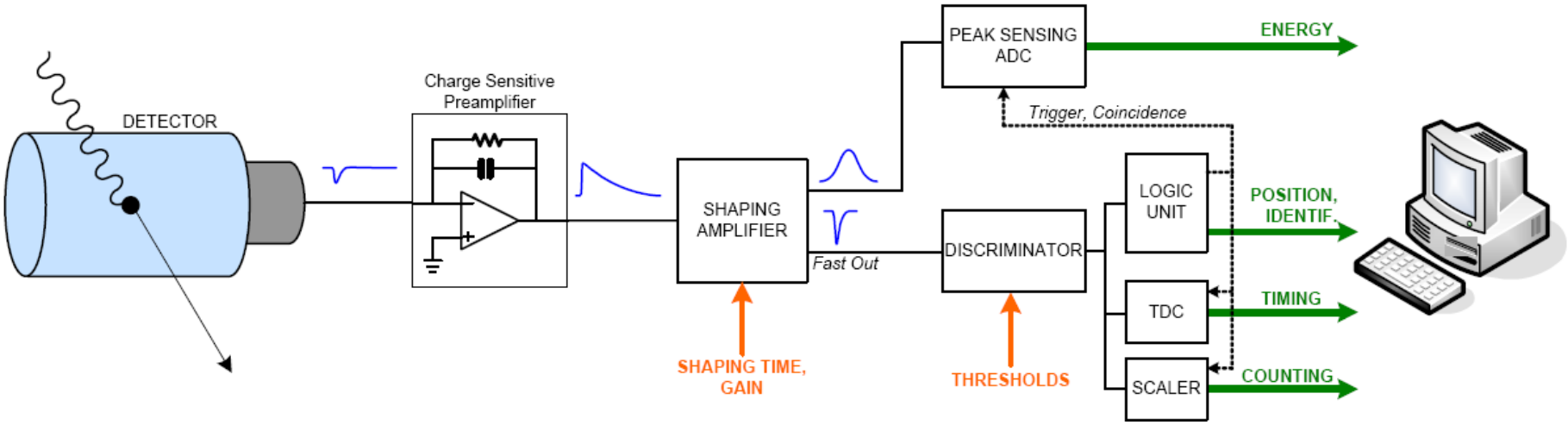
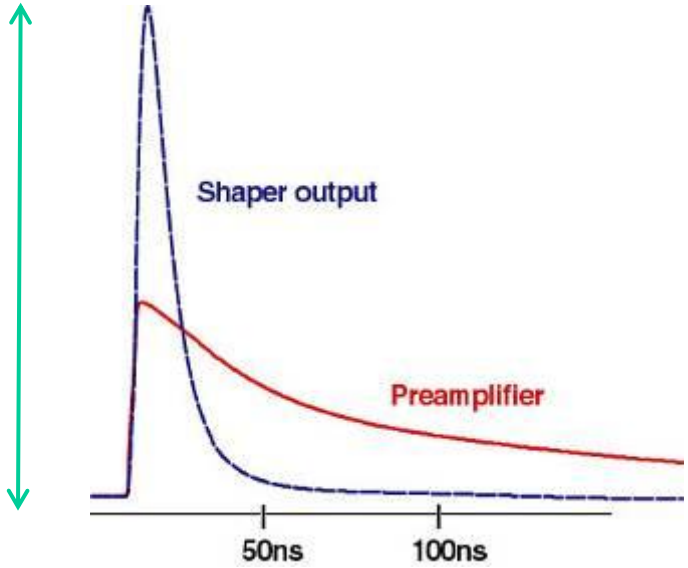
European Organization for Nuclear Research

Elettronica di readout: digital pulse processing (DPP) con FPGA

Traditional analog chain

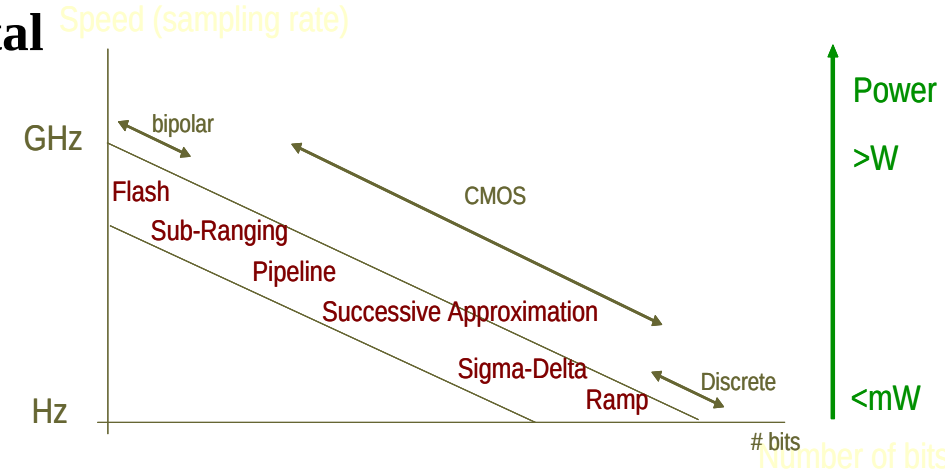


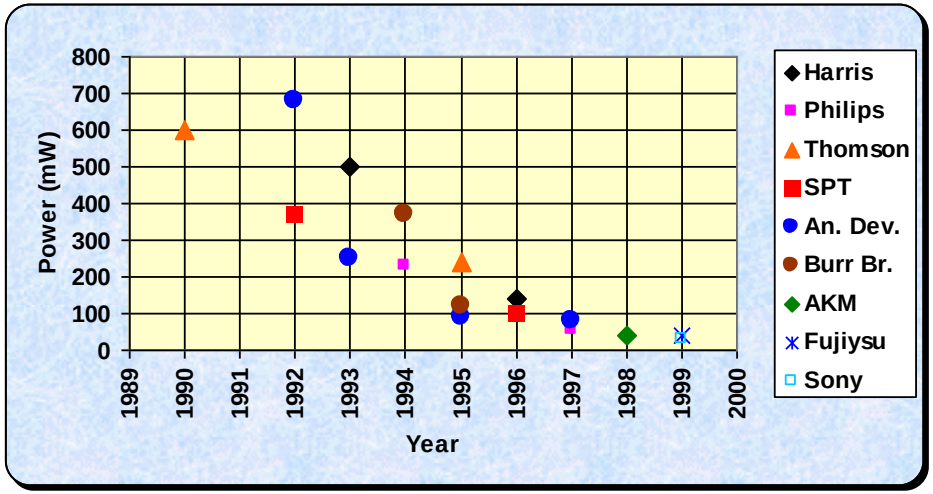
peak amplitude
=
energy



Analog to digital conversion

- There is clearly a tendency to **go digital as early as possible**
- The “cost” of the ADC determines which architecture is chosen
  - Strongly depends on speed and resolution
- Cost is here
  - **Power consumption**
  - Silicon area
  - Availability of radiation hard ADC
- Input frequencies must be limited to max Fs/2.
  - Otherwise this will fold in as additional noise.
- High resolution ADC also needs low jitter clock to maintain effective resolution





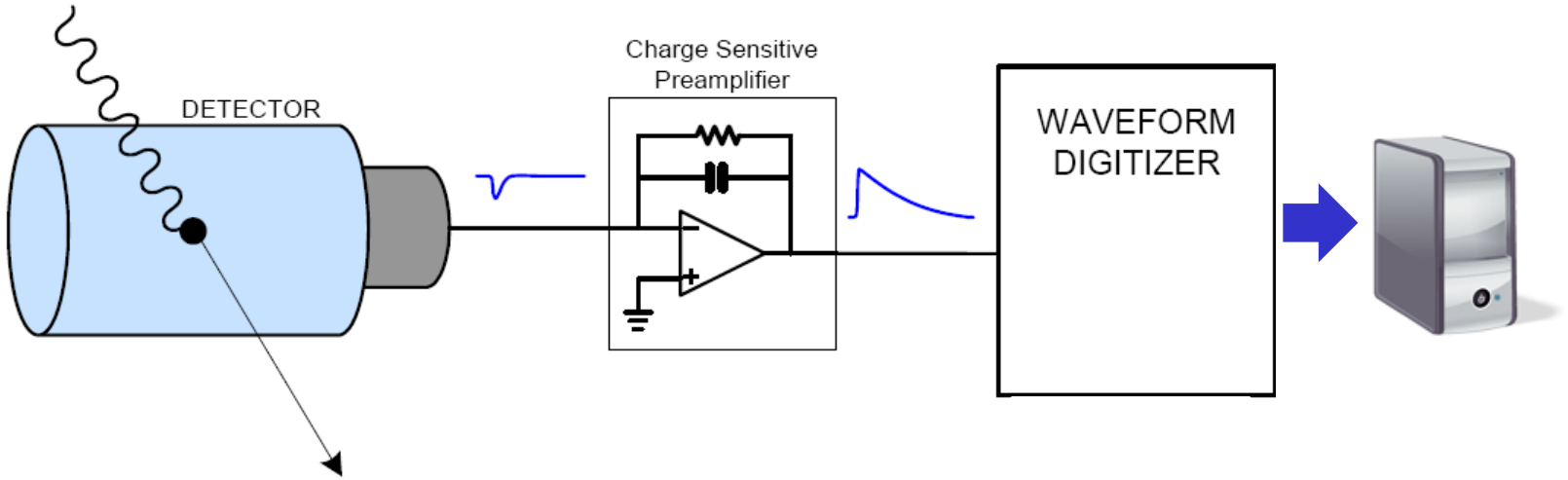
CAEN waveform digitizers main features

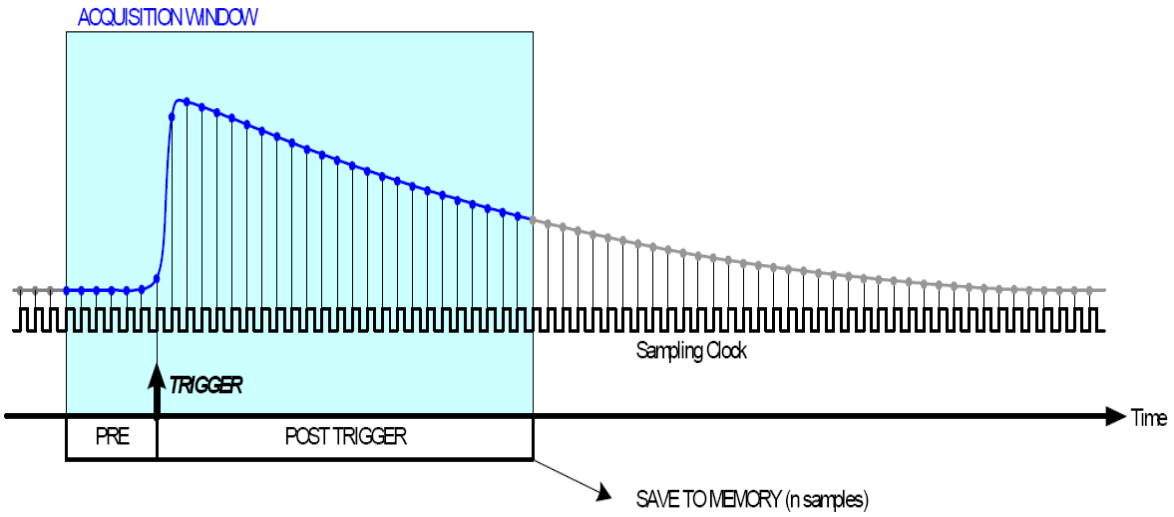
Series	Max sampling rate (MS/s)	Resolution (bits)	Memory (MS/ch)
724	100	14	0.5/4
720	250	12	1.25/10
721	500	8	2
731	500-1000	8	2-4
740	65	12	0.19-1.5
751	1000-2000	10	1.8-3.6
742	5000	12	0.128

Available form factors:

- VME64
- PCI Express

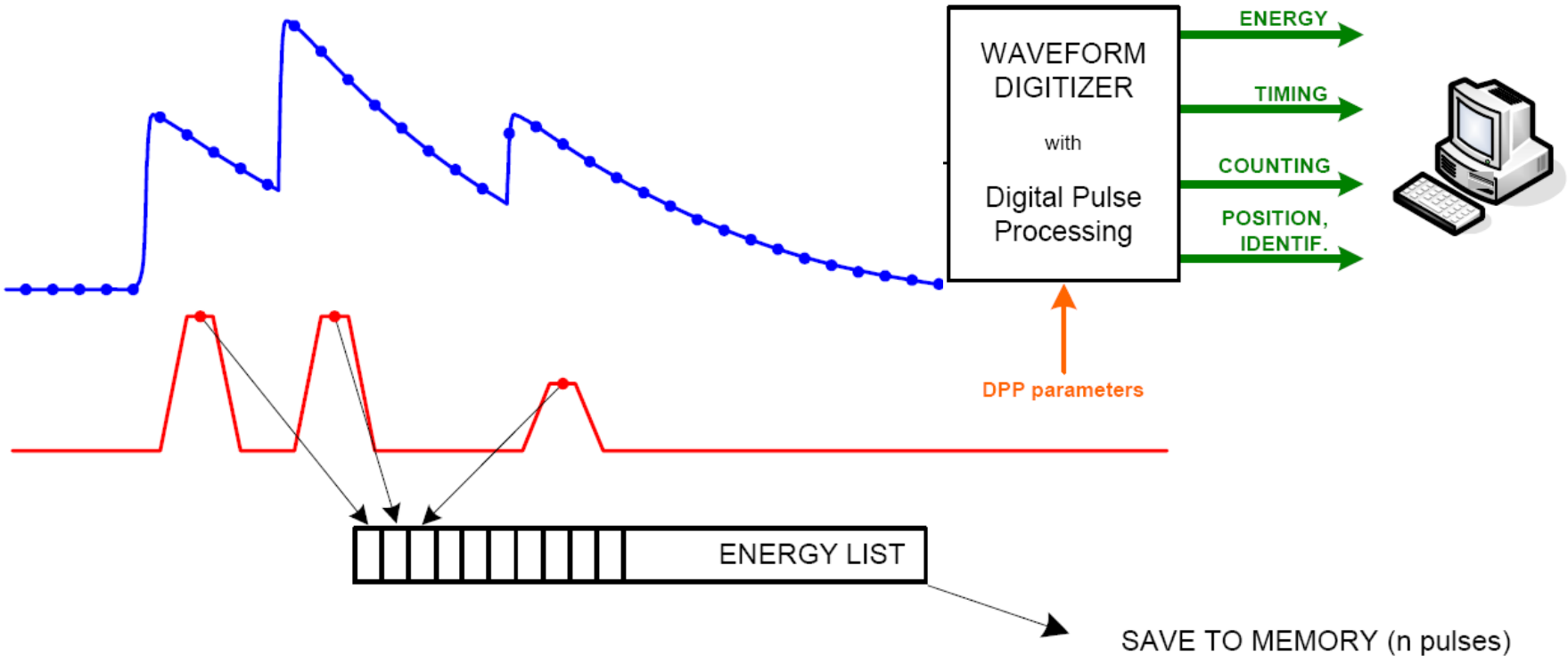
Waveform digitizer used in oscilloscope mode





- circular buffer of programmable size
- when a channel is triggered, the current buffer is saved
- the acquisition can continue without dead time with a new buffer
- high output data throughput

On-line digital pulse processing (DPP)



- the digitized signal is processed **on-line** and the acquisition is continuous
- the quantities of interest are calculated and saved in a digital buffer
- very small amount of data with respect to reading all wave samples

Digital versus analog pulse processing

Advantages:

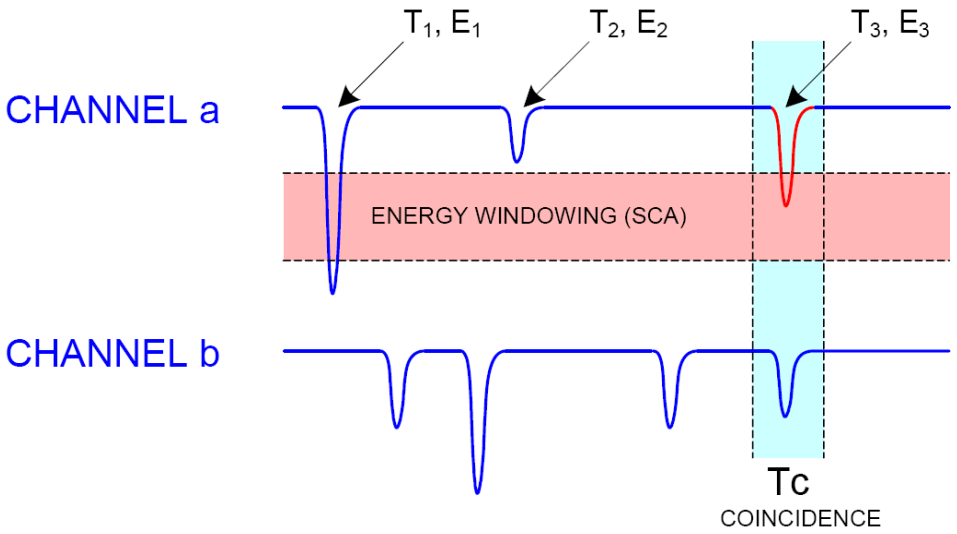
- low cost and high reliability
- good linearity and stability (reproducibility)
- flexibility
- faster and automatic tuning and calibration

Disadvantages:

- digital algorithms knowledge required
- customization requires low-level knowledge
- loss of resolution with fast signals

DPP for counting

- read the time-tag and energy list from the ADC and select only the pulses within a certain energy range
- on-line perform coincidence-anticoincidence



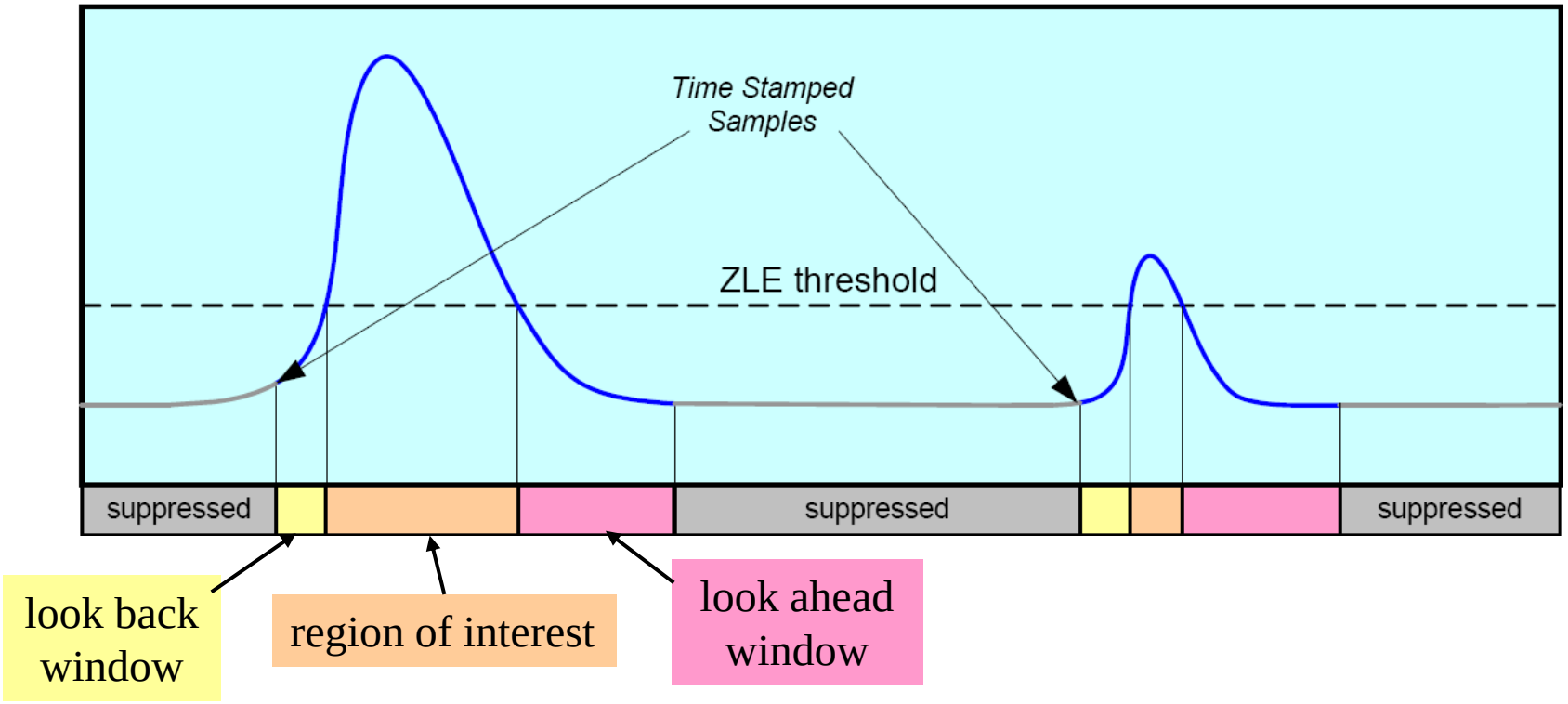
Ta,1	Ea,1	Ta,2	Ea,2	Ta,3	Ea,3

Tb,1	Eb,1	Tb,2	Eb,2	Tb,3	Eb,3	Tb,4	Eb,4

Time coincidence: $|T_{a,3} - T_{b,4}| < T_C$

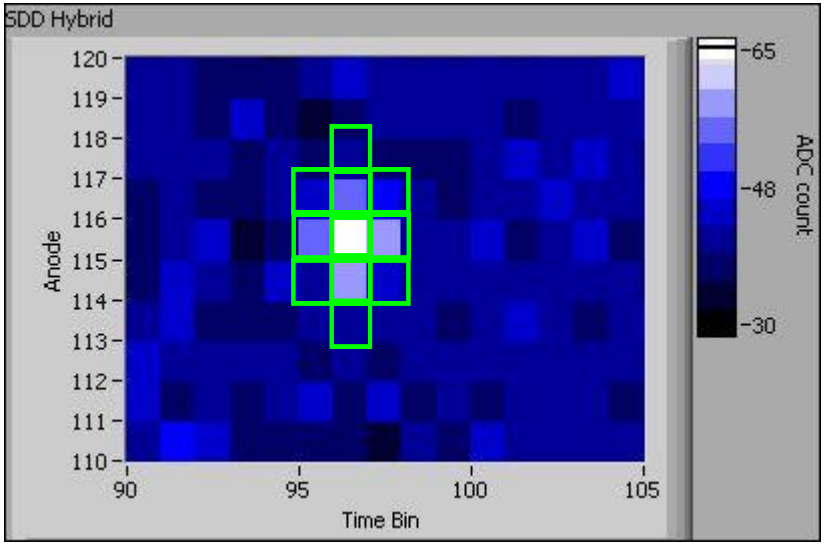
Energy windowing: $E_{THRlow} < E_{a,3} < E_{THRhigh}$

DPP for zero suppression



- Data are discarded if below the programmable threshold
- Thresholds and windows are programmable
- Any data compression algorithm can be encoded and applied

DPP for 2D zero suppression



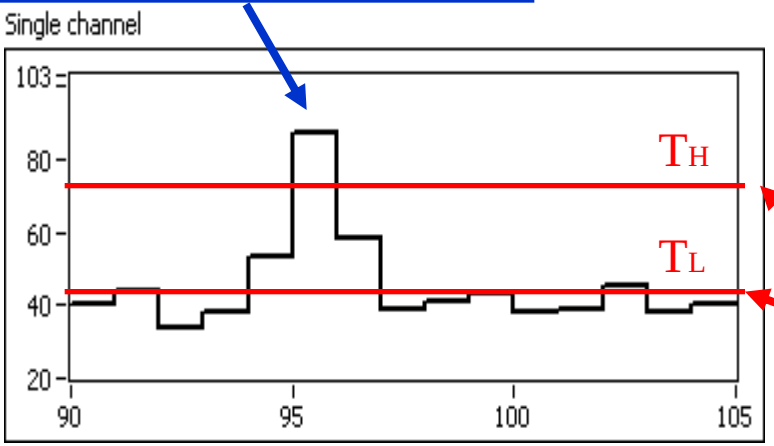
The central sample belongs to a cluster if the cross contains at least:

- one value > T_H
- two values > T_L

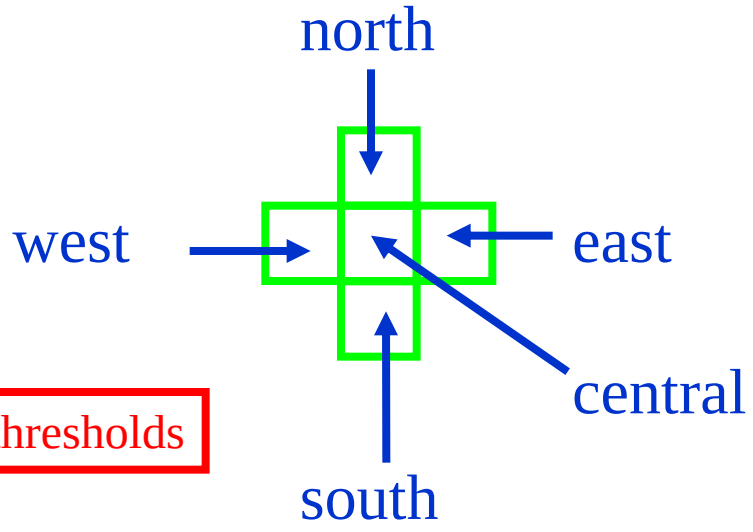
T_H: for cluster selection

T_L: so to collect information around the selected cluster

Cluster on channel 115

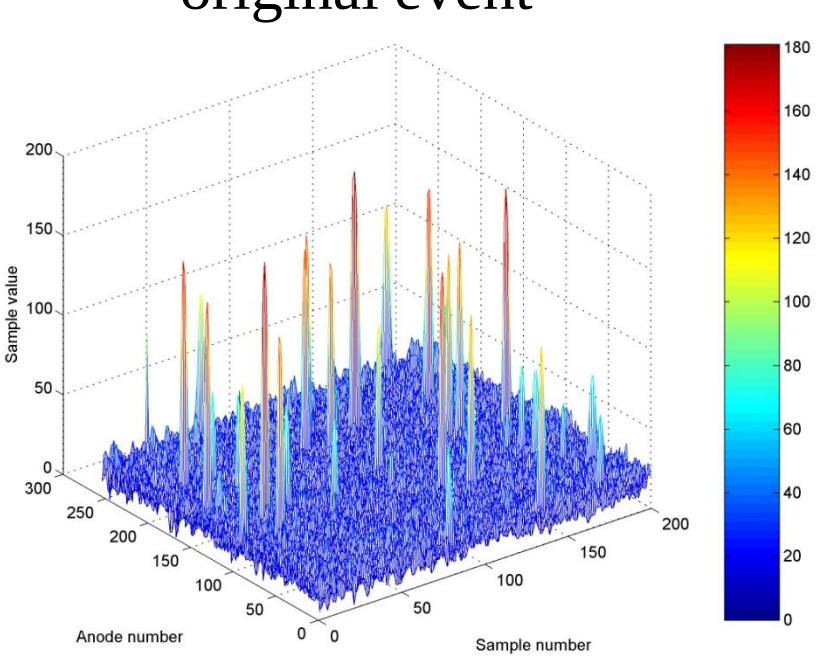


Two thresholds

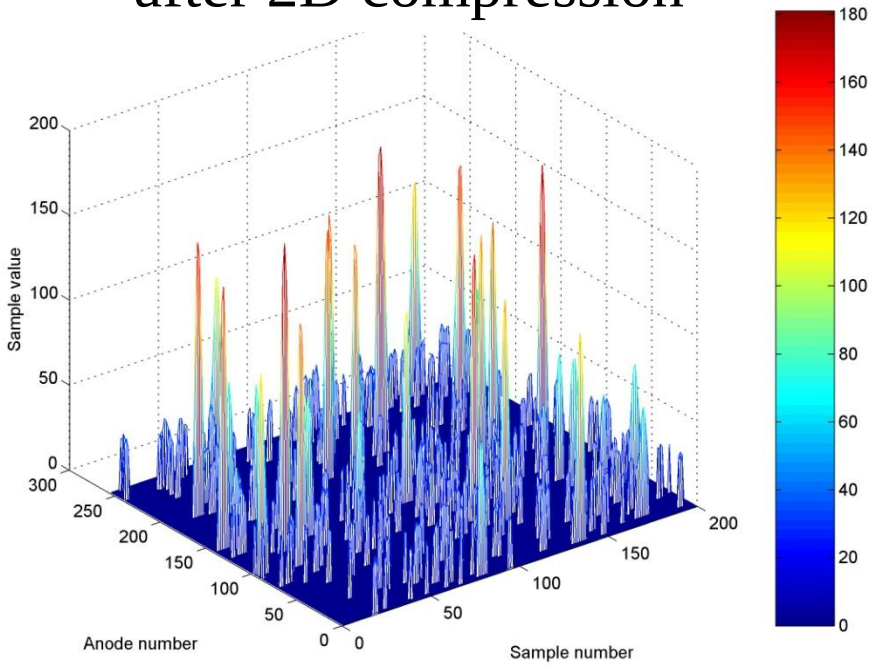


DPP for 2D zero suppression

original event

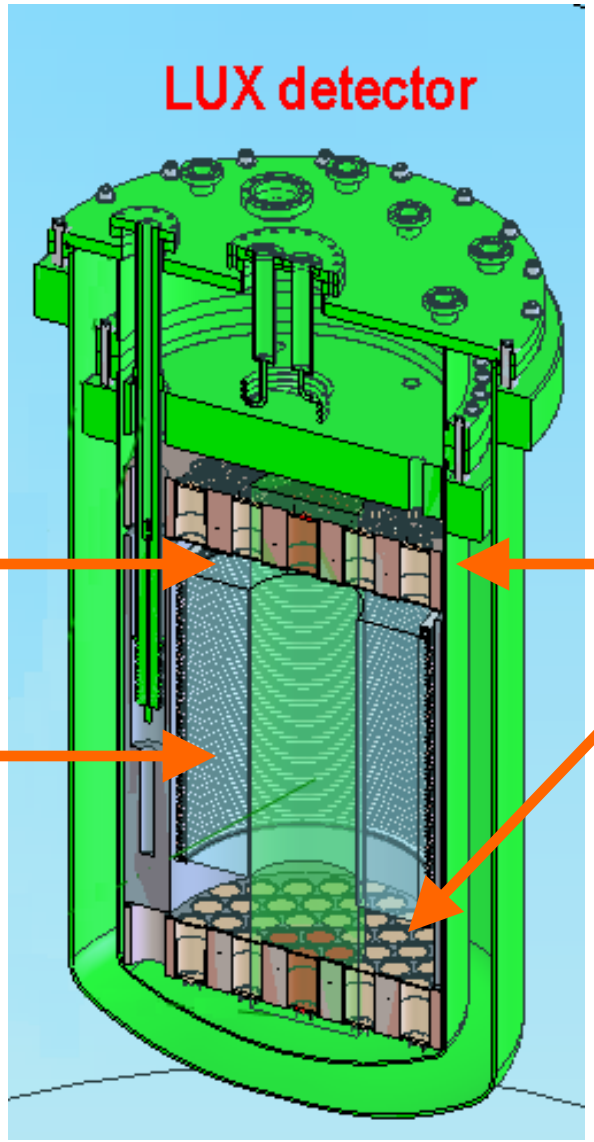


after 2D compression



The noise has been removed and only the clusters of interest are saved

Real case: LUX detector



Liquid xenon scintillates when hit by particles (e.g. photons, neutrons and potentially dark matter)

→

dark matter candidate: the Weakly Interacting Massive Particle (**WIMP**)

Gas Xe

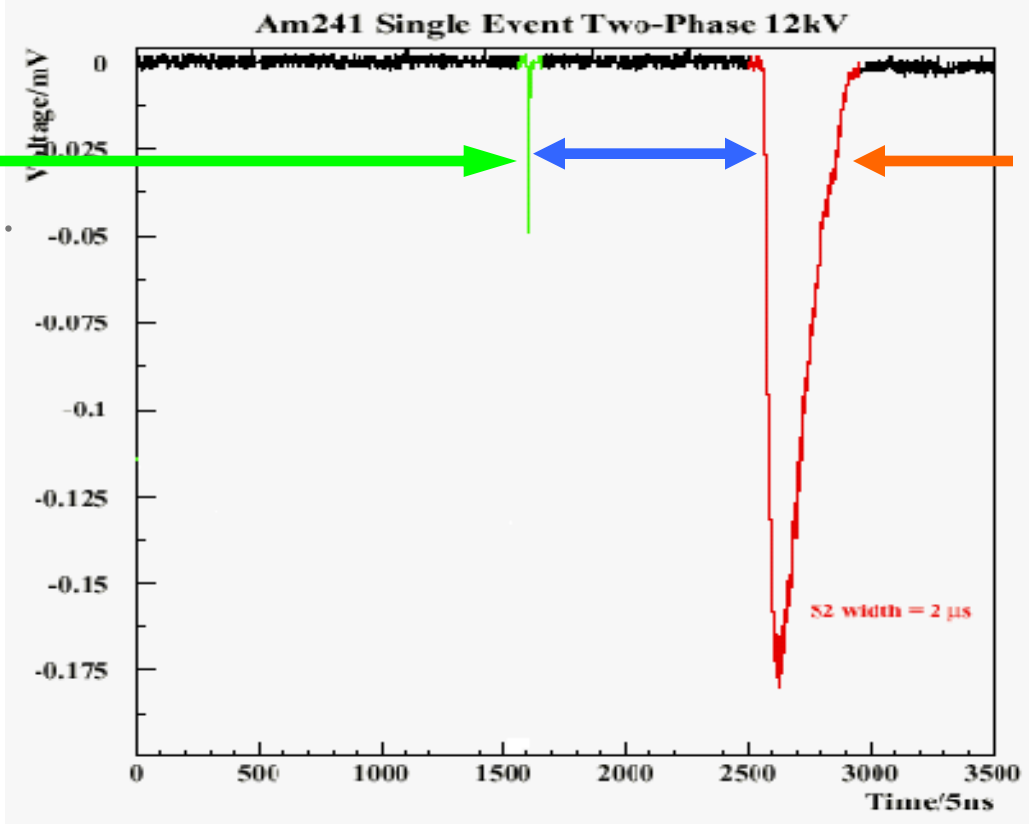
Liquid Xe

144 phototubes

Each phototube requires an independent ADC and the data-processing channel

Signal processing in a single channel Xe detector

Primary scintillation in liquid phase.



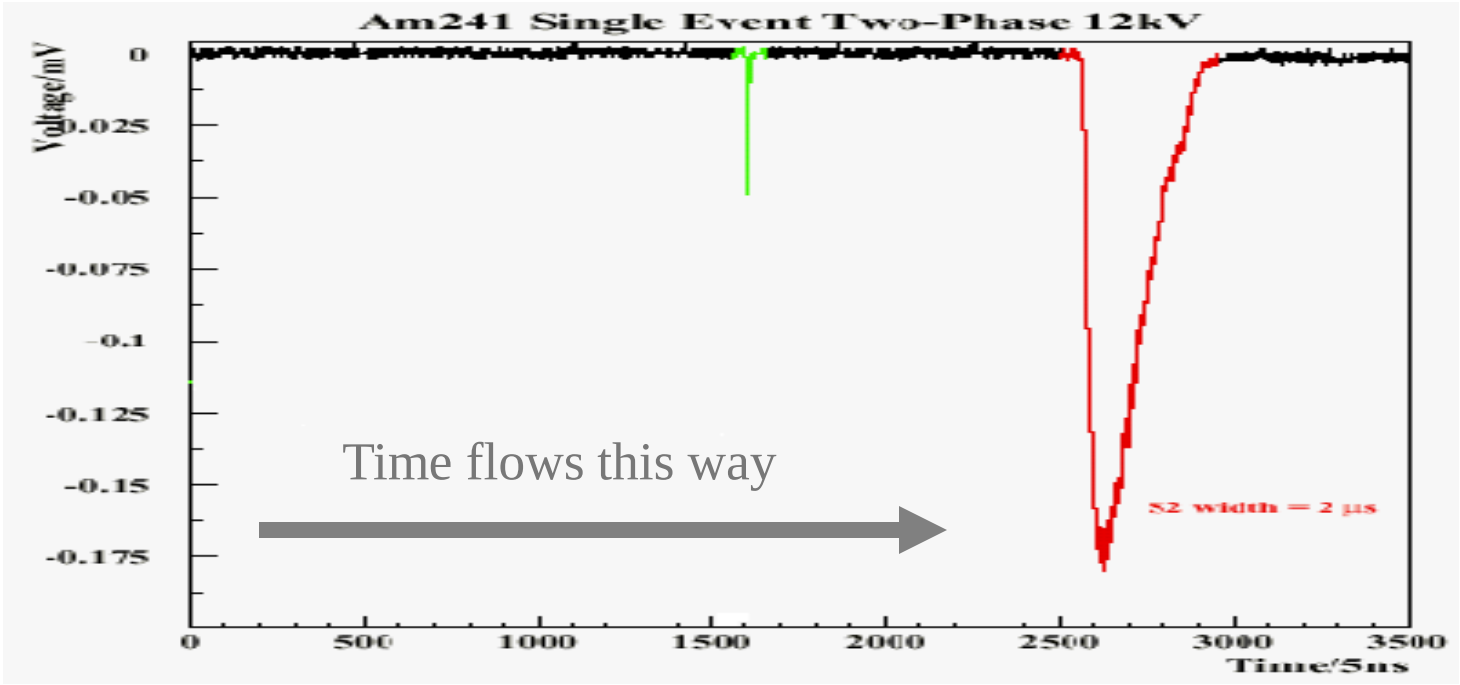
Secondary scintillation in gas phase (electroluminescence).

- Extract the areas under S1, S2, and the separation time between the S1 and S2.
- Time-stamp the data in order to correlate pulses in different channels.

Figure from: T.J.Sumner *et. al.*,
http://astro.ic.ac.uk/Research/Gal_DM_Search/report.html

Which data is interesting?

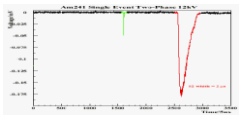
Baseline, not useful → Useful data → Not useful



- Select useful data (so-called *events*) and reject *baseline data*.

Figure from: T.J.Sumner *et. al.*,
http://astro.ic.ac.uk/Research/Gal_DM_Search/report.html

A single DPP channel



Analog signal input

Analog input stage

Nyquist filter

ADC

Sampling clock

Sample processor

Waveform memory

Trigger

Pulse information output

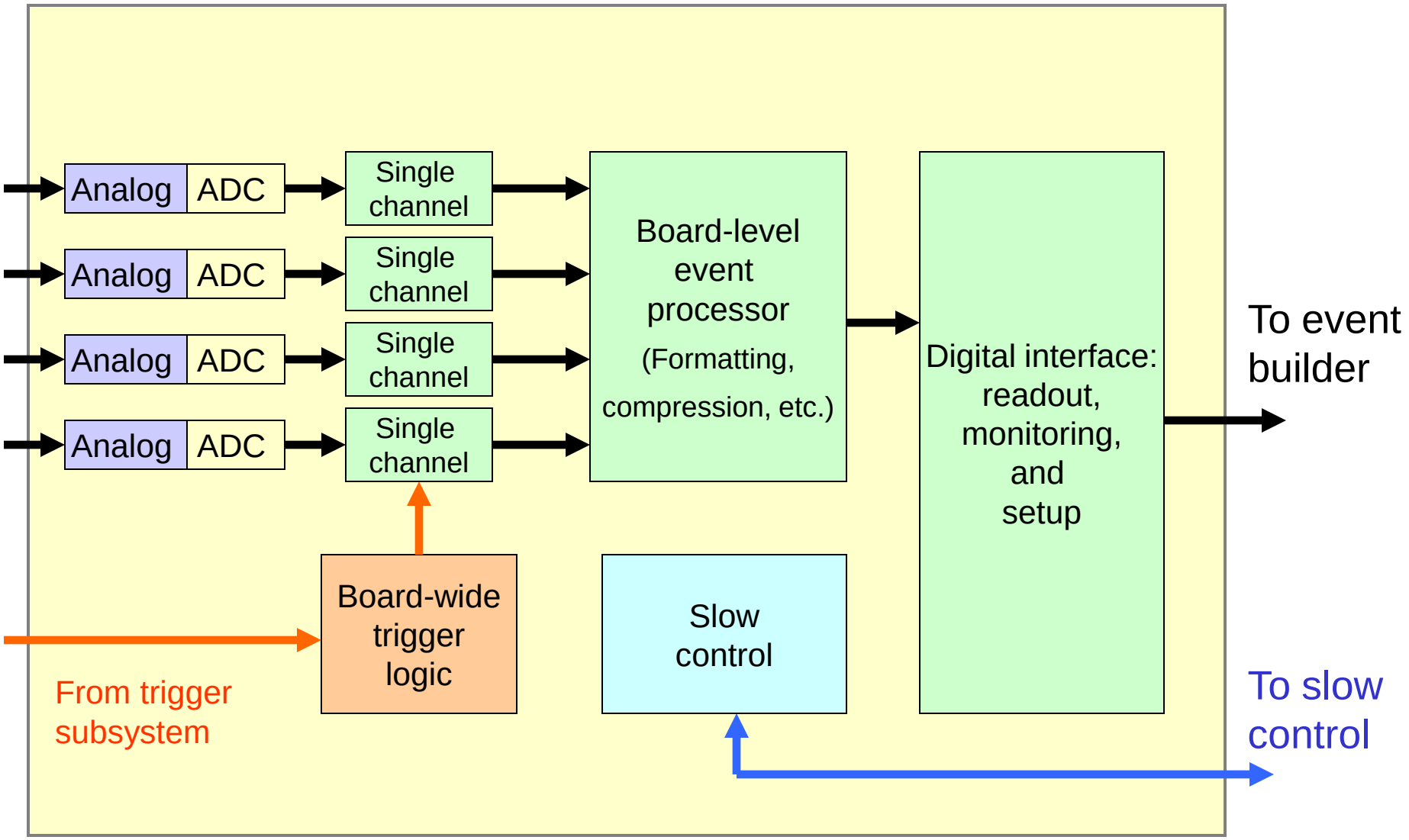
Optional external trigger in/out

analog

digital

Trigger subsystem pre-selects only "good data" to be recorded

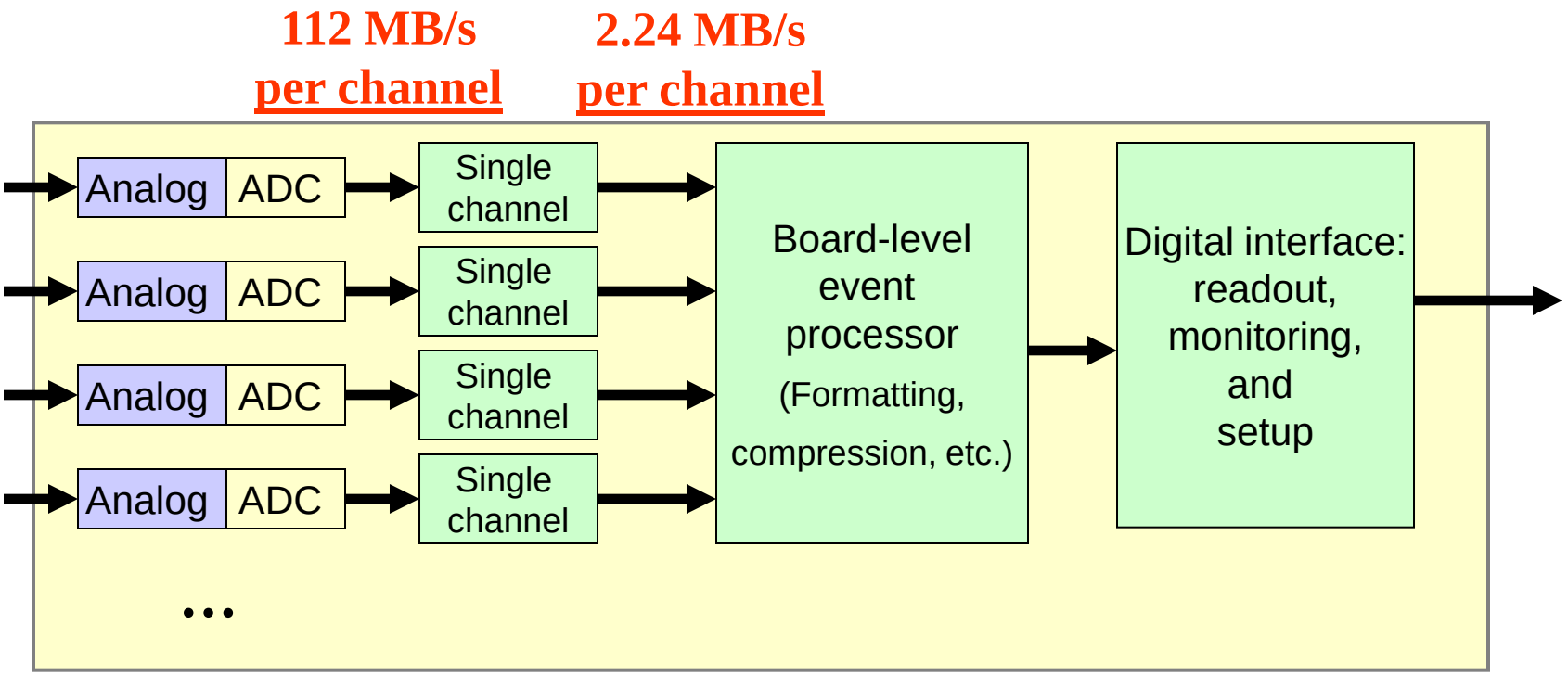
A multichannel DPP board



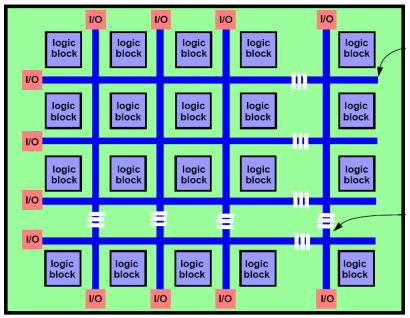
A multichannel DPP board

Assumptions:

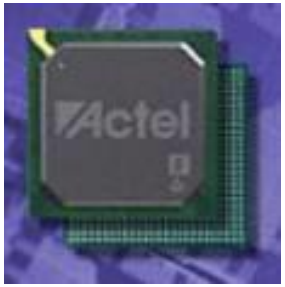
- 8 channels, 14-bit ADCs @ 64 MHz ➔ 8x112 MB/s = **896 MB/s**
- 200 μs ADC trace per event and 100 Hz ➔ 8x2.24 MB/s = **17.92 MB/s**
- Trigger compression factor = 50



How to implement DPP algorithms for real time applications ?



FPGA: Field Programmable Gate Array



FPGA: Field Programmable Gate Array

Born in the ‘80s from the CPLD (Complex Programmable Logic Devices) the main manufacturers are:

Xilinx: SRAM based devices

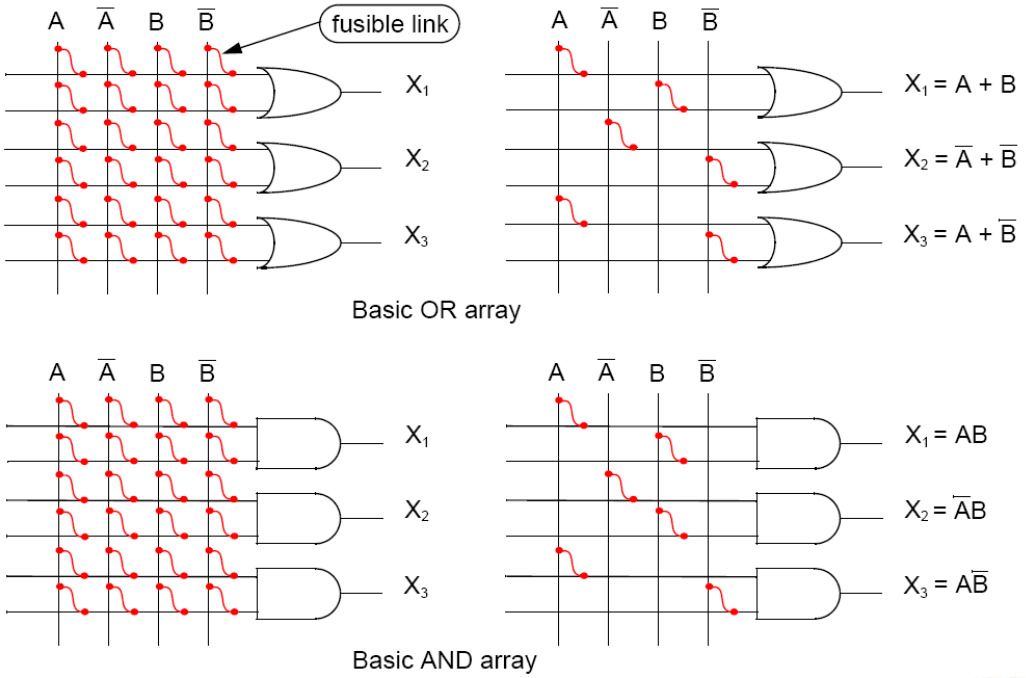
Altera: SRAM based devices

Actel: FLASH based devices

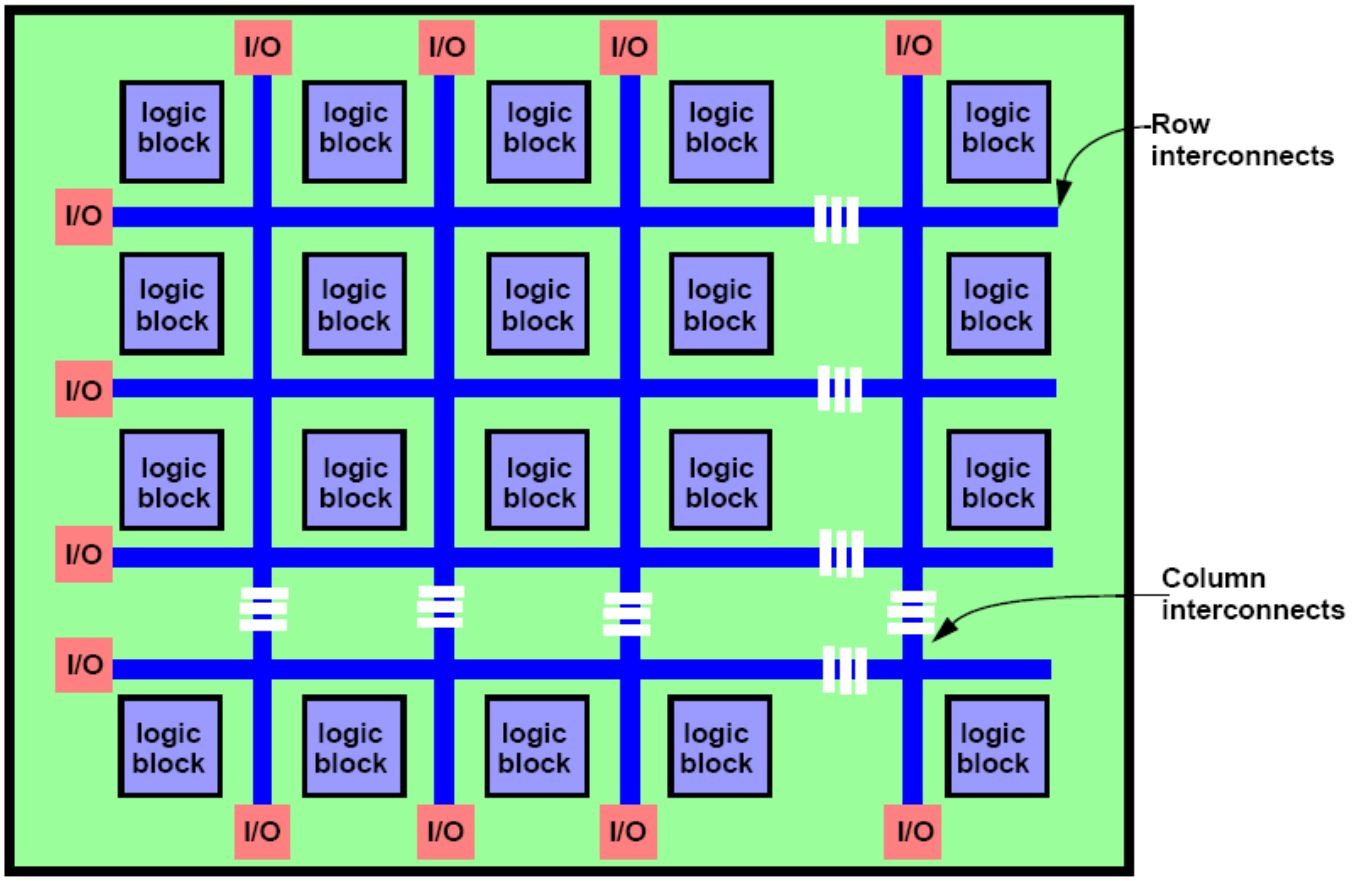
Historical Introduction

- The first programmable chips were PLAs (Programmable Logic Arrays): two level structures of AND and OR gates with user programmable connections.

- Programmable Array Logic devices were an improvement in structure and cost over PLAs. Today such devices are generically called Programmable Logic Devices (PLDs).

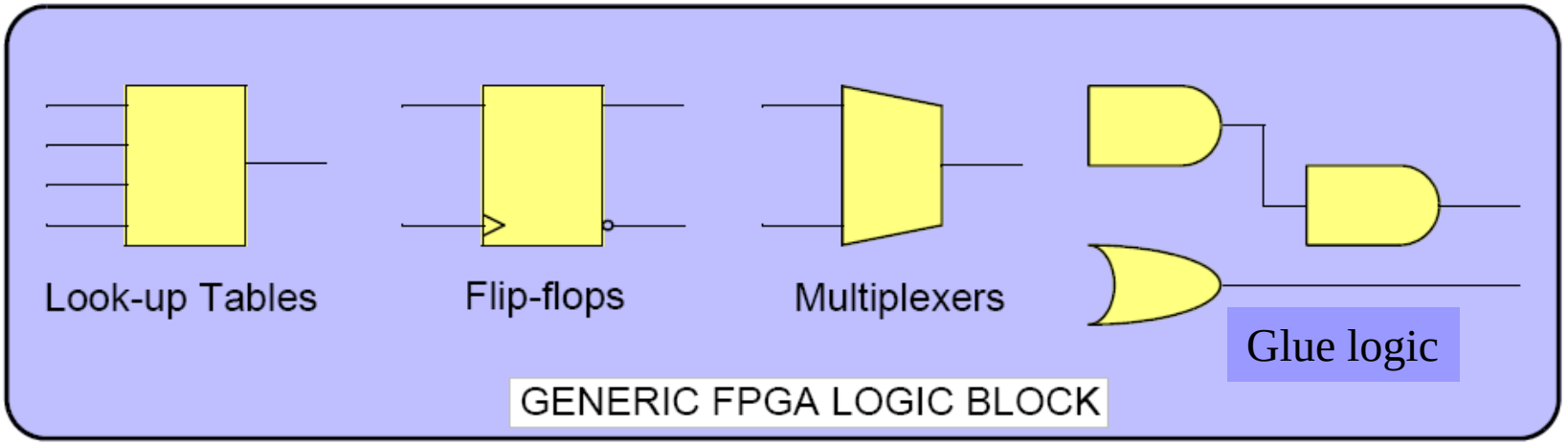


Architecture of a FPGA

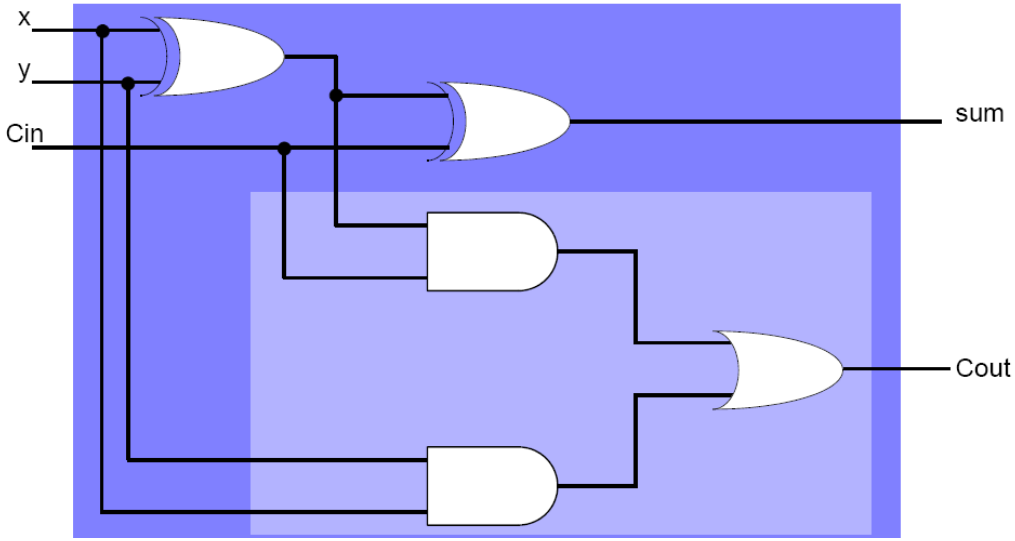


It is a user-programmable matrix of logic blocks with programmable interconnections that can implement **any** logic function or algorithm.

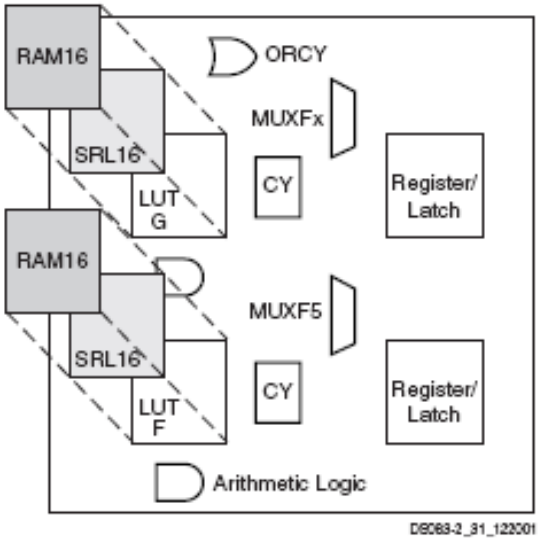
The logic block: a summary view



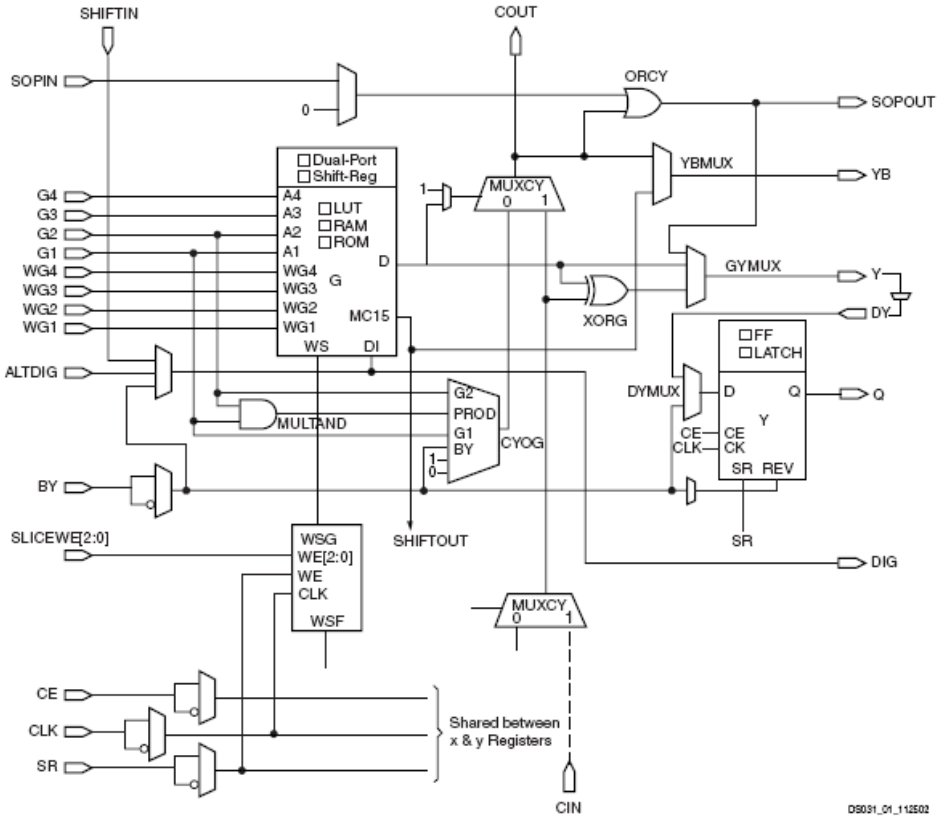
Example: using a LUT as a full adder.



A practical example: Xilinx Virtex II Pro family

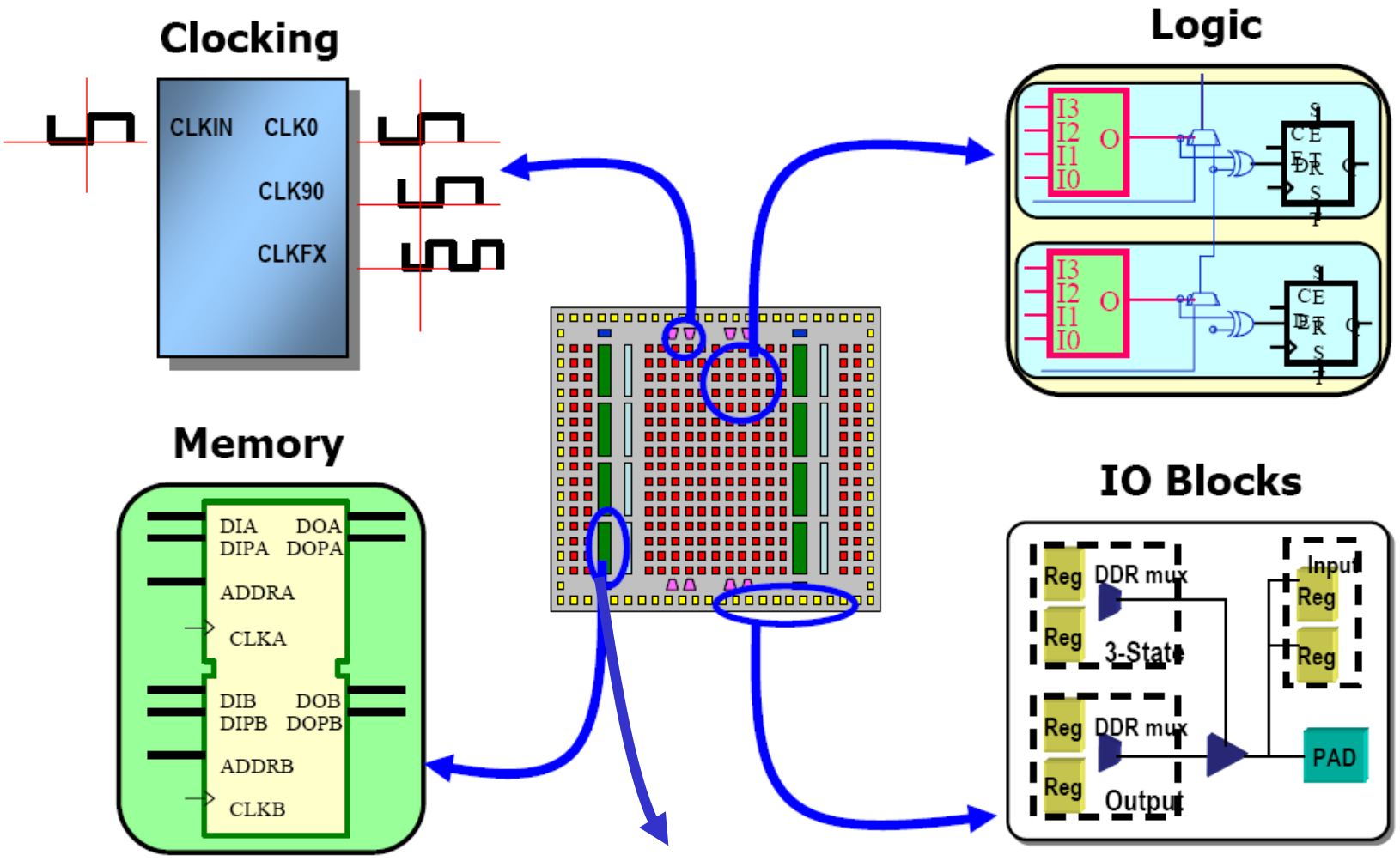


Slice



Detail of half-slice

Not only logic on larger and newer devices



DSP48E **MULTIPLIER + ACCUMULATOR**

Xilinx Virtex5 FXT family

Device	Array	Slices	DSP48E	Block RAM (Kb)	PowerPC	RocketIO	I/O banks	User I/O
FX30T	80x38	5120	64	2448	1	8	12	360
FX70T	160x38	11200	128	5328	1	16	19	640
FX100T	160x56	16000	256	8208	2	16	20	680
FX130T	200x56	20480	320	10728	2	20	24	840
FX200T	240x68	30720	384	16416	2	24	27	960

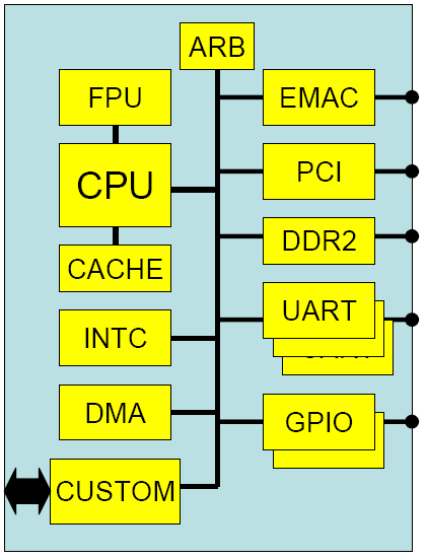
- **1 slice**: 4 LUTs and 4 flip-flops
- **1 DSP48E**: 1 25x18 multiplier, an adder and an accumulator
- **RocketIO** devices are designed to run fro 150 Mb/s to 6.5 Gb/s

Cost may be an issue: FX70T price for instance is about 500 EUROs

FPGA state of the art

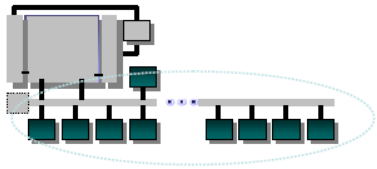
- In addition to logic gates and routing, in a modern FPGA you can find:
  - Embedded processors (soft or hard).
  - Multi-Gb/s transceivers with equalization and hard IP for serial standards as PCI Express and Gbit Ethernet.
  - Lots of embedded MAC units, with enough bits to implement single precision floating point arithmetic efficiently.
  - Lots of dual-port RAM.
  - Sophisticated clock management through DLLs and PLLs.
  - On-substrate decoupling capacitors to ease PCB design.
  - Digitally Controlled Impedance to eliminate on-board termination resistors.

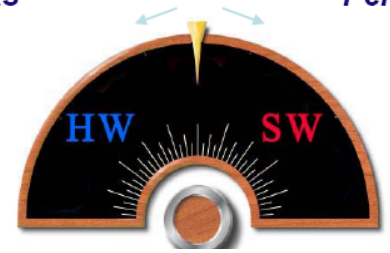
Why use embedded processors?



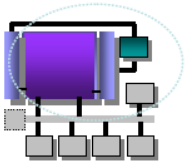
Customization: take only the peripherals you need and replicate them as many times as needed. Create your own custom peripherals.

Performing some software tasks in hardware can be expensive



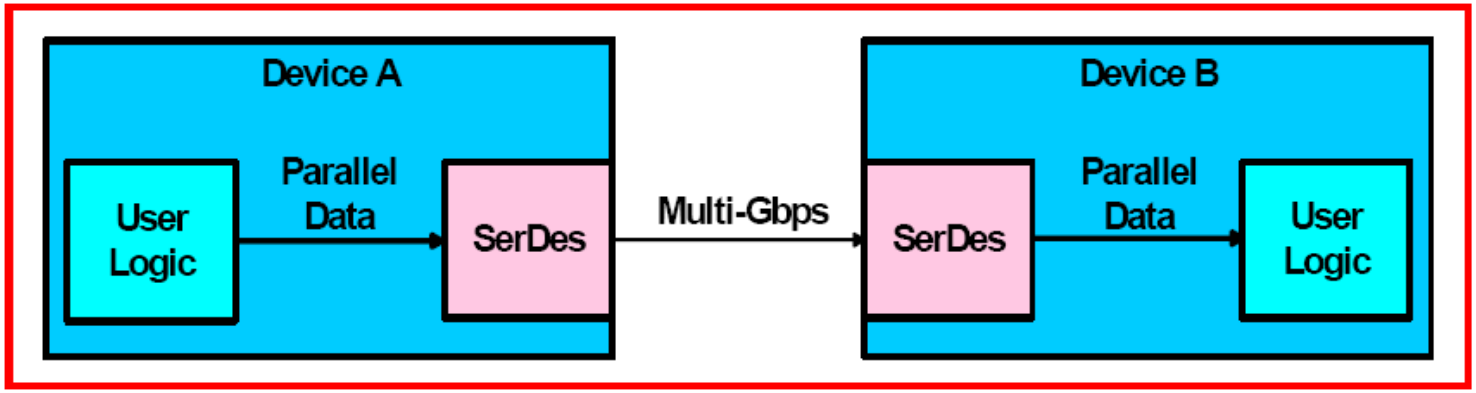


Performing some hardware tasks in software can be slow



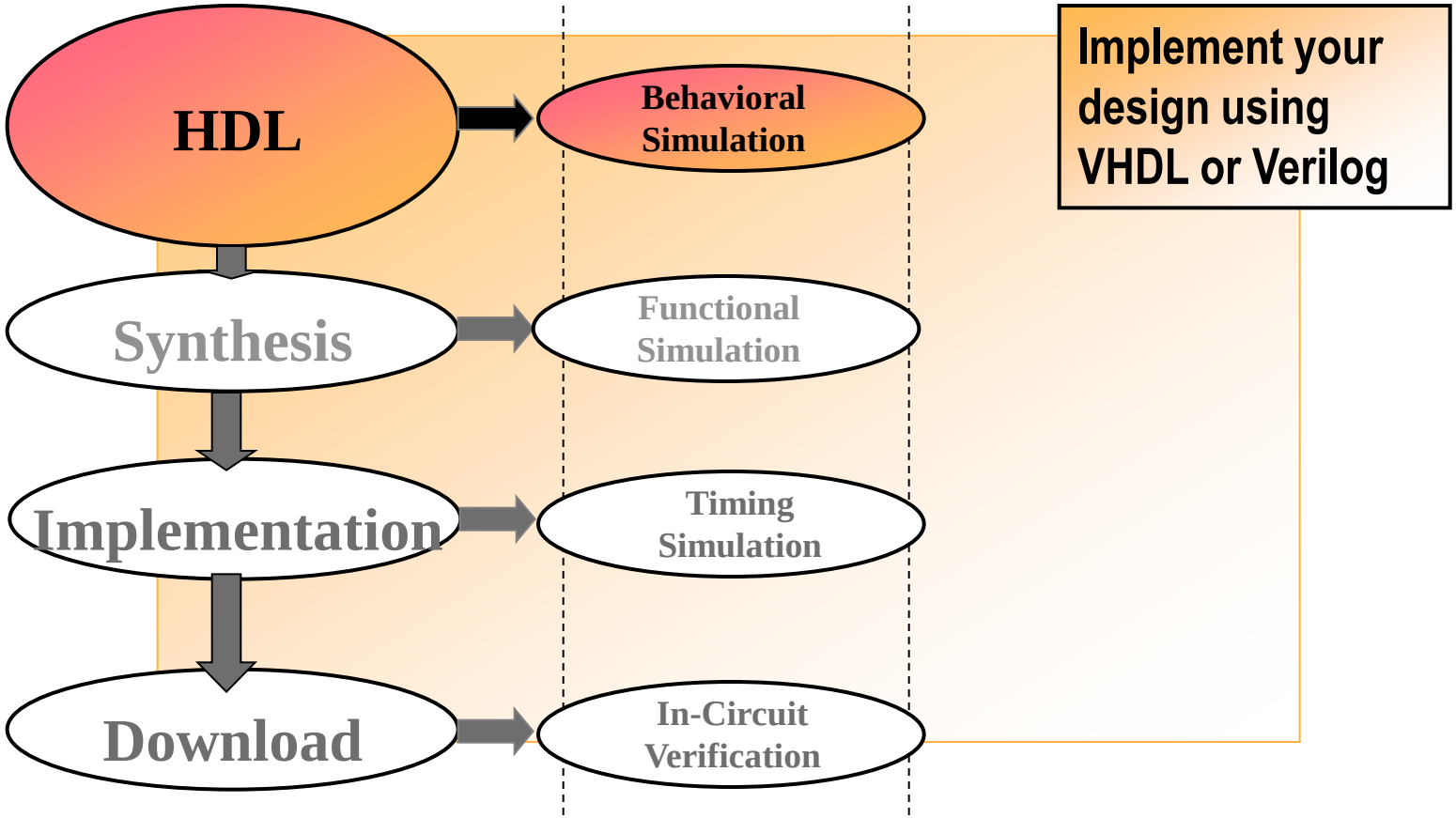
Strike optimum balance in system partitioning.

Serial signaling

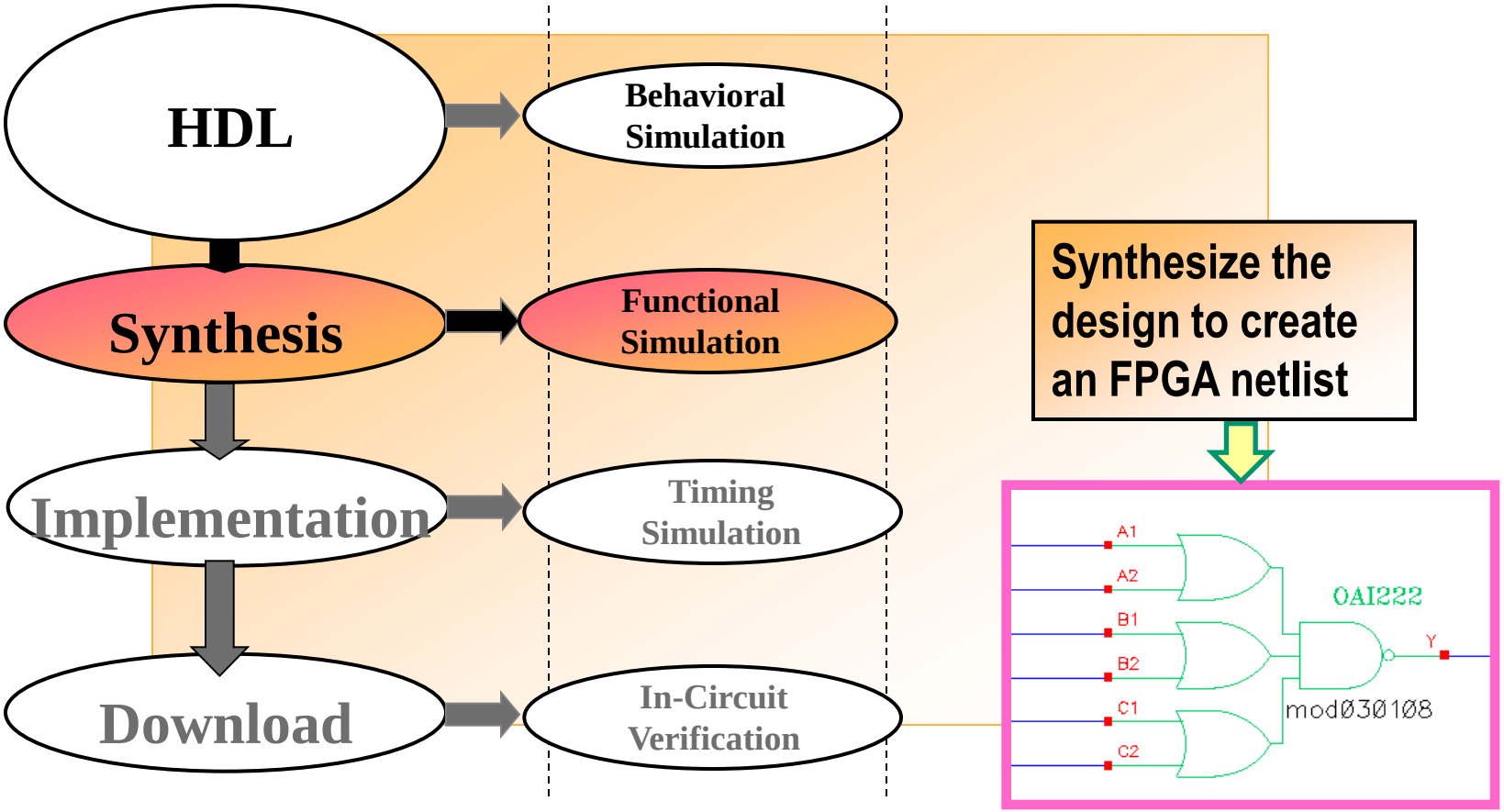


- Avoids clock/data skew by using embedded clock.
- Reduces EMI and power consumption.
- Simplifies PCB routing.

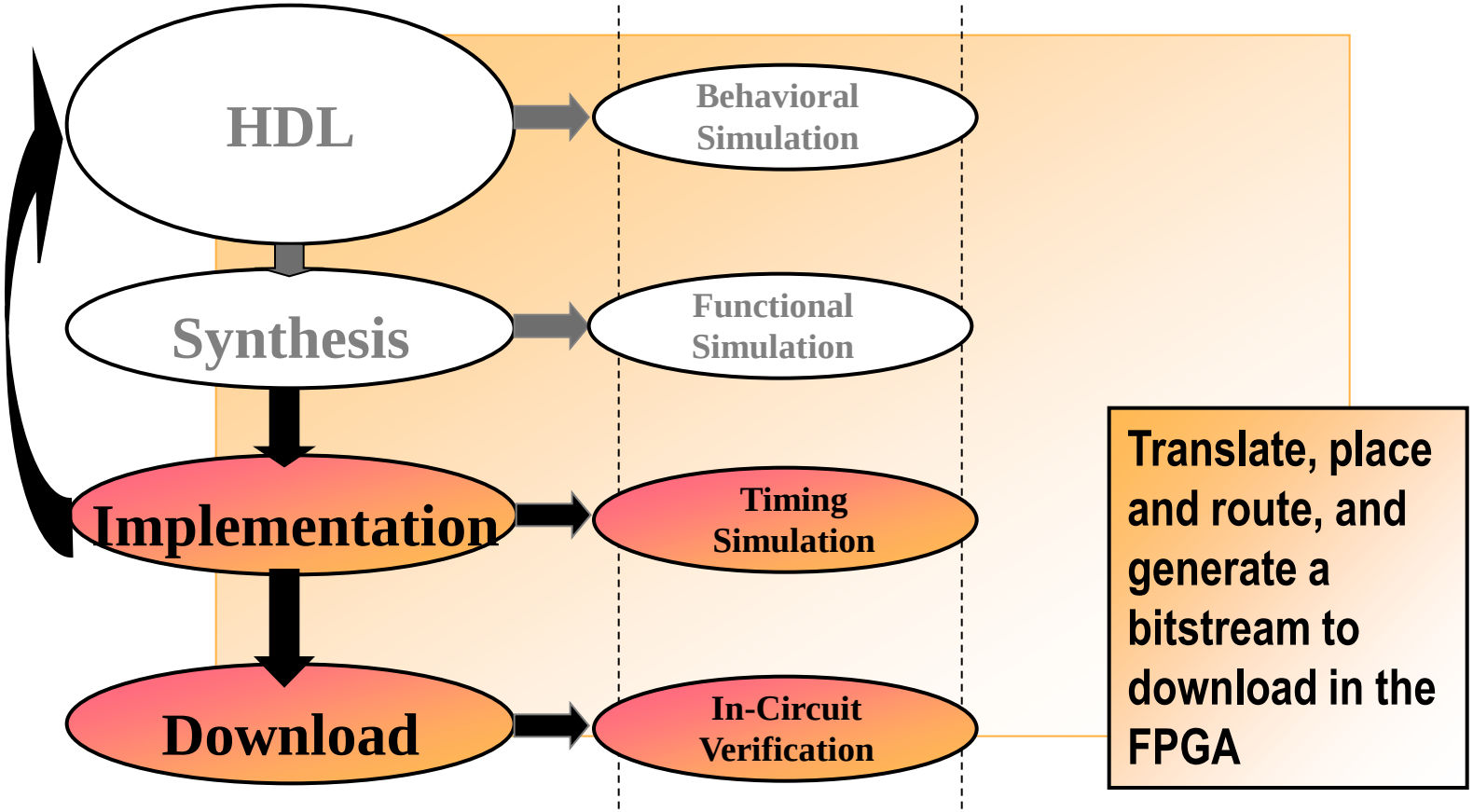
FPGA design flow



FPGA design flow



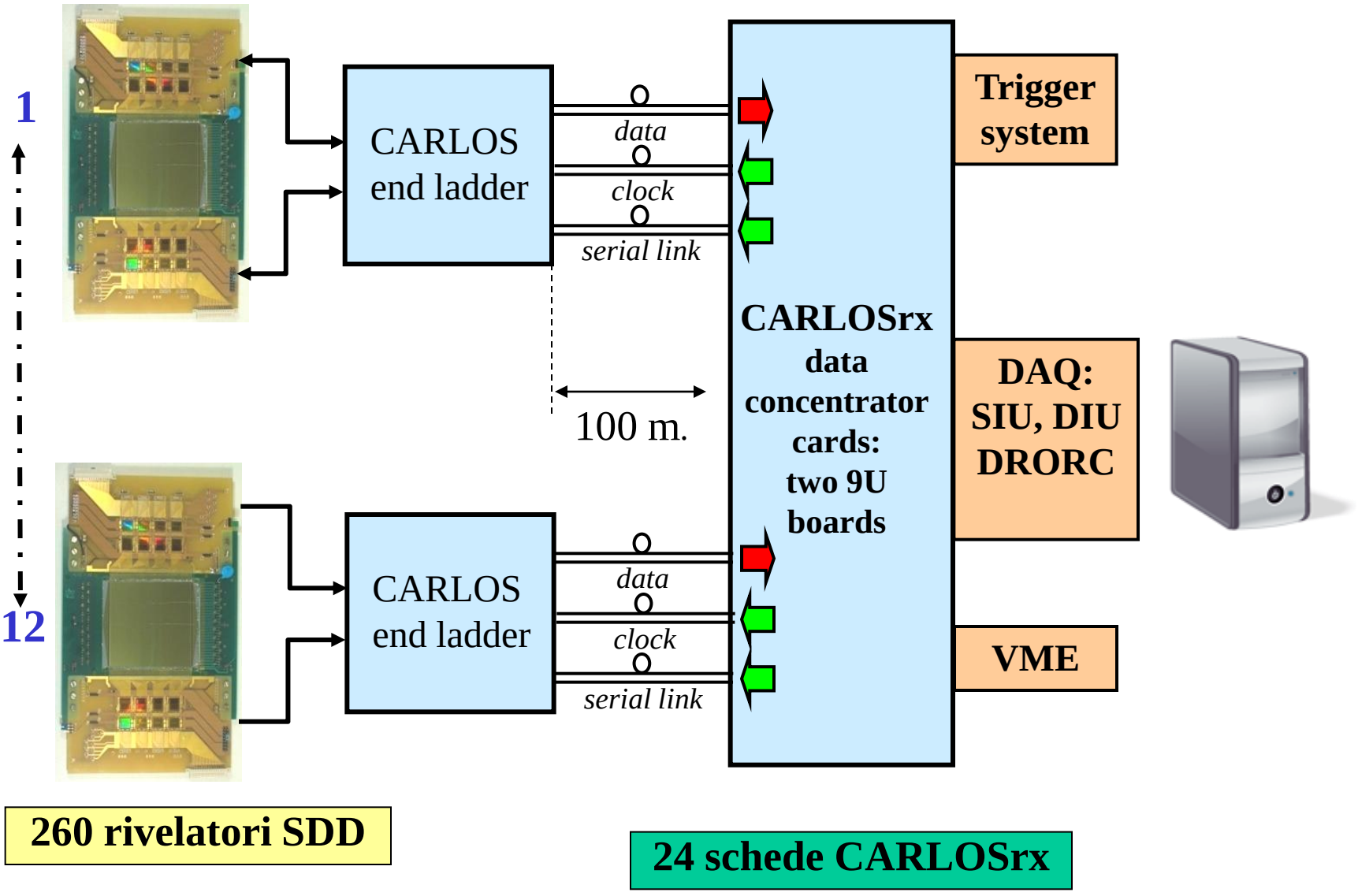
FPGA design flow



Now the debug phase begins: debug tools such as Xilinx Chipscope allow to watch signals inside the FPGA while working in real conditions and allow to reduce debug times.

Case study:
readout board for ALICE SDD

SDD readout chain



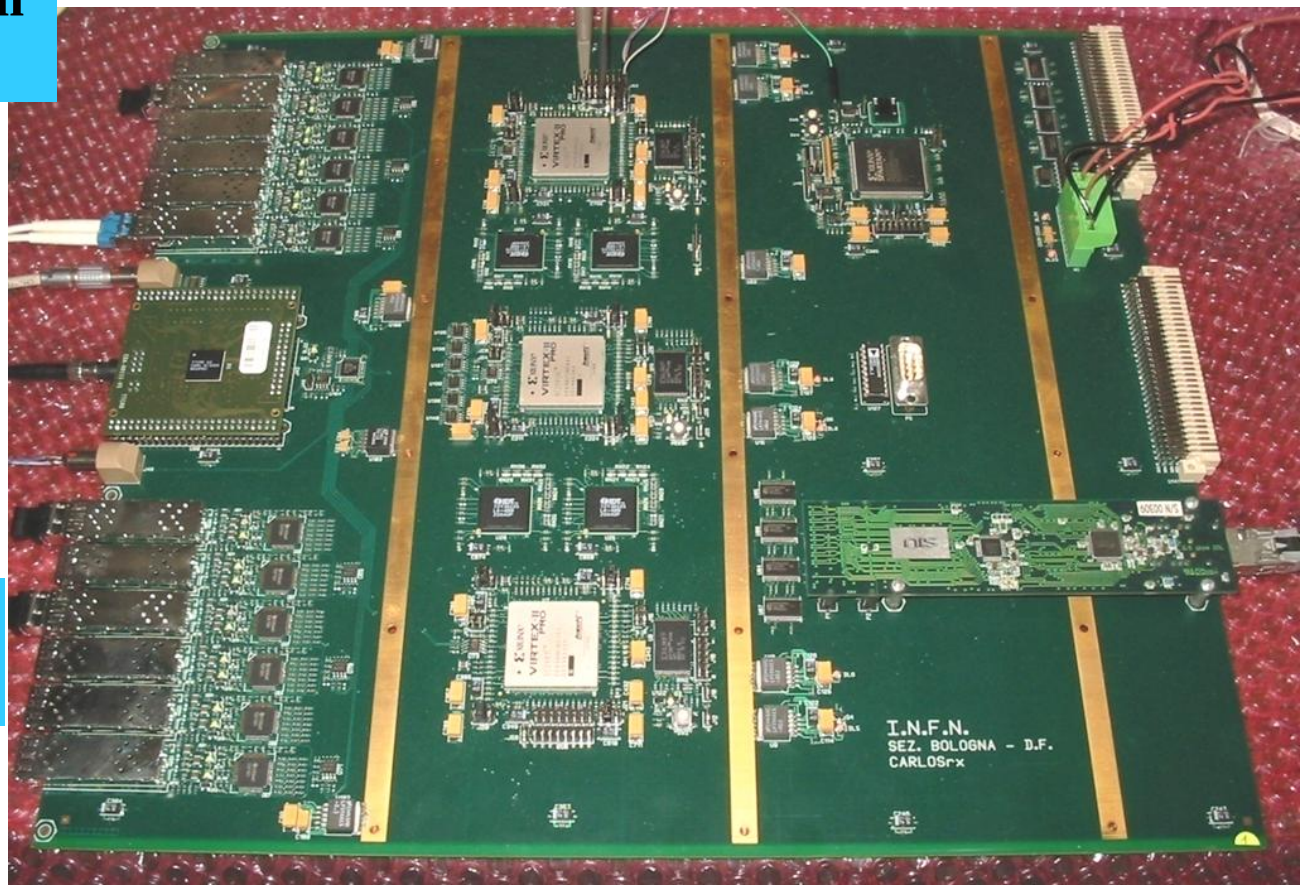
Scheda di readout CARLOSrx

6 moduli SDD

busy

TTCrq

6 moduli SDD

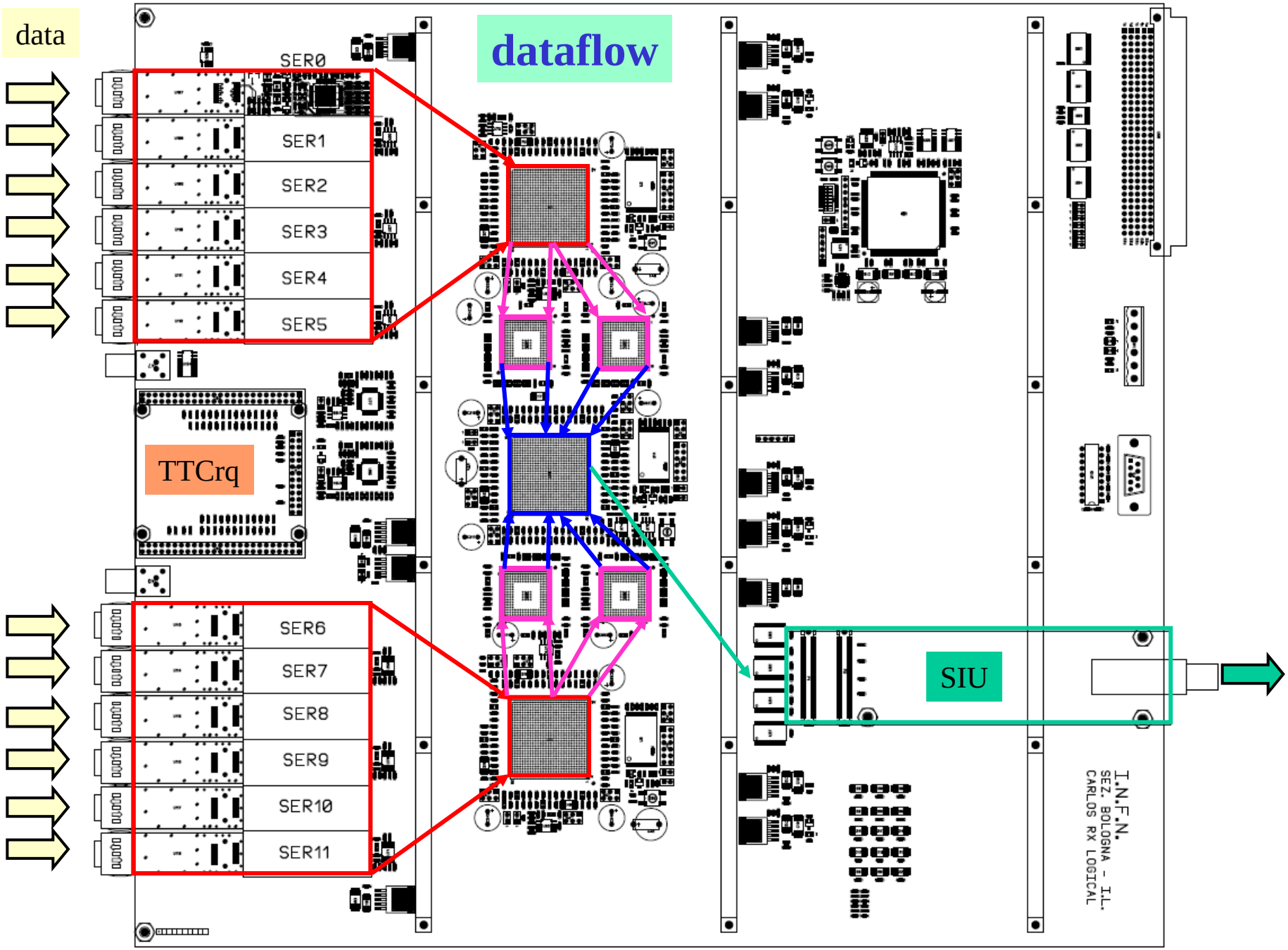


DDL

Input bandwidth: 12 x 800 Mbit/s

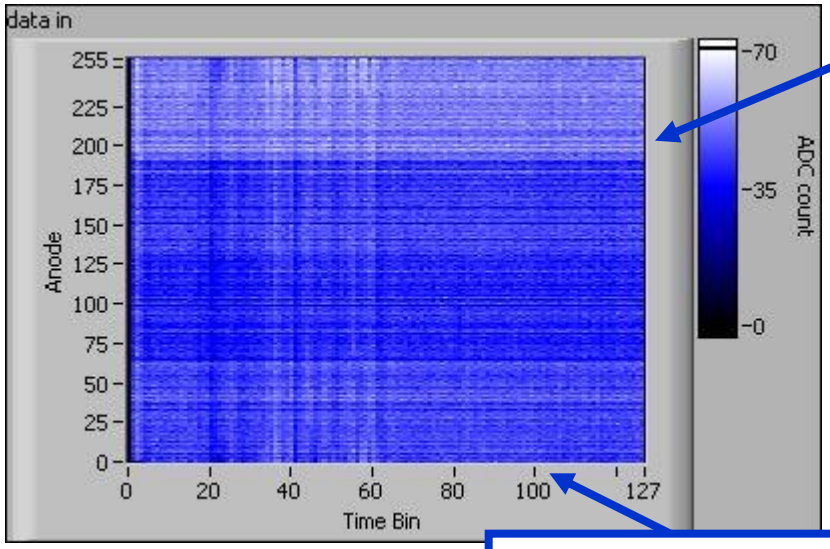
Output bandwidth: 2 Gbit/s

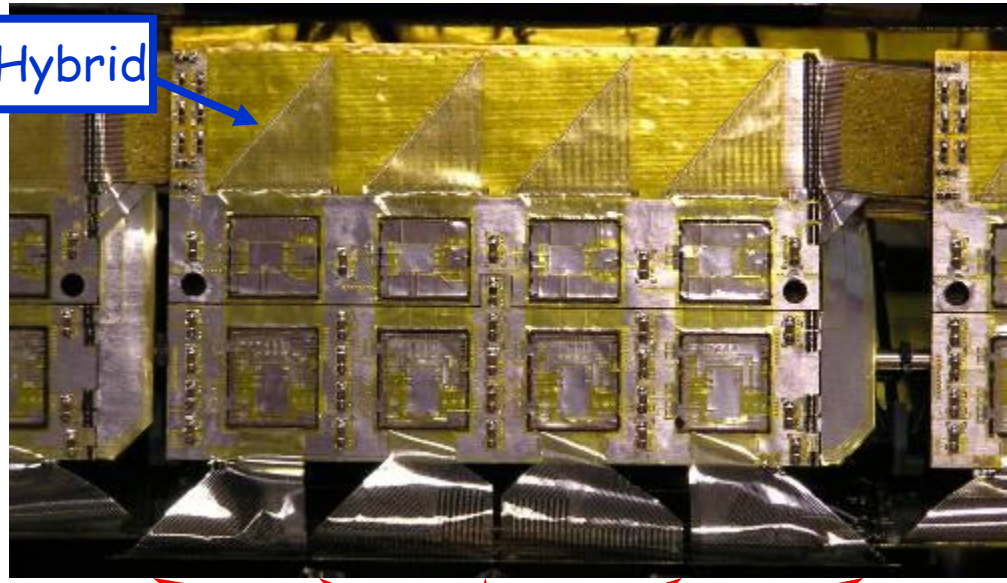
On-line data decoding, formatting and re-encoding with a different format

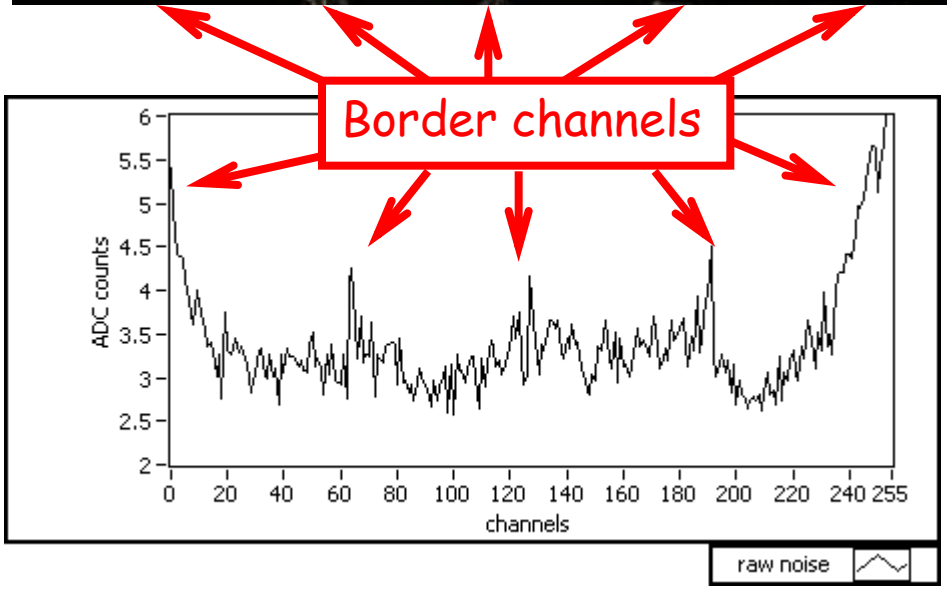

data
dataflow
SER0
SER1
SER2
SER3
SER4
SER5
TTCrq
SER6
SER7
SER8
SER9
SER10
SER11
SIU
I.N.F.N.
SEZ. BOLOGNA - I.L.
CARLOS RX LOGICAL

Work in progress:
common noise rejection with FPGAs

Common mode noise



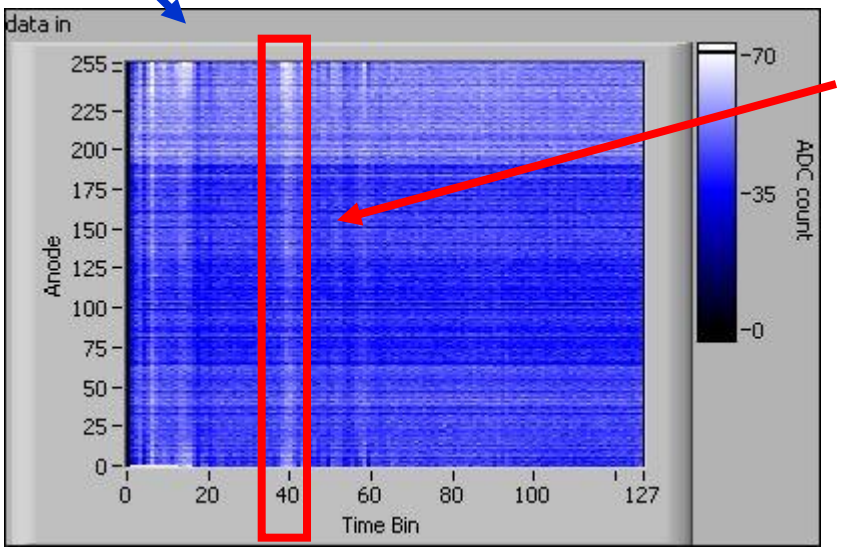




- The common mode noise is a coherent fluctuation of a group of electronic channels induced by external sources
- Vertical bands on Raw Data Plot
- Border channel effect due to microcables proximity

The Algorithm

Single Hybrid



i

j

Common mode shift for each time bin

pedestal

$$p_i = \frac{\sum_{j=1}^{TB} a_{ij}}{TB}$$

$$s_j = \frac{\sum_{i=1} (a_{ij} - p_i)}{m}$$

$$m \le n = 256$$

Average over a number of events

no hits and dead channels

Evaluated on-line event-by-event

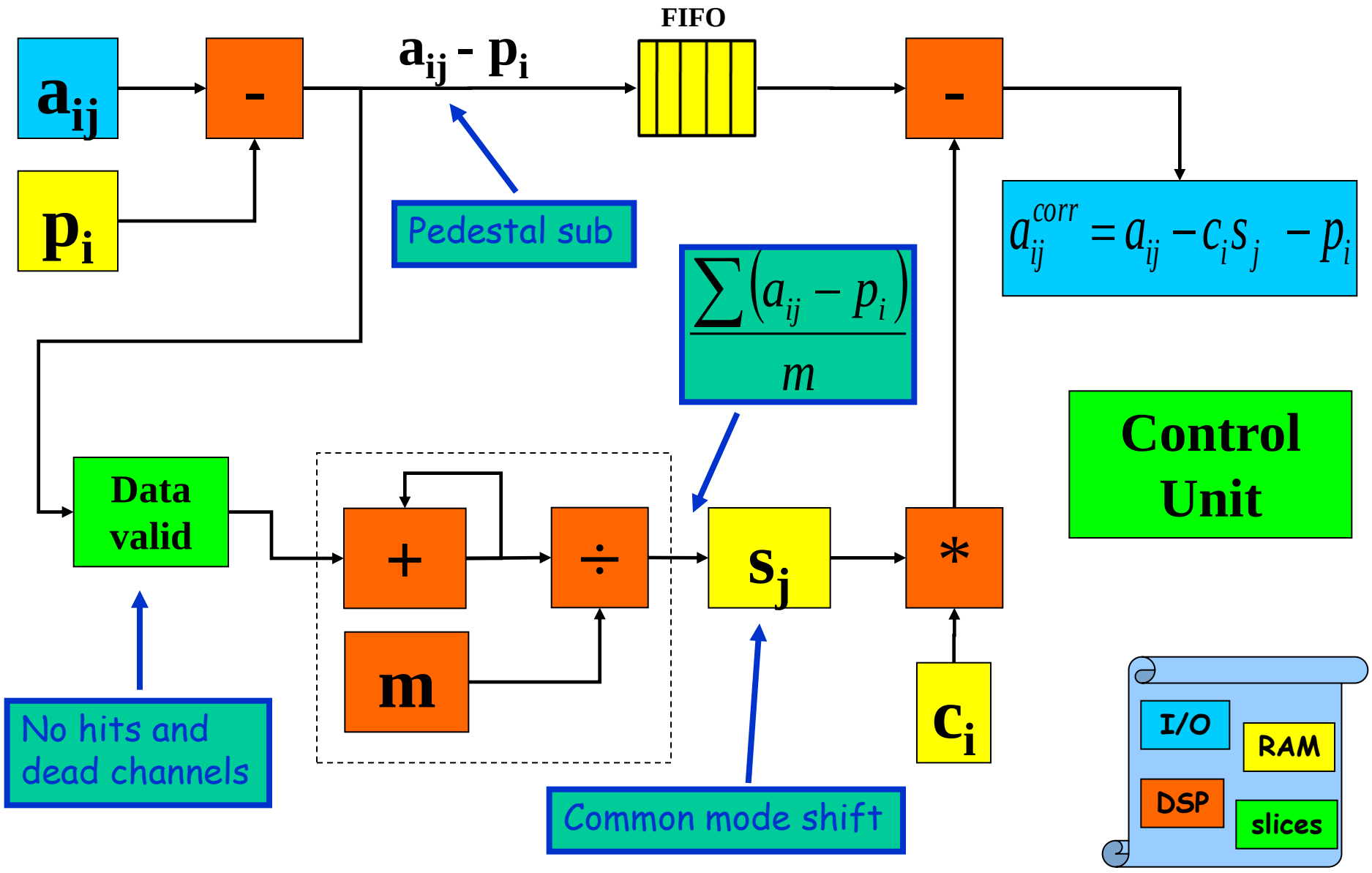
Correlation coefficient for each channel

$$a_{ij}^{corr} = a_{ij} - c_i s_j^* - p_i$$

$$c_i = \frac{\sum_j (a_{ij} - p_i) s_j}{\sum_j s_j^2}$$

common mode and pedestal subtraction

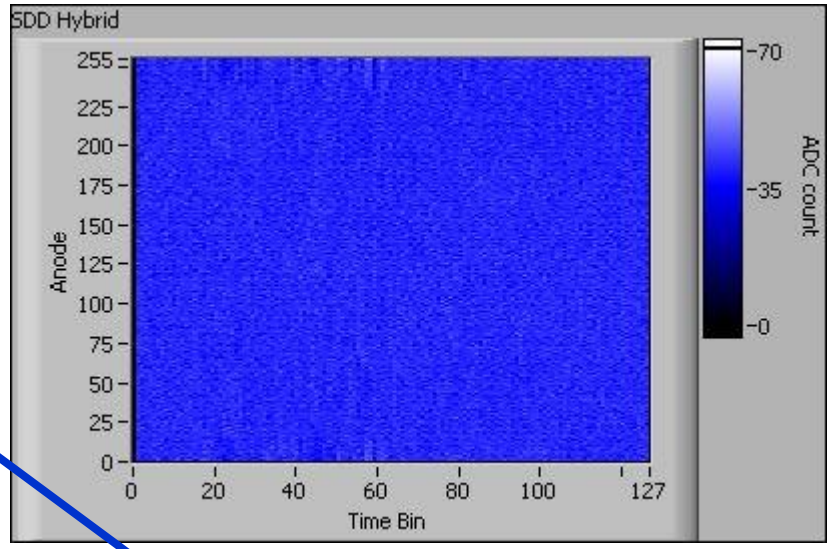
FPGA Implementation

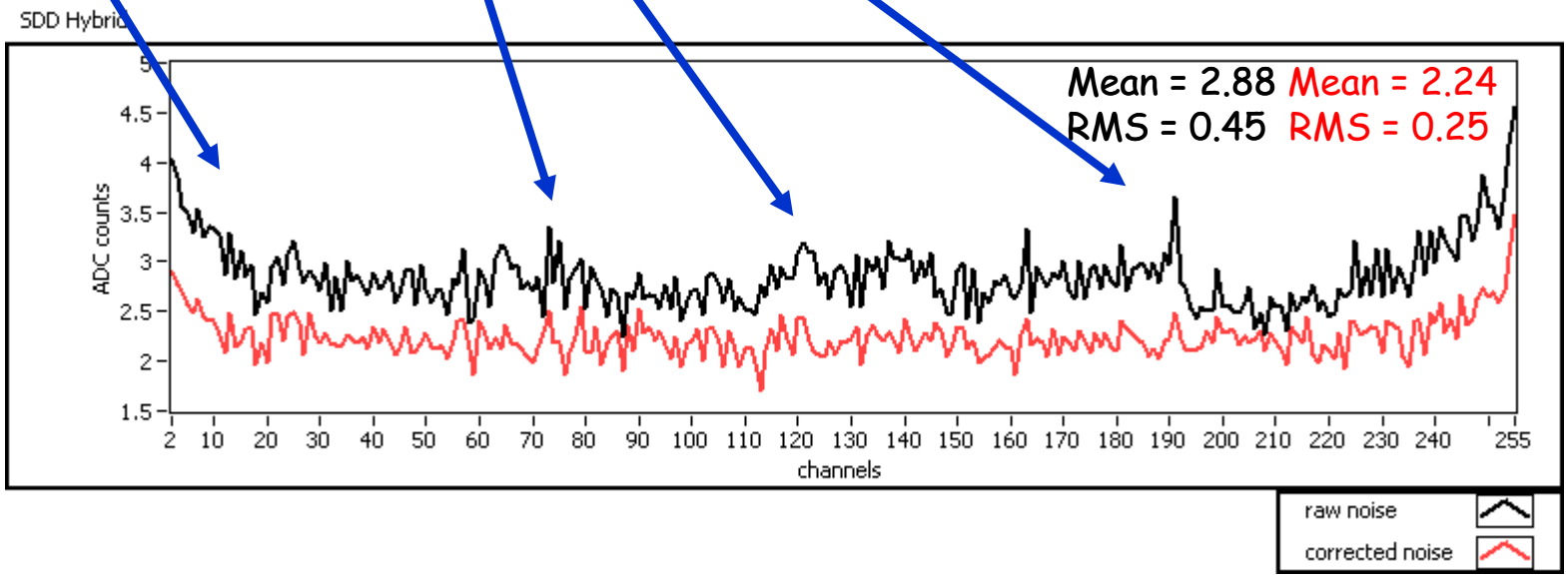


Single Hybrid Test results

Minimize border channel effects

work in progress…
calculate common mode noise for a restricted group of channels

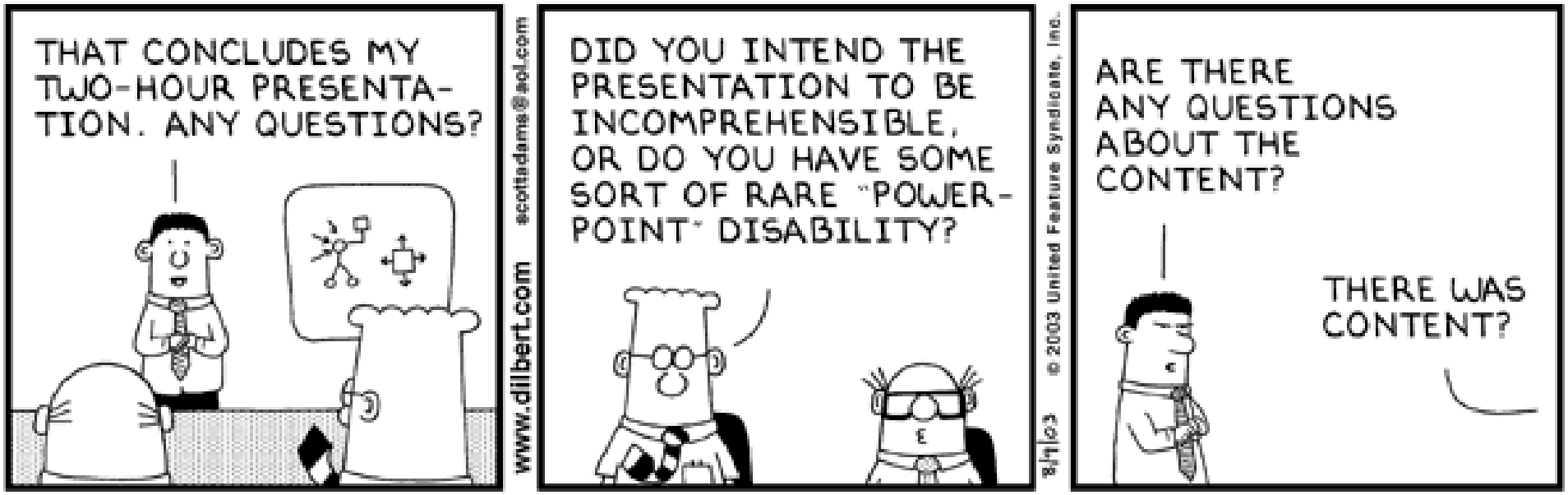




Conclusions

Up to date, FPGAs are the most suitable device for implementing DPP. In fact:

- FPGAs feature enormous logic power with DSP optimized blocks (for MAC operations);
- FPGAs can manage in real time data throughput of the order of magnitude of GBit/s on several channels at the same time;
- FPGAs are flexible, they can be reprogrammed anytime
- The design flow is straight-forward (if you know VHDL)
- IP cores are provided by manufacturers and can easily be found over the Web.
- Price is an issue (large FPGAs are very expensive), but you can choose the one who is tailored to your needs.



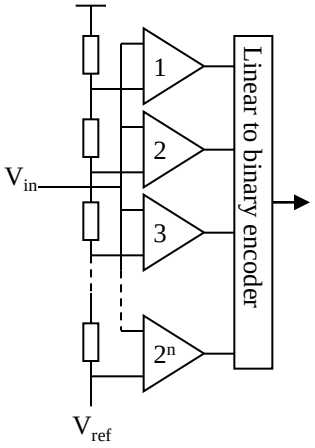
References

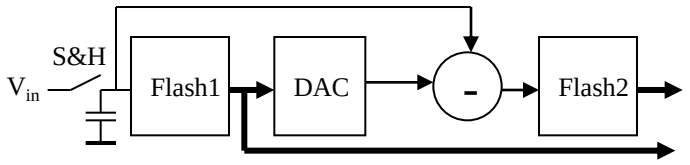
• "**The scientist and Engineer's Guide to Digital Signal Processing**" by Steven W. Smith, PhD
• "**Understanding Digital Signal Processing**" by Richard G. Lyons
• For FPGA:
  • http://www.xilinx.com
  • http://www.altera.com
  • http://www.actel.com
• For DSP:
  • http://www.ti.com
  • http://www.analog.com
• N07-4: "**A Digital Filter with Common Mode Noise Rejection for ALICE Silicon Drift Detector**", NSS-MIC2009

Backup slides

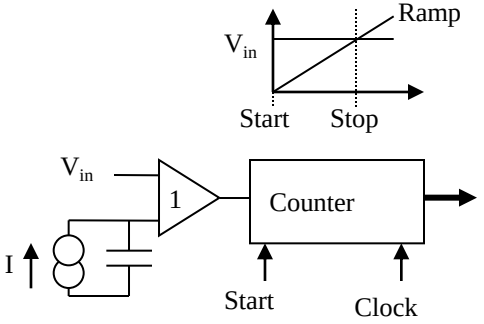
ADC architectures

- Flash
  - A discriminator for each of the 2^n codes
  - New sample every clock cycle
  - Fast, large, lots of power, limited to ~8 bits
  - Can be split into two sub-ranging Flash $2x2^{n/2}$ discriminators: e.g. 16 instead of 256 plus DAC
    - Needs sample and hold during the two stage conversion process



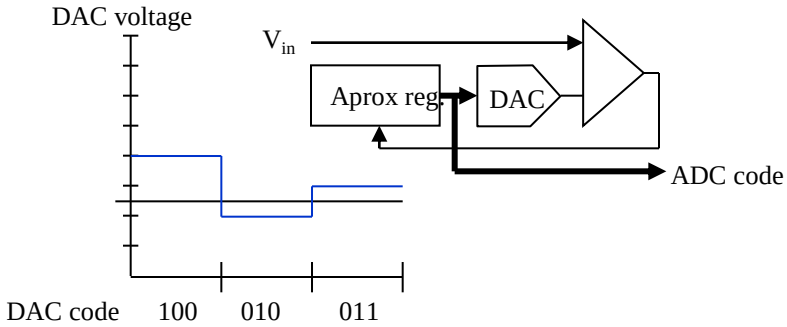


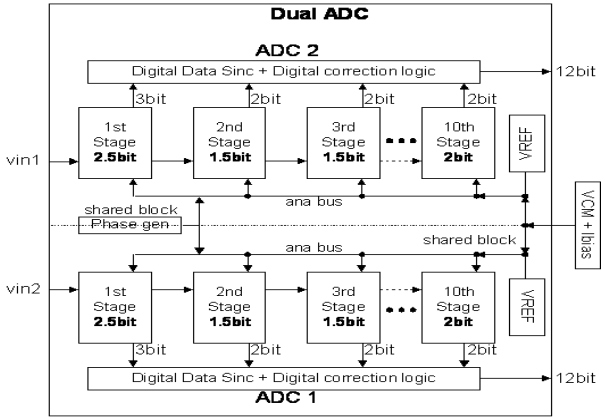
- Ramp
  - Linear analog ramp and count clock cycles
  - Takes 2^n clock cycles
  - Slow, small, low power, can be made with large resolution

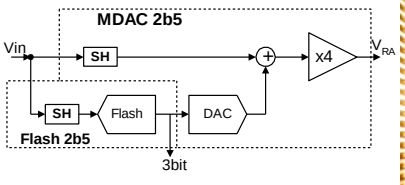


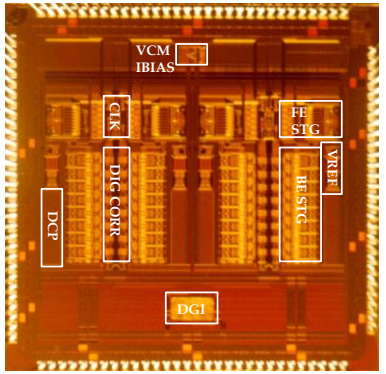
ADC architectures

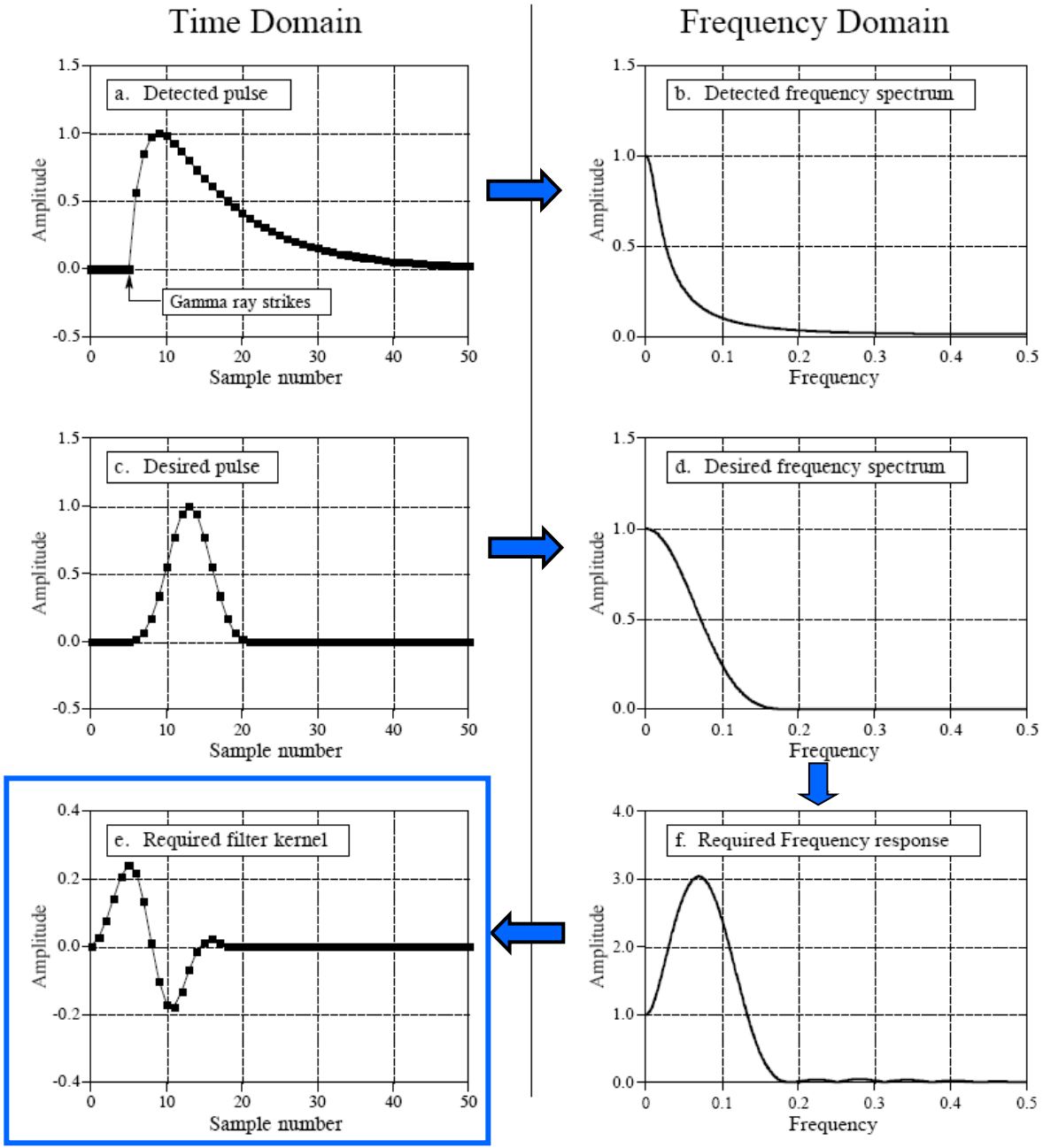
- Successive approximation
  - Binary search via a DAC and single discriminator
  - Takes n clock cycles
  - Relatively slow, small, low power, medium to large resolution
- **Pipelined**
  - Determines “one bit” per clock cycle per stage
    - Extreme type of sub ranging flask
  - n stages
  - In principle 1 bit per stage but to handle imperfections each stage normally made with ~2bits and n*2bits mapped into n bits via digital mapping function that “auto corrects” imperfections
  - Makes a conversion each clock cycle
  - Has a latency of n clock cycles
    - Not a problem in our applications except for very fast triggering
  - Now dominating ADC architecture in modern CMOS technologies and impressive improvements in the last 10 years: speed, bits, power, size



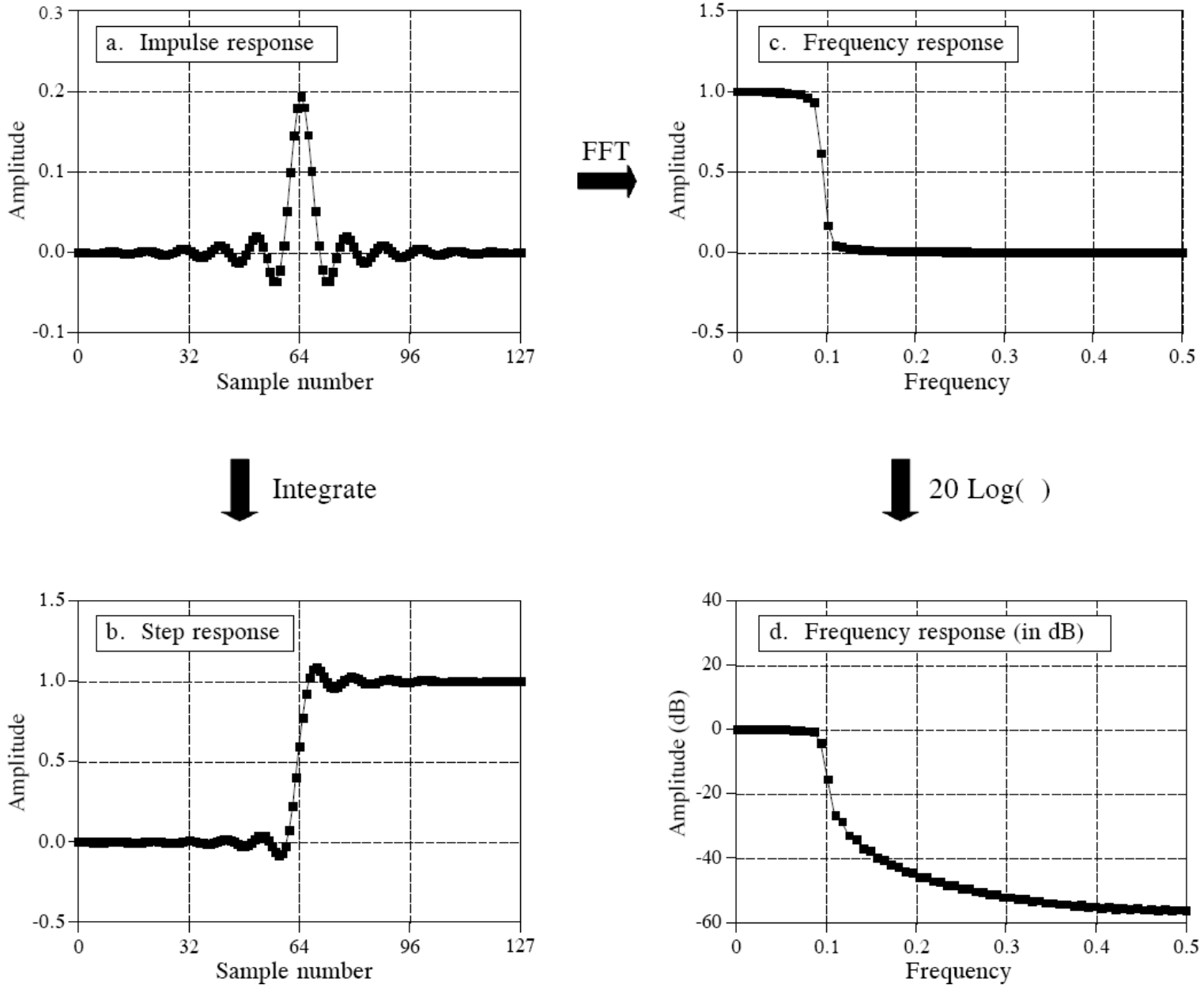






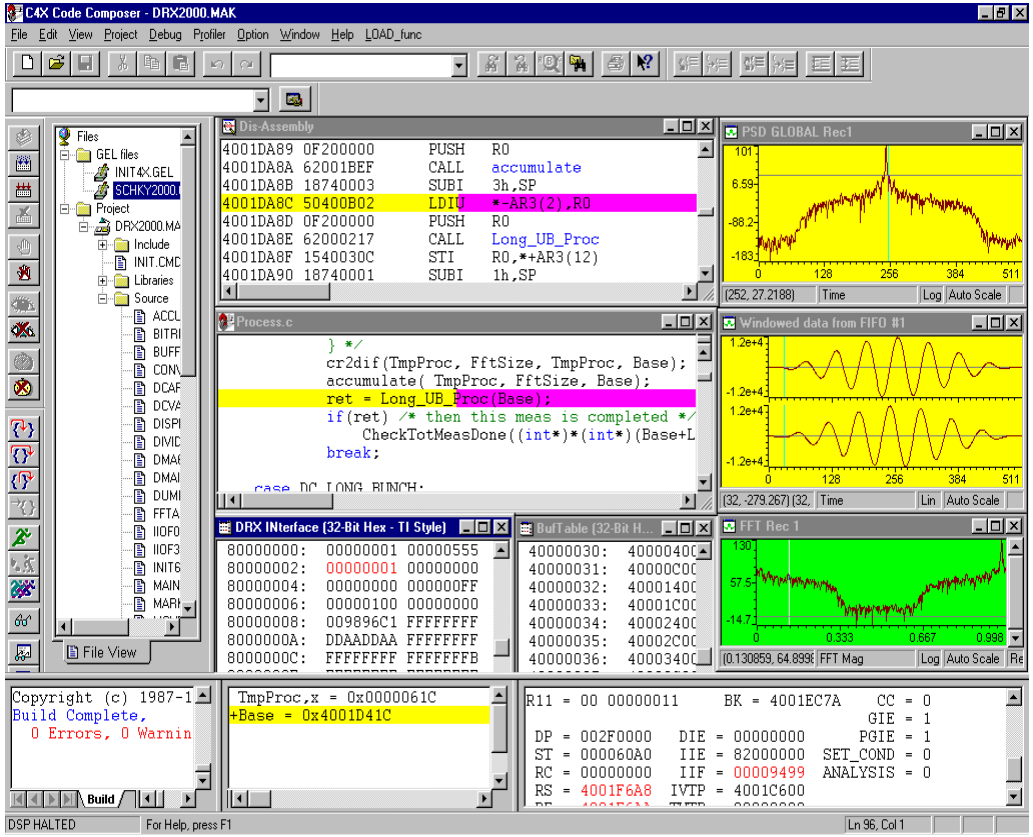

Time Domain
Frequency Domain
a. Detected pulse
Gamma ray strikes
Amplitude
Sample number
b. Detected frequency spectrum
Frequency
c. Desired pulse
d. Desired frequency spectrum
e. Required filter kernel
f. Required Frequency response

Filter parameters



2.1 DSP evolution: s/w tools

- Spectacular evolution!
- Advanced compilers
  - Deal with h/w complexity
  - Efficient high-level languages
- Graphical programming
  - MATLAB
  - NI LabVIEW DSP Module (Hyperception RIDE)



Code Composer for TI 'C40 DSPs (CERN, 1999)

- High-performance simulators, emulators & debugging facilities
  - High-visibility into target (~no interferences)
  - Multiple DSP development & debugging in same JTAG chain.

DSPs evolution: device integration

	1980	1990	2000	≥ 2010
Die size [mm]	50	50	50	5
Technology [µm]	3	0.8	0.1	0.02
MIPS	5	40	5000	50000
MHz	20	80	1000	10000
RAM [Bytes]	256	2000	32000	1 million
Price [$]	150	15	5	0.15
Power [mW/MIPS]	250	12.5	0.1	0.001
Transistors	50000	500000	5 million	60 million
Wafer size [in.]	3	6	12	12

PROJECTED VALUES

Current mainstream DSPs

3 main manufacturers: **Texas Instruments** (TI), **Analog Devices** (ADI) & **Freescale semiconductor** (formerly Motorola).

Mostly used for accelerators – TI & ADI

TI DSP families: TMS320Cxxxx

- **'C2x**: digital signal controller.
- **'C5x**: power-efficient.
- **'C6x**: high-performance.

ADI DSP families:

- **SHARC**: first ADI family (now 3 generations).
- **TigerSHARC**: high-performance for multiprocessor systems.
- **Blackfin**: high-performance, low power.

DSP core architecture: intro

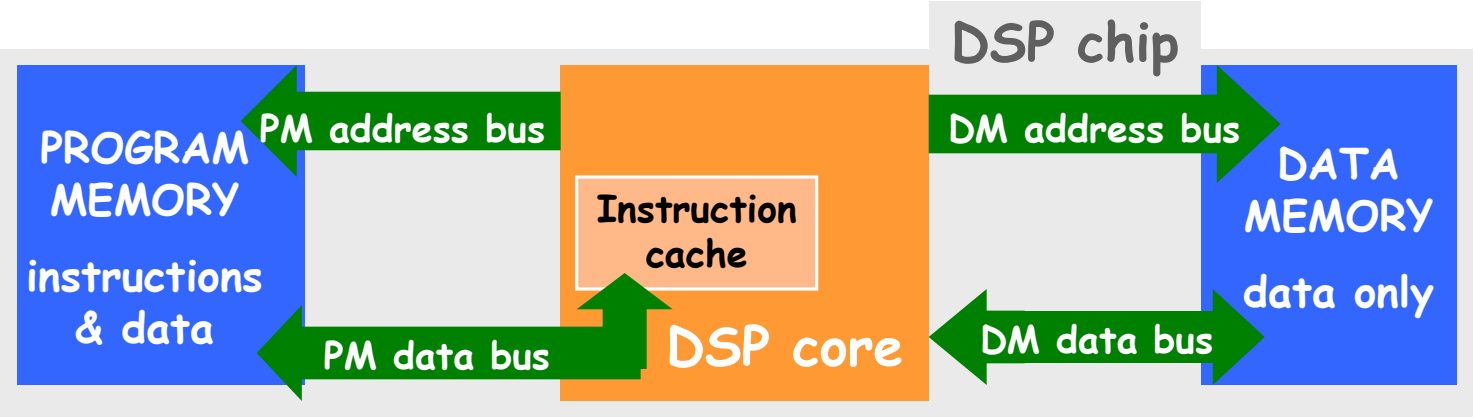
Shaped by **predictable real-time DSPing** !

ex: FIR $y(n) = \sum_{k=0}^{M} a_k \cdot x(n-k)$

Requirements	How
3.2 Fast data access	High-BW memory architectures. Specialised addressing modes. Direct Memory Access (DMA).
3.3 Fast computation	MAC-centred. Pipelining. Parallel architectures (VLIW, SIMD).
3.4 Numerical fidelity	Wide accumulator regs, guard bits ..
3.5 Fast-execution control	H/w assisted zero-overhead loops, shadow registers …

Fast data access

a) High-BW memory architectures



Harvard + cache = Super Harvard Architecture → SHARC

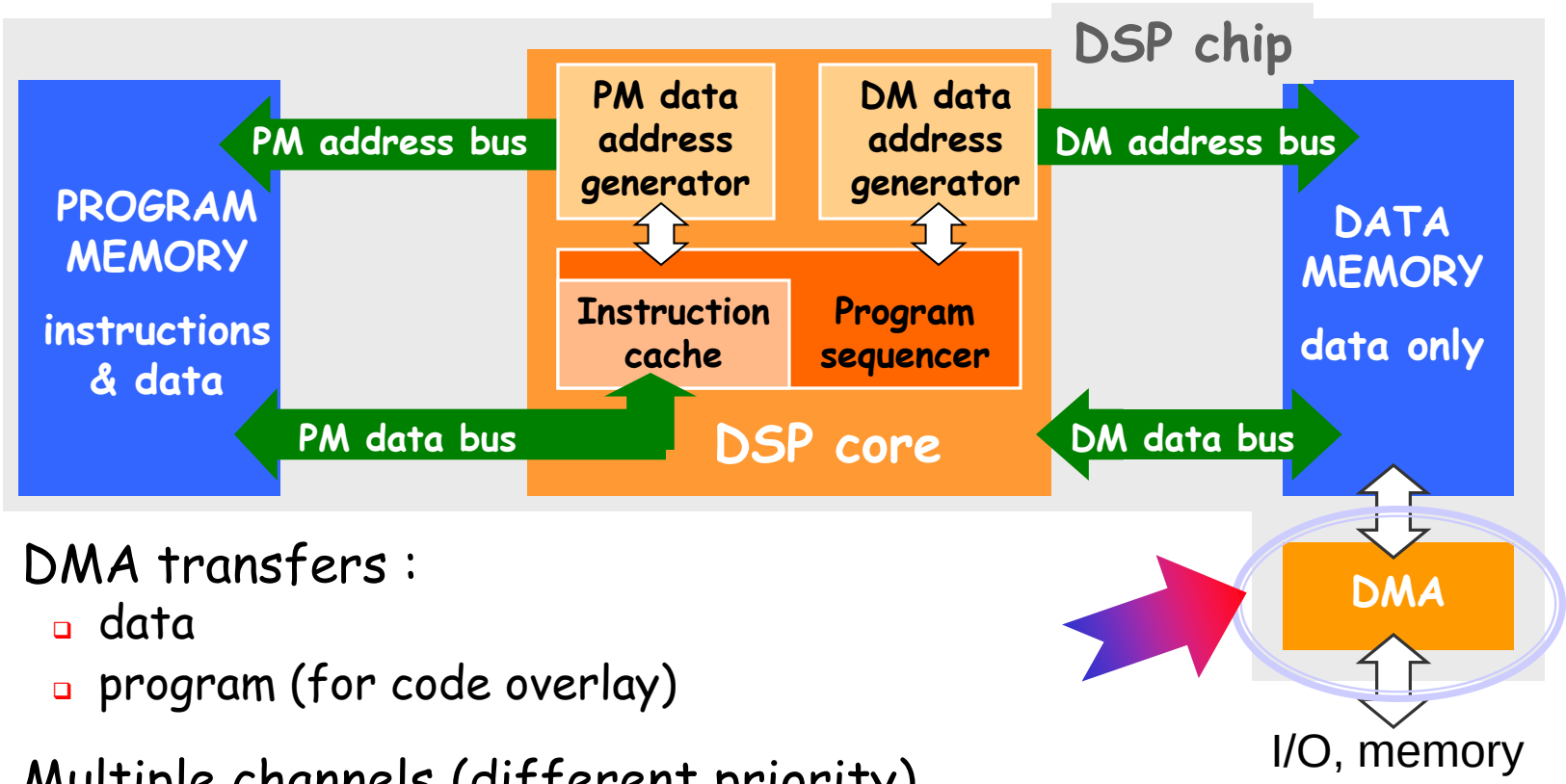
- Builds upon Harvard architecture & improves throughput.
- More buses than for Von Neumann: efficient *but* expensive.
- Instruction cache: loops instructions pre-fetched & buffered.

  → memory BW used for data fetch.
- Data cache on newer DSPs.

Fast data access

c) Direct Memory Access (DMA)

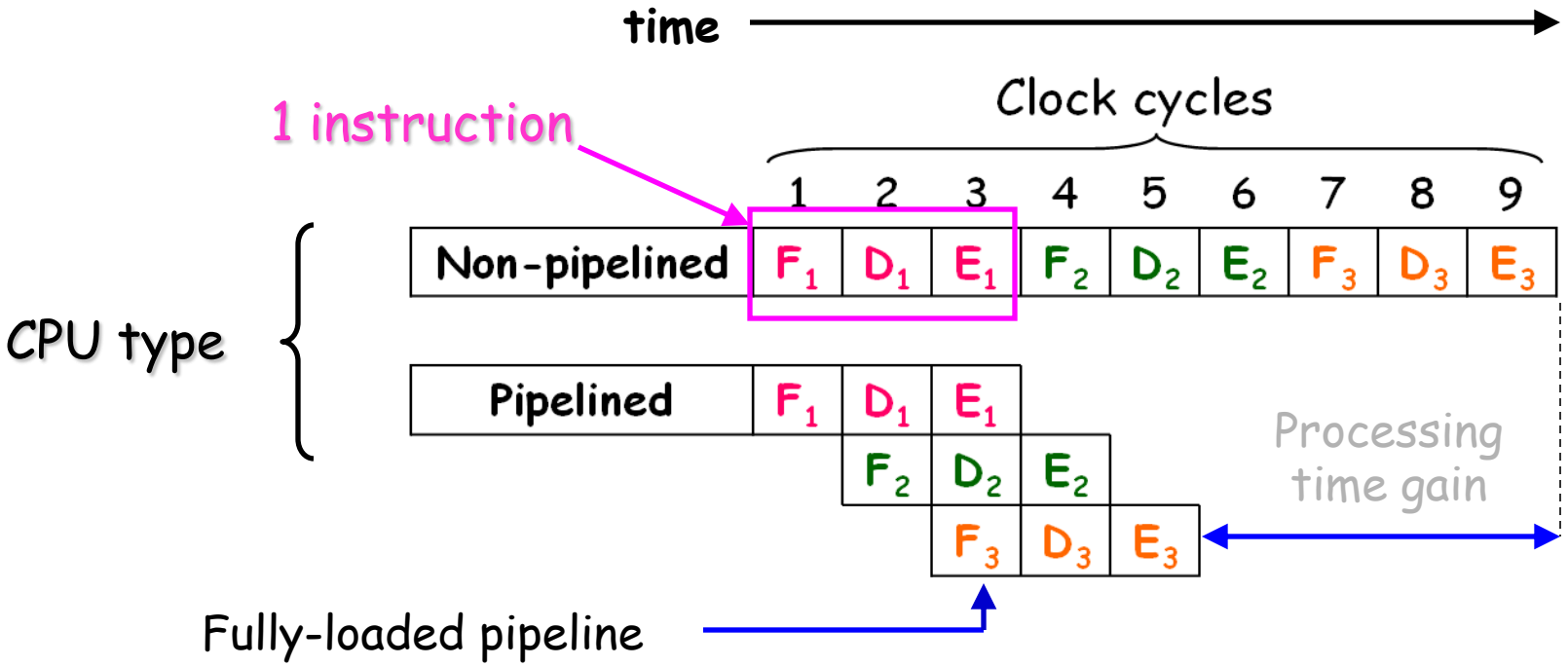
DMA coprocessor: memory transfers without DSP core intervention



- DMA transfers :
  - data
  - program (for code overlay)
- Multiple channels (different priority).
- Arbitration DSP core-DMA for colliding memory access.

Fast computation

b) Pipelining – cont'd



Fast computation

c) Parallel architectures

Increased parallelism improves performance.

Very Long Instruction Words (VLIW):

- **Instruction-level parallelism (ILP)**: multiple execution units, each executes its own instruction.
- Innovative architecture – first used in 'C62xx (1997) VelociTI.
- "Multi issue" DSPs: many instructions issued @same time.

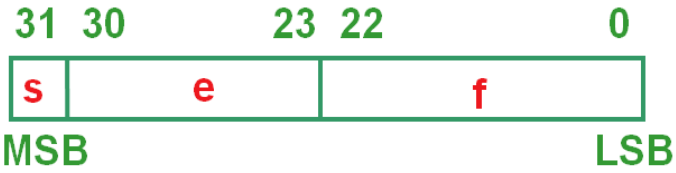
Single Input Multiple Data (SIMD):

- **Data-level parallelism (DLP)**: one instruction performs same operation on multiple data sets.
- Technique used within other architectures (ex: VLIW).
- "Single-issue": one instruction issued @same time.

Numerical fidelity

- Wide accumulators/registers for precision & overflow avoidance: *guard bits*.
- Overflow/underflow flags.
- Saturated arithmetic when overflowing.
- **Floating point arithmetic**: high dynamic range/precision.

IEEE 754 standard
(normalised single precision)



e = exponent, offset binary, -126 < e < 127

s = sign, 0 = pos, 1 = neg

f = fractional part, sign-magnitude + hidden bit

Coded number $x = (-1)^s \cdot 2^e \cdot 1.f$

$$\text{Dynamic range}_{dB} = 20 \log_{10}\left[\frac{\text{largest value}}{\text{smallest value}}\right]$$

Fixed point ~ 180 dB

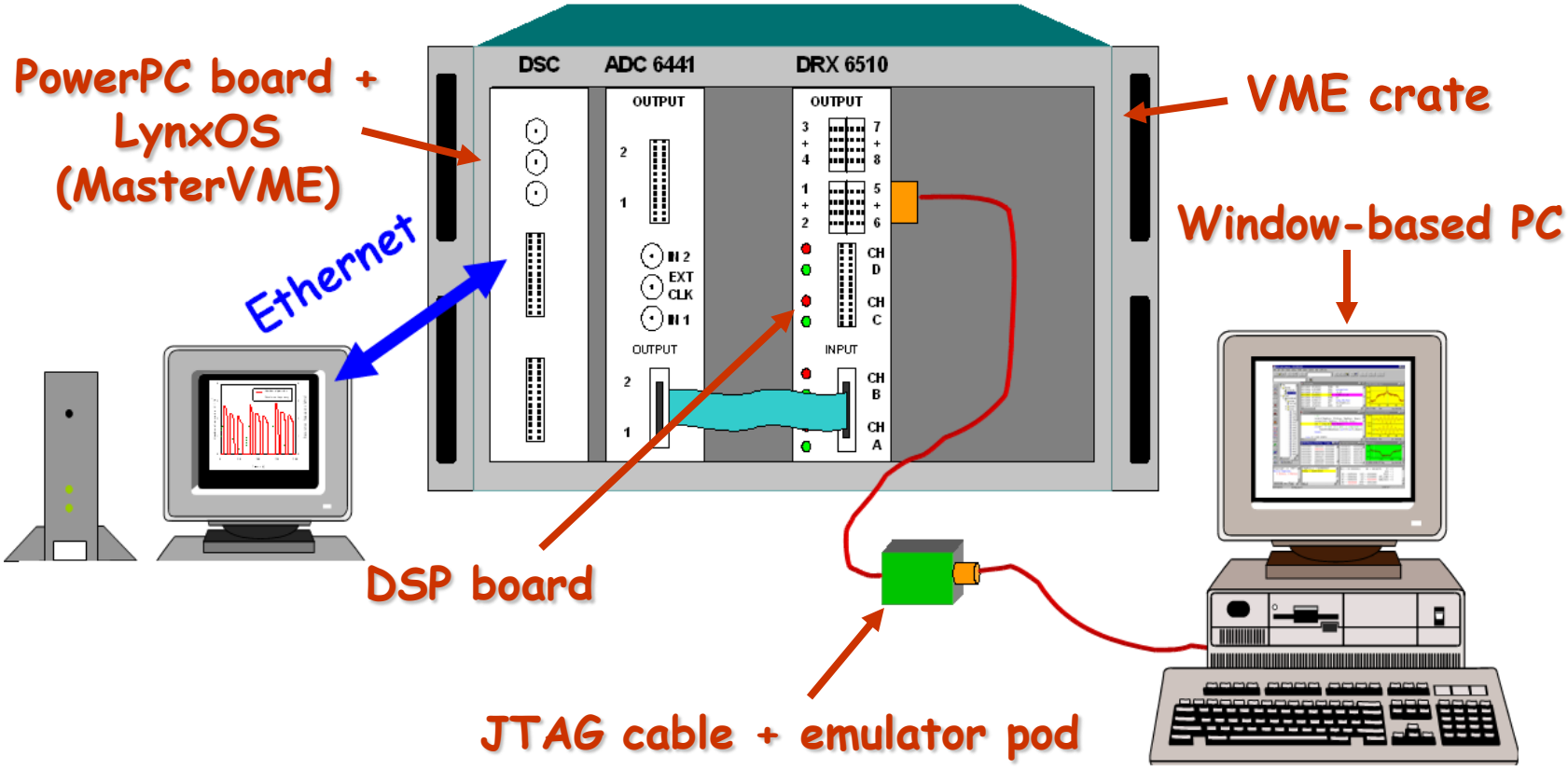
Floating point ~1500 dB

32 bits

Readout chain with DSPs

System use from Control Room

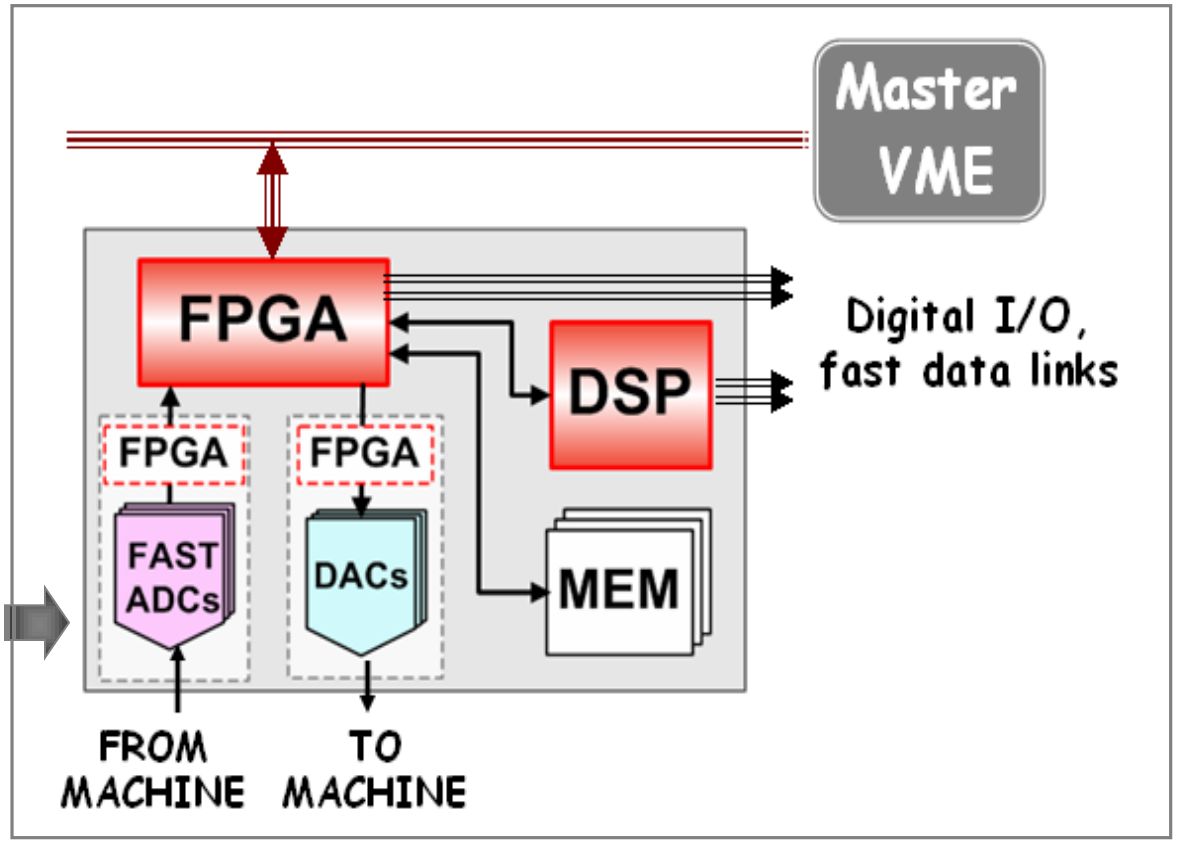
DSP code development/debugging



Code development setup. Example: AD beam intensity measurement (TI 'C40 DSP), CERN '98.

Readout chain with DSPs and FPGAs

Digital system: typical building blocks



Fixed point (integers)

Unsigned integer		Sign & magnitude		2's complement	
Decimal	Bit pattern	Decimal	Bit pattern	Decimal	Bit pattern
15	1111	7	0111	7	0111
14	1110	6	0110	6	0110
13	1101	5	0101	5	0101
12	1100	4	0100	4	0100
11	1011	3	0011	3	0011
10	1010	2	0010	2	0010
9	1001	1	0001	1	0001
8	1000	0	0000	0	0000
7	0111	0	1000	-1	1111
6	0110	-1	1001	-2	1110
5	0101	-2	1010	-3	1101
4	0100	-3	1011	-4	1100
3	0011	-4	1100	-5	1011
2	0010	-5	1101	-6	1010
1	0001	-6	1110	-7	1001
0	0000	-7	1111	-8	1000

Fixed point (fractional numbers)

digit worth								decimal value
$-(2^2)$	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	
-4	**2**	**1**	**0.5**	**0.25**	**0.125**	**0.0625**	**0.03125**	
0	0	0	0	0	0	0	1	0.03125
0	0	0	0	0	0	1	0	0.0625
1	0	1	0	0	0	0	0	-3.0
1	1	0	0	0	1	1	1	-1.78125
1	1	1	1	1	1	1	1	-0.03125

Example: 3 integer bits and 5 fractional bits

Floating point designs

- To work in floating point you (potentially) need blocks to:
  - Convert from fixed point to floating point and back.
  - Convert between different floating point types.
  - Multiply.
  - Add/subtract (involves an intermediate representation with same exponent for both operands).
  - Divide.
  - Square root.
  - Compare 2 numbers.
- The main FPGA companies provide these in the form of IP cores. You can also roll your own.

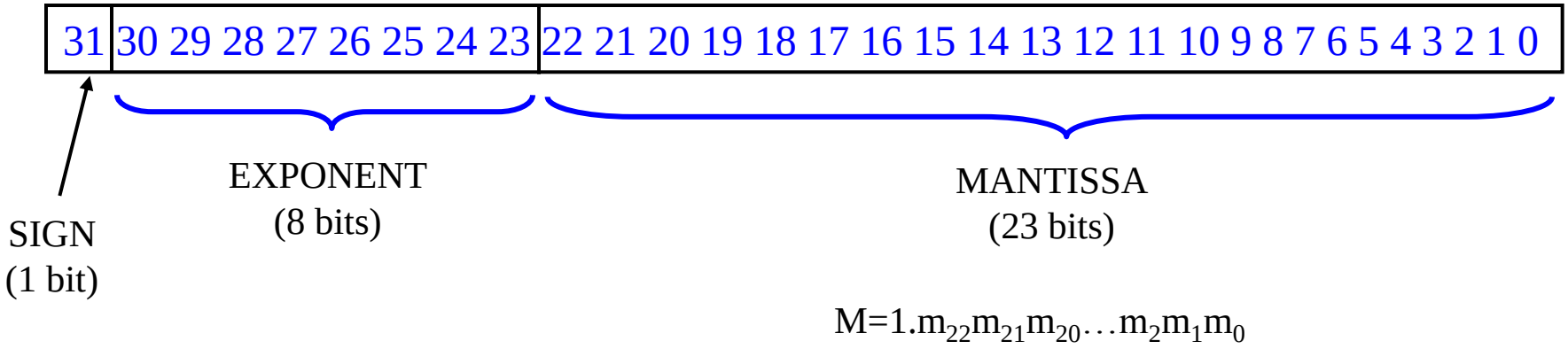
Some performance figures (single precision)

Table 26: **Characterization of Single-Precision Format on Virtex-5 FPGA**

Operation	Resources				Maximum Frequency (MHz)[1]
	Embedded		Fabric		Virtex-5
	Type	Number	LUTs	FFs	-1
Multiply	DSP48E (max usage)	3	88	177	450
	DSP48E (full usage)	2	126	209	429
	DSP48E (medium usage)	1	294	390	375
	Logic	0	641	698	357
Add/Subtract	DSP48E (speed optimized, full usage)	2	267	375	410
	Logic (speed optimized, no usage)	0	429	561	395
	Logic (low latency)	0	536	625	372
Fixed to float	Int32 input		181	226	398
Float to fixed	Int32 result		218	237	373
Float to float	Single to double		44	101	466
Compare	Programmable		80	24	393
Divide	C_RATE=1		788	1,370	365
	C_RATE=26		227	233	316
Sqrt	C_RATE=1		542	787	398
	C_RATE=25		175	204	388

1. Maximum frequency obtained with map switches -ol high and -cm speed, and par switches -pl high and -rl high.

Floating point binary numbers

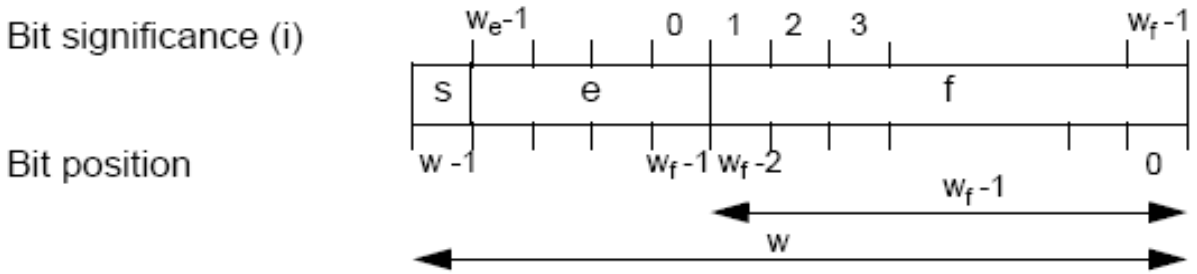


$$\text{Value} = (-1)^S \times M \times 2^{E-127}$$

Max value: $\pm (2-2^{23}) \times 2^{128} = \pm 6.8 \times 10^{38}$

Min value: $\pm 1.0 \times 2^{-127} = \pm 5.9 \times 10^{-39}$

Floating point format



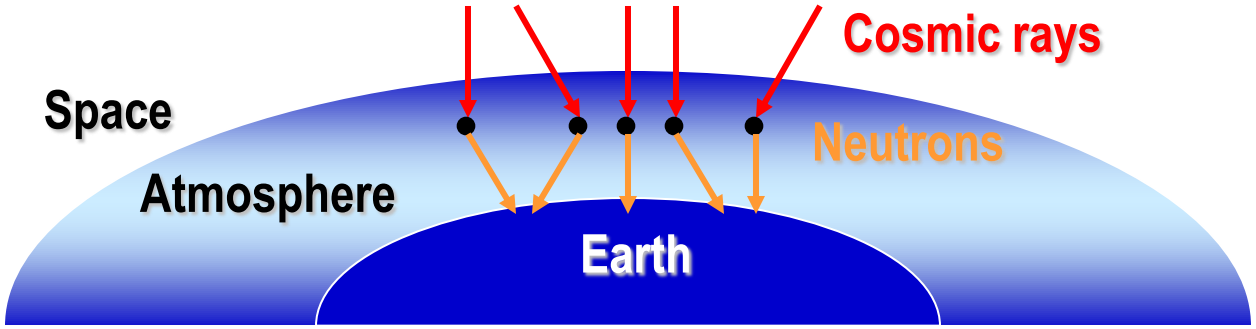
s: sign.
e: exponent.
f: fractional part ($b_0.b_1b_2b_3b_4...b_{w_f-1}$)
Convention: normalized numbers have $b_0=1$

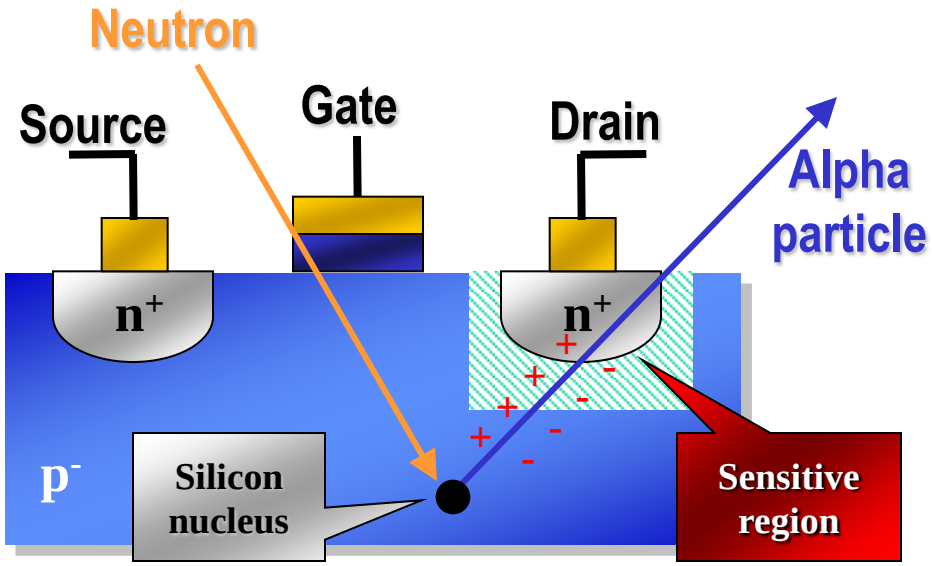
Exponent value: $E = e-(2^{w_e-1}-1)$ $\qquad e = \sum_{i=0}^{w_e-1} e_i 2^i$

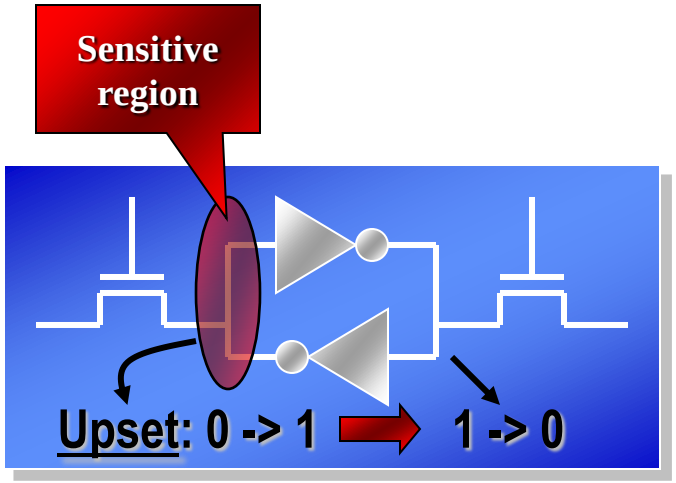
Total value: $v = (-1)^s 2^E b_0.b_1b_2\ldots b_{w_f-1}$

IEEE-754 standard single format: 24-bit fraction and 8-bit exponent (w=32 and w_f=24 in the figure).
IEEE-754 standard double format: 53-bit fraction and 11-bit exponent.

Single Event Effects (SEE) created by neutrons



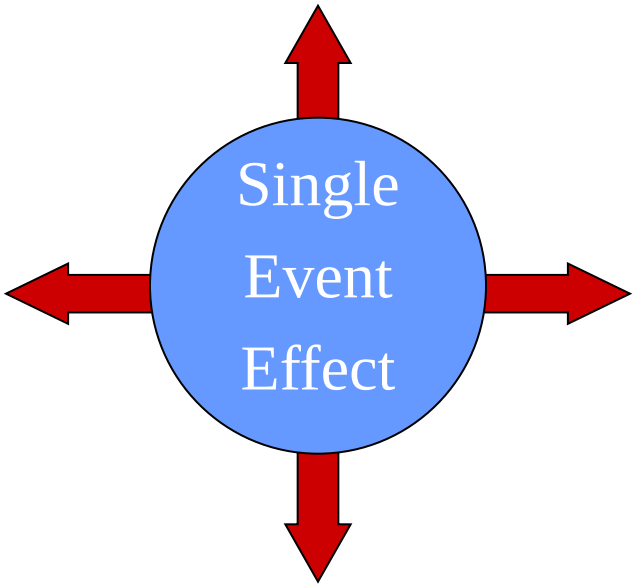




Memory Cell: CMOS Configuration Latch (CCL)

Classification of SEEs

Single Event Transient (**SET**)



Single Event Latch-Up (**SEL**)

Parasitic transistors activated in a device, causing internal short

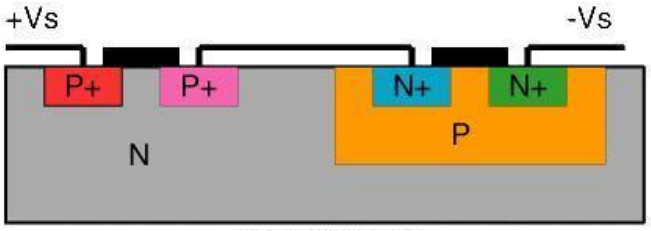
Single Event Upset (**SEU**)

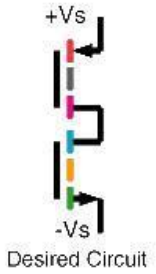
Bit-Flip Somewhere

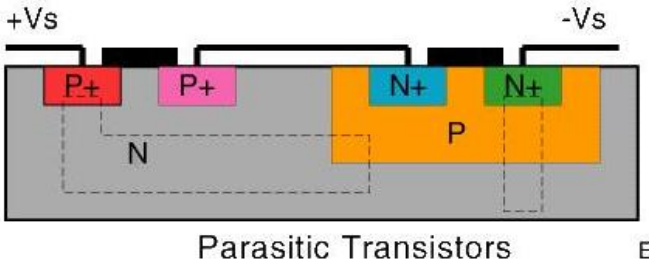
Single Event Functional Interrupt (**SEFI**)

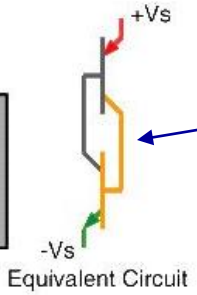
Bit-Flip specifically in a control register – POWER ON RESET/JTAG etc.

SEL description









Activation of either of these transistors causes a short from V+ to V-

- Has virtually disappeared in new technologies (low Vccint not enough to forward bias transistors).
- Only cure used to be epitaxial substrate (very expensive).

SEU Failures in Time (FIT)

- Defined as the number of failures expected in 10^9 hours.
- In practice, configuration RAM dominates. Example:

Virtex XCV1000 memory Utilization

Memory Type	# of bits	%
Configuration	5,810,048	97.4
Block RAM	131,072	2.2
CLB flip-flops	26,112	0.4

- Average of only 10% of FPGA configuration bits are used in typical designs
  - Even in a 99% full design, only up to 30% are used
  - Most bits control interconnect muxes
  - Most mux control values are “don’t-care”
- Must include this ratio for accurate SEU FIT rate calculations.