Status nuove schede di acquisizione FPGA

Federico Lazzari

DE0-Nano

- Al momento gli studenti hanno a disposizione le schede Terasic DE0-Nano.
- Diversi problemi noti:
  - 8 input digitali → Sciami estesi richiede fino a 12 input.
  - Input rate di pochi Hz → Mu decay spesso ha rate maggiori.
  - Risoluzione temporale 20 ns → Vita media del μ^- in presenza di nuclei di Al = 880ns.
  - Distanza tra due acquisizioni analogiche 5 μs → Bassa frequenza di campionamento per Mu life.
  - “Cronometro” con reset parziale → Letture temporali a doppio dente di sega.
  - Lettura tramite JTAG → Instabilità su lunghi periodi di presa dati.
  - Letture analogiche e digitali non in contemporanea.

DE10-Nano

- In passato sono state acquistate diverse schede Terasic DE10-Nano (almeno 1 per gruppo), ma mai messe in condizioni da poter esser usate dagli studenti.
- La DE10-Nano è notevolmente più versatile della della DE0-Nano:
  - FPGA di più recente generazione (firmware più elaborato e con clock maggiore).
  - ADC a 500 ksps.
  - CPU ARM integrata con SO Linux (comunicazione diretta con l'FPGA).
- Nei due mesi passati ho:
  - messo a punto il firmware per le acquisizioni digitali (poi utilizzato nell'esperienza Mu decay),
  - quasi completato la variante digitale + analogico.

Risorse utili

Manuale utente:
drive del laboratorio (consultabile dagli studenti)
Manuali/DE10-Nano/

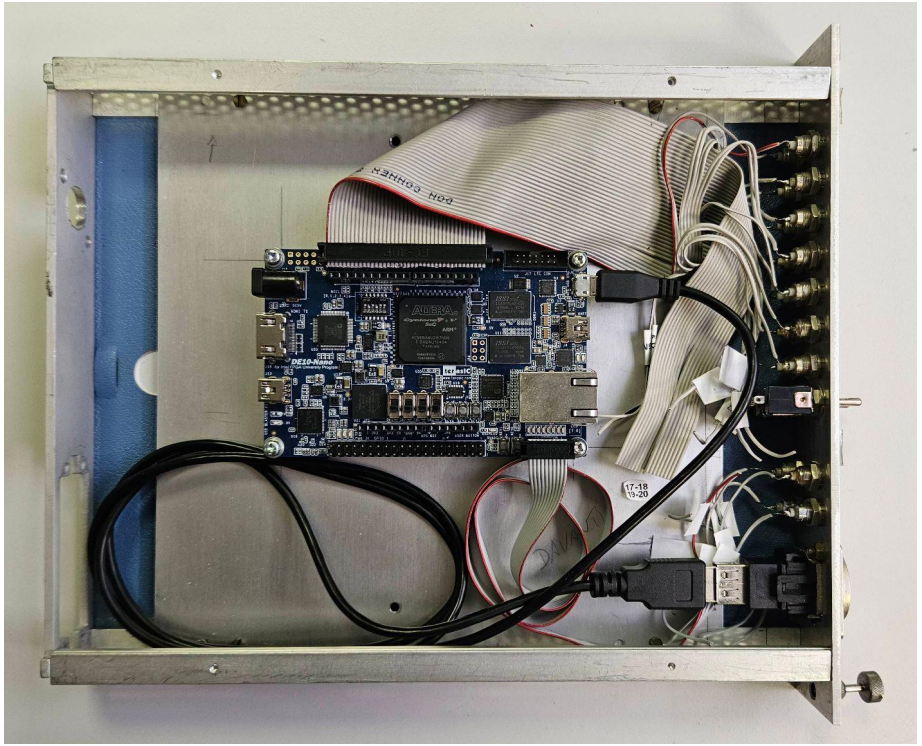
Software per acquisizione dati ed decodifica output:
https://baltig.infn.it/lazzari/de10-nano-software

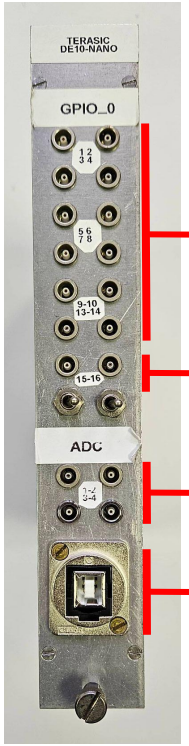
Sorgente firmware:
https://baltig.infn.it/lazzari/de10-nano-hardware

Archivio firmware e backup:
drive del laboratorio (accesso ristretto ai membri)
dispositivi/DE10-Nano/

Modulo NIM

- Prototipo di modulo NIM con DE10-Nano.





12 ingressi digitali
(terminati a 50 Ohm)

2 uscite digitali
(non terminate)

4 ingressi analogici.

Comunicazione USB con il SO

Firmware digitale

- Clock 200 MHz → risoluzione temporale 5 ns.
- 12 canali digitali in ingresso e 2 in uscita (usabili come trigger per i DRS).
- Input rate O(10kHz).
  - In caso di superamento dei limiti perdita di dati e avvisi, non blocco dell'acquisizione.
- Tempo morto sullo stesso canale 5 ns, non tempo morto su canali differenti.
- Reset cronometro ogni ~5,37 s (2^{30} * 5 ns).
  - Ogni reset riporta il cronometro a 0.
  - Viene tenuta traccia del reset nel file in output.

Firmware digitale

- Firmware di fatto completo e testato.
  - Riflessioni del segnale nei limiti dello standard TTL (terminazioni ~54 Ohm).
- Utilizzato anche dagli studenti.
  - Nessun comportamento anomalo segnalato.

- Aggiustamenti fini sul software:
  - Output dati anche su terminale.
  - File binario invece che testuale (minori scritture su scheda di memoria).
  - Conversione file output in formato comprensibile ad occhio.

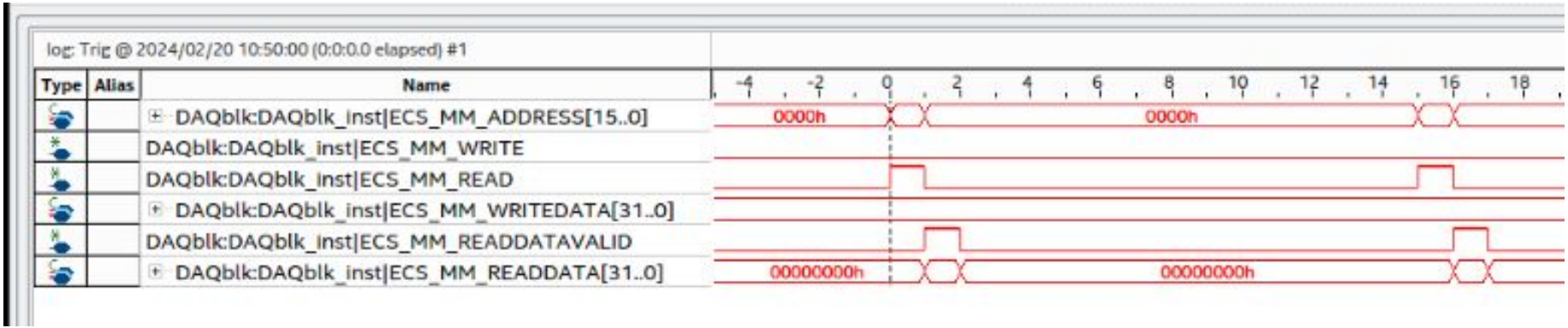
Limiti di lettura

- Potenzialmente sarebbe possibile leggere i registri ogni 300 ns (3.33 MHz).

  → input rate O(1 MHz).

- Nei fatti limitato dalla potenza della CPU integrata.

  → Meno operazioni deve compiere e meglio è.



Output files

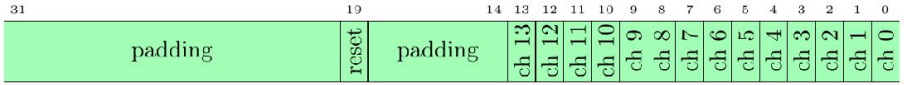


Figure 1: Channel word format.

Figure 2: Counter word format.

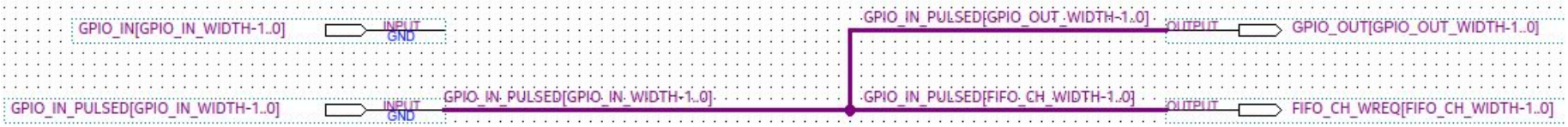
```
1       5700          -> ch 0 (2^0 = 1) clk cycle 5700
2       5701          -> ch 1 (2^1 = 2) clk cycle 5701
3       15985         -> ch 0 e 1 (2^0 + 2^1 = 3) clk cycle 15985
...
3       1073729347    -> ch 0 e 1 clk cycle 1073729347
3       1073739650       -> ch 0 e 1 clk cycle 1073739650
524288 2147483648        -> reset (2^19 = 524288) id 0 (2147483648 % 2^30 = 0)
3       8130          -> ch 0 e 1 clk cycle 8130 (+ 2^30 * (reset_id+1))
3       18433         -> ch 0 e 1 clk cycle 18433 (+ 2^30 * (reset_id+1))
...
```

Logica custom

- Il blocco “custom_logic” è un ambiente sicuro dove Implementare le funzioni logiche desiderate.
- Input: input digitali puri e ridotti a impulso.
- Output: richiesta scrittura FIFO DAQ e output digitali
- Attualmente è un bypass, ma è programmabile con il tool grafico di Quartus o direttamente in VHDL.

```
-----------------------------------------
-- CUSTOM LOGIC
-----------------------------------------

custom_logic : entity work.custom_logic
  GENERIC MAP (
    FIFO_CH_WIDTH    => FIFO_CH_WIDTH,
    GPIO_IN_WIDTH    => GPIO_0_IN_WIDTH,
    GPIO_OUT_WIDTH   => GPIO_0_OUT_WIDTH
  )
  PORT MAP (
    CLK              => CLK,
    NRESET           => nreset_reg,
    GPIO_IN          => gpio_0_in_d,
    GPIO_IN_PULSED   => gpio_0_pulsed,
    FIFO_CH_WREQ     => fifo_ch_wreq,
    GPIO_OUT         => gpio_0_out
  );
```



Firmware analogico

- Stesse funzioni del firmware digitale → acquisizione contemporanea di segnali analogici e digitali.
- Acquisizione analogica disabilitabile → non è necessario cambiare firmware.
- Digitalizzazione di un singolo canale analogico (ch 0).
- Ampiezza segnale [0, 4.095] V (DE0-Nano 2.5 V).
- Digitalizzazione triggerata da un segnale canale digitale 0, ritardo[25, 45] ns.
- 8 digitalizzazioni in sequenza: una ogni 2 us (500 ksps).
- Nell'output viene salvato (in word differenti)
  - tempo a cui è arrivato il trigger
  - le coppie valore-tempo di fine acquisizione.
- Trigger rate stimato O(1 kHz).

Firmware analogico

Valore analogico incorporato nella word dei canali:

```
1          640889595   -> segnale trigger (ch 0 al colpo di clock 640889595)
1907097600 640889996   -> valore letto 3179: (1907097600 / 2 ^18) % 2^12
1915224064 640890396   -> valore letto 3210 dopo 400 colpi di clock (2 µs)
1077936128 640890796   -> valore letto 16
1076363264 640891196   -> valore letto 10
1075576832 640891596   -> valore letto 7
1075314688 640891996   -> valore letto 6
1075314688 640892396   -> valore letto 6
1075314688 640892796   -> valore letto 6
```

Da fare:

- Ripulire un po' il codice.
- Scrivere un software d'esempio per la decodifica dell'output.
- Aggiornare il manuale.

Backup