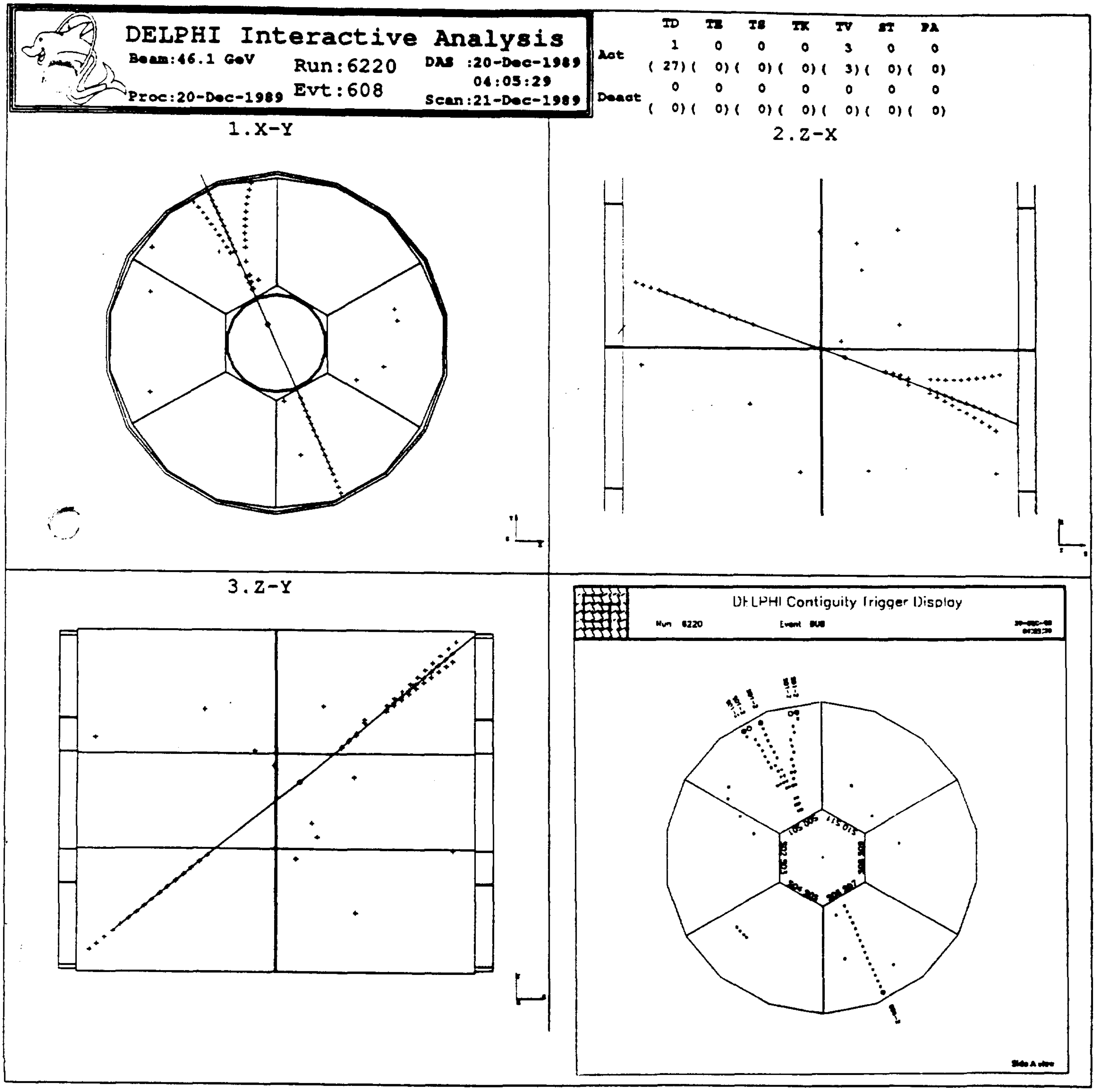
IdG 24/5/90 P251-1 8/5/90 IEEE 18/11/90

33

A TPC event: (tau-candidate)

- (1) TPC XY display, (2) TPC XZ display, (3) TPC YZ display.
- (4) Contiguity Trigger display:
  dots: Image Memories,
  circles: Contiguity Mask Network results (found tracks) .



DELPHI Interactive Analysis

Beam:46.1 GeV Run:6220 DAS :20-Dec-1989
04:05:29
Proc:20-Dec-1989 Evt:608 Scan:21-Dec-1989

	TD	TE	TS	TK	TV	ST	FA
Act	1	0	0	0	3	0	0
	(27)	(0)	(0)	(0)	(3)	(0)	(0)
Deact	0	0	0	0	0	0	0
	(0)	(0)	(0)	(0)	(0)	(0)	(0)

TRIGGER: T1 AT LHC

PROBLEM(S):

- BCO INTERVAL : 15 ns
- ON AVERAGE 30 INTERACTIONS / BCO !!

  ↓ NOTHING ONE CAN DO

SOLUTION FOR 15 ns : PIPELINING

1- PIPELINING DATA

DATA MUST WAIT FOR T1 DECISION
(I.E. ≃ 1000 BCOS IN THE PIPE)

2- PIPELINING TRIGGER

TRIGGER IMPLEMENTED IN STEPS OF 15 ns
(OR IF "FARM OF TRIGGER", STEPS OF N# 15 ns

ANALOG PIPELINING ? DIGITAL PIPELINING ?

BIG ISSUE : TIMING CONTROL
SYNCHRONISATION

(15 ns IS 3 m OF CABLE !)

TRIGGER: H/W (Cont'd)

- CENTRAL DECISION LOGIC
    - LOOK UP TABLES

THE IDEA IS TO USE N BOOLEAN INFORMATIONS TO MAKE A SINGLE DECISION: YES/NO

→ USE A RAM OF 2^N BITS

EXAMPLE: $N=3$

EM	μ	TR

∅	∅	∅	No track, no EM
∅	∅	1	Track, no μ
∅	1	∅	μ, track inefficient
∅	1	1	μ track
1	∅	∅	EM, no track
1	∅	1	EM, track, no μ
1	1	∅	EM, μ, track ineffic
1	1	1	EM, μ track

HENCE TO SET A TRIGGER E.G ON "SINGLE PHOTONS":

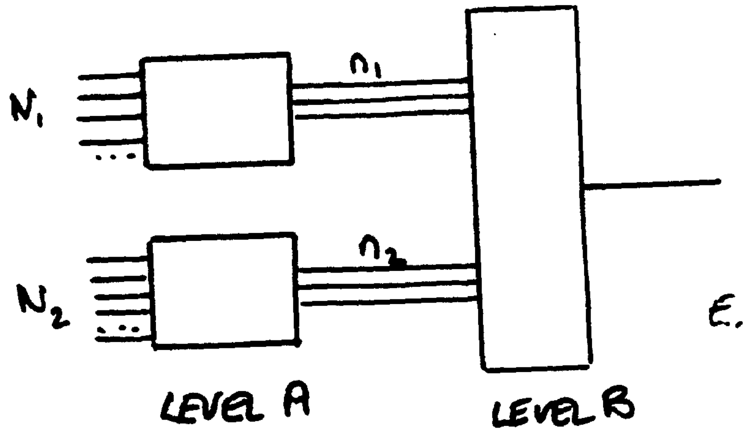
∅, ∅, ∅, ∅, 1, ∅, ∅, ∅

ON "INCLUSIVE μ's":

∅, ∅, 1, 1, ∅, ∅, 1, 1

→ PROBLEM: IF N IS LARGE, THIS BECOMES PROHIBITIV

⇒ USE SEVERAL CHAINED L.U.T:



MEMORY NEEDED:

$$n_1 2^{N_1} + n_2 2^{N_2} + 2^{n_1+n_2}$$

E.G IF $N=20$

$N_1=10$, $N_2=10$, $n_1=n_2=4$

$$4.2^{10}+4.2^{10}+2^{8}=8448$$

INSTEAD OF 1,048,576 (1 Mbi

The Central Decision Module

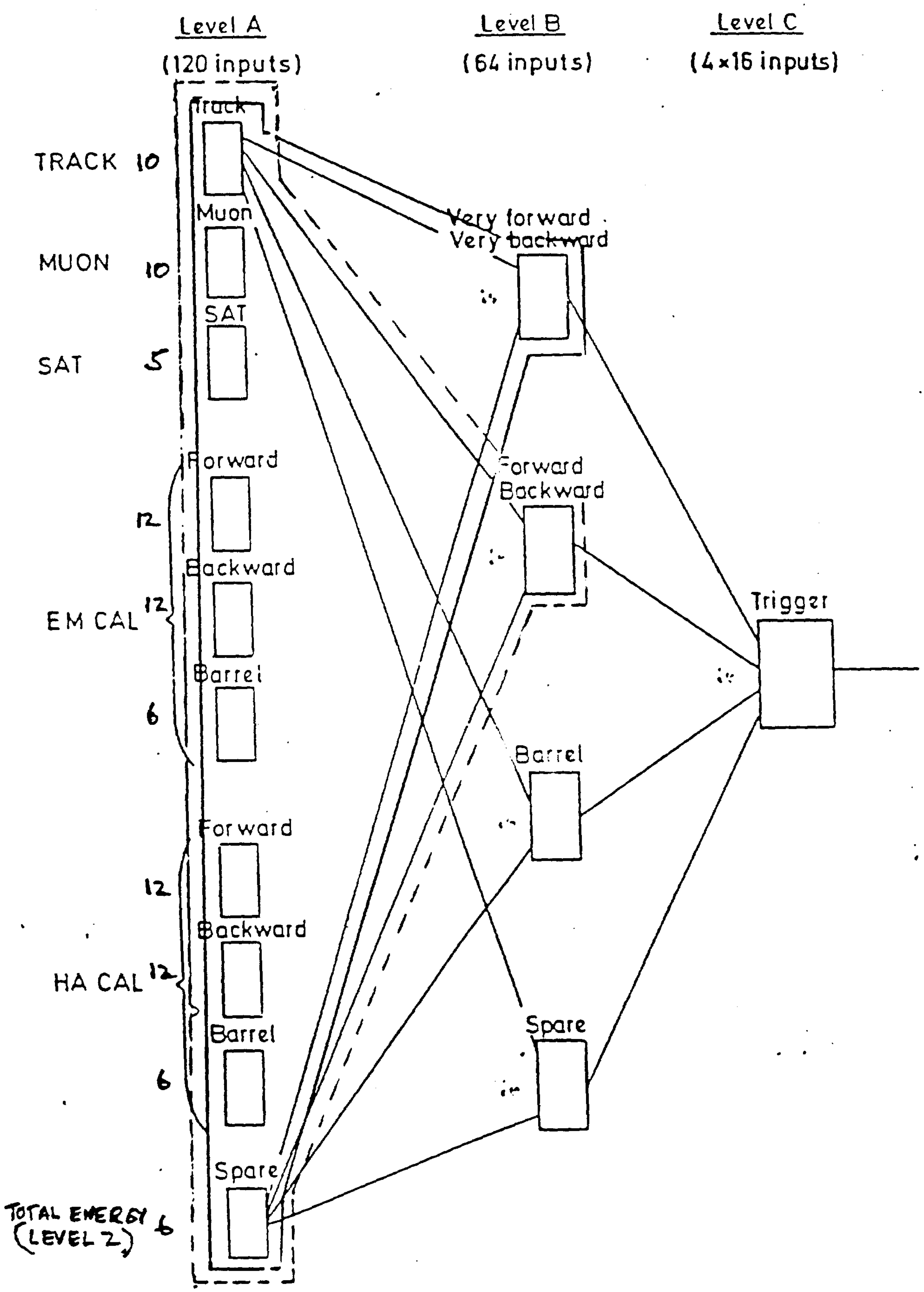


Fig. 2

TRIGGER : S/W TRIGGERS

MAIN CHARACTERISTICS: • ASYNCHRONOUS WITH THE EVENT

• EMBEDDED INSIDE THE DAQ SYSTEM

• USE EMDEDDED PROCESSORS

SOME DECISION IS MADE BY THE READOUT PROCESSORS THEMSELVES, OR BY A DEDICATED PROCESSOR, UNDER CONTROL OF THE READOUT

EXAMPLES: DELPHI T3 → 68000/68020 EMBEDDED PROCESSOR

DELPHI T4 }
L3 T3 } → EMULATORS INTERFA TO FASTBUS

• USE "FARMS" CONNECTED TO THE MAIN DAQ COMPUTE

- FARMS OF μProcessors
- FARMS OF μVAX CPUs (ALEPH T3)
- FARMS OF APOLLOS (OPAL: DN10000 NAWL FOR TAGGING)

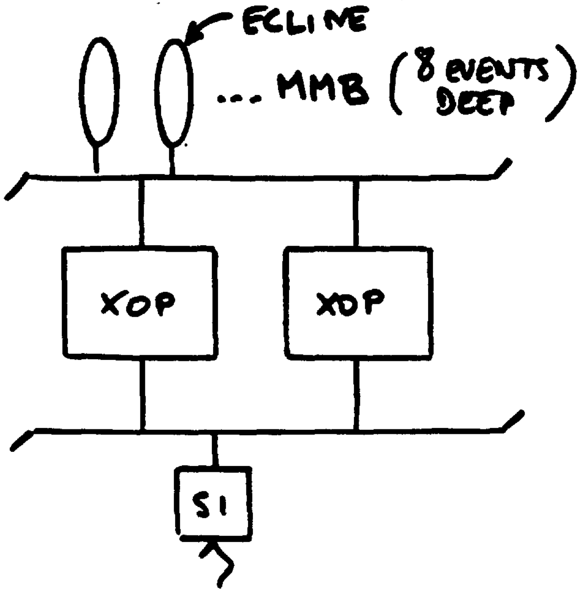
- FAST DECISION PROCESSORS

INBETWEEN "H/W" & "S/W" TRIGGERS

CHARACTERISED BY :

- HIGH SPEED (~ 1 ms)
- LOW LEVEL OF PROGRAMMATION
- DIRECT INTERFACE TO H/W ON INPUT SIDE

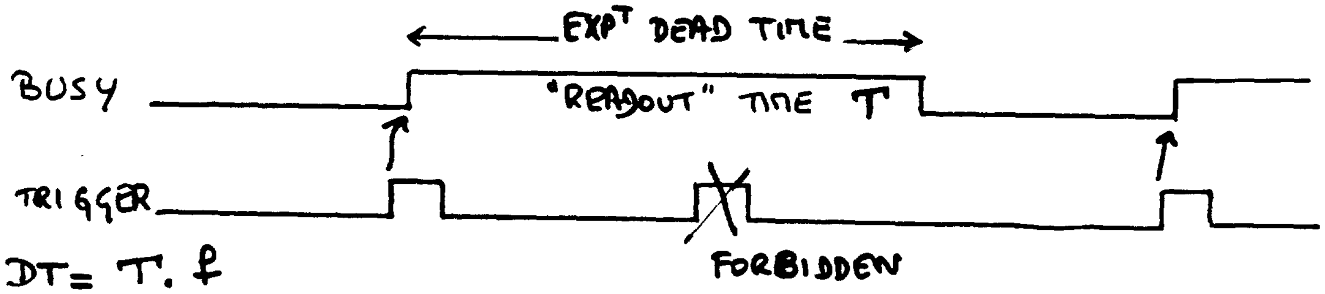
TYPICAL EXAMPLE: XOP PROCESSORS FOR LEVEL 2 IN L3



- OPERATING ON A STREAM ≠ FROM MAIN DATA STREAM
- LOW NUMBER OF INFOS (< 256/MMB)
- VERY HIGH PERFORMANCE ON FASTBUS
  < 560 μs TO FETCH A FULL EVENT
- FAST DECISION TIME
  ≃ 10 MS

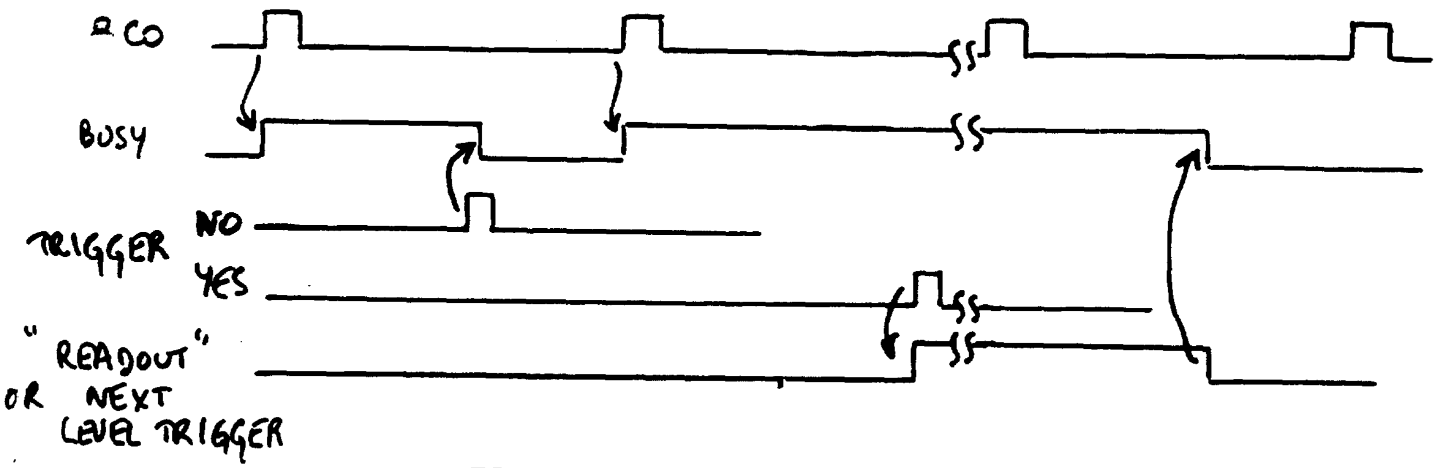
Trigger : Control

Goal :
- Transmit to Front End (FE) electronics timing signals & trigger decisions
- Handle the locking of triggers while FE's are busy
- Provide detectors with standalone test facilities



$DT = T \cdot f$ ↳ trigger frequency

Dead time on a DC machine
(e.g. fixed target exp^T)



$$DT = \sum_{\text{Levels}} (\text{Number of BCO's lost})_{\text{level } i} \cdot f_i \Big/ \text{BCO\_FREQUENCY}$$

Dead time on a bursted machine
(e.g. LEP)

TRIGGER : CONTROL (CONT'D)

EXAMPLES :

ALEPH :
$$DT = \left(\overset{\text{BCOs lost at T1-YES}}{4}\cdot f_1 + \overset{\text{BCOs lost at T2-YES}}{1000}\cdot f_2\right)/\underset{\text{BCO frequency}}{44500}$$

With $f_1 = 8\ \mathrm{Hz}$, $f_2 = 1.5\ \mathrm{Hz}$:

$$DT = 3.4\%$$

DELPHI:
$$DT = (1.\, f_1 + 300.\, f_2)/44500$$

With $f_1 = 12\ \mathrm{Hz}$, $f_2 = 1.5\ \mathrm{Hz}$

$$DT = 1.0\%$$

IF THE BCO FREQUENCY INCREASES, NO CHANGE I THE DEADTIME, EXCEPT WHEN BCO TIME BECOMES CLOSE TO T1 DECISION TIME ....

TRIGGER CONTROL DEVICES

- HIGHLY EXPERIMENT DEPENDENT.
- RELY ON A WELL DEFINED "TRIGGER PROTOCOL"
- FOLLOW THE HIERARCHICAL STRUCTURE OF THE EXPERIMENT

CENTRAL UNIT (MAIN TRIGGER SUPERVISOR)

- RECEIVES MACHINE TIMING SIGNALS :
  - WARNING EJECTION FOR FIXED TARGET
  - BCO FOR COLLIDER
- CONNECTS TO THE CENTRAL TRIGGER DECISIO
- HANDLES DETECTOR'S BUSY SIGNALS
- SENDS TIMING SIGNALS + SYNCHRO TO LOCAL UNITS

TRIGGER: CONTROL (CONT'D)

LOCAL UNITS (LOCAL TRIGGER SUPERVISORS)

- RECEIVES C.U. SIGNALS
- TRANSMITS TO LOCAL READOUT PROCESSORS
- HANDLES LOCAL BUSY PROTOCOL & TRANSMITS TO CENTRAL UNIT
- ALLOWS TO RUN LOCAL DETECTOR IN STANDALONE MODE (TEST TRIGGERS)

READOUT PROCESSORS

- RECEIVES L.T.U. SYNCHRO SIGNALS
- REPORT BUSY STATUS TO L.T.U

THIS LINKS TO THE NEXT TOPIC OF THESE LECTURES:

DATA ACQUISITION

DAQ

- DAQ ARCHITECTURE

  THIS IS A SECRET : "ALL LEP DETECTORS HAVE GOT THE SAME !"

- DAQ IMPLEMENTATION

  OR : WHERE THE EXPERIMENTS REALLY ARE DIFFERENT

  WHEN HARDWARE IS CLOSER TO ARTISTIC CREATION THAN ANYWHERE ELSE

  OR : HOW TO MAKE A CHOICE ?

DAQ : GOALS

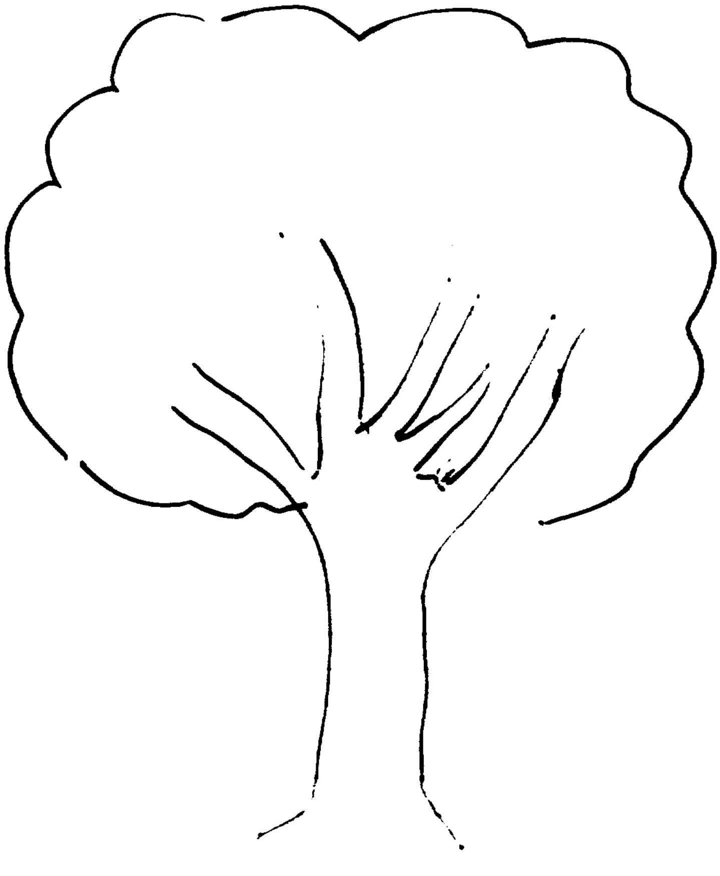
* GATHER INFORMATIONS FROM ALL DETECTOR

- AS FAST AS POSSIBLE
- AS COMPACT AS POSSIBLE
- AS RELIABLY AS POSSIBLE

* TRANSFER THEM ONTO A MASS STORAGE MEDIUM

THE PROBLEM IS ALWAYS THE SAME
THE SOLUTIONS ARE ALL DIFFERENT

... BUT THE BASIC IDEAS ARE ALL THE SAME



DAQ: ARCHITECTURE

1) LEVELS

- IDENTIFY SEGMENTATION LEVELS IN THE TREE

  * FRONT END ELECTRONICS

  - RECEIVES DIRECTLY SIGNALS FROM THE DETECTOR + TRIGGER & TIMING SIGNALS
  - PRODUCES DIGITIZED INFORMATION
  - DON'T FORGET TRIGGER MODULES

  [* CRATE OR CRATE CLUSTER LEVEL]

  NEEDED IF TOO MANY FE MODULES

  * DETECTOR LEVEL

  - MERGES DATA FROM SEVERAL FE/CRATES

  → USUALLY THERE IS A NEED TO OPERATE ALSO EACH DETECTOR INDEPENDENTLY OF OTHERS + MONITORING OF DATA

  * CENTRAL DAQ

  - MERGES DETECTORS DATA → BUILDS EVENTS
  - PERFORMS T3/T4 REJECTION
  - LOGS DATA ONTO MASS STORAGE

  ** DAQ CONTROL

  - PERFORM THE OVERALL CONTROL OF DAQ (CONFIGURATION, RUN CONTROL, LOGGING CONTROL, ERROR REPORTING)

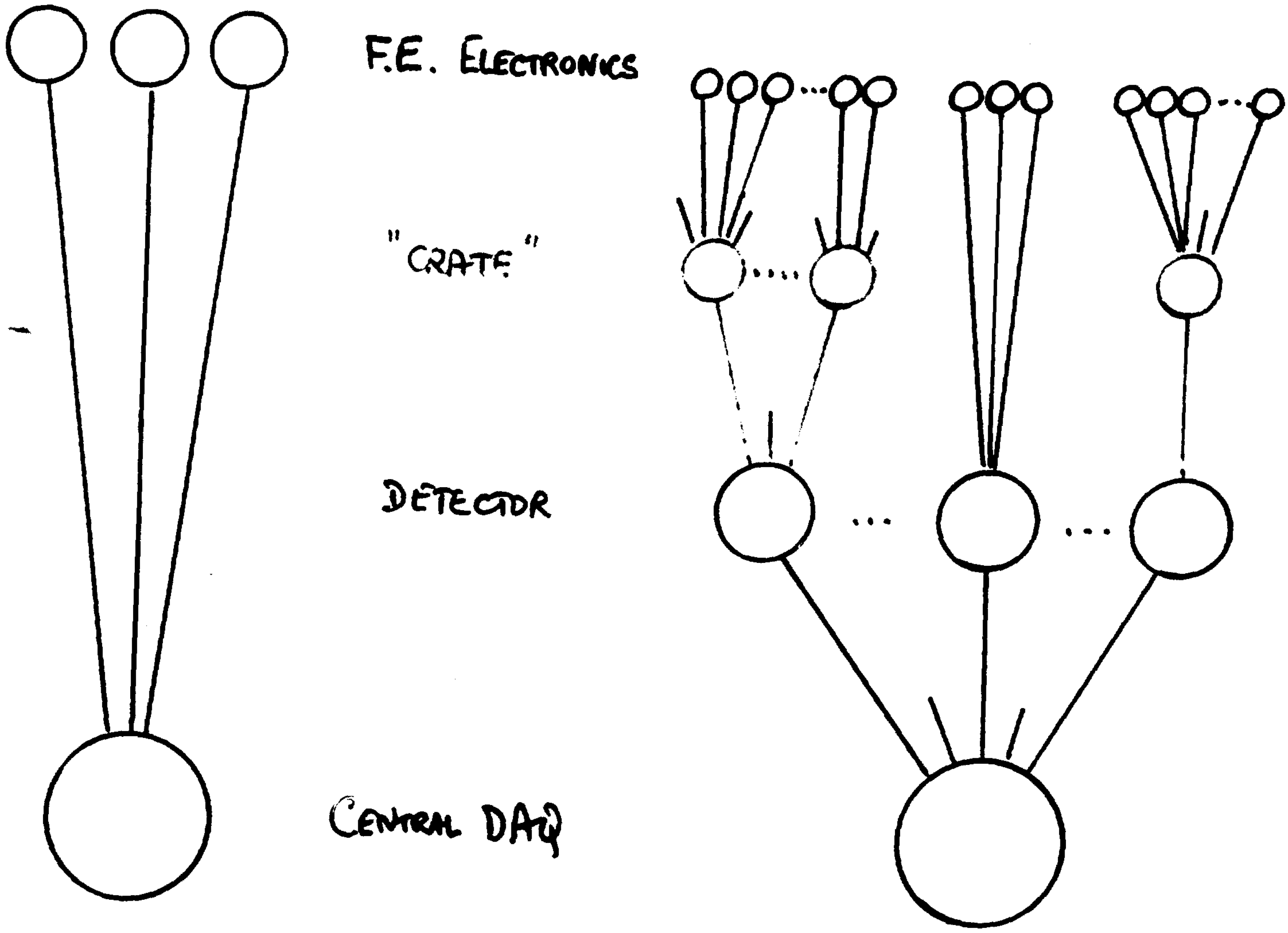
DAQ : ARCHITECTURE (CONT'D)

- EXAMPLES
  - SMALL EXPERIMENT (FEW CAMAC/VME/FB CRAT

    2 LEVELS : 1) FE
    2) CENTRAL DAQ

  - LEP EXPERIMENTS

    4 LEVELS
    1) FE
    2) CRATES (NOT FOR ALL DETECTORS)
    3) DETECTOR
    4) CENTRAL DAQ



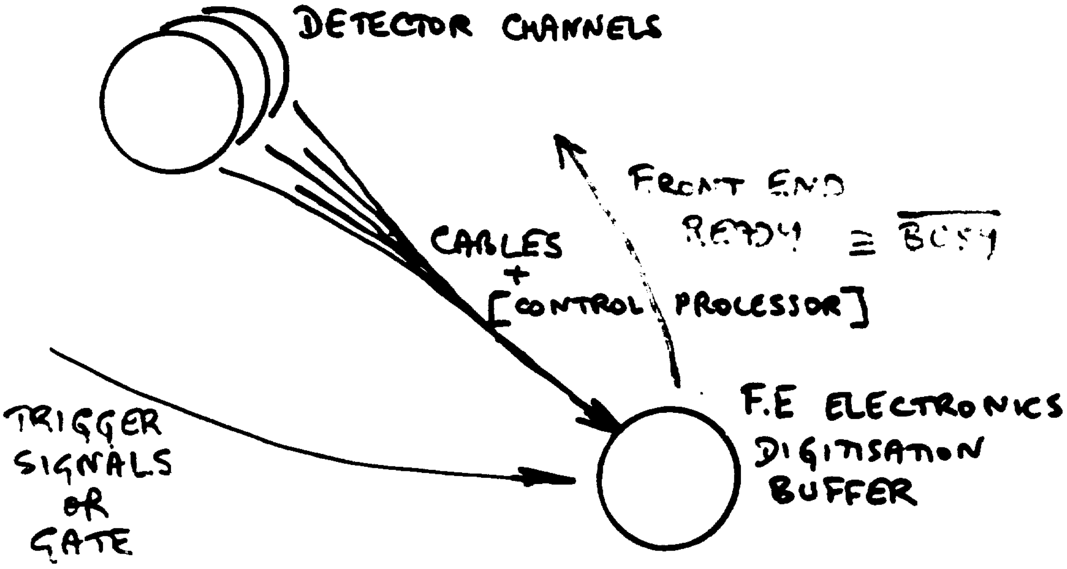
DAQ : ARCHITECTURE

2) ELEMENTARY CELL

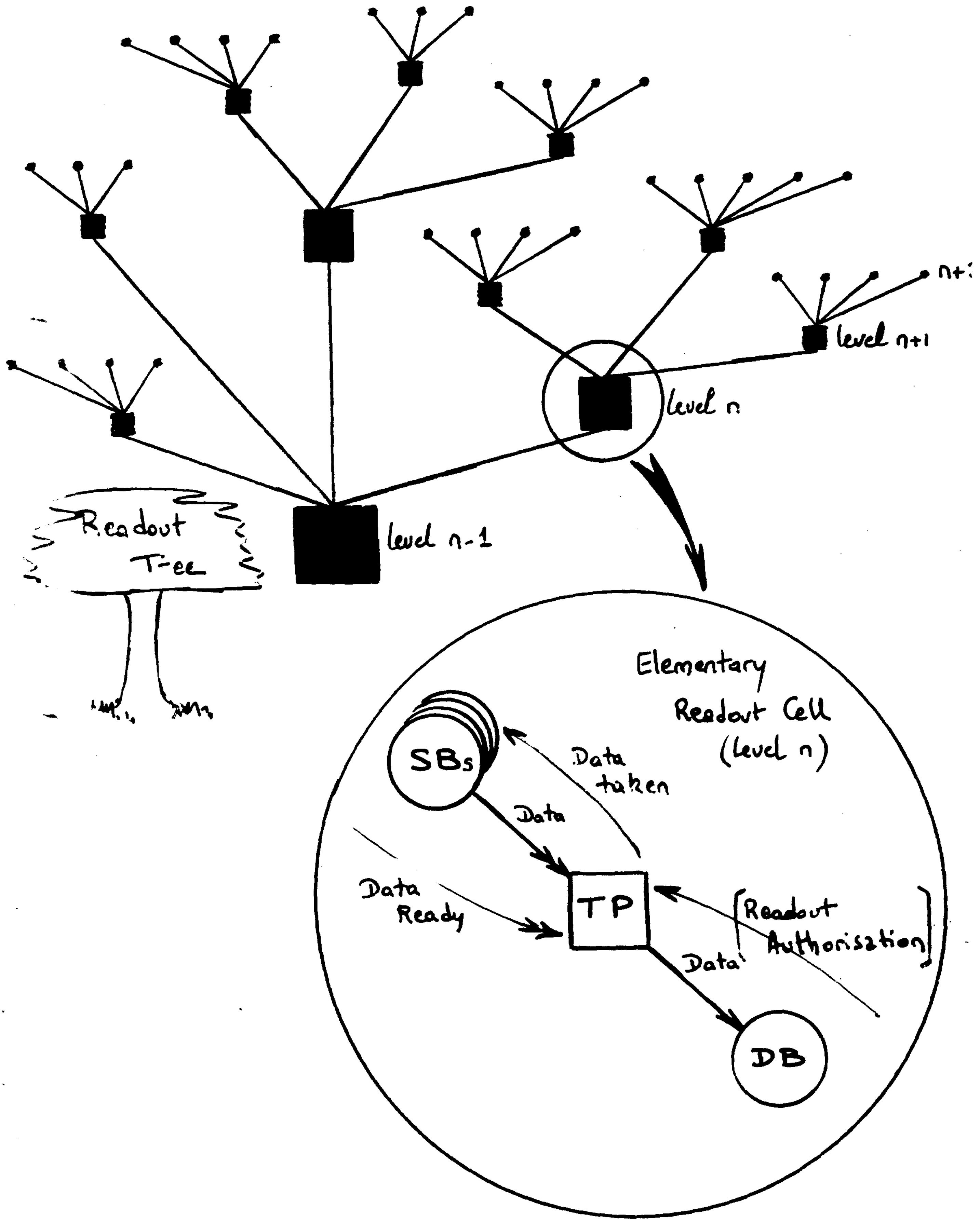
- AT EACH LEVEL, THERE IS ONE ELEMENTARY CELL WHICH CAN BE DESCRIBED IN TERMS OF
  - BUFFERS (Source Buffer /Destination Buffe
  - TRANSFER PROCESSOR (TP)
  - CONTROL MESSAGES (Data-Ready / taken/[Author.

*(next transparency)

- SPECIAL CASE OF FE LEVEL
  - SAME PATTERN
  - NO "UPSTREAM" PROCESSOR
  - NOT NECESSARILY A "TRANSFER PROCESSOR"


DETECTOR CHANNELS
FRONT END READY ≡ $\overline{\text{BUSY}}$
CABLES + [CONTROL PROCESSOR]
TRIGGER SIGNALS OR GATE
F.E ELECTRONICS DIGITISATION BUFFER

Control of Transfers

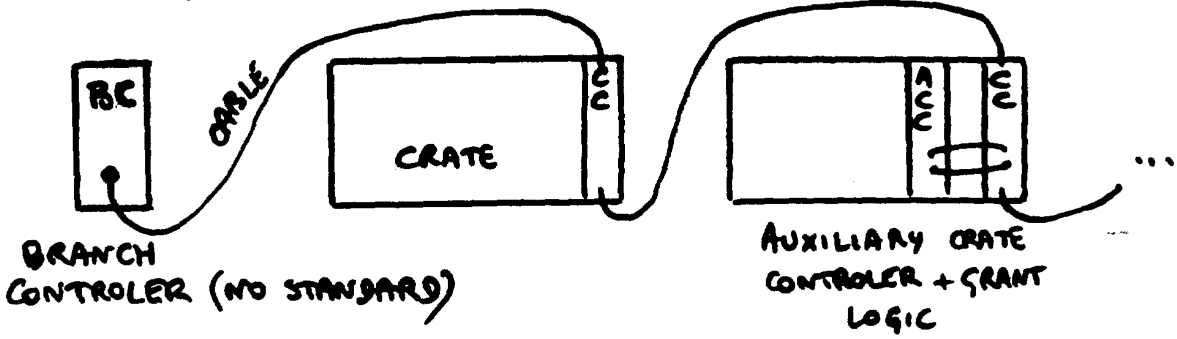


DAQ : H/W IMPLEMENTATIO

1) CHOICE OF (A) STANDARD(S)

- **CAMAC** (FROM EARLY '70s)

BASICALLY A 2 LEVEL BUS STANDARD


BC
CABLE
CRATE
CC
ACC
CC
...
BRANCH CONTROLER (NO STANDARD)
AUXILIARY CRATE CONTROLER + GRANT LOGIC

- SLAVE MODULE
- A SLAVE MAY BE AUXILIARY CRATE CONTROLER BUT NO CRATE TO CRATE LINK
- A SLAVE MAY BE BRANCH CONTROLER
- NO BUS RESERVATION, NO SLAVE LOCKING
- SMALL SIZE FOR BOARDS
- LIMITED TO 7 CRATES ON A BRANCH, 23 STATIONS IN A CRATE

PERFORMANCES : ≃ 3 µs FOR 24 bits

- NO REAL "BLOCK TRANSFER" MODE DEFINED IN THE SPECS

USE : VERY ROBUST SYSTEM STILL IN USE IN MANY SMALL/MEDIUM SIZE EXP^{ts}.

USED FOR SPECIFIC F.E. IN SEVERAL LEP EXP^{ts}

AVAILABILITY : • LARGE VARIETY OF F.E. MODULES

- INTERFACED TO MANY DAQ COMPUTERS
- SOME µ-PROCESSORS ACC (BIT SLICE/68

DAQ : H/W - BUS STANDARD

• VME (≈ '80)

ORIGINALLY SINGLE CRATE BUS STANDARD FOR INDUSTRIAL CONTROL SYSTEMS

- 32 BIT BUS
- MULTI SLAVE / MASTER BUS
- SECONDARY BUS (USB)

PERFORMANCES : ≃ 16 Mb/s (250 ns/32 bit WORD)

AVAILABILITY OF MODULES :

VERY LARGE VARIETY OF MODULES:

- CPU BOARDS : 68K, 68020, 68030
- MEMORIES : RAM UP TO 16 Mb
- ETHERNET INTERFACES
- DISK INTERFACES
- HOST COMPUTER INTERFACES (VAX, MacIntosh...)
- INTERFACES TO OTHER BUSES (CAMAC, FASTBUS
- F.E. ELECTRONICS : LESS THAN CAMAC, BUT EASY TO DESIGN + BUILD DEDICATED BOARDS

CRATE INTERCONNECTION :

NOT PART OF THE STANDARD ORIGINALLY

⤳ SEVERAL "HOME MADE" IMPLEMENTATIONS (e.g. VIP FOR OPAL)

USE : UA1 } MAINLY FROM "DETECTOR LEVEL" ONWARDS
OPAL

<u>DAQ</u> : H/W - BUS STANDARD

- <u>FASTBUS</u> (END OF '80's)

DESCRIPTION

- 32 BIT BUS (ADDRESS & DATA MULTIPLEXED)
- MULTI-SEGMENT (CRATES & CABLES)
- MULTI-MASTER (WITH BUS RESERVATION + SLAVE LOCKING)
- BLOCK TRANSFER ORIENTED
- TWO SPACES SLAVES (DATA + CONTROL)
- COMPLEX H/W SPECIFICATIONS
- S/W SPECIFICATION (NOT MANDATORY)
- LIMITED (I.E. POOR) IN CONTROL SIGNALS (ONLY <u>ONE</u> SERVICE REQUEST LINE + LIMITED INTERRUPT MESSAGE SYSTEM)

PERFORMANCES

- CRATE INTERNAL TRANSFERS ≃ 40 Mb/sec MA
  USUAL IMPLEMENTATIONS: ≃ 16 Mb/s in B.T
- SEGMENT TO SEGMENT TRANSFERS :
  ADD : 2 * 20 ns FOR EACH Segment Inter
  2 * 6 ns FOR EACH METER OF CA

e.g. FOR 2 SIs + 50 m of cable + 250 ns SLA
≃ 4 Mbytes/s

AVAILABILITY OF MODULES

RATHER FEW COMMERCIAL MODULES :

- SI + ALC + ANC (ANCILLARY LOGICS)
- μP boards : AEB, GPM, FIP
- FE Electronics : Almost all custom design

FASTBUS (CONT'D)

ADVANTAGES:

- LARGE BOARDS (LIMITS THE NUMBER OF INTERFACE FOR A GIVEN NUMBER OF CHANNELS
- FAST BLOCK TRANSFERS
- ALLOWS COMPLEX ARCHITECTURES (NOT ONLY TREES

DISADVANTAGES:

- LARGE BOARDS → EXPENSIVE FOR SIMPLE APPLICATIONS
- NOT THAT FAST FOR SINGLE WORD READ/WRITE + LIMITED CONTROL SIGNALS
- ALLOWS COMPLEX ARCHITECTURES (BUT WE USUALLY DO NOT NEED IT, TREES ARE SUFFICIENT)
- LARGE S/W OVERHEAD FOR ERROR & BUS CONTENTION CASES

- FUTUREBUS, SUPERBUS, OMNIBUS ....

IN THE COMING YEARS, THIS FIELD WILL EVOLVE MORE RAPIDLY THAN IN THE PAST 20 YEARS, TAKING ADVANTAGE OF:
- NEW TECHNOLOGY
- EXPERIENCE

I HOPE THAT A BUS WILL COME OUT MERGING ALL ADVANTAGES OF EXISTING BUSES, ESPECIALLY COMMERCIAL AVAILABILITY OF MODULES

CHALLENGE OF THE 90'S

DAQ : H/W - BUS STANDARD

- HOW TO CHOOSE ?

  1) THERE IS NO TRUTH

  2) THERE ARE FASHIONS, PREFERENCES, SUBJECTIVITY

ALTHOUGH FOR LARGE EXPERIMENTS AN HYBRID SYSTEM SHOULD BE THE BEST, TRY AND MINIMISE THE NUMBER OF STANDARDS.

• UA1	:	CAMAC FOR F.E.
		VME FOR CENTRAL DAQ
• OPAL	:	CAMAC, FASTBUS, VME FOR F.E
		VME FOR DETECTOR LEVEL + CENTRAL D
• ALEPH	:	FASTBUS
• L3	:	FASTBUS
• DELPHI	:	FASTBUS + A FEW CAMAC CRATES FOR F.E.
• UA2	:	FASTBUS

DON'T FORGET : WHATEVER YOU CHOOSE, YOU WILL
REGRET IT AT LEAST ONCE !

DAQ : H/W IMPLEMENTATION

2) CHOICE OF MODULES

COMPROMISE BETWEEN FUNCTIONALITY, LONG TERM AVAILABILITY/MAINTENANCE, MANPOWER ... AND PRICE

- CAMAC, VME : * CHOOSE ON THE PRICE LISTS
  * BUILD WHAT IS MISSING

- FASTBUS : * ALSO LOOK IN THE PRICE LISTS... ... BUT THEY ARE RATHER POOR
  * DEFINE VERY PRECISELY YOUR NEEDS IN TERMS OF FUNCTIONALITY - IN PARTICULAR FOR μ-PROCESSOR BASED MODULES :
    - FASTBUS FUNCTIONS (SLAVE, MASTER, SINGLE OR MULTI-SEGMENT
    - OPERATING SYSTEMS SUPPORTED BY THE μP.
    - LANGUAGES " " " "
    - MEMORY SIZE (EPROM + RAM)
    - SECONDARY CONNECTIONS TO THE OUTSIDE WORLD (RS232, ETHERNET ...)
    - FASTBUS HANDLING (IN PARTICULAR ERROR CONDITIONS + MULTITASKING ASPECTS)

↓

FB IS A SINGLE USER BUS : BE CAREFUL WITH MULTITASKING OPERATING SYSTEMS

HERE ALSO: THERE IS NO TRUTH

ANYWAY : SINCE IN SOFTWARE, EVERYTHING IS FEASIBLE!..

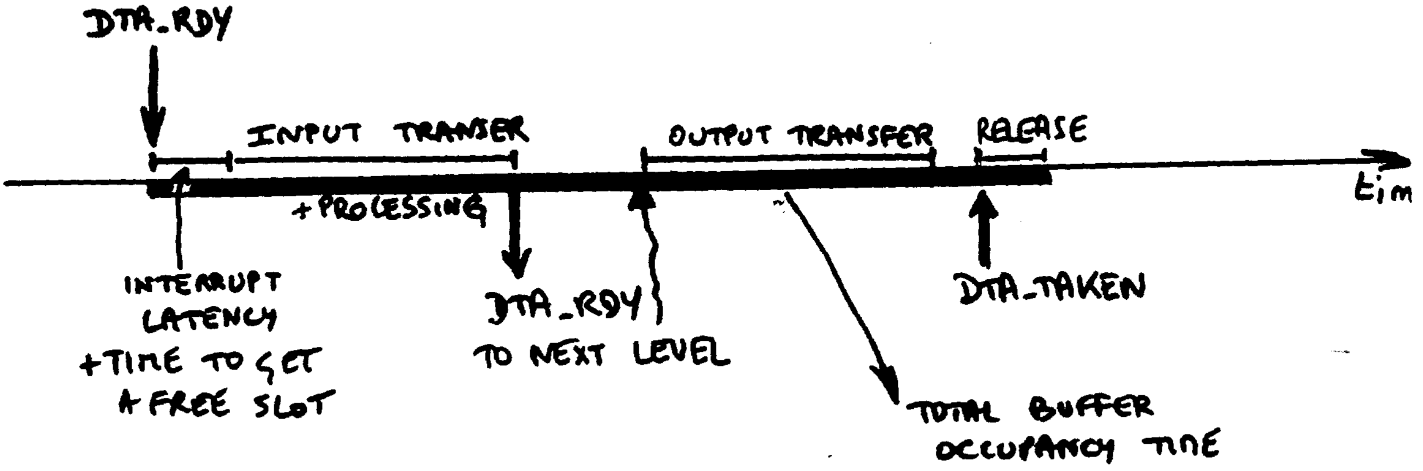
DAQ : H/W IMPLEMENTATI

3) BUFFER SIZES (OR : "TIME IS MONEY")

THE SIZE OF EACH BUFFER (I.E. THE NUMBER OF EVENTS WHICH IT CAN CONTAIN) DEPENDS MAINLY ON THE TIME THE EVENT STAYS THE

BUFFER OCCUPANCY TIME :

IT IS THE TIME ELAPSED BETWEEN THE "RESERVATION" OF THE BUFFER SLOT AND THE "RELEASE"



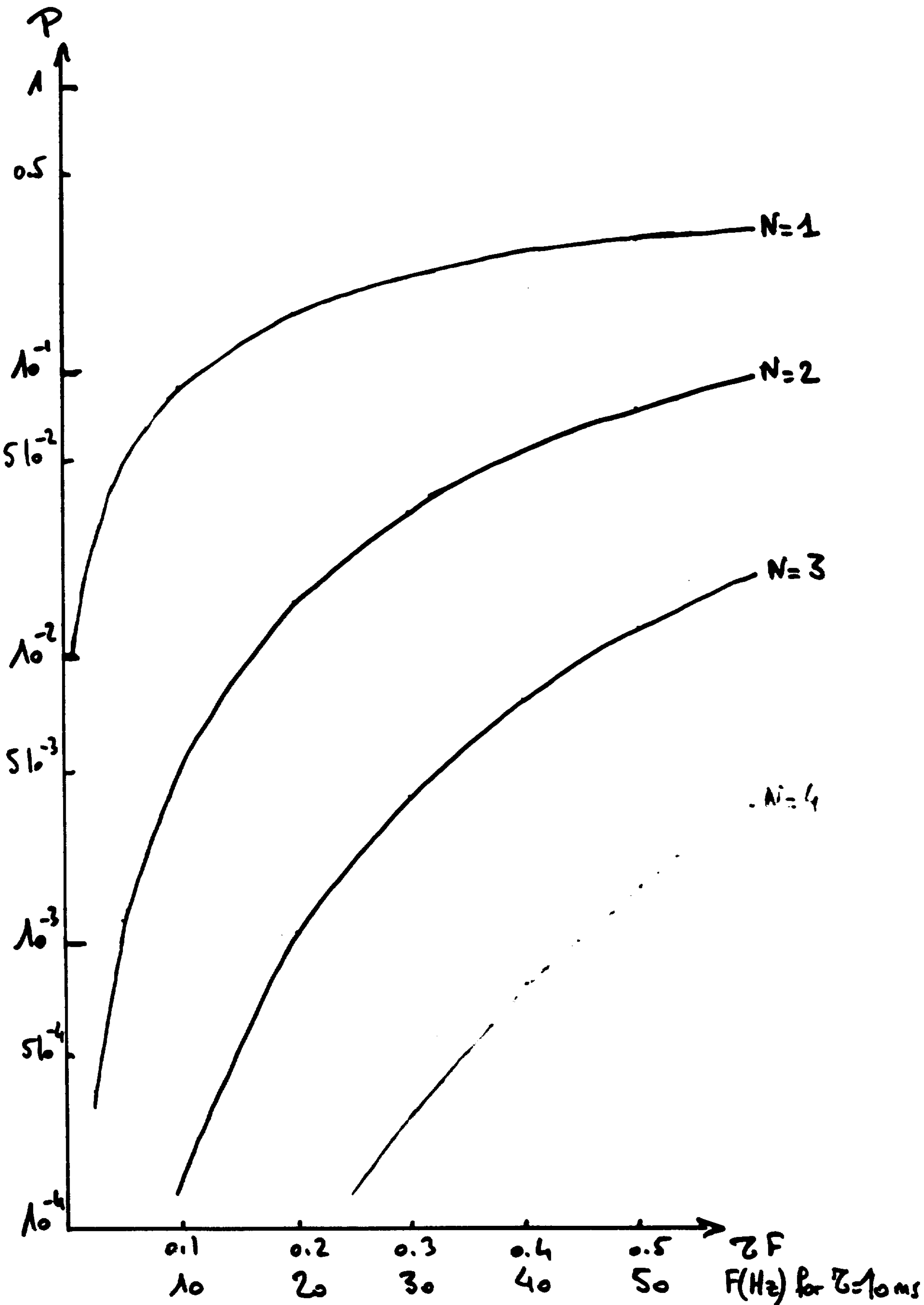
- AVERAGE CONDITION : $N^{max}_{events} > \tau_{mean} \cdot F_{input}$

MEAN OCCUPANCY TIME

INPUT FREQUENC

NECESSARY, BUT NOT SUFFICIENT! FAR FROM THAT

- ONE MUST "DERANDOMIZE" THE ARRIVAL TIMES IN ORDER TO COPE WITH "BURSTS" OF EVENTS



Probability of getting N events in time interval τ if the frequency is F

DAQ : H/W - BUFFER SIZE

ROUGH ESTIMATE OF HOW BIG BUFFERS MUST BE.

ONE STARTS HAVING TROUBLE WHEN N EVENTS COME IN BEFORE THE FIRST ONE HAS BEEN READ OUT, AND FILL THE BUFFER, HENCE BLOCKING THE "PRODUCER"

IF $1/F_{input}$ IS LARGE COMPARED TO τ_{mean}

THE NUMBER OF EVENTS COMING IN DURING A TIME τ FOLLOWS A POISSON DISTRIBUTION:

$$P(N) = e^{-\tau F} \frac{(\tau F)^N}{N!}$$

(SEE FIGURE)

THE EXACT ESTIMATION IS MORE COMPLEX, DEPENDS ON THE S/W IMPLEMENTATION, AND ON THE INPUT & OUTPUT TIMES ⟹ NEEDS SIMULATION (NOT EASY...)

SUMMARY : THE MORE TIME ONE USES TO TRANSFER EVENTS, THE LARGER BUFFERS MUST BE

⇓

TIME COSTS MONEY !

DAQ : S/W DESIGN

1) CHOICE OF AN OPERATING SYSTEM

IMPORTANT PROPERTIES TO TAKE INTO ACCOUNT:

1) USER FRIENDLYNESS
2) PACKAGES FOR COMMUNICATION (TCP-IP, RPC, CATS, TPU
3) SUPPORTED LANGUAGES (C, FORTRAN?, ASSEMBLER;
4) INTRINSIC PERFORMANCE (CONTEXT SWITCHING, INTERRUPT LATENCY ....).

AT LEP & LAA, MOST PROCESSORS HAVE A 68K FAMILY PROCESSOR

2 OPERATING SYSTEMS USED (MAINLY)

- MONICA (SINGLE TASK OS) L3, UA2
- OS9 (MULTITASK OS) ALEPH, DELPHI, OPAL

ADVANTAGES OF MULTITASKING:

1) EASY QUEUE HANDLING
2) SUPPORT FOR: RAM, FLOPPY, HARD DISKS
   REMOTE DISKS (E.G. OS9 + ALEPH NETWOR DISKS)
3) SUPPORT FOR COMMUNICATION TOOLS:
   - REMOTE LOGIN
   - FILE TRANSFER
   - RPC

   etc...

DAQ : S/W DESIGN

2) SYSTEM DESIGN

...DIFFERENT APPROACHES...

BUT FOR LARGE SYSTEMS, I CONSIDER AN SASD* - LIKE APPROACH AS VERY USEFUL

- COMMON DISCUSSION LANGUAGE
- SIMPLE DESCRIPTION OF THE NEEDS

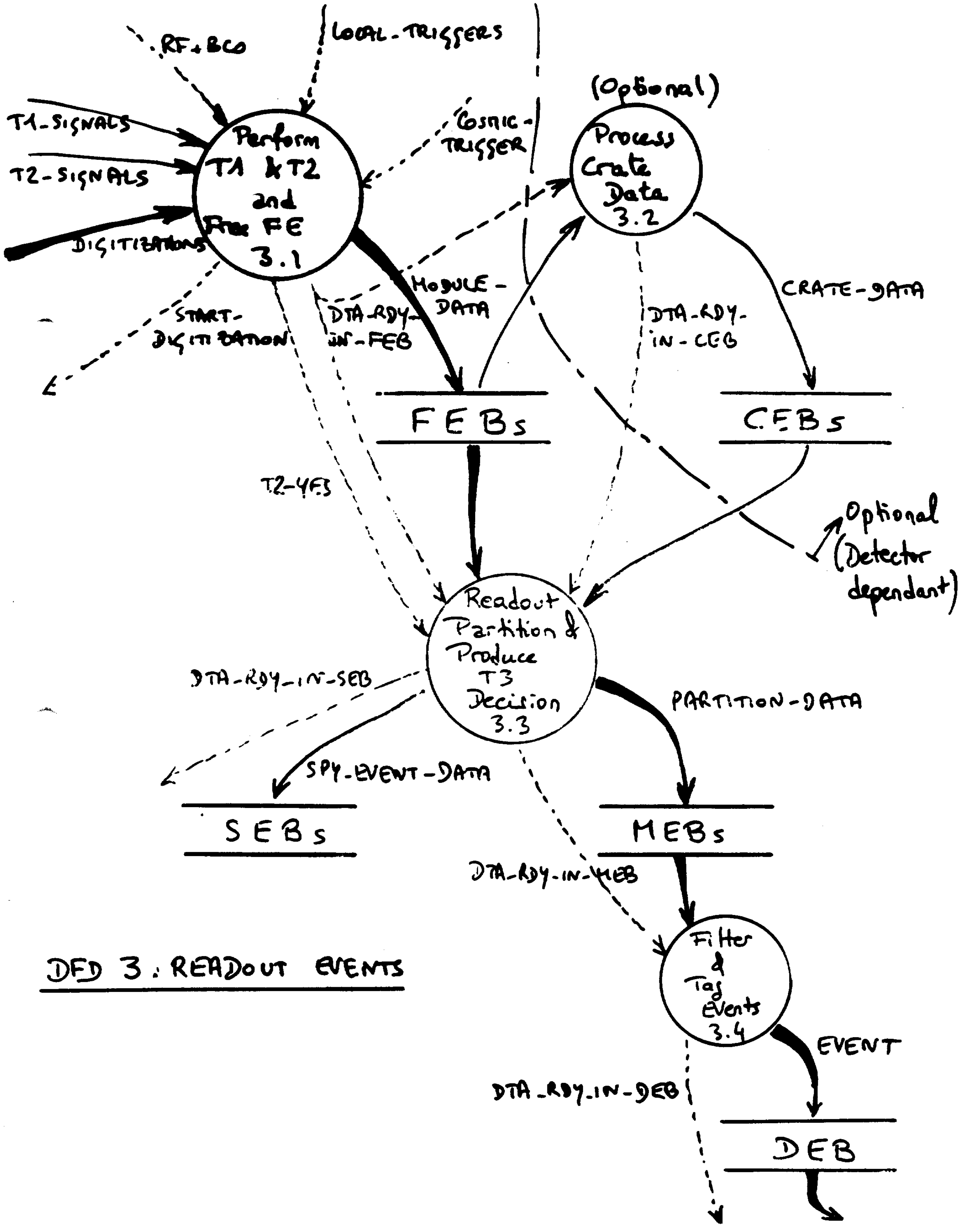
WHAT IT DOES NOT PROVIDE:

- DESCRIPTION OF COMMUNICATION PROTOCOL
- HOW TO STORE/RETRIEVE INFORMATION
- HOW TO WRITE EFFICIENT CODE
- HOW TO WRITE BUG FREE CODE

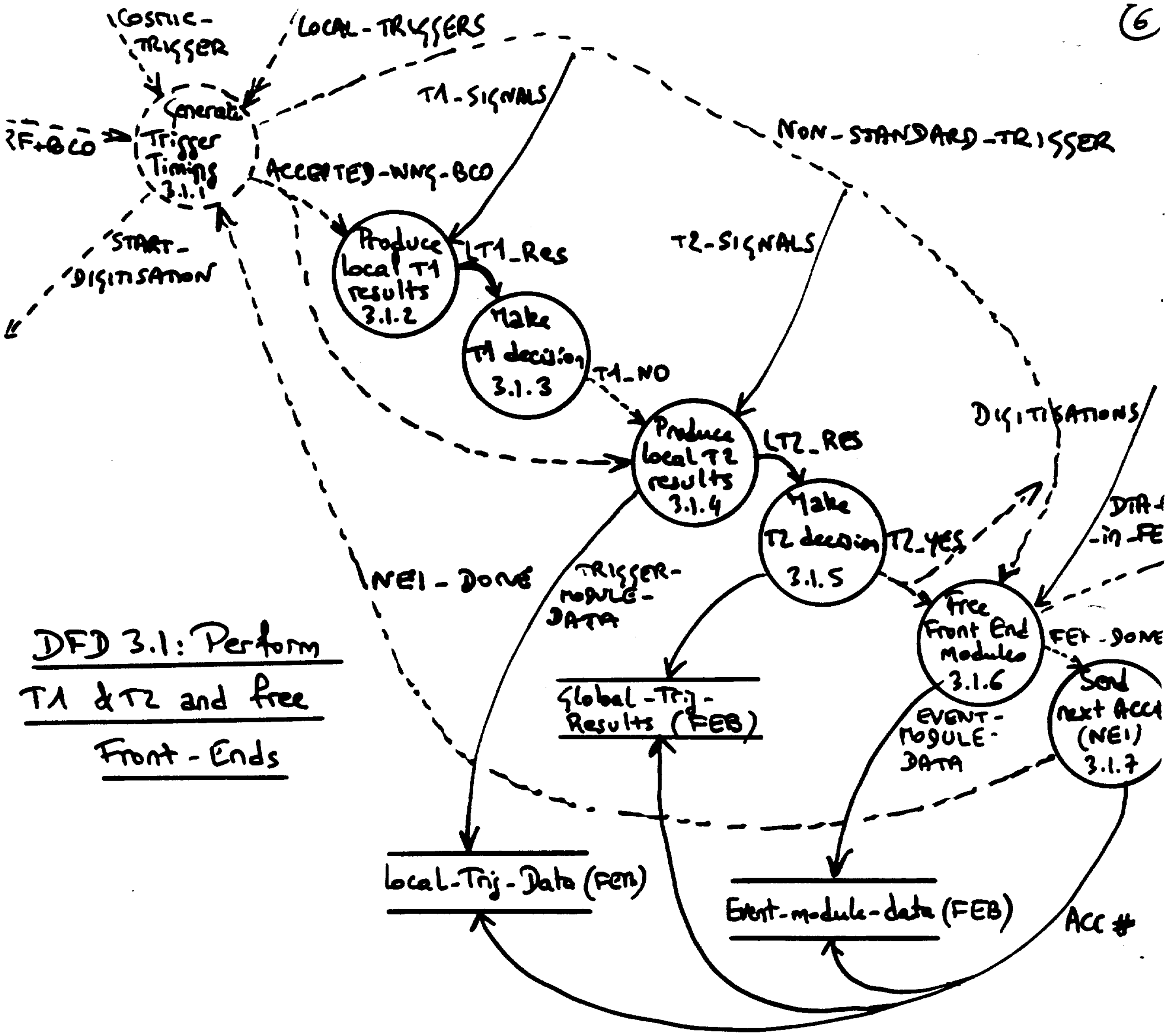
WHAT IT GIVES YOU AS A CONSTRAINT:

- CONTINUOUSLY UPDATE DESCRIPTION (IF POSSIBLE BEFORE MODIFYING)

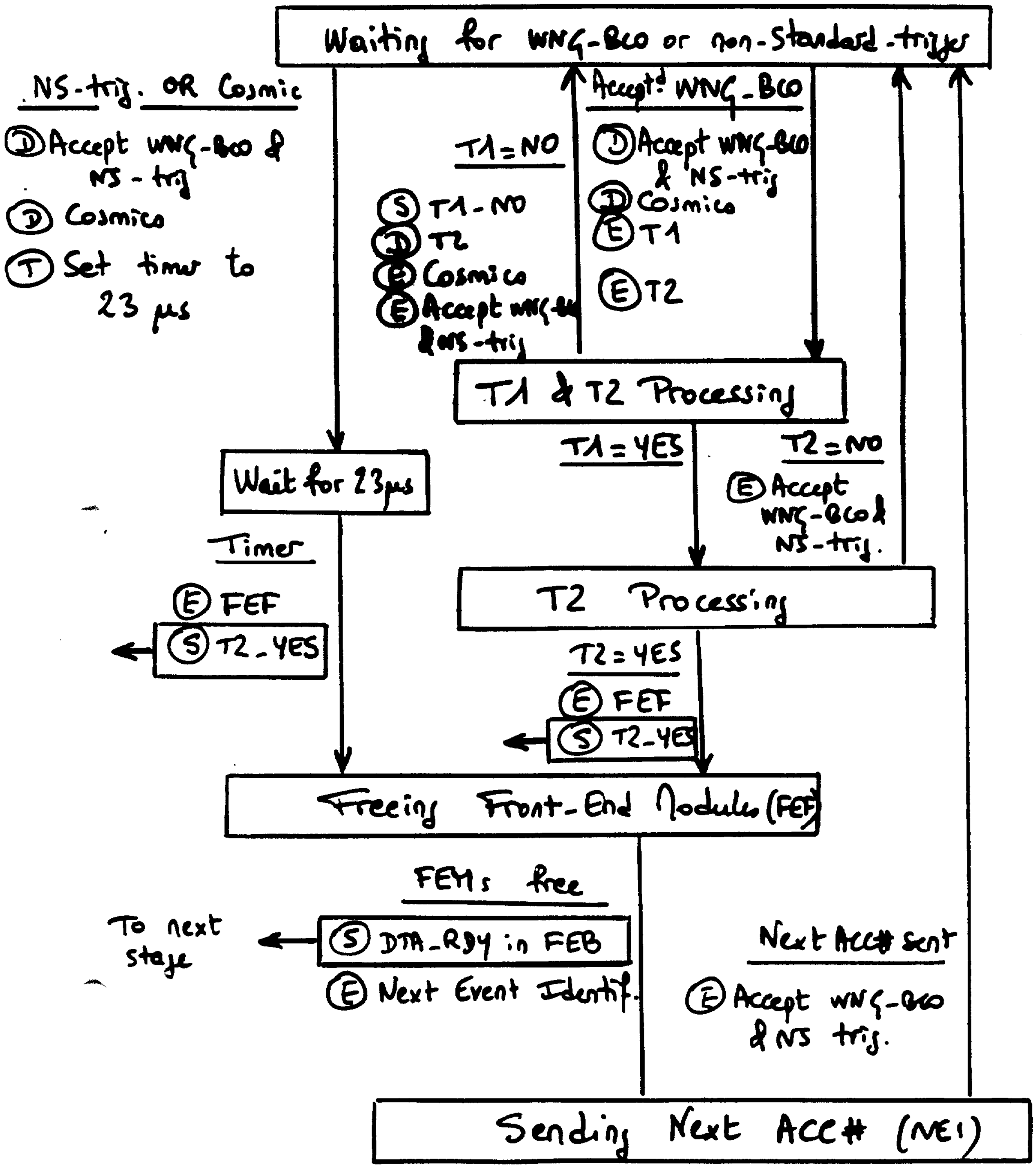
*STRUCTURED ANALYSIS & STRUCTURED DESIGN



DFD 3: READOUT EVENTS



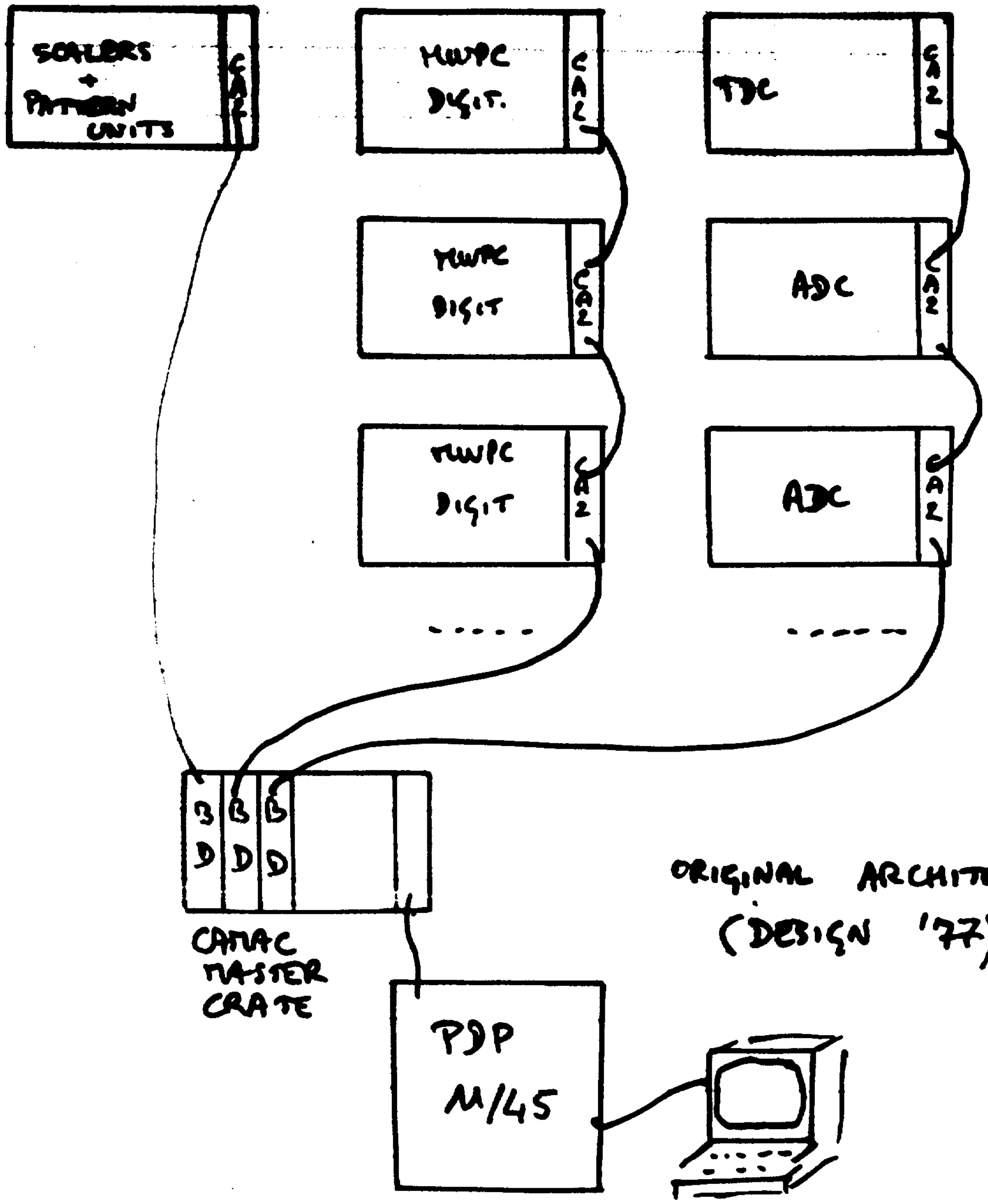
DFD 3.1: Perform T1 & T2 and free Front-Ends



STD 3.1.1 : Triggering and (FEF + NEI)

DAQ : THE (GOOD?) OLD DAYS

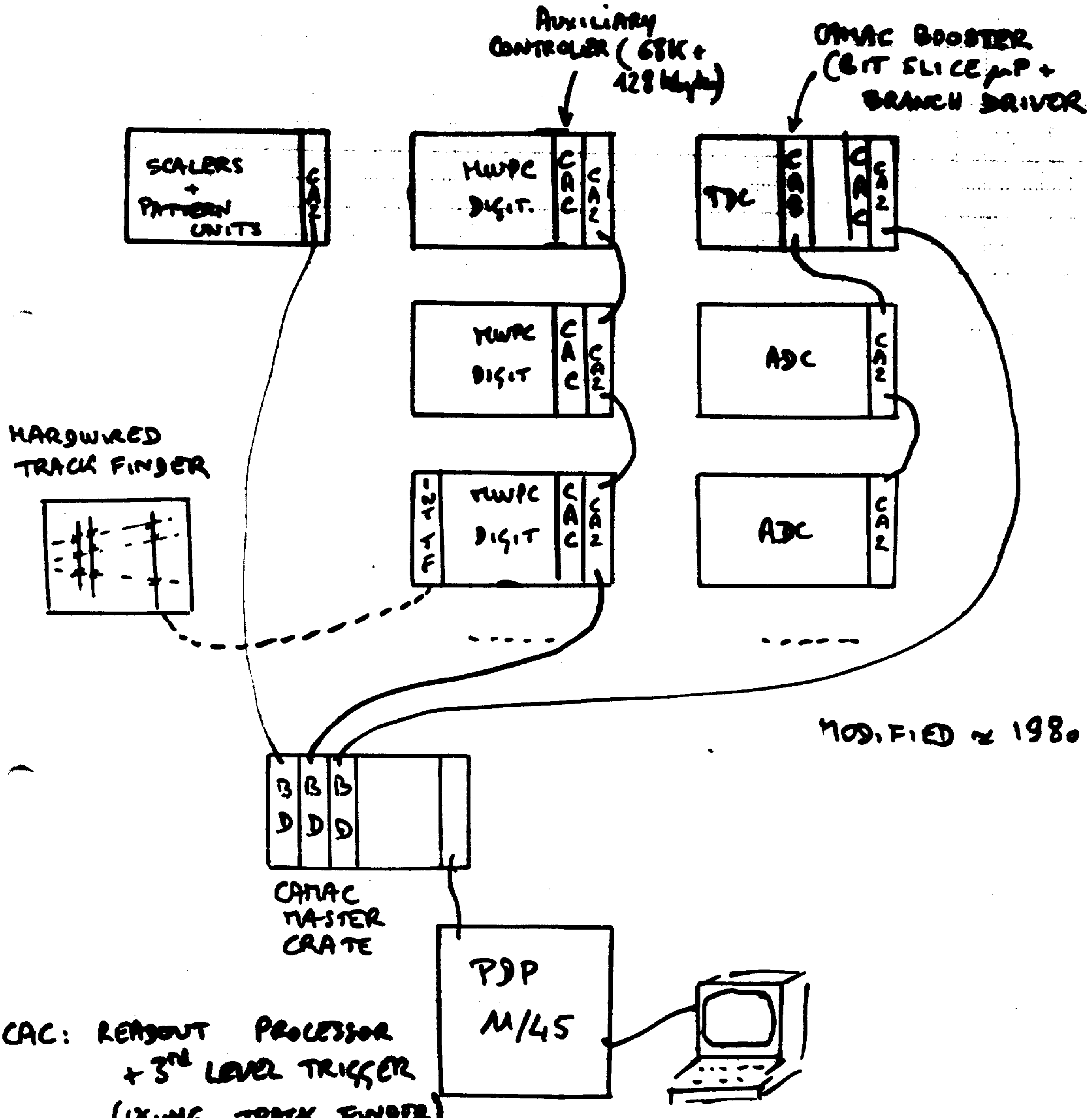
≃ 10 ÷ 12 YEARS AGO (NA3 EXPT)



ORIGINAL ARCHITECTURE (DESIGN '77)

DAQ : THE (GOOD?) OLD DAYS

≃ 10 ÷ 12 YEARS AGO (NA3 EXPT)



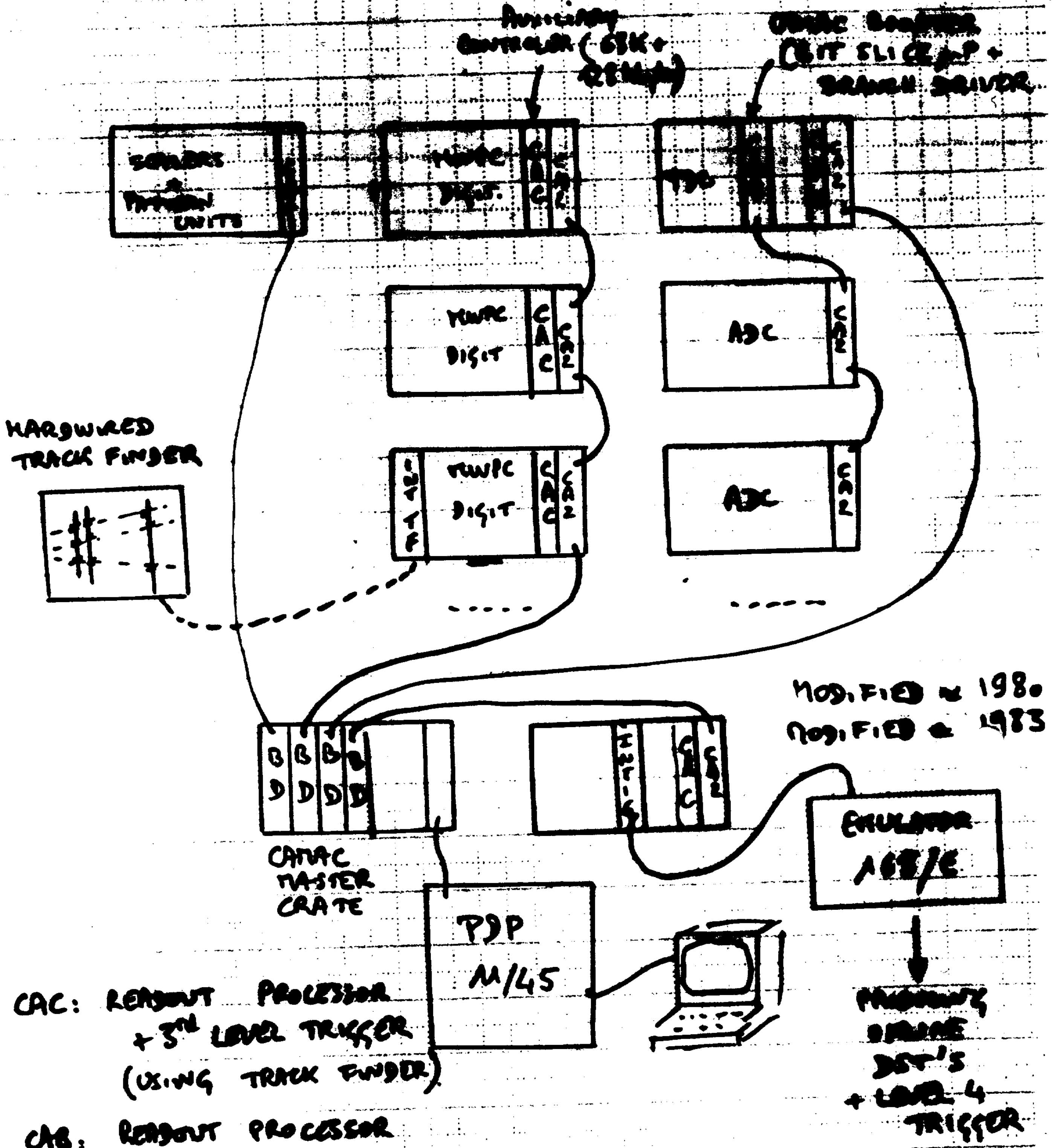
MODIFIED ≈ 1980

CAC: READOUT PROCESSOR + 3rd LEVEL TRIGGER (USING TRACK FINDER)

CAB: READOUT PROCESSOR

DAQ : THE (GOOD?) OLD DAYS

≃ 10 + 12 YEARS AGO (NA3 EXPT)

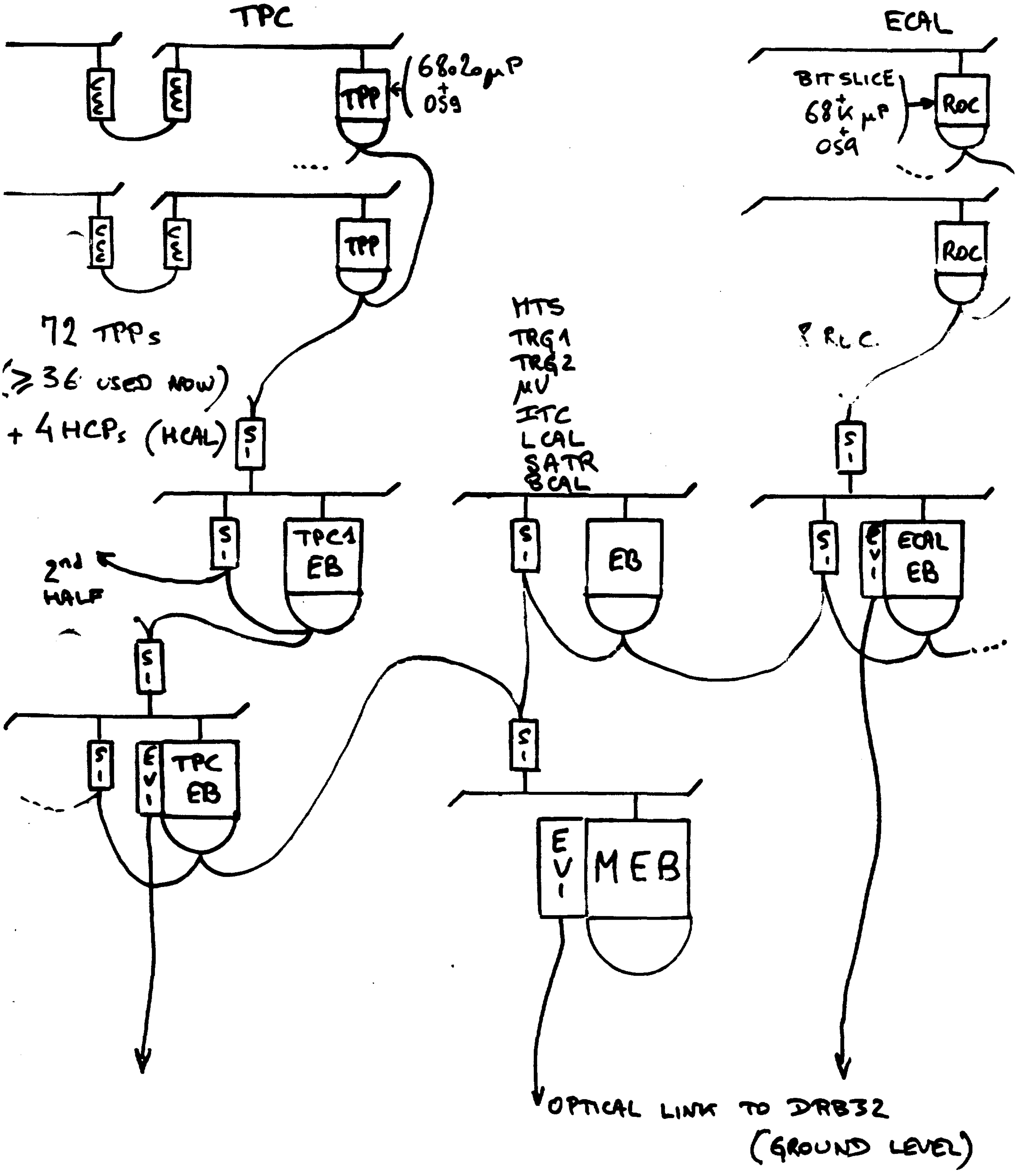


CAC: READOUT PROCESSOR + 3rd LEVEL TRIGGER (USING TRACK FINDER)

CAB: READOUT PROCESSOR

DAQ : H/W Examples

ALEPH



ALEPH (CONT'D)

- 3 DETECTORS WITH READOUT CONTROLERS → 84 R.O.C.S
  (TPC SPLIT IN TWO HALVES)

- 11 EVENT BUILDORS AT "DETECTOR LEVEL"
  +2 E.B FOR TPC (1 PER HALF)

- 1 MAIN EVENT BUILDER

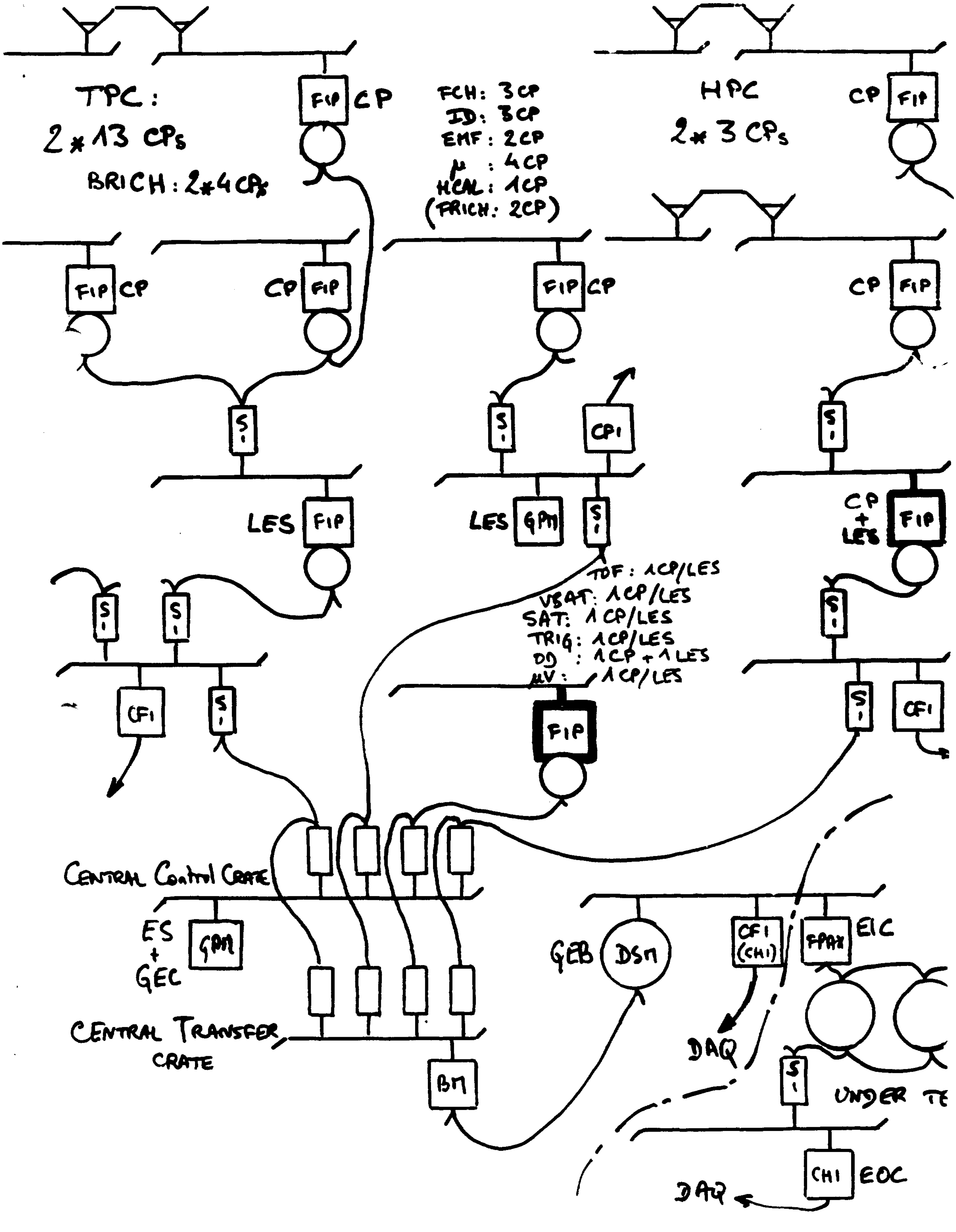
- 3 DETECTORS OWNING THEIR DEDICATED "EQUIPMENT - COMPUTER"

- UNDERGROUND ↔ SURFACE: OPTICAL LINK WITH EVI INTERFACE
  (EVENT BUILDERS DIRECTLY INTERFACED)

- S/W STANDARDIZATION: * AT E.B LEVEL, SOME S/W IS STANDARD
  * NO STANDARD AT R.O.C. LEVEL

ONLY CONSTRAINT: * R.O.C + E.B. FOLLOW TRIGGER CONTROL PROTOCOL (H/W + S/W)
* R.O.C + E.B FOLLOW READOUT PROTOCOLS (S/W)

DELPHI


TPC :
2 * 13 CPs
BRICH : 2 * 4 CPs
FIP
CP
FCH : 3 CP
ID : 3 CP
EMF : 2 CP
μ : 4 CP
HCAL : 1 CP
(FRICH : 2CP)
HPC
2 * 3 CPs
CP
FIP
FIP
CP
CP
FIP
FIP
CP
CP
FIP
S
S
CFI
S
S
LES
FIP
LES
GPM
S
CP
+
LES
FIP
TOF : 1 CP/LES
VSAT : 1 CP/LES
SAT : 1 CP/LES
TRIG : 1 CP/LES
DD : 1 CP + 1 LES
μV : 1 CP/LES
S
S
CFI
S
FIP
S
CFI
S
CENTRAL Control CRATE
ES
+
GEC
GPM
GEB
DSM
CFI
(CHI)
FPAX
EIC
DAQ
CENTRAL TRANSFER
CRATE
BM
S
UNDER TE
CHI
EOC
DAQ

DELPHI (CONT'D)

- H/W STANDARDIZATION :

  * ALL DETECTORS HAVE CRATE PROCESSOR (S) :
  FASTBUS INTERSEGMENT PROCESSORS (FIP)
  - 68020 μP (16MHz) + FB MASTER
  - 1Mb DRAM + EXTENSIONS
  - 1/4 Mb SRAM DUAL PORTED
  - SIMPLEX S1
  - ETHERNET/ CHEAPERNET INTERFACE

  * TOTAL OF 54 CPs, 7 CP/LES, 5 LES → FIPs
  6 LESs → GPMs

  * COMPLEX EVENT BUILDER : ES+GEC (2 GPMs) + BM
  (TO BE REPLACED BY GES (FIP) + BM)

  * ALL DETECTORS HAVE A CONNECTION TO EQUIPMENT COMPUTER (μVAX II) VIA CFI

  * CENTRAL DAQ READOUT : CFI + μVAX II
  (TO BE REPLACED BY CHI + OPTICAL LINK)

  * TRIGGER LEVEL 4 : SET OF EMULATORS 3081/E (W.I.T INPUT / OUTPUT CONTROLERS)

DELPHI (CONT'D)

- S/W STANDARDISATION

  * ALL CP S/W BASED ON THE SAME SKELETON + DETECTOR DEPENDENT ROUTINES

  * ALL LES S/W STANDARD + FEW DETECTOR DEPENDENT FORMATTING ROUTINES

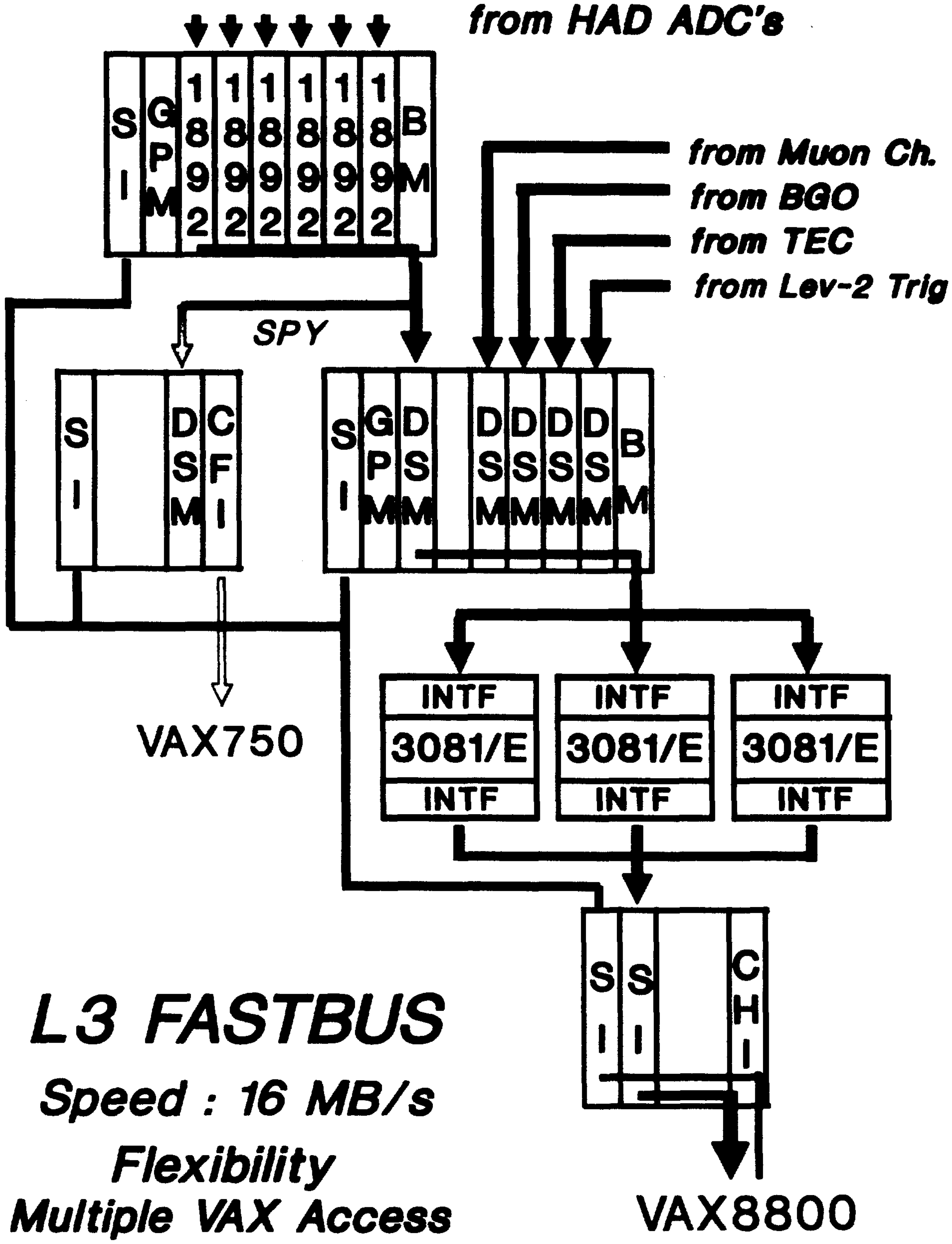
  * SERVICES: - STANDARD PACKAGE (LIBRARY + SERVER) FOR FASTBUS MESSAGES

  - ALL PROCESSORS (EXCEPT CHI) RUNNING OS9 (WITH DELPHI EXTENSIO

  - ALL FIPs RUNNING ALEPH NETWORK DISK S/W

  - TCP/IP BEING IMPLEMENTED
    ↓
    TELNET, RPC etc.

- MINIMIZATION IN MANPOWER INVESTMENT, IN PARTICULAR FROM EACH DETECTOR GROUP



L3 FASTBUS

Speed : 16 MB/s

Flexibility

Multiple VAX Access

Hardware Description

The Subdetector Event Builder

All subdetector branches are identical. The main modules used in the acquisition are

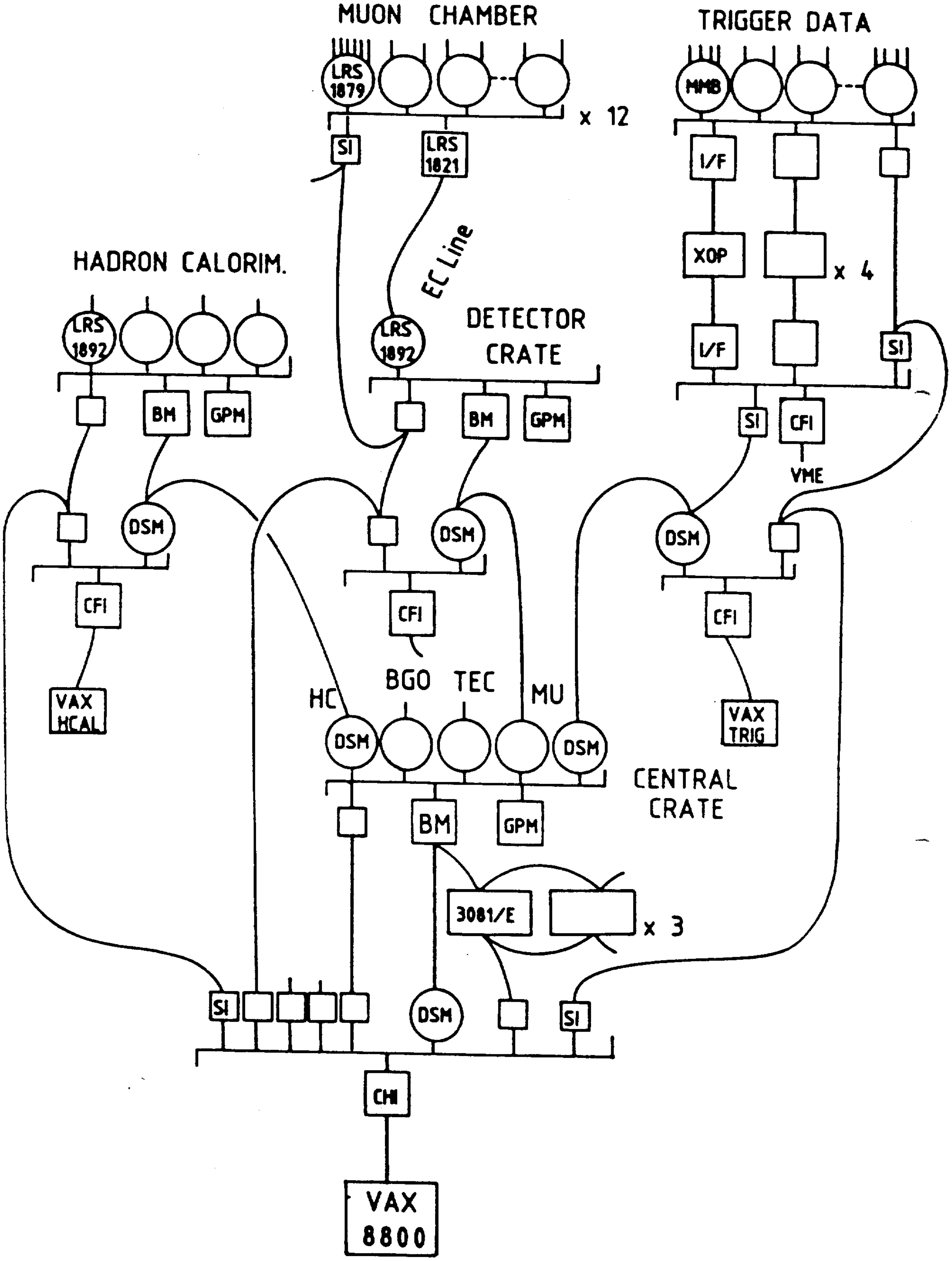
- (several) input memories (LeCroy 1892)
- (one) destination memory (Struck DSM)
- (one) 68000-based master (Struck GPM)
- (one) Blockmover (Kinetics F930)

Data are moved from the input memories to one destination memory. The task of moving and formatting data is splitted between the GPM and the BM. The GPM can execute very flexible programs and is therefor used to

- check event number and trigger pattern
- report any errors on a terminal in the control room
- calculate total event length
- prepare the BM registers

The BM on the other side

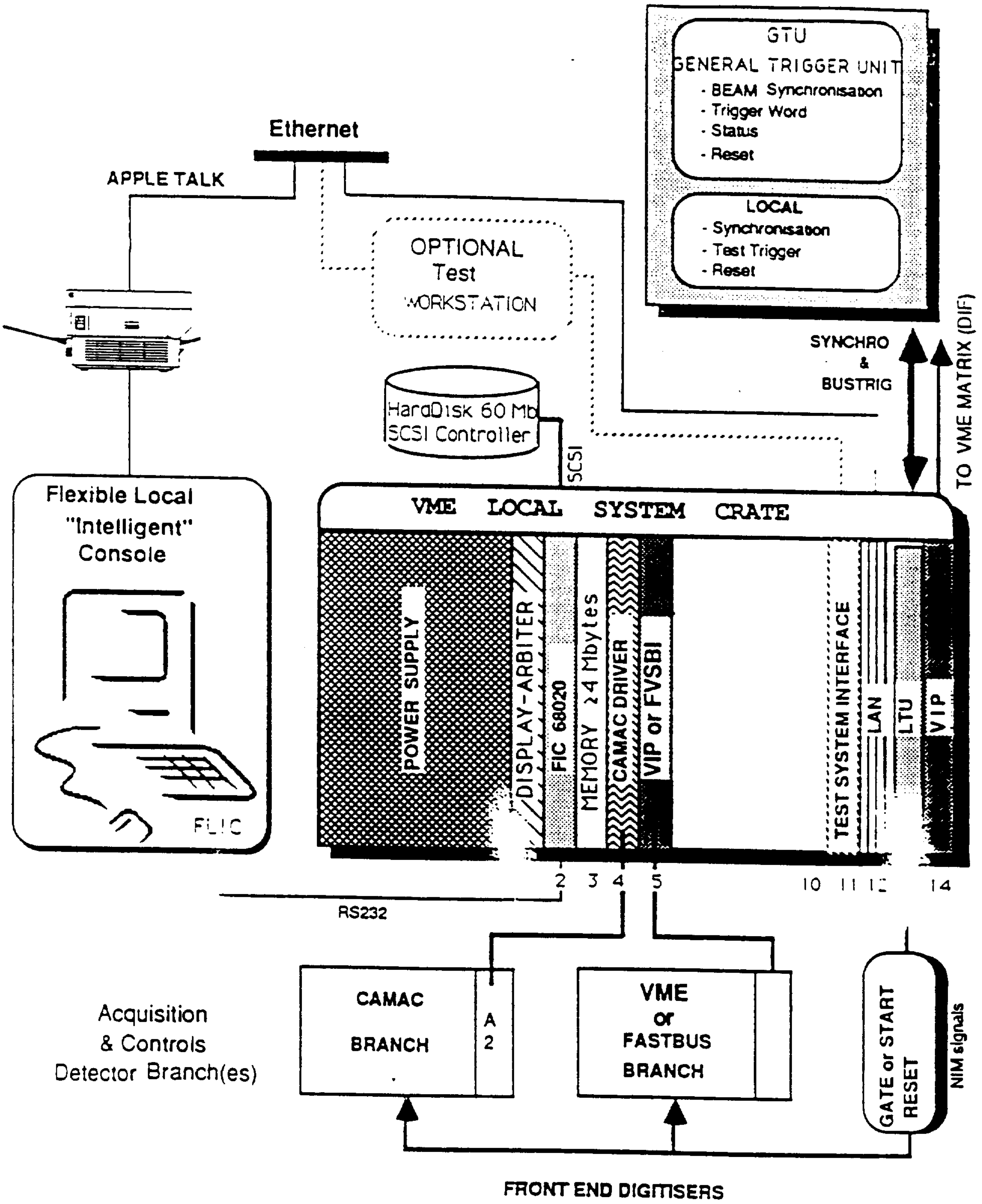
- moves data from several sources to one destination at very high speed
- can generate wordcounts and insert them in the data stream
- runs simple microprograms

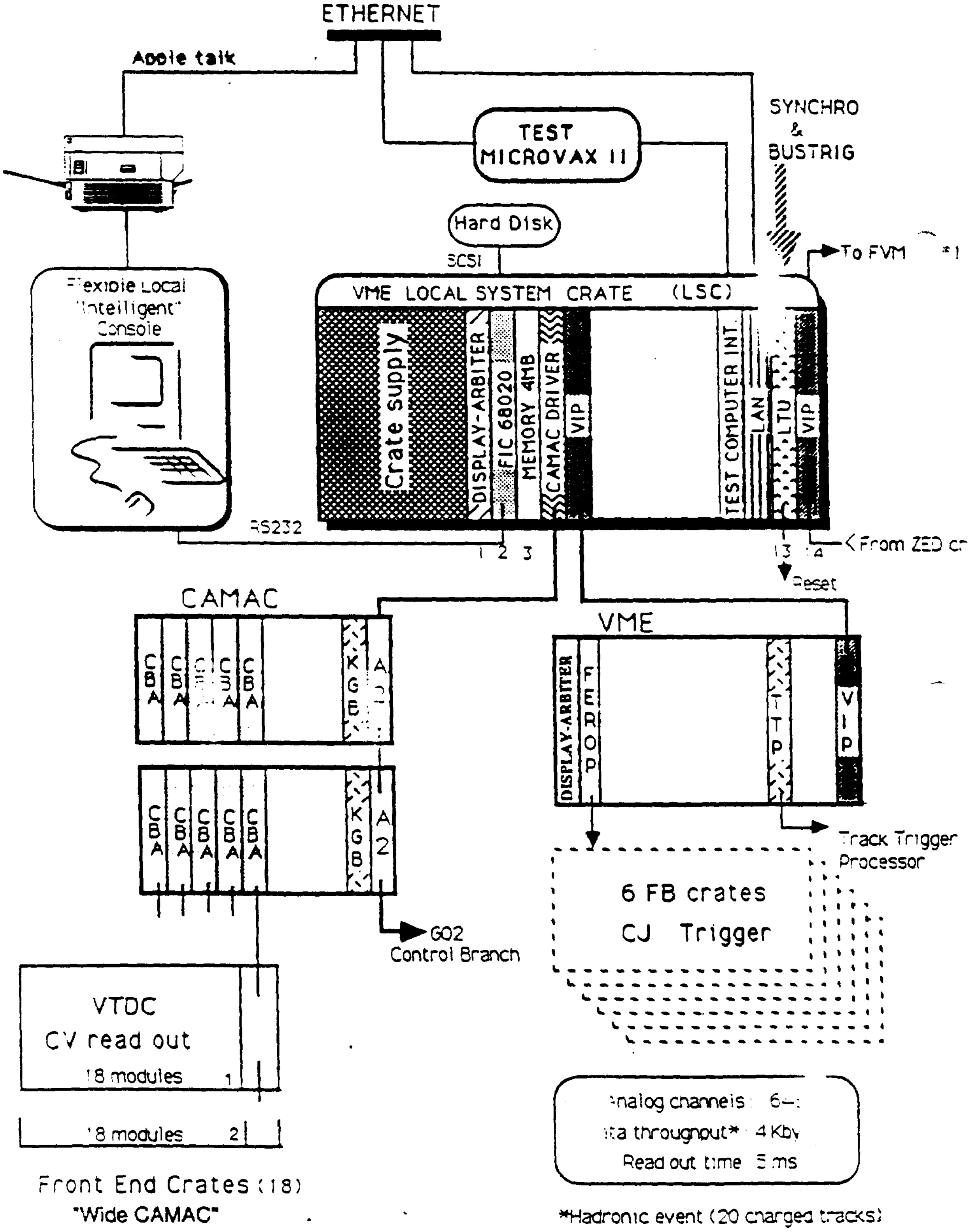

MUON CHAMBER
LRS 1879
x 12
SI
LRS 1821
EC Line
TRIGGER DATA
MMB
I/F
XOP
x 4
I/F
SI
SI
CFI
VME
HADRON CALORIM.
LRS 1892
BM
GPM
DSM
CFI
VAX HCAL
LRS 1892
DETECTOR CRATE
BM
GPM
DSM
CFI
DSM
CFI
VAX TRIG
HC
BGO
TEC
MU
DSM
DSM
CENTRAL CRATE
BM
GPM
3081/E
x 3
SI
DSM
SI
CHI
VAX 8800

OPAL

EACH DETECTOR MUST HAVE A "LOCAL SYSTEM CRATE" (LSC)

16 LSCs (INCLUDING 2 FOR TRIGGER)





*Hadronic event (20 charged tracks)

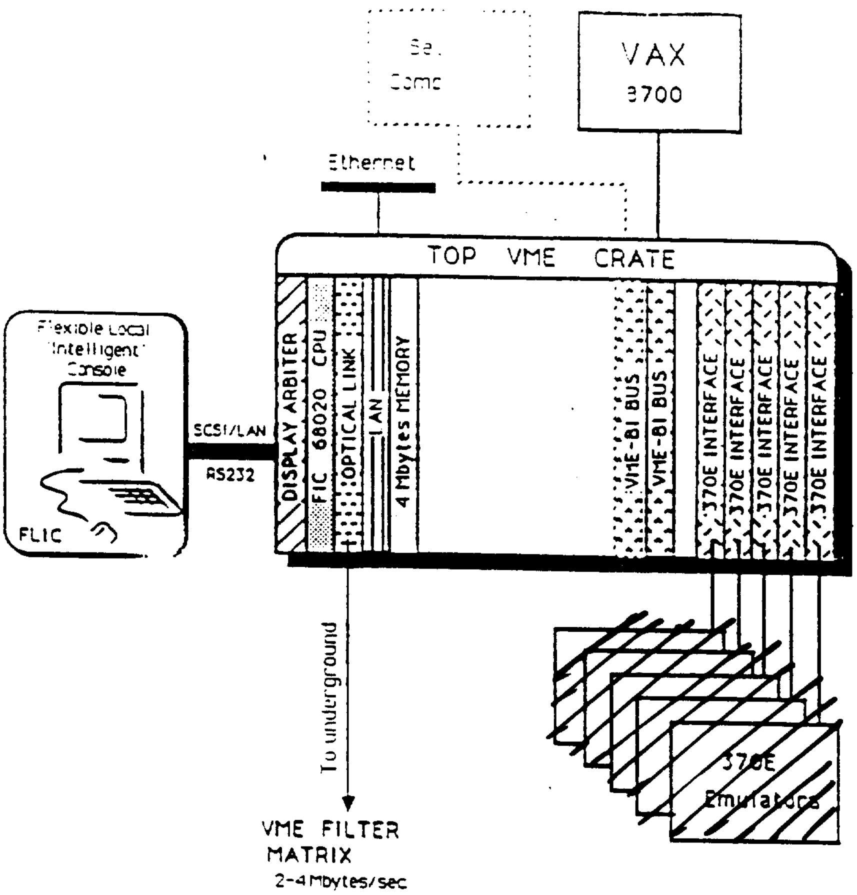


fig 7. Le Top VME Crate

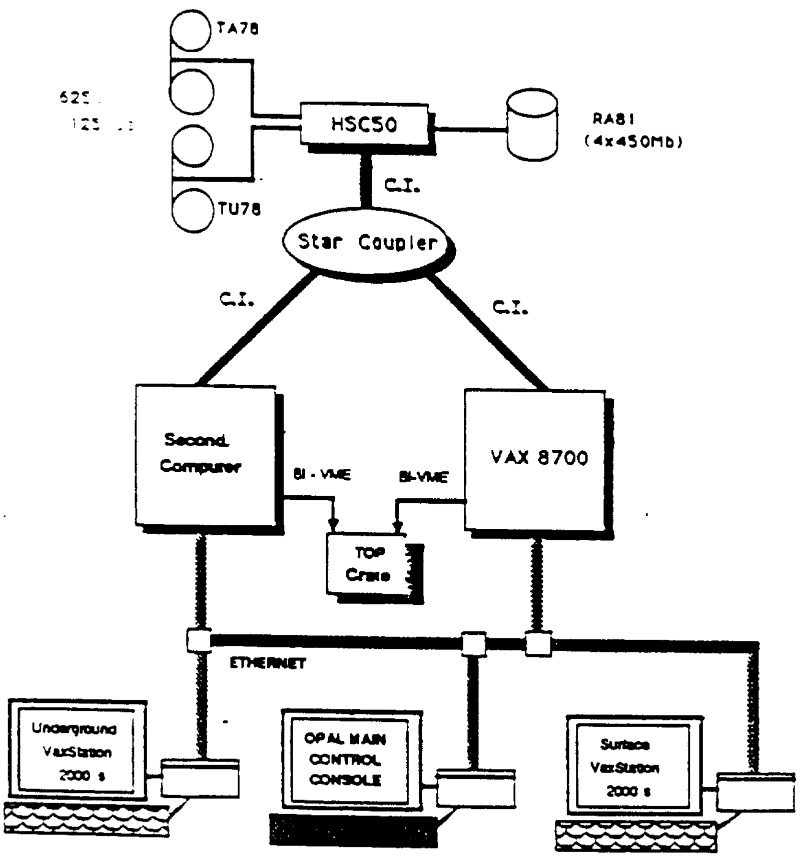
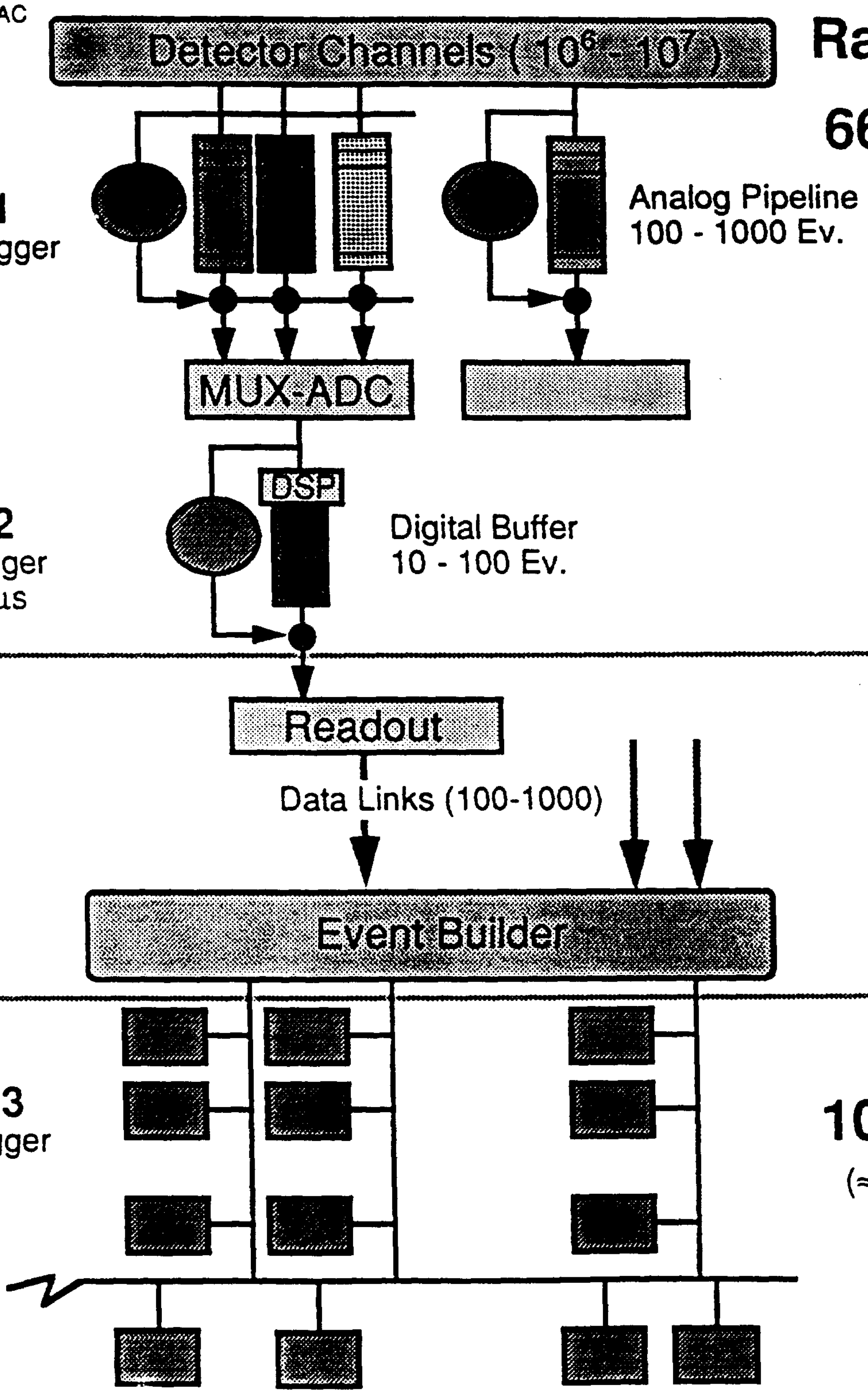


fig 8. La configuration du système VAX

LHC General-Purpose DAQ

E. Barsotti, E. Gaines/FNAL
A. Landford/SLAC



Detector Channels (10^6 - 10^7)

Rate [Hz]

$66 \cdot 10^6$

LEVEL 1
Prompt Trigger
1 - 10 µs

Analog Pipeline
100 - 1000 Ev.

MUX-ADC

DSP

LEVEL 2
Digital Trigger
10 - 1000 µs

Digital Buffer
10 - 100 Ev.

$\approx 10^5$

≈ Tbyte/s

Readout

Data Links (100-1000)

$\approx 10^3$

≈ Gbyte/s

Event Builder

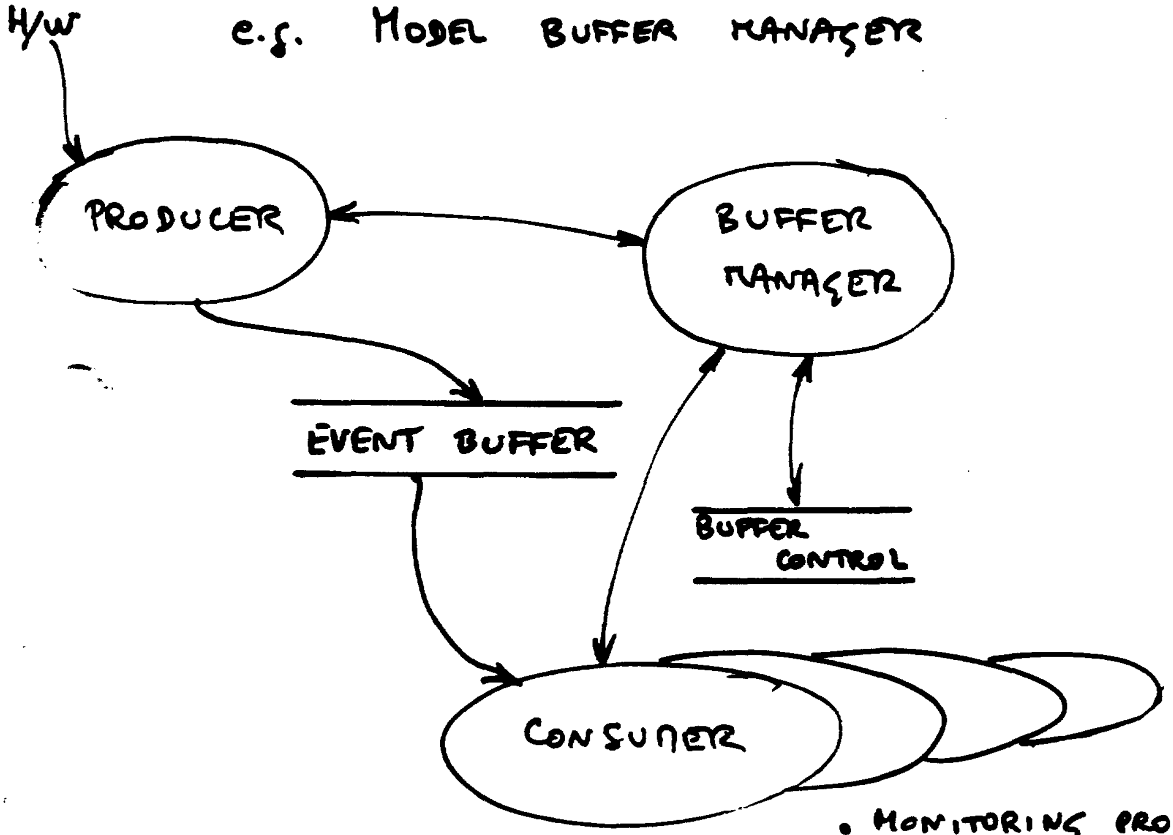
LEVEL 3
Event Trigger

10 - 100
(≈ Mbyte/Ev.)

Mass Storage & Analysis

DAQ: Final readout & Data logging

- Uses rather powerful computers (VAX 8700)
- Different kinds of interfaces
  - VME : VME-BI interface
  - FASTBUS: CFI ($\Delta\phi$)
    CHI (L3, $\Delta\phi$)
    EVI (N)
- Event handling
  e.g. Model buffer manager



- Monitoring processes / DISPLA
- Data logger

DAQ : FINAL READOUT & DATA LOGGING (CONT'D)

- DATA LOGGING

  * LOGGING ON DISK (ALEPH, DELPHI)
  + TRANSFER ON CARTRIDGES (160 Mb)
  FROM MAIN. DAQ COMPUTER

  * LOGGING ON CARTRIDGES (OPAL, L3)

- ONLINE PROCESSING

  * EMULATORS (L3, DELPHI, UA1)
  - INTEGRATED IN FASTBUS/VME READOUT
  - TAGGING + HIGH LEVEL FILTERING

  * PROCESSOR FARM (ALEPH)
  μVAX CPUS DIRECTLY CONNECTED TO BI
  ON THE DAQ VAX 8700

  * APOLLO FARM (OPAL)
  2 * DN 10000 READING OUT DATA FROM
  TOP VME CRATE
  WRITING DSTs FROM VME CARTRIDGE
  INTERFACE

DAQ: CONTROL & Monitoring

- S/W LOCATED IN THE MAIN DAQ COMPUTER (USUALLY VAX OR μVAX) & TEST COMPUTERS
- CONTROL DAQ
  - ALLOWS RECONFIGURATION OF DAQ SYSTEM
  - RUN CONTROL (BEGIN, PAUSE, STOP, CONTINUE...)
  - DATA LOGGER CONTROL

EXAMPLE OF GENERAL PURPOSE CONTROL:

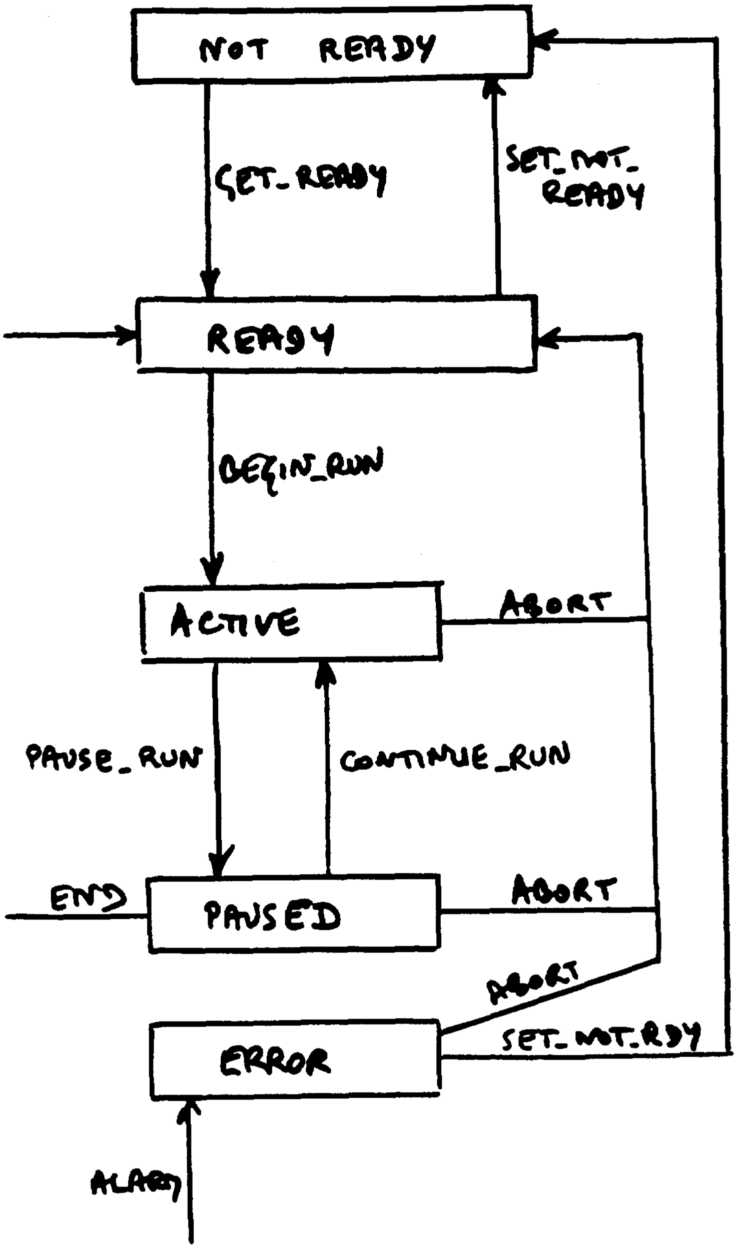
STATE MANAGEMENT INTERFACE (SMI)

- INITIATED BY DELPHI (M.JONKER)
- DEVELOPPED BY "DD-OC"

- USE A HIGH LEVEL LANGUAGE TO CONTROL THE BEHAVIOUR OF OBJECTS WHICH MAY BE IN CERTAIN STATES AND IN WHICH ONE MAY TRIGGER ACTIONS
- S/W TRANSPOSITION OF SASD "STATE TRANSITION DIAGRAMS"

DAQ : CONTROL

EXAMPLE OF SMI USE FOR RUN CONTROL



SIMPLIFIED STATE TRANSITION DIAGRAM FOR OBJECT "LC" (LOCAL CONTROL)

- EACH ACTION ON OBJECT "LC" MAY BE DECOMPOSED INTO MORE ELEMENTARY ACTIONS ON ELEMENTARY OBJEC

  CFI_SERVER
  LES

- EACH CHANGE OF STATE IS TRIGGÉRING STATE CHANGES ON THE HIGHER LEVEL OBJEC

  E.G. IF "LES" GOES TO "ERROR" → "LC" GOES TO "ERROR" AS WELL

- EACH LOWER LEVEL OBJECT IS ASSOCIATED WITH AN "ELEMENTARY PROCESS" CONNECT TO H/W

Imprimé au CERN.   Prix :   4.10 Fr.

```
object : RUN

    state : NOT_READY
!
        ● action : GET_READY                              !End_states : READY
!
                    if (not CFI_SERVER in_state ACTIVE) then
                        do CONNECT CFI_SERVER
                    end if
!
                    if (not CFI_SERVER in_state ACTIVE) then
                        do SET_CFI_SERVER ERROR
                        terminate_action/state = ERROR
                    end if
!
!
                    do RESET LES

                    do CONFIGURE_LOCAL LES

                    if (not LES in_state CONFIGURED) then
                        do SET_LES ERROR
                        terminate_action/state = ERROR
                    end if
!
                    do RO_SETUP LES

                    if (not LES in_state READY) then
                        do SET_LES ERROR
                        terminate_action/state = ERROR
                    else
                        terminate_action/state = READY
                    end if
!
    state : READY
            when (not CFI_SERVER in_state ACTIVE) do SET_CFI_SERVER_
            when (not LES in_state READY) do SET_LES_ERROR
!
        ● action : BEGIN                           !End_states : PAUSED
!                   if (not TAPE in_state LOGGING) then
!                           do WARNING DISPLAY
!                   end if

                    do BEGIN LES

                    if (not LES in_state PAUSED) then
                            do SET_LES ERROR
                            terminate_action/state = ERROR
                    else
                            terminate_action/state = PAUSED
                    end if
!
        ● action : ABORT                           !End_states : READY
!
                do ABORT LES
!
                if (not LES in_state READY) then
                    do SET_LES ERROR
                    terminate_action/state = ERROR
                else
                    terminate_action/state = READY
```

```
                    end if
!
            action : SET_NOT_READY
                    terminate_action/state = NOT_READY
!
    state : PAUSED
            when (not CFI_SERVER in_state ACTIVE) do SET_CFI_S
            when (not LES in_state PAUSED) do SET_LES_ERROR
        ● action : CONTINUE                          !End_states : RUNNING
!
                    do CONTINUE LES
!
                    if (not LES in_state RUNNING) then
                        do SET_LES ERROR
                        terminate_action/state = ERROR
                    else
                        terminate_action/state = RUNNING
                    end if

        ● action : ENDRUN                             !End_states : READ
!
                    do ENDRUN LES
!
                    if (not LES in_state READY) then
                        do SET_LES ERROR
                        terminate_action/state = ERROR
                    else
                        terminate_action/state = READY
                    end if
!
        ● action : ABORT                          !End_states : READY
!
                    do ABORT LES
!
                    if (not LES in_state READY) then
                        do SET_LES ERROR
                        terminate_action/state = ERROR
                    else
                        terminate_action/state = READY
                    end if
!
        ● action : SET_NOT_READY
                    terminate_action/state = NOT_READY
!
    state : RUNNING
            when (not CFI_SERVER in_state ACTIVE) do SET_CFI_SE
            when     (not LES in_state RUNNING)
                  and(not LES in_state PAUSED) do SET_LES_ERROR
            when  ( LES in_state PAUSED) do SET_PAUSED
        ● action : SET_PAUSED
            terminate_action / state = PAUSED
        ● action : PAUSE                             !End_states : PAUSED
                    do PAUSE LES
!
                    if (LES in_state PAUSED) then
                            terminate_action / state = PAUSED
                    end if
                    do SET_LES ERROR
!
                    terminate_action/state = ERROR
```

```
        ● action : ABORT                          !End_states : READY  / |
                    do ABORT LES

                    if (not LES in_state READY) then
                            do SET_LES ERROR
                            terminate_action/state = ERROR
                    else
                            terminate_action/state = READY
                    end if
!
        ● action : SET_NOT_READY
                terminate_action/state = NOT_READY
!
    state : ERROR
!
        ● action : SET_NOT_READY
                do RESET ERROR
                terminate_action/state = NOT_READY
!
        ● action : ABORT
!
            if   ( LES in_state READY)
              or ( LES in_state PAUSED)
              or ( LES in_state RUNNING) then
               do ABORT LES
            end if
!
            if (LES in_state READY) then
               do RESET ERROR
               terminate_action / state = READY
            end if
!
            do SET_LES ERROR
            terminate_action / state = ERROR
!
!
```

SMI PROGRAM
CORRESPONDING TO THE
STATE TRANSITION DIAGRAM
(DELPHI EXPT)