerosi emendamenti

che dire della sensitivity list del secondo processo? l'autore riferisce di un clock implicito... la simulazione gli dà ragione, però la compilazione dà un warning al riguardo

```
library ieee;
use ieee.std_logic_1164.all;
entity fsm is
   port(
      clk, rst, choice : in std_logic;
      out1 : out std_logic_vector(1 downto 0)
   );
end entity fsm;

architecture simple of fsm is
      type state_type is ( s0, s1, s2, s3 );
      signal current, next_state : state_type;
begin
   process ( clk )
   begin
      if ( clk = '1' ) then
         current <= next_state;
      end if;
   end process;
   process ( current )
   begin
      case current is
         when s0 =>
            out1 <= "00";
            if ( rst = '1') then
               next_state <= s1;
            else
               next_state <= s0;
            end if;
         when s1=>
            out1 <= "01";
            if ( choice = '0') then
               next_state <= s3;
            else
               next_state <= s2;
            end if;
         when s2=>
            out1 <= "10";
            next_state <= s0;
         when s3=>
            out1 <= "11";
            next_state <= s0;
      end case;
   end process;
end;
```

esperienza di laboratorio

obiettivo di questa esperienza è il codesign HW/SW e la cosimulazione, nella piattaforma Gezel gplatform, di un modello FSMD di calcolo del delay di traiettorie di Collatz

la funzionalità del datapath hardware è simile a quella del circuito per il delay di traiettorie di Collatz presentato nella seconda lezione ed elaborato nella seconda esperienza di laboratorio, tuttavia con alcune differenze dovute al protocollo di comunicazione con il software e a decisioni di progetto relative alla cosimulazione; in particolare:

- la configurazione hardware include un microcontrollore 8051 con porte parallele da 8 bit P0, P1, P2, P3
- la dimensione di segnali e registri di interfaccia per scambio dati è perciò dimezzata a 8 bit, come pure è dimezzata a 16 bit la dimensione di segnali e registri interni per il calcolo della traiettoria
- il protocollo di handshake per lo scambio di dati è simile a quello mostrato nell'esempio visto di codesign in FSMD/C, ma con una differenza importante: in questo caso vi sono due handshake distinti, uno per l'invio del dato di inizio della traiettoria dal software al datapath, l'altro per l'invio del risultato dal datapath al software
- il software gioca il ruolo del circuito di collaudo nel testbench sviluppato nella terza esperienza di laboratorio: pilotare il datapath con i numeri da 1 a 255 quali punti di inizio delle traiettorie
- il componente hardware è da descrivere mediante una FSMD in cui, per ogni doppio scambio di dati con il software, la FSM di controllo attraversi in sequenza quattro stati: iniziale, handshake di ricezione del dato di inizio traiettoria, calcolo del delay, handshake di invio del risultato; per tornare quindi allo stato iniziale
- mediante uso appropriato della direttiva di cosimulazione $display, va mostrato il ciclo di clock a ogni inizio e fine del calcolo della traiettoria (ingresso e uscita nel/dal terzo stato di cui sopra), assieme ai valori del dato di inizio della traiettoria e del ritardo calcolato

riferimenti

letture raccomandate:

Schaumont (2012) Cap. 3, Sez. 3.3; Cap. 5, Sez. 5.4.4-5.5

Wilson (2015) Cap. 22

letture per ulteriori approfondimenti:

Vahid (2006) Cap. 9

Zwolinski (2004) Sez. 5.5-6, 7.1-2