Acquisizione Dati

Roberto Ferrari

giugno 2009

Contenuti

- digitalizzazione di segnali in fisica delle A.E.
- gestione di eventi scorrelati, bufferizzazione
- rappresentazioni di numeri in base 2
- esempi di convertitori analogico-digitali
- elementi di elettronica digitale

Digitalizzazione di segnali

- segnali impulsati e segnali "quasi stazionari"
- vantaggi e svantaggi della digitalizzazione
- campionamento / integrazione
- tempi (T), tensioni (V), cariche (Q)

  → TDC, PADC, QADC
- live time

In fisica delle alte energie …

Misure di segnali impulsati con:

- tempi << 1 ms
- scorrelati
- alto rate
- rapporto segnale/rumore variabile (rivelatore)
- basso rapporto segnale/fondo (fisica)

Tempo Vivo …

f = frequenza di arrivo degli eventi

τ = tempo necessario ad acquisire un evento

Quanto vale l'efficienza di presa dati ?

$$\left[\int_{exp} £dt = \varepsilon \cdot \int_{beam} £dt \right]$$

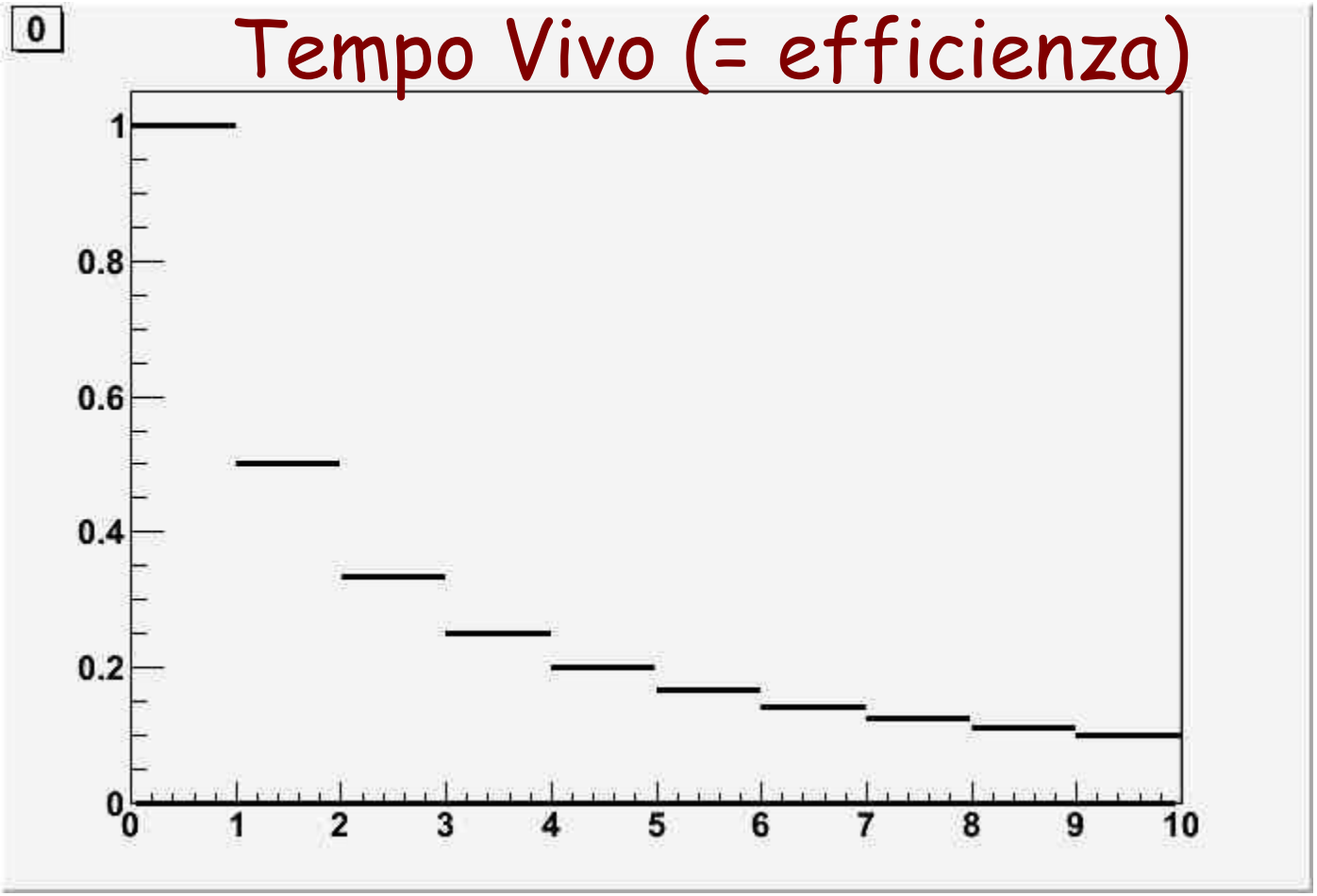
Segnali correlati (sincroni):

se $\tau < 1/f \rightarrow \varepsilon = 100\%$

se $2/f > \tau > 1/f \rightarrow \varepsilon = 50\%$

...

Eventi sincroni



$x = f \cdot \tau$

Eventi scorrelati

Rate di acquisizione = ν

→ % tempo morto = $\nu \cdot \tau$

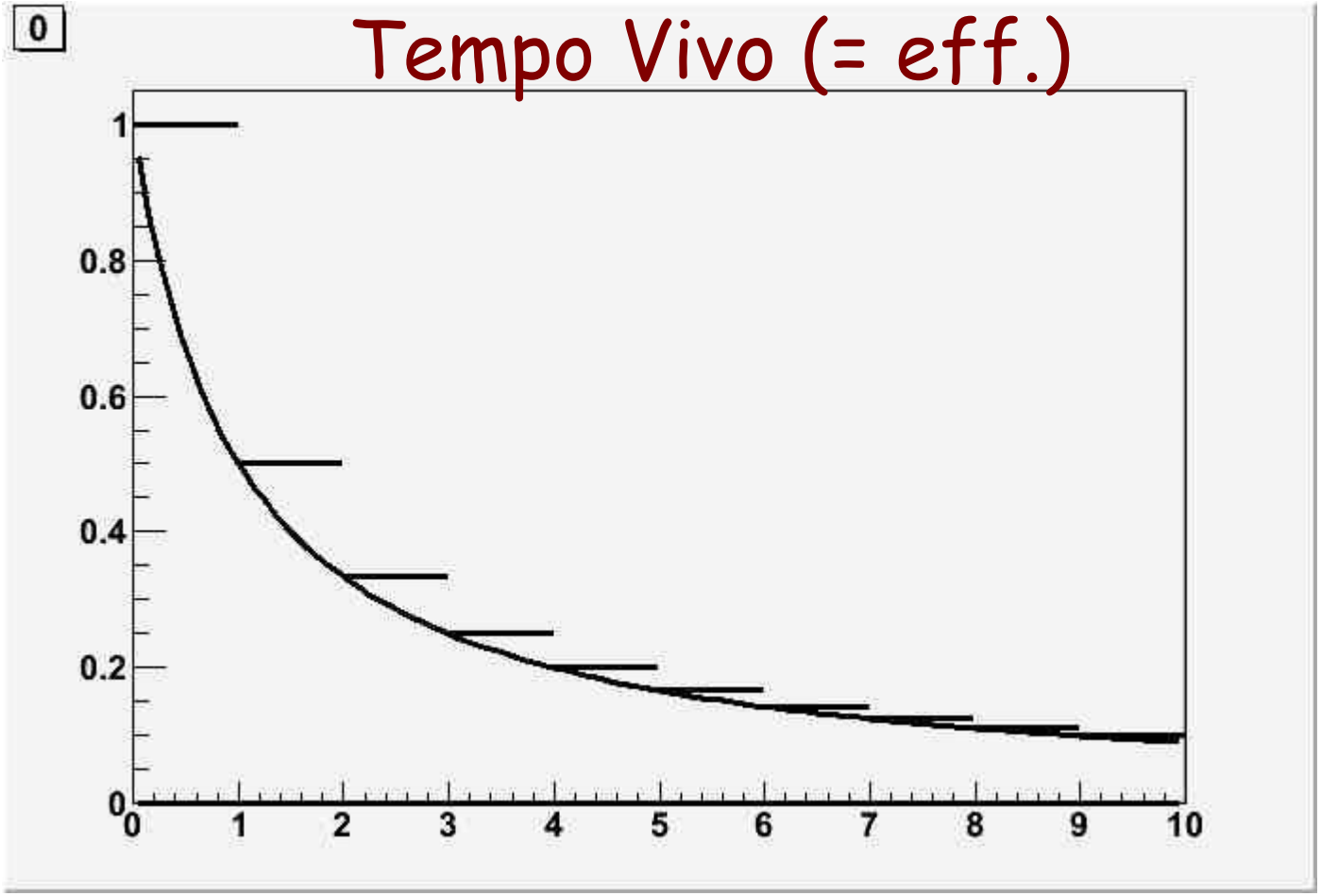
→ % tempo vivo = $1-\nu \cdot \tau$

→ rate eventi = $f \cdot (1-\nu \cdot \tau) = \nu$

→ $\nu = f/(1+f \cdot \tau)$

→ $\varepsilon = (\nu / f) = 1/(1+f \cdot \tau)$

Eventi scorrelati (2)



$x = f \cdot \tau$

Eventi scorrelati (3)

Se $x=(f\cdot\tau) \approx 1 \rightarrow \varepsilon \approx 50\%$!

$$x=(1-\varepsilon)/\varepsilon$$

$\varepsilon \approx 100\% \rightarrow x \approx (1-\varepsilon) \ll 1, \quad \tau \ll 1/f$

$\varepsilon \approx 95\% \rightarrow \tau \approx 0.05/f \quad (F=20\cdot f)$

$\varepsilon \approx 98\% \rightarrow \tau \approx 0.02/f \quad (F=50\cdot f)$

Derandomizzazione

Buffer con capacità N eventi:

... teoria delle code:

P_j = % tempo con j eventi in pancia

Rate transizione (j -> j+1) = $f \cdot P_j$

Rate transizione (j+1 -> j) = P_{j+1} / τ

All'equilibrio:

$$f \cdot P_j = P_{j+1} / \tau \quad \rightarrow \quad P_{j+1} = (f\tau) \cdot P_j = x \cdot P_j$$

Derandomizzazione (2)

$P_1 = x \cdot P_0$, $P_2 = x^2 \cdot P_0$, ... $P_N = x^N \cdot P_0$, $(x = f \cdot \tau)$

Normalizzazione $\Sigma P_k = 1 \rightarrow P_0 = 1/\Sigma x^k = (1-x)/(1-x^{N+1})$

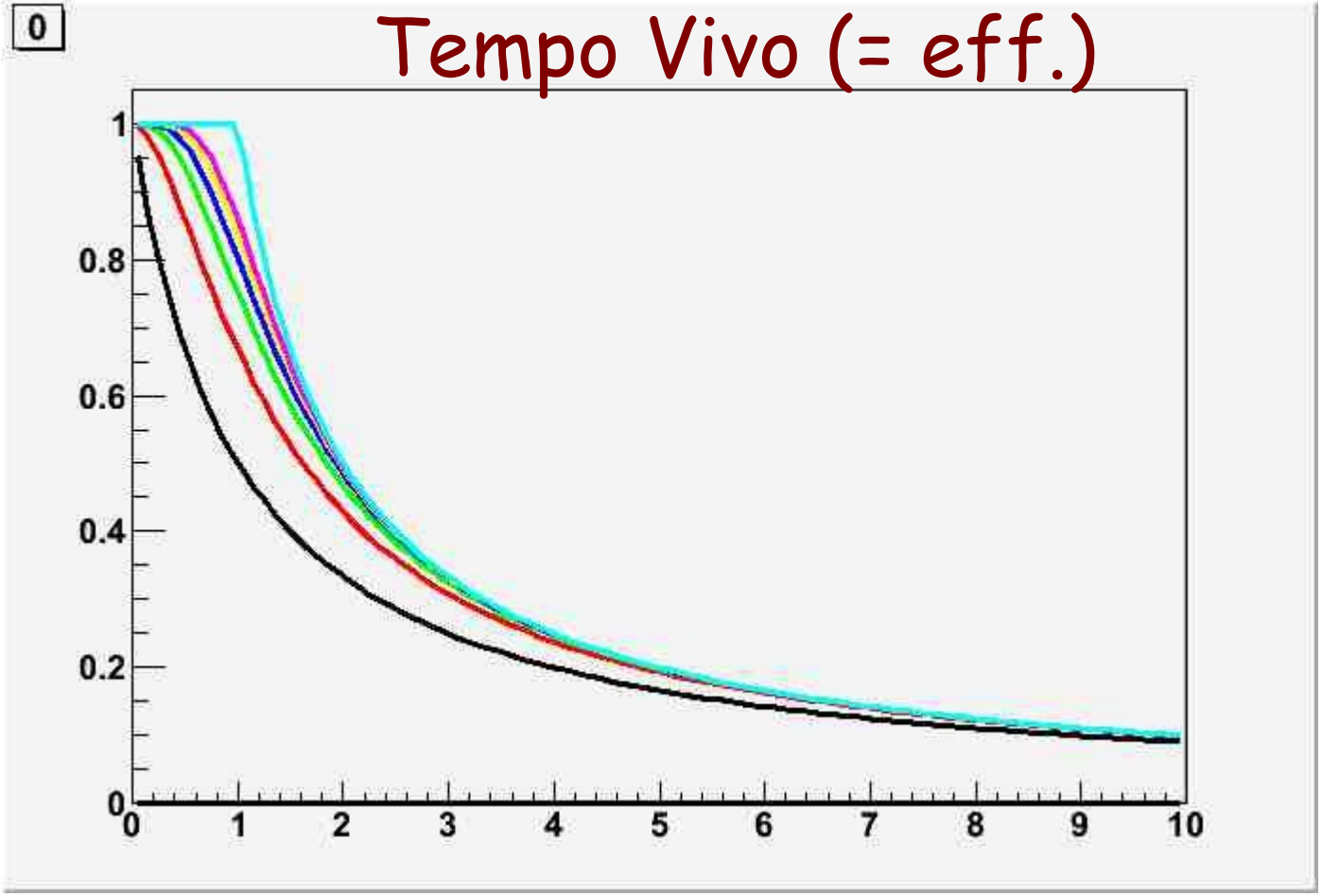
Tempo morto = $P_N = x^N \cdot (1-x)/(1-x^{N+1})$

Efficienza $\varepsilon_N = (1-P_N) = (1-x^N)/(1-x^{N+1})$

$(x<1)$: $\lim_{N\to\infty} \varepsilon_N = 100\%$

$(x>1)$: $\lim_{N\to\infty} \varepsilon_N = x^{-1}$

Efficienza con buffer N



$x = f \cdot \tau$

Derandomizzazione (3)

Se $x=(f\cdot\tau)=1 \rightarrow P_0=P_1=\ldots=P_N=1/(N+1)$

Tempo morto = P_N

Efficienza $\varepsilon = N/(N+1)$

$\varepsilon \approx 95\% \rightarrow N \approx 20$

$\varepsilon \approx 98\% \rightarrow N \approx 50$

Derandomizzazione (4)

In modo analogo si analizzano casi più complessi (es: N buffer in parallelo) ...

Testi online di teoria delle code:

http://web2.uwindsor.ca/math/hlynka/qonline.html

Conversioni Analogico-Digitali

ADC: analog-to-digital converter

Scelta dipende da:

- tipo di segnale
- velocita'
- risoluzione (numero di bit)

Molto diverso leggere un calorimetro o uno scintillatore o un tracciatore al silicio o un rivelatore proporzionale ...

- linearita' ?

Precisione/Risoluzione

Normalmente:

ADC Output $\propto$ [S(input)] (linearita')

- N bit : 2^N configurazioni da 0 a 2^N-1
- Risoluzione = 1 bit = (fondo scala)/2^N

N = 8 : 256 → 0.4 %

N = 10 : 1024 → 0.1 %

N = 12 : 4096 → 0.025 %

N = 16 : 65536 → 16 ppm

Codifica binaria

Numeri naturali (solo positivi): ogni bit rappresenta una potenza successiva di 2

Interi con segno: rappresentazione in complemento a 2

Complemento (operatore "~" : y = ~x):

~1001 1001 = 0110 0110 (complemento bit a bit)

ovvero ~0x99 = 0x66 (su 8 bit)

(base 2 <-> base 16): e' sufficiente convertire i bit a 4 a 4

Complemento a 2

Facile dimostrare che vale sempre:

$(x + \sim x) = 0xff;$

$\rightarrow \; (x + \sim x + 1) = 0x100 = 0$ (su 8 bit)

Essendo, per definizione, (-x) il numero per cui:

$(x + (-x)) = 0 \quad \rightarrow \quad (-x) = \sim x + 1$

Ovvero, nel nostro caso:

$-0x66 = 0x9a$

$-0x9a = 0x66$

Chi fra 0x9a e 0x66 e' negativo ?

Complemento a 2 (2)

0X9a = 1001 1010

0x66 = 0110 0110

Convenzione:

bit piu' significativo (msb) = bit di segno:

(msb == 1) <==> numeri negativi

(msb == 0) <==> numeri positivi

Valore numerico:

(msb == 0) <==> v = x ("as usual")

(msb == 1) <==> v = -(~x+1)

Qualche esempio

1001 1010 (0x9a) = - 0110 0110 (- 0x66)

1111 1111 (0xff) = - 0000 0001 (- 1)

1111 1110 (0xfe) = - 0000 0010 (- 2)

1000 0001 (0x81) = - 0111 1111 (- 0x7f)

1000 0000 (0x80) = - 1000 0000 (- 0x80) ???

c'e' un numero che rappresenta anche il proprio opposto ... succede con qualsiasi numero di bit.

Per convenzione, rappresenta il piu' piccolo numero negativo: -128 (8 bit), -32768 (16 bit), ...

Numeri con e senza segno

Proprieta' notevoli:

- ordinamento identico (eccetto ai "confini")
- aritmetica identica !

Range:

8 bit: 0 .. 255 -> -128 .. +127

12 bit: 0 .. 4095 -> -2048 .. +2047

16 bit: 0 .. 65535 -> -32768 .. +32767

32 bit: 0 .. $2^{32}-1$ -> -2^{31} .. $+2^{31}-1$

Operazioni

Somma: as usual ... ma ... riporto con segno (CS) != riporto senza segno (SS):

(CS): 0xff + 0x2 = 1 (nessun riporto)

(SS): 0xff + 0x2 = 1+0x100 → fuori dal range

riporto (carry over) di 0x100

Sottrazione: si somma il complemento a 2 ...

0x2 – 0x3 = 0x2 + 0xfd = 0xff

(CS): non richiede prestiti (borrow)

(SS): necessario un prestito di 0x100

Riporti/Prestiti

A seconda del caso (CS o SS) ci puo' essere o ci puo' non essere un problema di riporti/prestiti:

0x70+0x70 = 0xd0 SS: ok, CS: riporto

0x90-0x40 = 0x50 SS: ok, CS: prestito

0xd0+0x70 = 0x40 CS: ok, SS: riporto

0x70-0xf0 = 0x80 CS: ok, SS: prestito

un registro (status register) della unita' aritmetico-logica tiene traccia di quanto succede ...

Propagazione del segno

Cambio di rappresentazione, da N a M bit (con M > N). Ci sono (M-N) bit aggiuntivi:

SS: vengono messi a zero

CS: vengono messi come il bit di segno

0x70 -> 0x0070 (in ogni caso)

0x90 -> 0x0090 (SS) oppure 0xff90 (CS)

Da M a N bit si tronca il numero di bit (il valore puo' cambiare !):

0x0090 (CS) != 0x90 (CS)

0x1234 != 0x34 (in ogni caso)

Bit Field

Operazioni con bit:

j-bit test:	x & (1<<j)	and aritmetico
j-bit set:	x | (1<<j)	or " "
j-bit clear:	x & ~(1<<j)	and " "
j-bit flip:	x ^ (1<<j)	xor " "

allocazione/deallocazione di risorse (es: semafori):

test-and-set / test-and-clear

Read-Modify-Write: nel processore Motorola 68000 implementata come singola istruzione

Convertitori Digitale-Analogico

DAC:

- circuito sommatore (N resistenze ognuna dal valore doppio della precedente)

$R_N = R_0 * 2^{N-1}$ rapporto $1:2^{N-1}$?!?!

... diagramma a blocchi ? ...

- alternativa: partitore R-2R (pag 616 H-H)

Convertitori Analogico-Digitali

Tracking-ADC

- funziona in ~ "continua"
- abbastanza lento
- risoluzione OK
- poco costoso (costo ~ N)

... diagramma a blocchi ? ..

Flash-ADC

- serve trigger
- molto veloce (100-500 MSPS)
- risoluzione non elevata (8-10 bit max)
- costoso (costo ~ 2^N)
- "faticoso" da leggere

... diagramma a blocchi ? ...

ADC ad approssimazioni succ.

- trigger
- abbastanza veloce
- risoluzione OK
- poco costoso (costo ~ N)

... diagramma a blocchi ? ..

TDC

Conteggio di tempi:

- start / stop (uno dei due e' il trigger)
- clock interno (multi-hit)

oppure

- carica/scarica di una capacita' (corrente nota, t = CV/I) (single-hit)

... diagramma a blocchi ? ..

Logiche digitali

Logiche combinatorie

Logiche sequenziali

Reti Sequenziali (Macchine a Stati)

Logiche Combinatorie

Not / And / Nand / Or / Nor / Xor

Sommatore

Mux / Demux

Encoder / Decoder

Matrici di Logiche Programmabili (PAL)

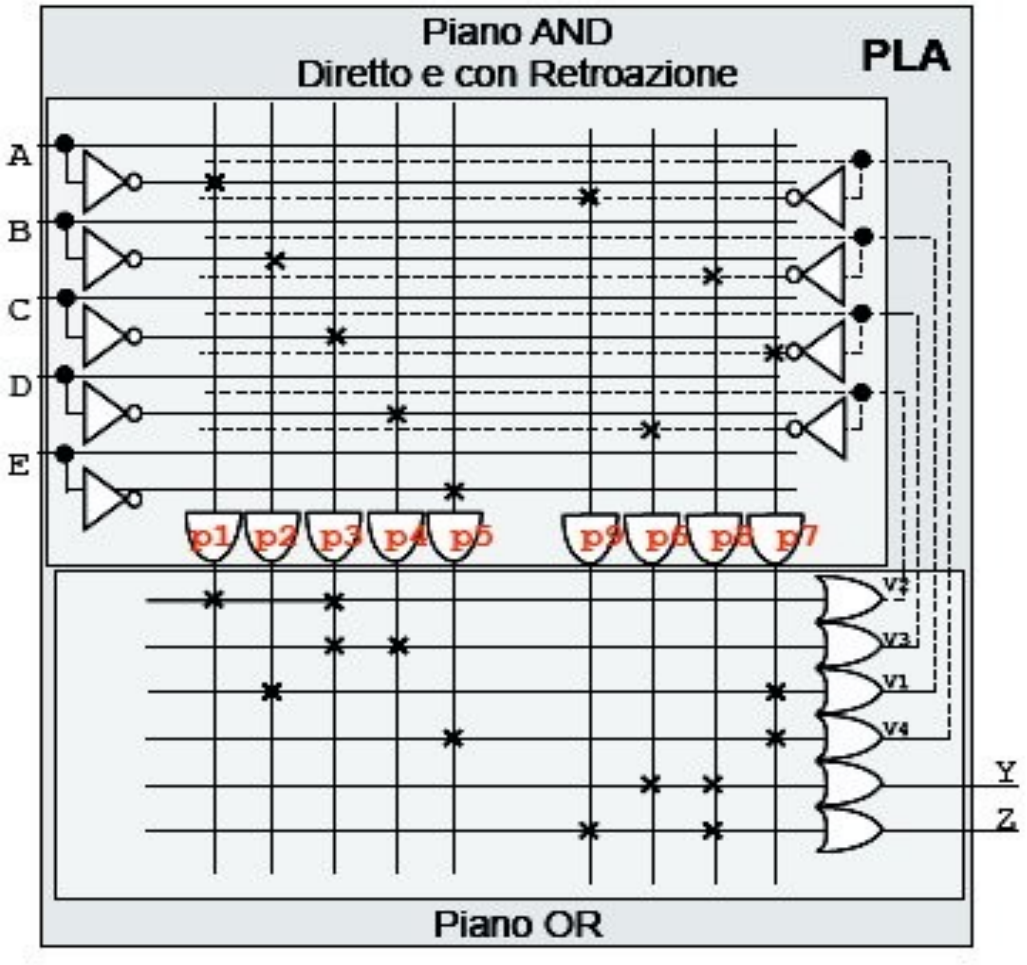
Unita' Aritmetico-Logiche (ALU)

Mux/Demux

Multiplexer/Demultiplexer

PAL

Esempio (PLA)

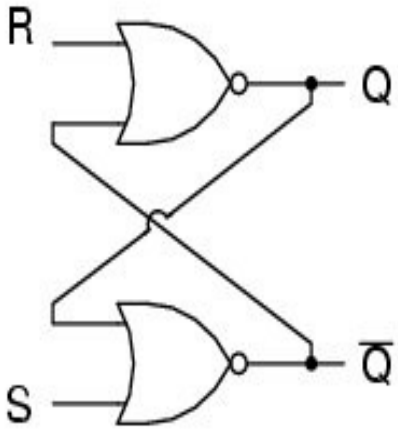


ALU

Struttura Unita' Aritmetico-Logica

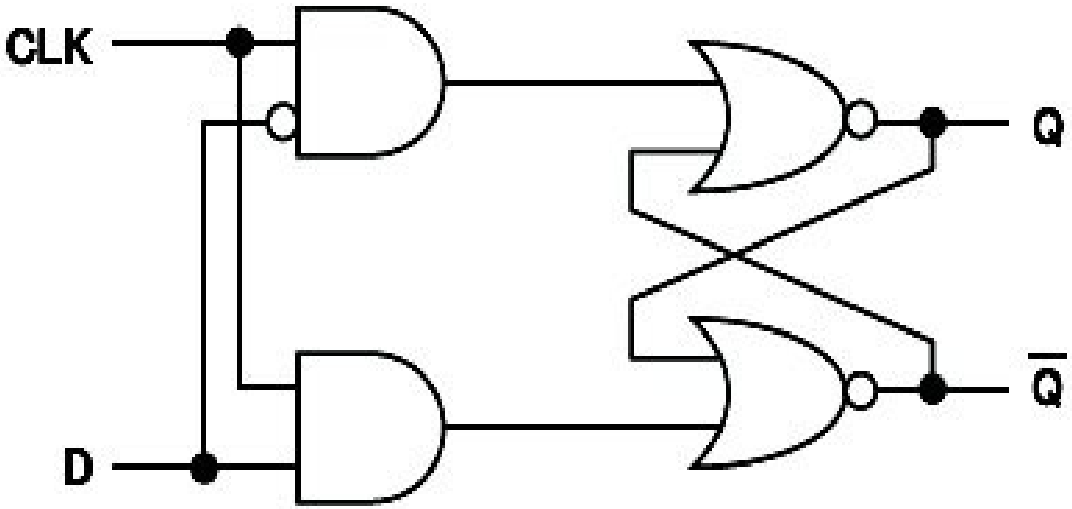
Logiche Sequenziali - Latch

Latch Set/Reset



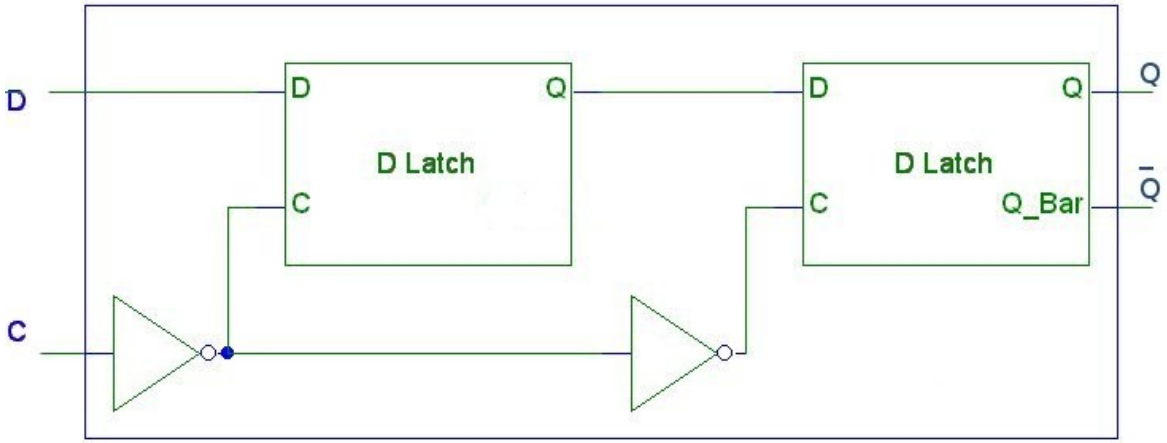
S	R	Q	$\overline{Q}$
0	0	latch	latch
0	1	0	1
1	0	1	0
1	1	0	0

Latch D

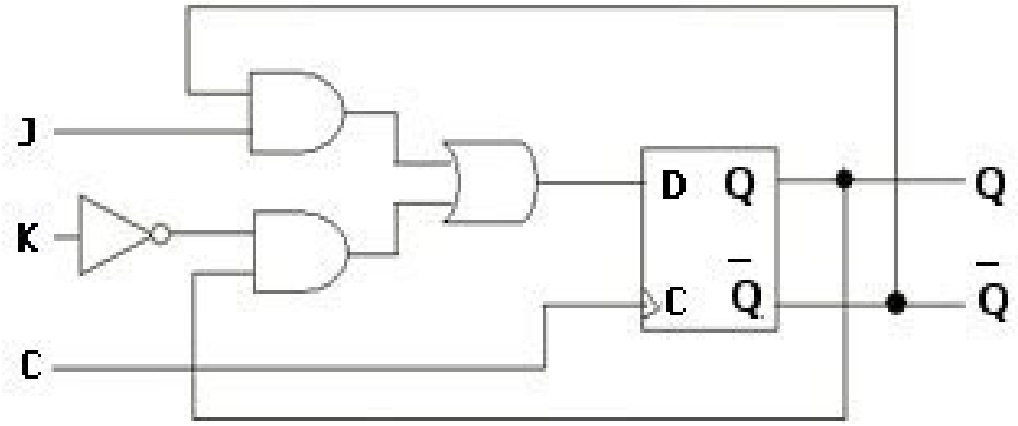


Flip-Flop

Flip-Flop
D



Flip-Flop
J/K



Registri e Contatori

Registri D

Registri a scorrimento

Contatori Ripple

Contatori Sincroni

Memorie

Memorie RAM e ROM

Memorie FIFO

Reti Sequenziali (FSM)

Sistema con memoria e stati

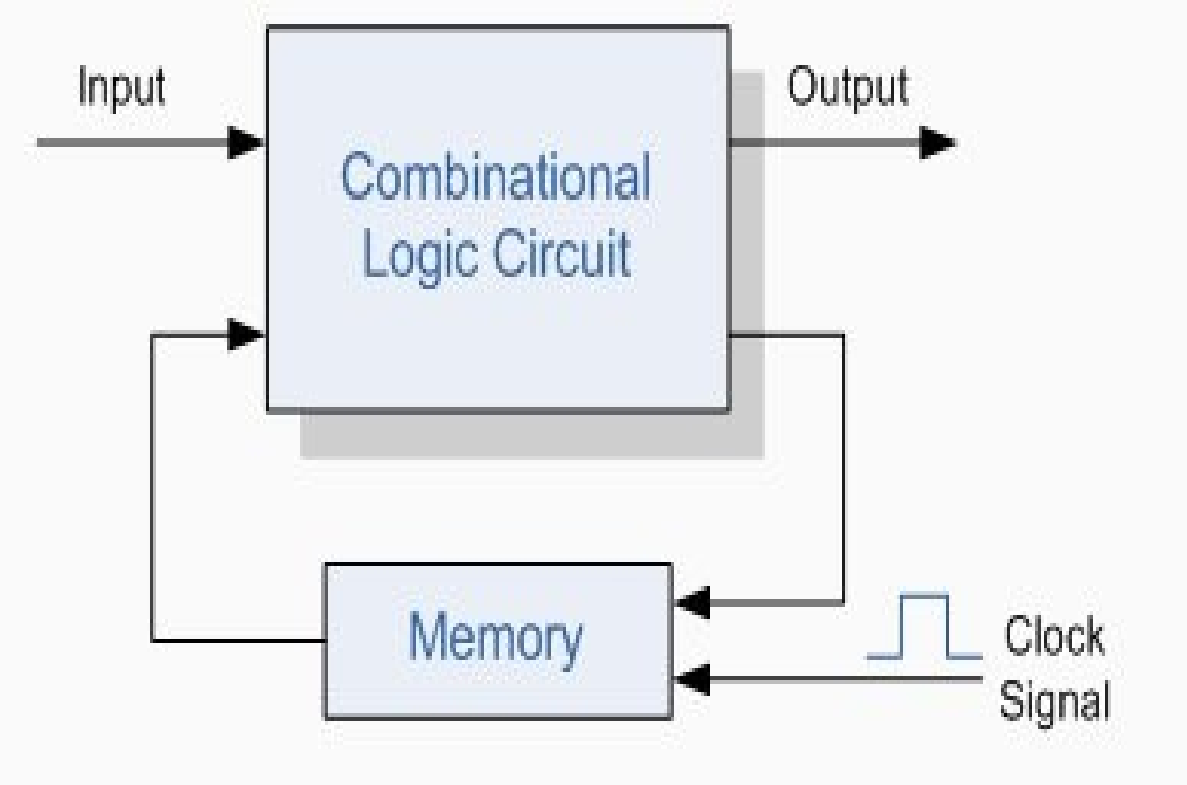
- sincrono o asincrono
- uscita (U) dipende da ingresso (I) e stato presente (S)

Macchina di Mealy

Sincrona:

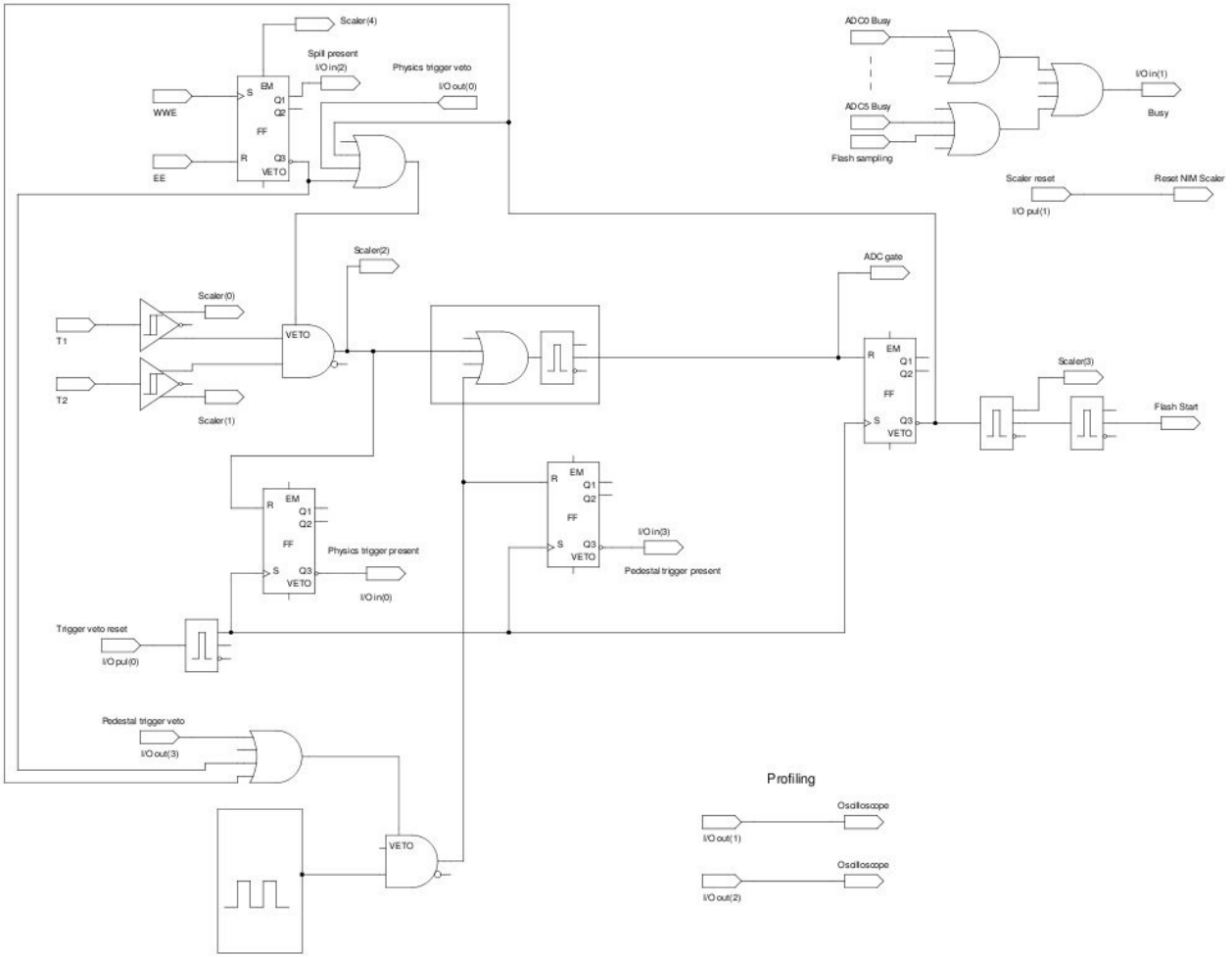
$U_{N+1} = f(I_N, S_N)$

$S_{N+1} = g(I_N, S_N)$



Logiche di Trigger

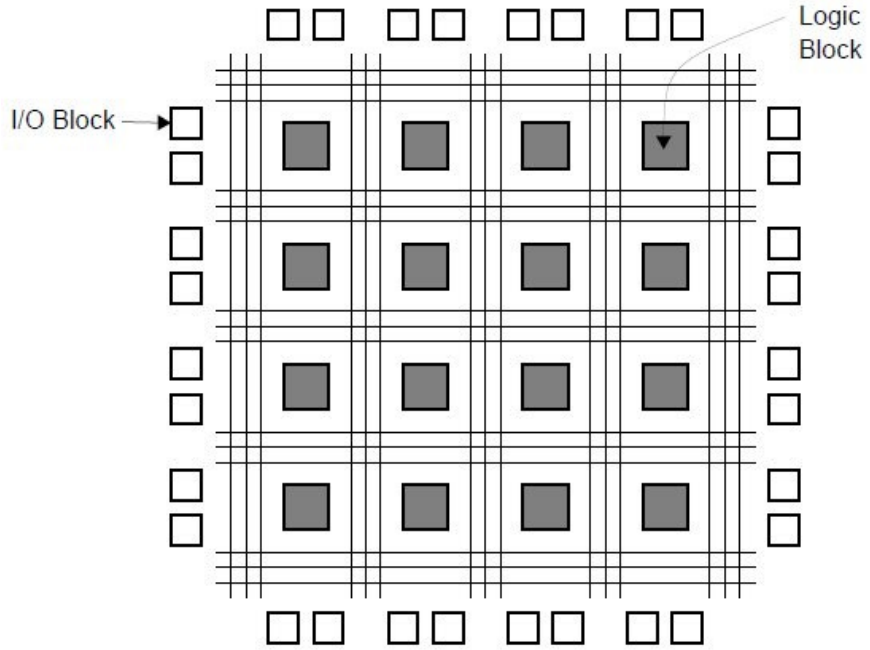
DREAM
TestBe



FPGA

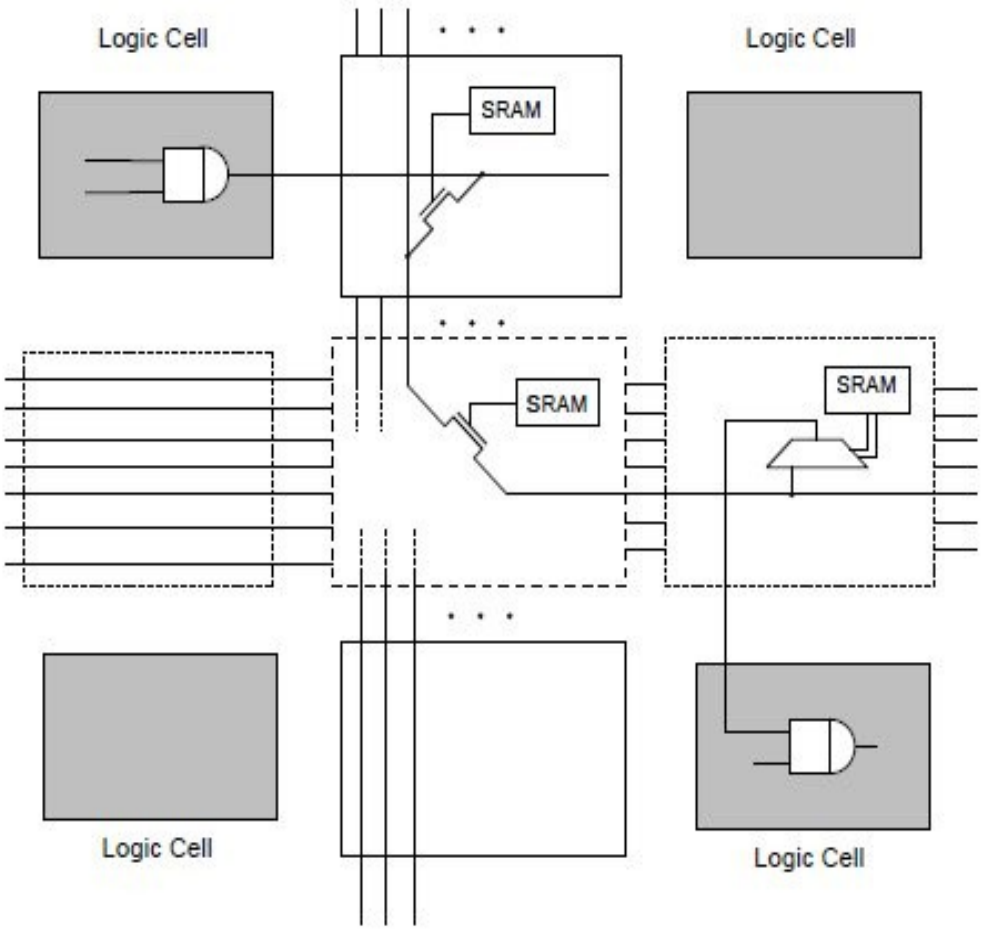
Field-Programmable Gate Array

10^2 / 10^3 / 10^4 unita' logiche interconnesse dinamicamente



FPGA (2)

Configurazione controllata da una RAM



FPGA (3)

Blocchi Logici Configurabili

multiplexer controlled by config program

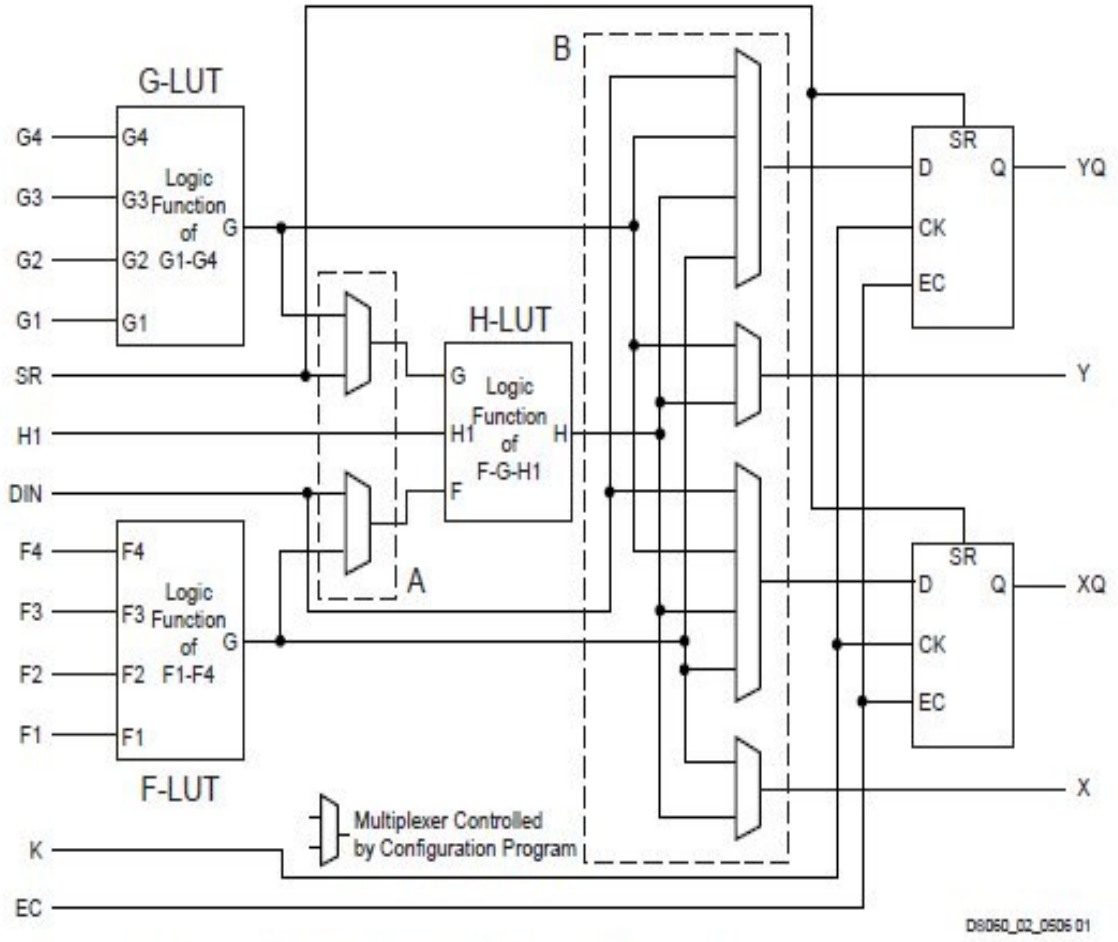


Figure 2: **Spartan/XL Simplified CLB Logic Diagram (some features not shown)**

FPGA .vs. ASIC

[ASIC = application specific integrated circuit]

Pro FPGA

- costo iniziale pressoche' nullo
- estremamente flessibile
- startup velocissimo
- ciclo di prototipaggio / debugging molto rapido

[dal sito http://www.fpga4fun.com: "you can design a circuit on your computer and have it running on your desk in minutes."]

FPGA .vs. ASIC (2)

Contro FPGA

- rapporto prezzo/prestazioni molto maggiore
- performance comunque peggiori
- percentuale di sfruttamento del circuito bassa

==>

- maggiore area occupata
- maggiore potenza dissipata (~10x)
- minore freq. di funzionamento (~1/10)

Programmazione FPGA

Hardware Description Languages (HDL) ... I piu' usati:

- Verilog
- VHDL

abbastanza equivalenti sotto molti aspetti.

Tutorial ed esempi si trovano facilmente in rete:

http://www.fpga4fun.com

-- Possibile codifica VHDL di un decoder a 2 ingressi

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY decoder_vhdl IS
  PORT (a, b            : IN BIT;
        u0,u1,u2,u3     : OUT BIT);
END decoder_vhdl;
ARCHITECTURE arch_decoder_vhdl OF decoder_vhdl IS
BEGIN
        -- attivazione (concorrente) dei segnali, non e' importante l'ordine
        u3 <= (a AND b) ;
        u2 <= ((NOT a) AND b);
        u1 <= (a AND (NOT b));
        u0 <= ((NOT a) AND (NOT b));
END arch_ decoder_vhdl;
```

Trigger e Temporizzazione

Segnale di trigger:

- efficiente per il segnale (S)
- inefficiente per il fondo (B) → purezza
- veloce

Parametrizzazione:

- ε (efficienza) : S(trigger)/S(true)
- **R** (reiezione) : B(true)/B(trigger)
- λ (latenza) : T(trigger)-T(true)

Fisica delle alte energie

a) Coincidenza e anticoincidenza (VETO) fra i segnali di piu' rivelatori (scintillatori / camere / moduli di calorimetri / ...)

b) Segnali di partenza analogici (?)

Problemi:

→ digitalizzazione veloce (possibilmente 0/1)

→ temporizzazione (istante di arrivo e durata di ogni segnale)

Logiche di Trigger

Discriminatori (comparatori con soglia programmabile)

Monostabili

Coincidenze (AND, OR)

Macchine a stati (Flip-Flop)

Comparatori

- Velocita'
- Stabilita'
- Immunita' al rumore

effetti di soglia:

rispari → isteresi (trigger di Schmitt)

Circuiti Monostabili

Anche chiamati: One-Shot o Timing Unit

- singoli impulsi "digitali"
- durata predefinita e stabile
- non-retriggerabili
- retriggerabili (→ piu' trigger ravvicinati)

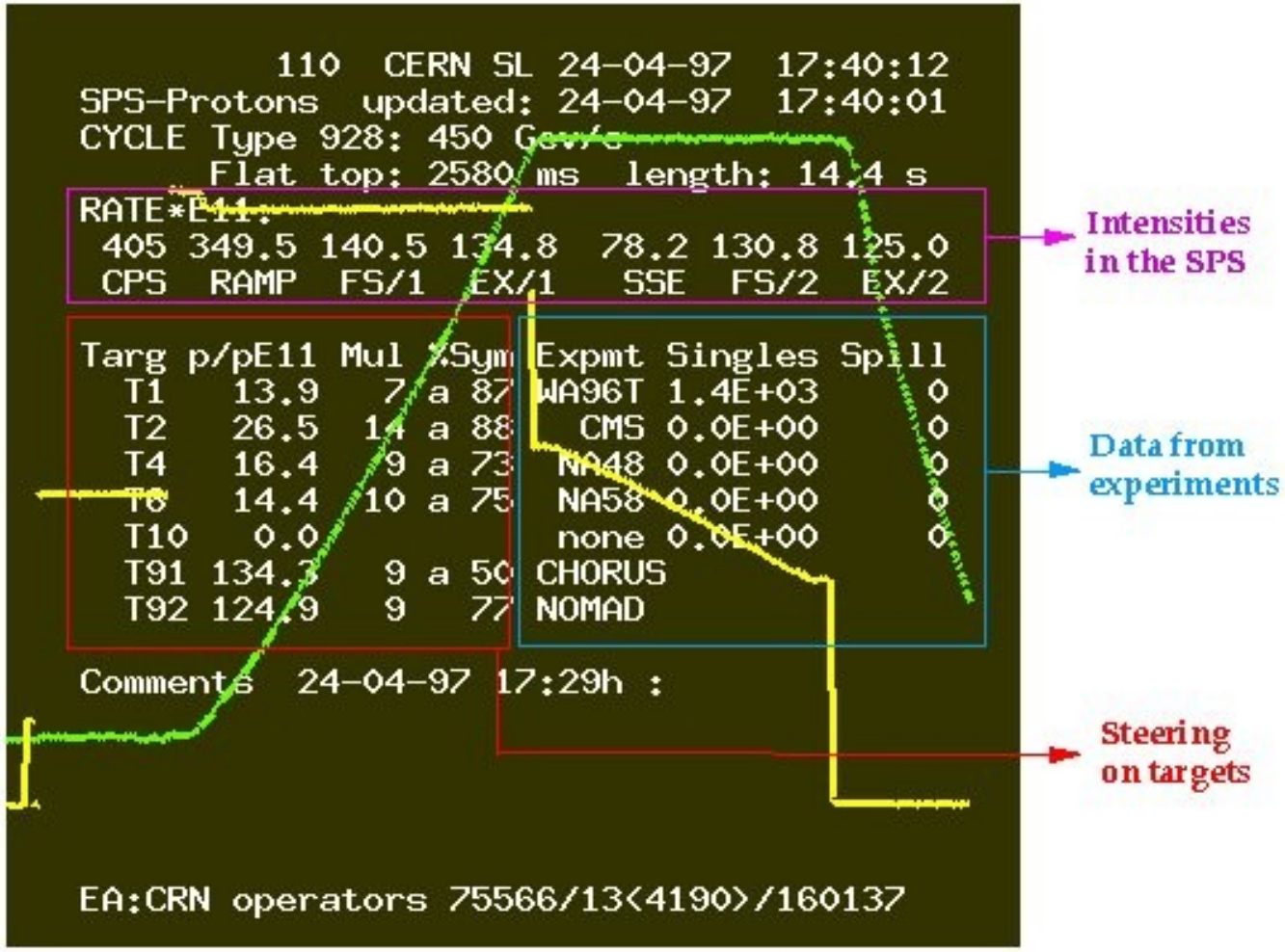
(flip-flop: "bistabili", oscillatori: "astabili")

Esempio – Testbeam Dream

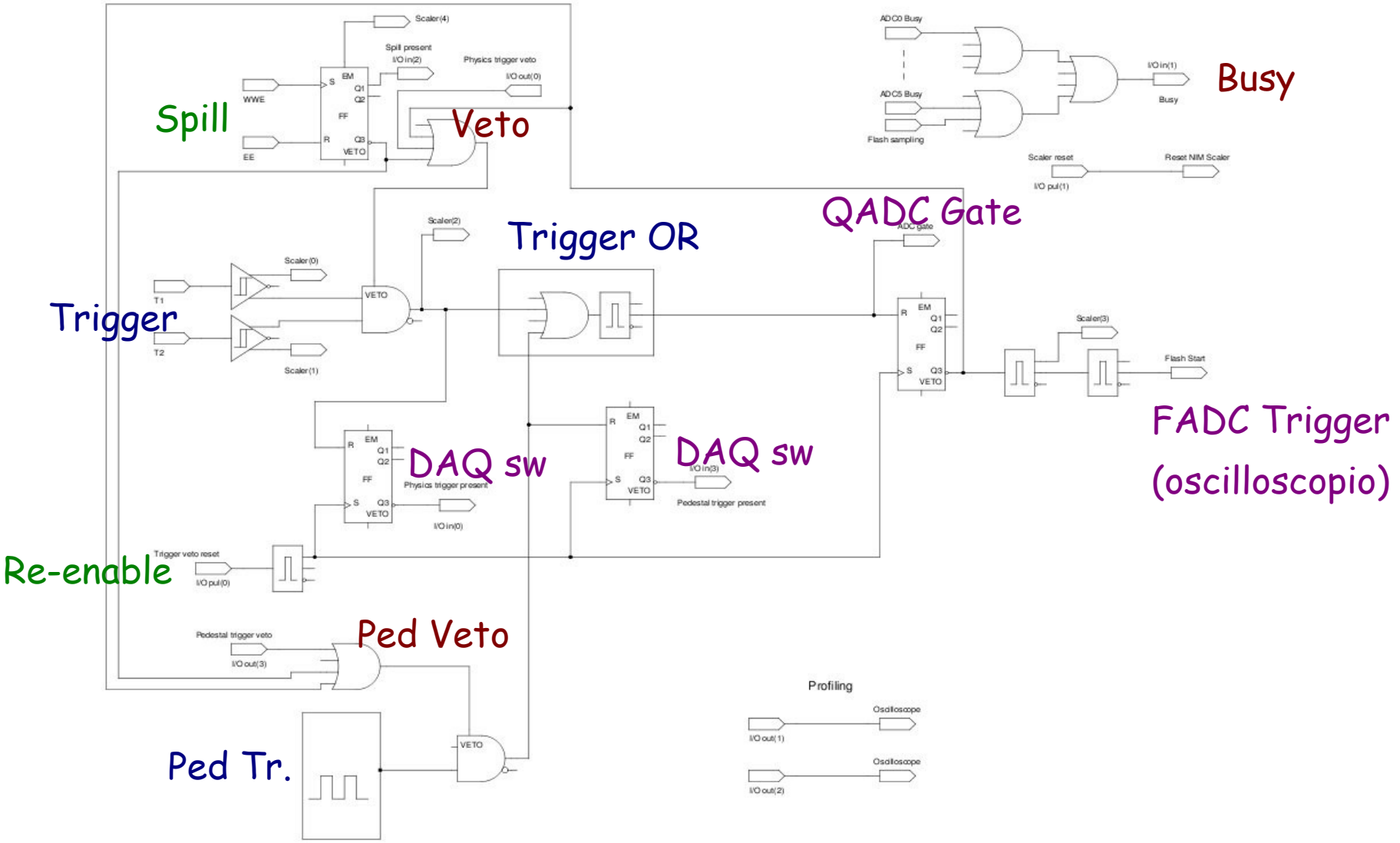
SpS Cycle

fascio:

2s ogni 14.4



Trigger "spill-driven"



Implementazione

- elettronica discreta (moduli NIM)
- puo' arrivare solo "in-spill"
- si auto veta
- pilota un ingresso asincrono (timing !)

→ meglio se pilota un clock

problema: per gli ADC serve un gate "prompt" (cavi veloci a bassa distorsione) ...

- "workaround" (= pezza): impulsi dai discriminatori quanto piu' possibile brevi

NIM (= Nuclear Instrumentation Module)

- moduli + crate (12 slot, fornisce solo alimentazioni)
- standard vecchio (1968-69) ma comunque di buone prestazioni (rivisto nel 1990)
- ingressi / uscite / cavi tutti con Z = 50 Ohm (!)
- abbastanza veloce (segnali ~ O(10 ns))
- Valori nominali: V(0) = 0, V(1) = -0.8 mV (-16 mA su 50 Ohm)
- moduli facilmente reperibili al pool del CERN
- tuttora in produzione (es.: CAEN)

Moduli

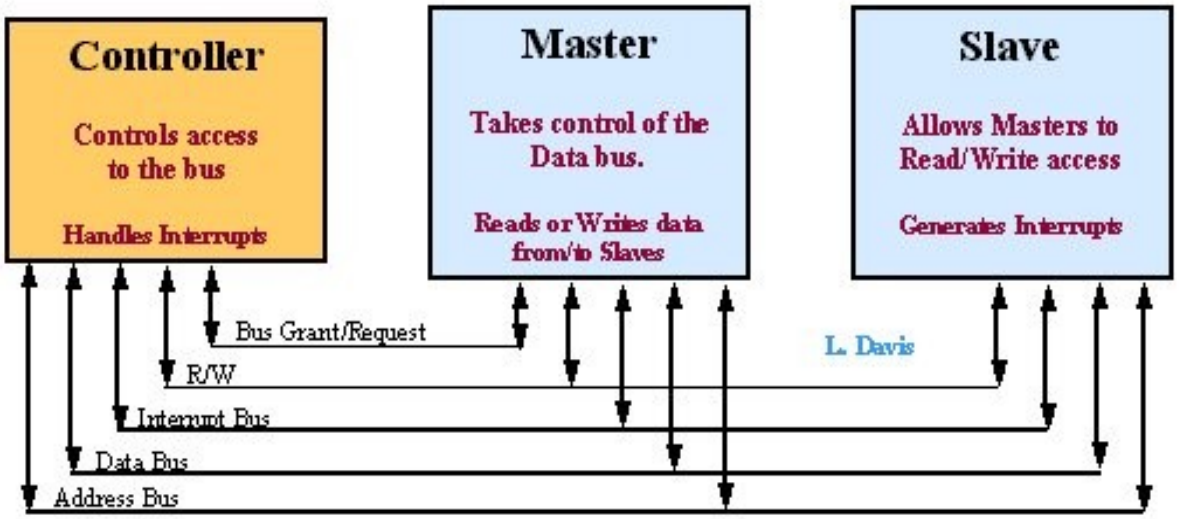
- amplificatori, attenuatori
- multicanali, ADC, discriminatori
- adattatori di livello (NIM<->ECL, NIM<->TTL)
- Fan In – Fan Out (or, nor), coincidenze (and, nand)
- timing unit (= monostabili, bistabili, clock)
- scaler
- unita' logiche configurabili (con switch o con segnali esterni)

VME (= Versa Module Eurocard)

- standard (aperto) definito nel 1981
- supporto per multiprocessamento in parallelo
- crate modulari di 21 slot interconnesse
- standard elettrico TTL: V(0)=+5V, V(1)=0V
- bus dati e di indirizzi a 32 bit
- portata: 40 – 80 (– 160 – 500) MB/s

rispecchia le caratteristiche del microprocessore 68000 della Motorola (... VERSAbus era stato definito, nel 1979, dalla Motorola proprio per il 68000)

VME bus



VMEbus System

- bus dati: 8-16-24-32-64 bit

- bus indirizzi: 16-24-32-40-64 bit (+ address modifier)

- trasferimenti asincroni (master/slave)

- arbitraggio (in slot 1)

- 7 livelli di interrupt

Trasferimento dati

Esempio: scrittura di un dato (A: 24 bit, D: 16 bit)

- il "Master" chiede il bus (bus request)
- l'arbitro ad un certo punto glielo concede (bus grant)
- il "Master" scrive address modifier, indirizzo, dato
- lo "Slave" riconosce il proprio indirizzo e la richiesta di scrittura, carica il dato, lo memorizza e segnala il tutto
- il "Master" rilascia il bus

in caso di lettura sara' lo "Slave" a pilotare il bus dati ...

In rete trovate tutto quanto ...

Interrupt

Richiesta (asincrona) di intervento di uno slave ... ("master cercasi" ... immaginate lo squillo di un telefono):

- lo slave attiva il livello X di IRQ (chiama uno di 7 telefoni ...)

- uno ed solo uno dei master deve rispondere (IACK): "Pronto chi e' che ha chiamato al telefono X ? " [X lo specifica sulle linee di indirizzo A1-A3 ---> inizia una operazione di lettura]

- lo slave si identifica scrivendo un "interrupt vector" sul bus dati

- il master legge il vettore, e chiama la funzione di gestione appropriata

- c'e' una ed una sola funzione di gestione per vettore di interrupt

Moduli VME

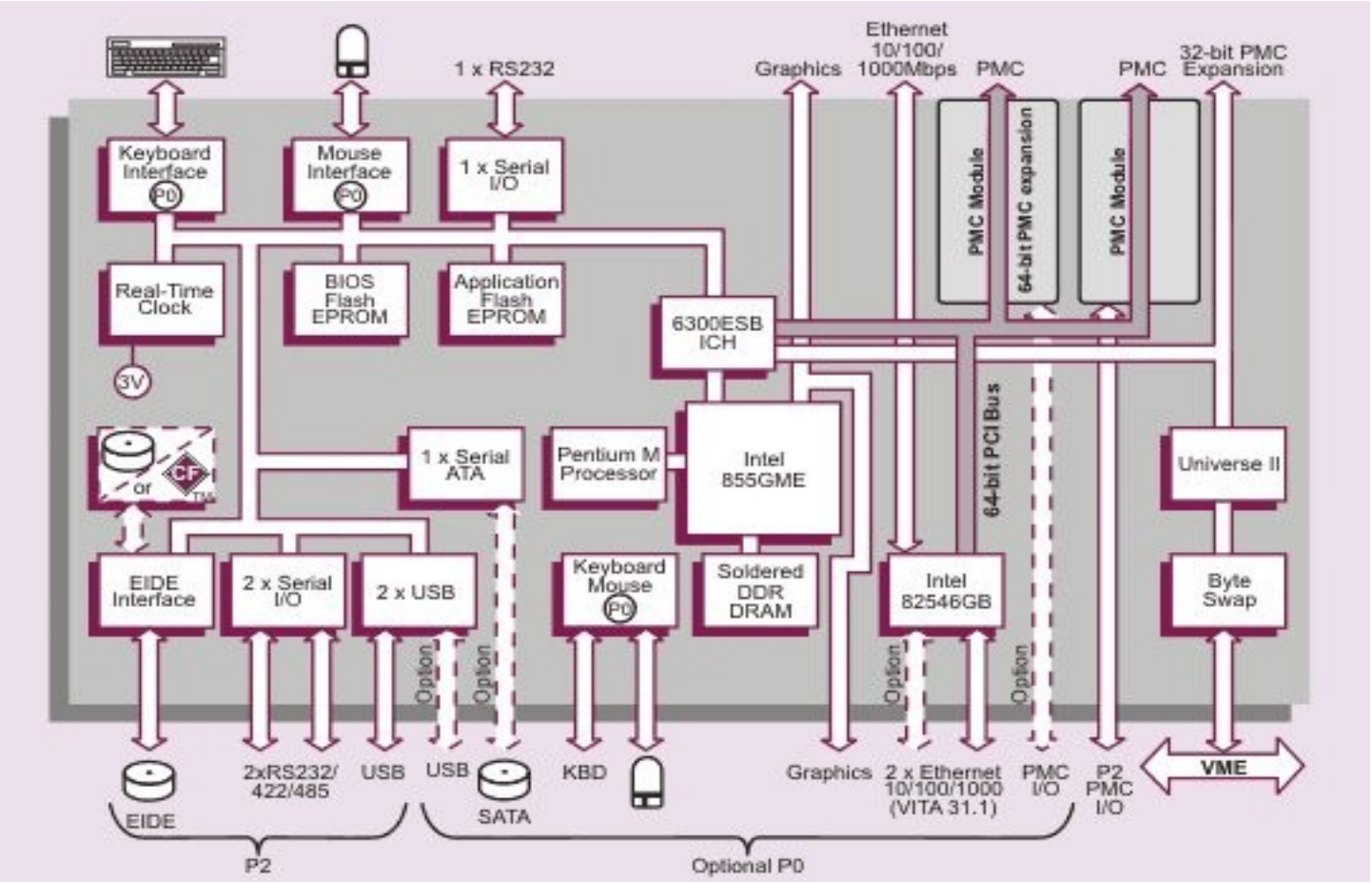
- FADC/PADC/QDC/TDC
- ... I/O Module ...

ma anche:

- processori (ATLAS usa le CT-VP110 single board computer)
- interfacce verso bus seriali e paralleli (ethernet, USB, PCI, ...)

... in realta' negli esperimenti moderni il bus VME e' usato solo per controllo, configurazione e monitoraggio (il flusso dei dati viaggia per altre strade ...: S-Link)

CT-VP315/VP317



Bridge VME<->PCI

Endianness:

VME big endian

PCI little endian

0x01234567:

BE: 0x01 0x23 0x45 0x67

LE: 0x67 0x45 0x23 0x01

byte 0 1 2 3

Trasferimento Dati

Sincroni/Asincroni:

telefonata ... (~ brevi distanze / latenze)

lettera ... (~ lunghe distanze / latenze)

Seriali/Paralleli:

fax, rete internet / PCI, VME

Point-to-point/broadcast → messaggi

Block Transfer

Operazione di lettura/scrittura sequenziale ininterrotta – in generale non controllata dalla CPU

Esempi:

- VME master
- device PCI
- ISA DMA controller
- North o South Bridge
- ...

DMA

Hardware dedicato (fuori dal microprocessore, ad es. nel chipset) che accede contemporaneamente alla memoria e ad un device hw:

BMA (Block Move Accelerator)

DMA (Direct Memory Access)

Sequenza:

- si caricano gli indirizzi di partenza (source, dest)
- si definisce il modo (es: until "fifo not empty")
- si da' il via

Software

Accesso risorse hw:

a) riservato al kernel

b) implementato in device driver + librerie dinamiche (codice c + assembler, in qualche caso)

Memory Mapping I/O:

1. indirizzi assoluti <-> indirizzi virtuali

   TLB ("translation lookaside buffer")

2. accesso diretto a tutti i registri hw

Memory Mapping

Esempio:

```
void* adr=mmap(NULL, size, prot, flags, fd, base);
munmap(adr, size);
```

dove:

```
prot = PROT_READ | PROT_WRITE;
flags = MAP_SHARED;
fd = open(dev, O_RDWR);
base = physical_base_address;
```

Real Time

Real Time kernel: esiste una massima latenza
(ovvero: interrupt handler rispondono entro <= TOT)

... scheduling piu' aggressivo ("preemption")

Unix (linux), Windows non sono Real Time

(latenze massime non definite, kernel non interrompibile)

Unix RT (Posix):

lynx-os (tcix), ..., RTLinux, ..., RTAI, ...

(anche il kernel e' interrompibile)

VxWorks, OS9

VMS (vax, anni 80): "event-driven"

Gestione Real Time

1. eventi asincroni (interrupt, segnali, eccezioni)
   - → es: signal(SIGNUM, sighandler)

     void (*sighandler)(int)
2. timer (temporizzazione)
   - → es.: getitimer, setitimer (→ SIGALRM)
3. multi-processing/threading
   - → es.: pthread_create, pthread_join
4. IPC
   - → semafori(mutex), messaggi, memorie condivise

Segnali

"interrupt" software → puo' essere mascherato

a) esterni (asincroni):

kill -SIGTERM pid ... kill -SIGUSR1 pid ...

ctrl-C → SIGINT (termina)

ctrl-Z → SIGSTOP (stop: non mascherabile)

ctrl-\ → SIGQUIT (quit: core dump)

...

SIGKILL: termina subito (non mascherabile)

b) interni ("sincroni"):

segm. fault (SIGSEGV), floating point excp. (SIGFPE)

Segnali(2)

kill -l:

1) SIGHUP 2) SIGINT 3) SIGQUIT 4) SIGILL
5) SIGTRAP 6) SIGABRT 7) SIGBUS 8) SIGFPE
9) SIGKILL 10) SIGUSR1 11) SIGSEGV 12) SIGUSR2
13) SIGPIPE 14) SIGALRM 15) SIGTERM 16) SIGSTKFLT
17) SIGCHLD 18) SIGCONT 19) SIGSTOP 20) SIGTSTP
21) SIGTTIN 22) SIGTTOU 23) SIGURG 24) SIGXCPU
25) SIGXFSZ 26) SIGVTALRM 27) SIGPROF 28) SIGWINCH
29) SIGIO 30) SIGPWR 31) SIGSYS

... fino a 64 (un paio non predefiniti)

Gestione Segnali

sigwait, sigtimedwait:

aspetta su piu' segnali, finche' uno non arriva

→ in ambienti multithreaded e' il modo piu' semplice per riceverli nel "posto giusto"

signal (ANSI C), sigaction (POSIX):

associano una funzione o una "azione" (informazione piu' completa)

Inter Process Communication

Memorie condivise (es. Unix system V)

'ipcs -m'

```
------ Shared Memory Segments --------
key          shmid     owner    perms   bytes  nattch status
0x000caffe 4653080 roberto 777     4096   0
```

```
int id=shmget( key, size, IPC_CREAT|0666);
void* ptr=shmat(id, NULL,mode);
shmdt(ptr);
shmctl(id, cmd, &shmpar);
```

Semafori (es. POSIX)

sem_t* sem=sem_open("/nome",O_CREAT,0666,value);

sem_unlink("/nome");

sem_post(sem); → incrementa sem di 1

sem_wait(sem); → decrementa sem di 1 (se diverso da 0)

sem_trywait(sem); → ... ci prova ...

sem_timedwait(sem,timeout); → ci prova fino a timeout

sem_getvalue(sem,&value);

Messaggi (es. Unix system V)

'ipcs -q'

```
int id=msgget( key, IPC_CREAT);
msgsnd(id, &message, size, mode);
msgrcv(id, &message, size, type, mode);
```

mode : IPC_NOWAIT, ...

MPI - CORBA

"Message Passing Interface":

protocollo per generazione e gestione messaggi in rete

"Common Object Request Broker Architecture":

scambio di oggetti indipendentemente da linguaggio e distribuzione nella rete dei soggetti ("broker": mediatore)

(MPI ha un set di modi di comunicazione piu' variegato)

PIPE

Named pipe (anche chiamate FIFO):

```
% mkfifo fifo1
% ls -l fifo1
prw-r--r-- 1 roberto roberto 0 2009-07-03 02:25 fifo1
```

un processo scrive nella pipe, uno legge.

Multi-threading

suddivisione di un processo in piu' branche in esecuzione parallela (thread)

→ intercomunicazione piu' facile

→ interferenze distruttive piu' facili

sincronizzazione, scheduling (POSIX pthread):

→ create, join, exit

→ mutex (lock, unlock, trylock, ...)

→ condition (wait, signal, ...)

Multi-core CPU

(un core == una unita' di processamento)

velocita' CPU limitata dal dissipamento:

potenza = $a*(\text{freq clock})^2$

scappatoia(!): + core su di un singolo chip

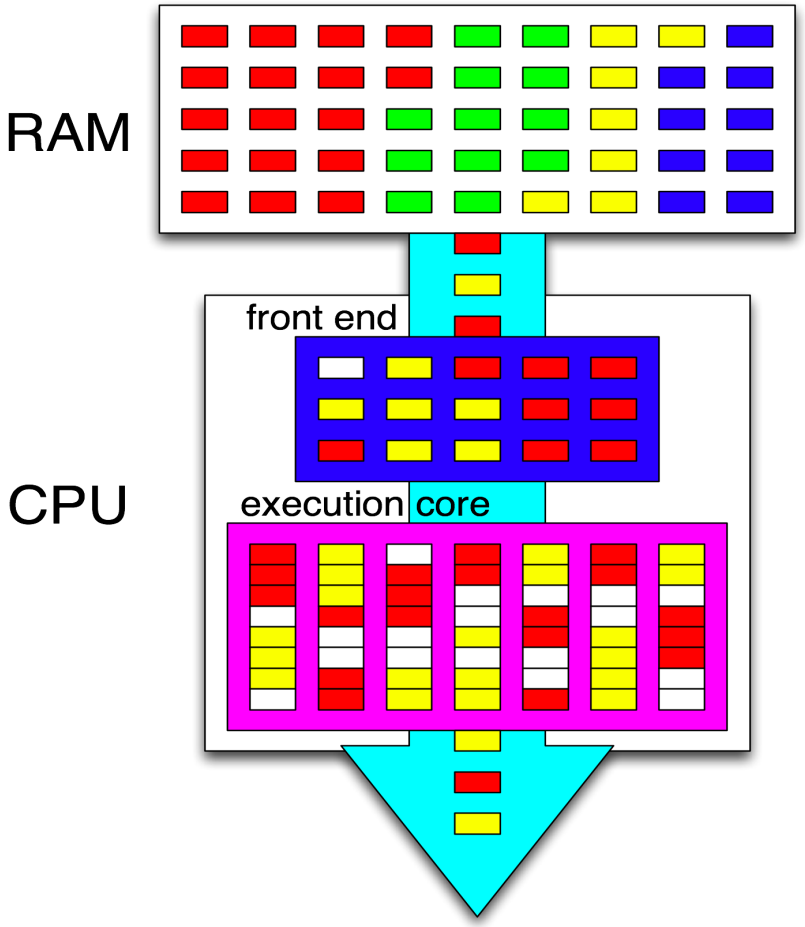
2-3-4-6-8-...

e + thread su un singolo core (tipicamente 2)

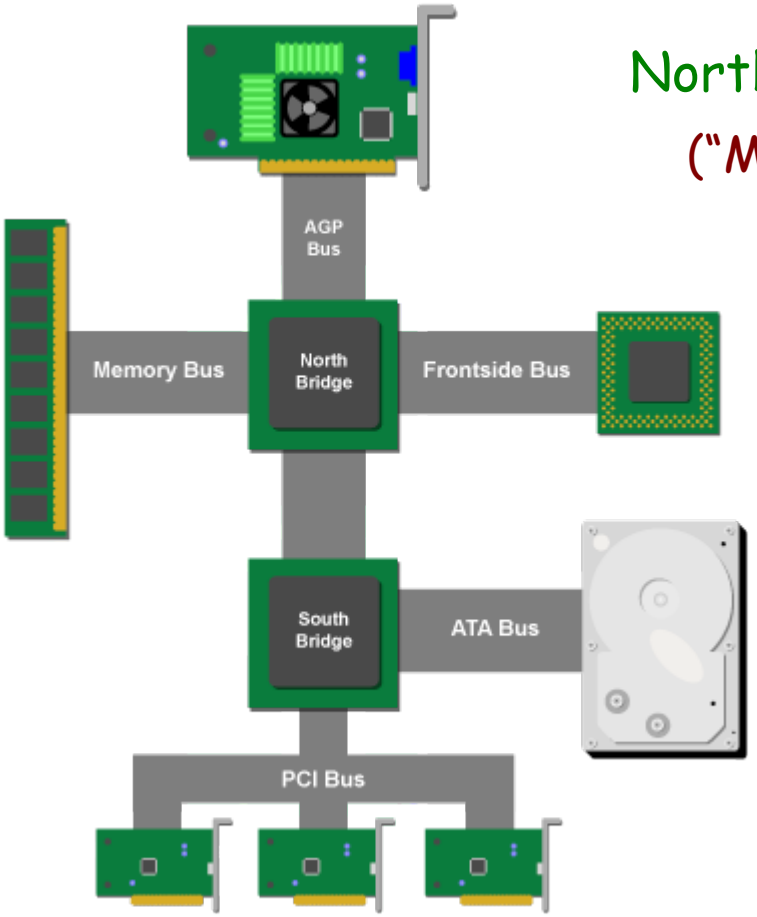
1 CPU : 2-4-8-16 thread in parallelo

HyperThreading CPU

1 CPU : 2-4-8-16
thread in parallelo



PC Architecture (intel)



NorthBridge (piu' veloce):

("Memory Controller Hub" - MCH)

RAM, AGP (video)

SouthBridge (piu' lento):

("I/O Controller Hub 2" - ICH2)

altri device (tastiera, mouse, ...)

Architetture CPU

CISC (Complex Instruction Set Computer):

- set di istruzioni del microprocessore ampio
- codice assembler piu' vicino al codice di alto livello
- programmi piu' compatti
- istruzioni lente, accessi ripetuti alla memoria

.... ma nel 90% del tempo la CPU utilizza sempre un ristretto sottoinsieme di istruzioni →

RISC (Reduced Instruction Set Computer):

- set limitato di istruzioni semplici e veloci
- cache locale (di primo e secondo livello) per dati e istruzioni
- pipeline (esecuzione parallela di piu' istruzioni)

Architetture CPU (2)

CISC: VAX, 68000, 80x86

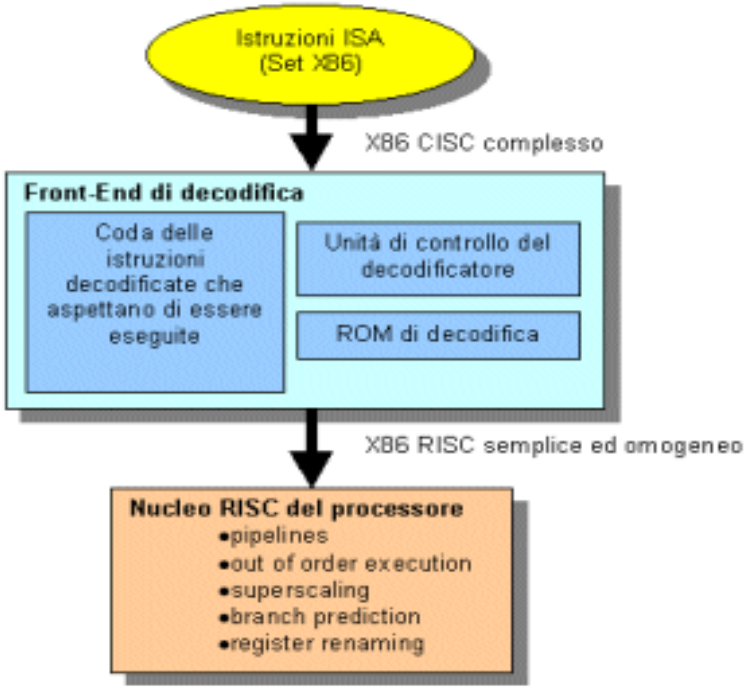
RISC: alpha, sparc, mips, powerpc, arm, pa-risc

Pentium4:

finge di essere un CISC

(compatibilita' x86)

ma lavora come un RISC

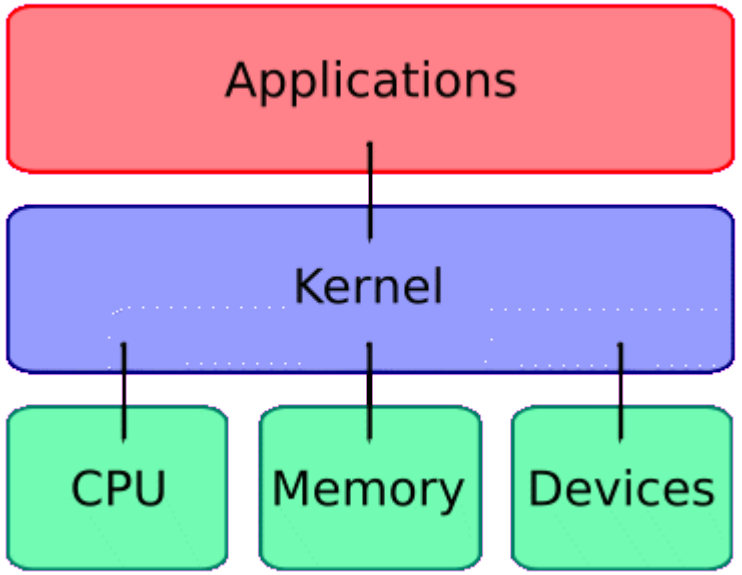


Kernel

Cuore del sistema operativo

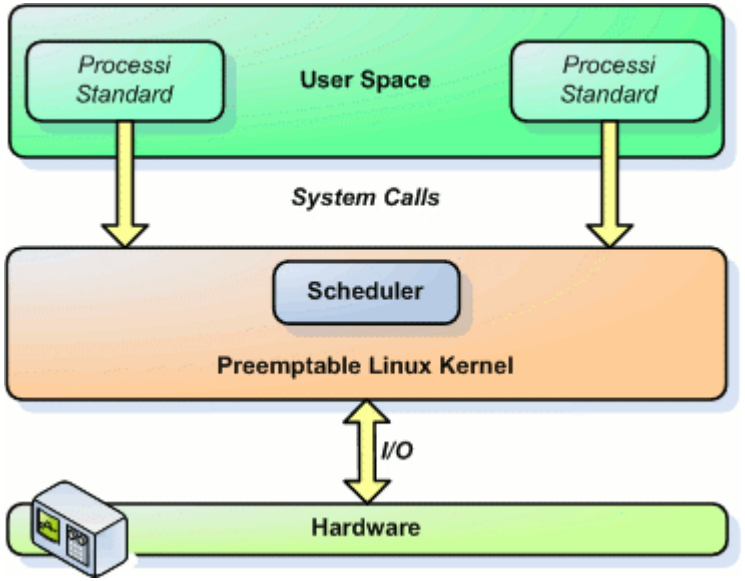
Gestione risorse hardware

Scheduling e gestione processi sw



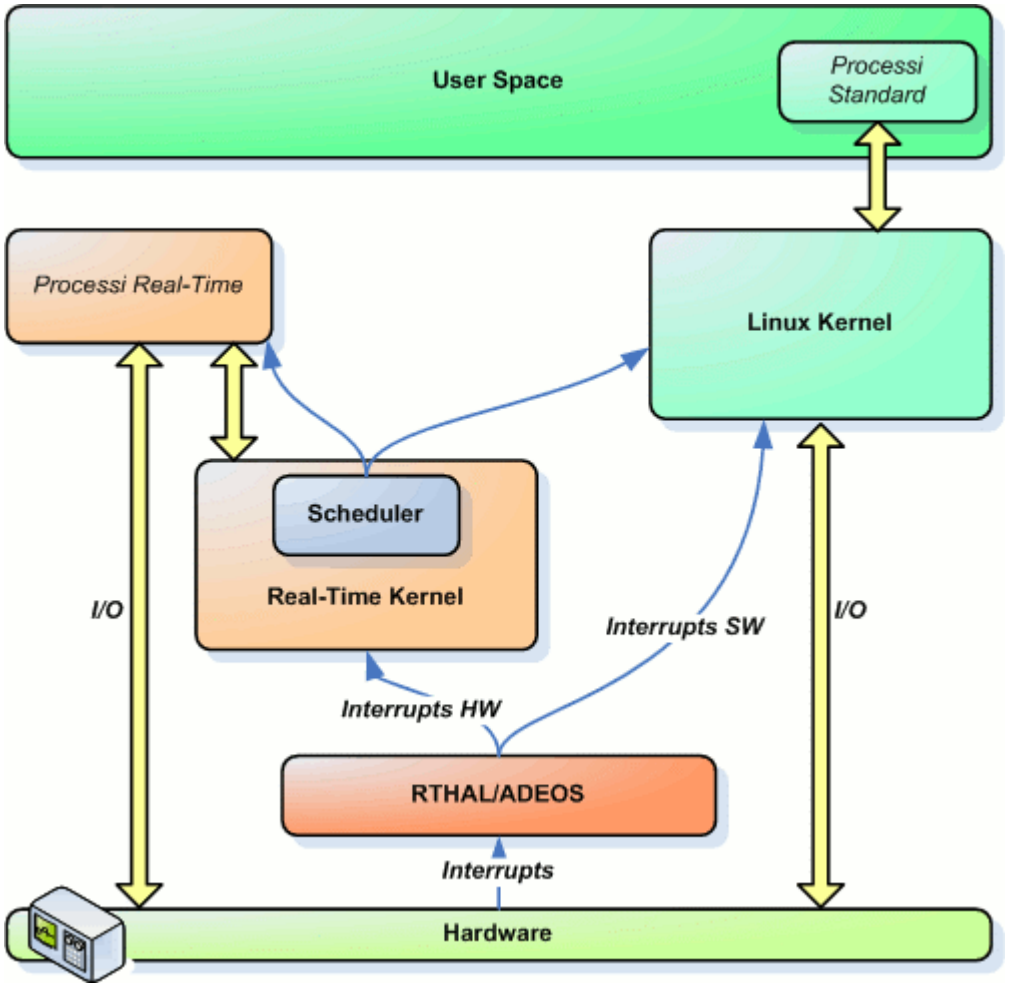
Real Time (1)

Low-latency patch (Ubuntu Studio): linux kernel interrompibile



Real Time (2)

RTAI: il kernel linux gira come una applicazione a priorita' piu' bassa



Architettura Trigger/DAQ

THE LHC CASE

Sistema complesso a molti livelli:

- configurazione hw e sw (db)
- controllo e U.I. (run control GUI !)
- lettura, formattazione e transferimento dati
- trigger, analisi e selezione eventi
- monitoring
- storage "persistente"

Controllo rivelatore (DCS)

Architettura (2)

ATLAS

Trasferimento dati fra (sub-)farm di processamento e buffering

a) Link "specializzati" fino a buffer di 1 livello

b) protocolli di rete (UDP, TCP, ...) oltre

c) message passing con CORBA

d) servizi/protocolli di livello superiore (specifici) implementati sopra questi

e) "scalabile"

ATLAS DAQ

Impementazione basata:

- C, C++, Java, Python
- linux (no RT)
- moduli multiprocessore, multicore, rack-mounted

gerarchia: modulo (~4-8-16 core, O(10-100) processi)

→ rack (~30 moduli, O(1000) processi)

In ognuno dei (50-100) rack:

- 1 local server (file system, proxy caching)
- 1 switch di rete (GB ethernet)

Dream DAQ

Molto piu' semplice:

- 2 PC standard (linux)
- 2 crate VME
- 2 interfacce pci-vme SBS 61x
- VME driver giapponese (“kinoko”)
  - → Mapped I/O, IRQ, DMA implementato
  - → abbiamo dovuto modificarlo per supportare insieme le nostre 2 interfacce

Dream DAQ (2)

3 processi di base:

- readout (singolo evento ...)
- dataWriter
- sampler

con una memoria condivisa in mezzo (buffer eventi FIFO)

In spill solo il readout e' attivo, a fine spill il dataWriter entra in azione, infine tocca al sampler

Readout

Classe base VME: apre e mappa il device (ADC, TDC, I/O, FADC, ...)

Classi derivate:

- LeCroy L1182 (charge ADC)
- LeCroy L1176 (TDC)
- CES RCB8047 (trigger module)
- KloeTdc
- Caen V260 (scaler), V262 (I/O reg), V513 (I/O reg)
- Caen V488 (TDC), V767A (TDC)
- Caen V792AC (charge ADC)
- SIS 3320 (FLASH ADC)
- Oscilloscopio Tektronix TDS 7254B

... more

- Readout time: ~300 μs (max rate: 3 kHz)
- Event size: fixed (soppressione zeri solo offline)
- Max nr di eventi limitato dall'oscilloscopio (buffer interno)
- Dati oscilloscopio trasferiti su disco a end-of-spill e "infilati" evento per evento nel buffer di readout
- ogni 10 eventi, 1 evento di "pedestallo"

...

monitoring ...

Monitoring

- processo a piu' bassa priorita'
- non deve introdurre tempo morto add.
- analisi su base statistica (campionamento)

→ istogrammazione (ROOT)