Computing on FPGA

S. F. Schifano

University of Ferrara and INFN-Ferrara

Advanced Workshop on Modern FPGA Based Technology for Scientific Computing

May 14, 2019

ICTP, Trieste, Italy

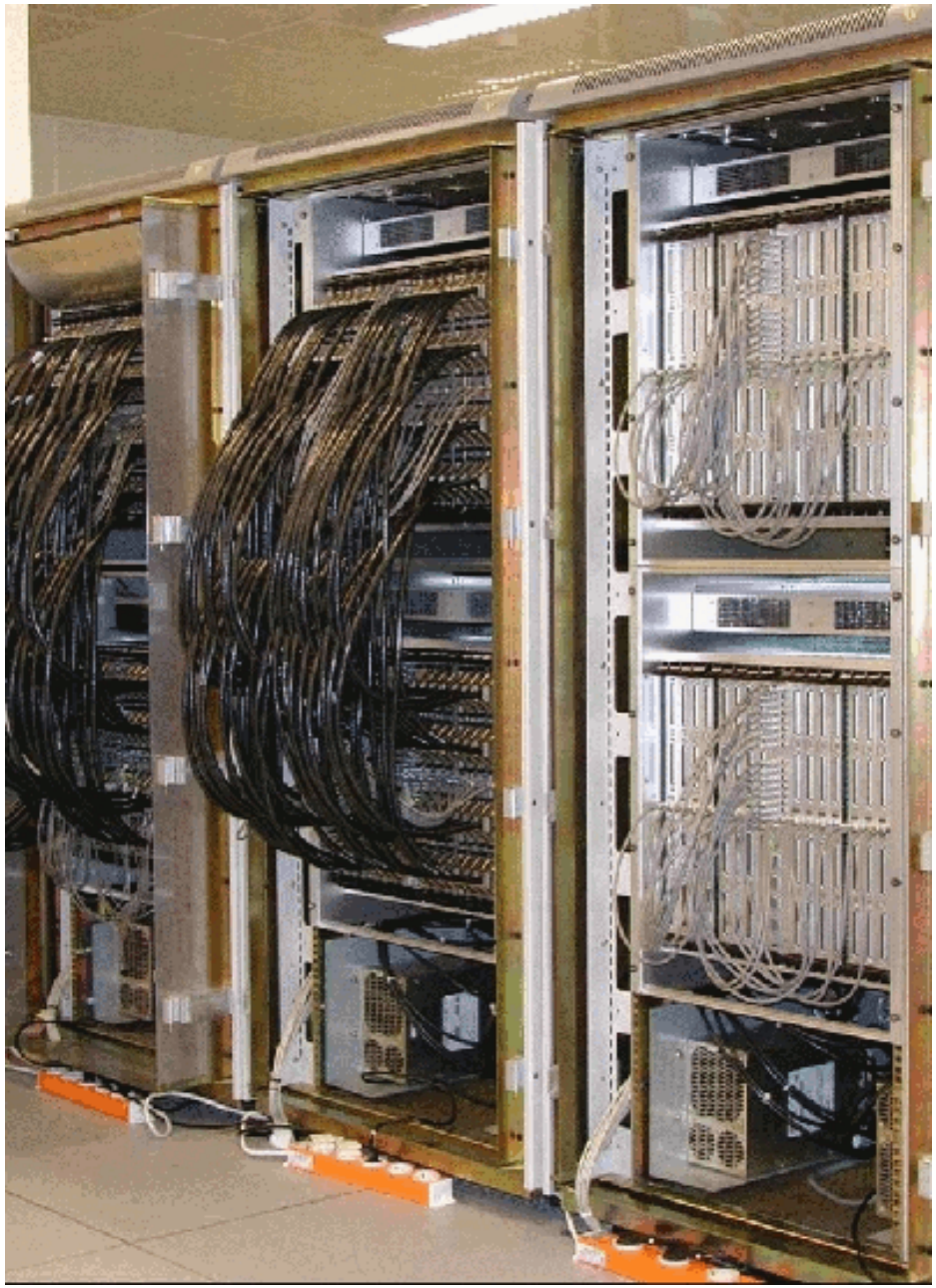
Outline

Background: Let me introduce myself

Development of computing systems optimized for computational physics:

- APEmille and apeNEXT: LQCD-machines, FPGA used to interface APE with standard commodity CPUs
- AMchip: pattern matching processor, installed at CDF, FPGAs to control configuration of the system
- Janus I+II: FPGA-based system for spin-glass simulations
- QPACE: Cell-based machine, mainly for LQCD apps, Network processor on FPGA
- AuroraScience: multi-core based machine, Network processor on FPGA
- EuroEXA: hybrid ARM+FPGA exascale system, accelerator on FPGA

APEmille e apeNEXT (2000 and 2004)



$a \times b + c \quad a, b, c \in \mathbb{C}$

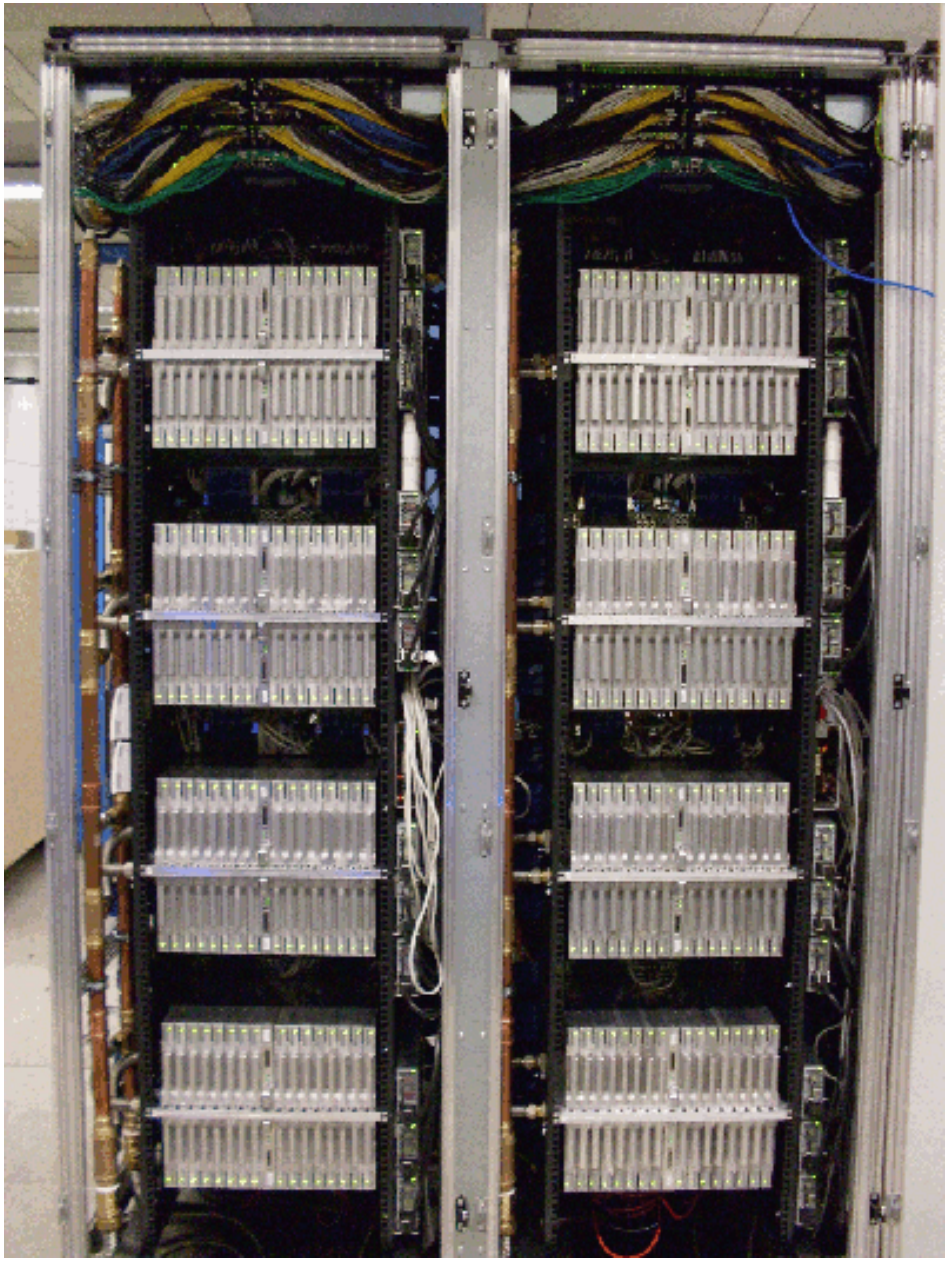
Janus I (2007)

- 256 FPGAs
- 16 boards
- 8 host PC

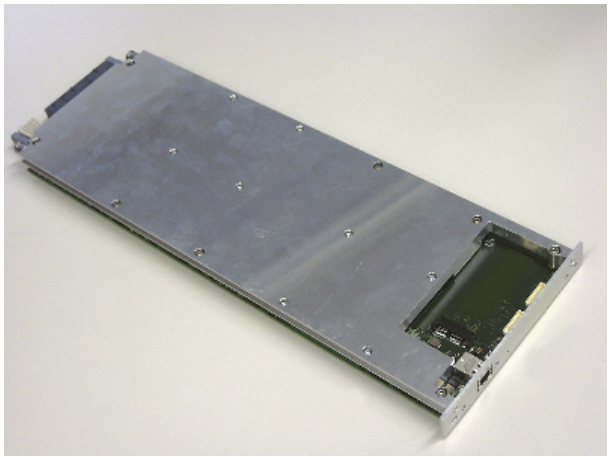
- Monte Carlo simulations of Spin Glass systems

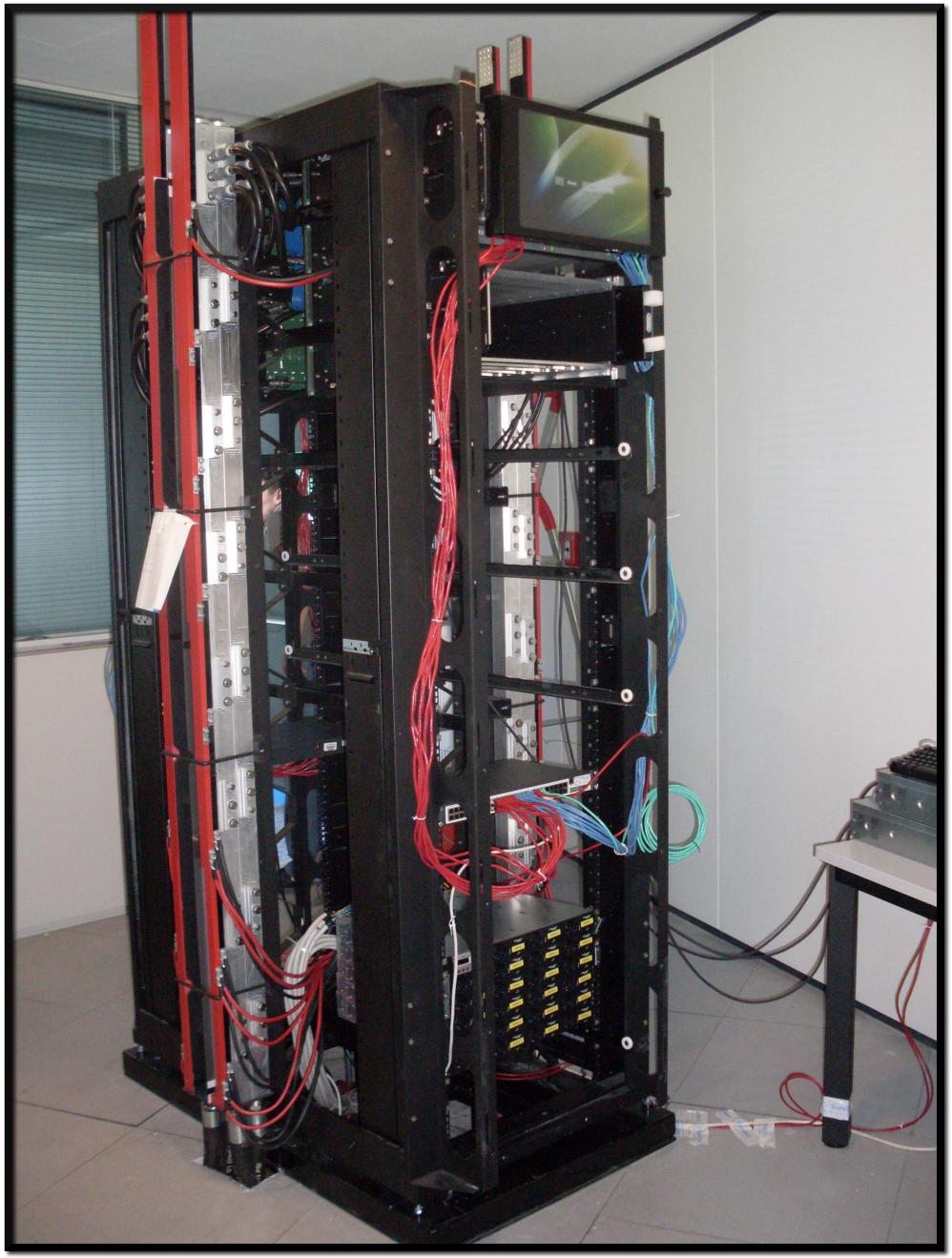
QPACE Machine (2008)

- Processor IBM PowerXCell8i, enhanced version of PS3
- 8 backplanes per rack
- 256 nodes (2048 cores)
- 16 root-cards
- 8 cold-plates
- 26 Tflops peak double-precision
- 35 KWatt maximum power consumption
- **773 MFLOPS / Watt**
- TOP-GREEN 500 in Nov.'09 and July'10

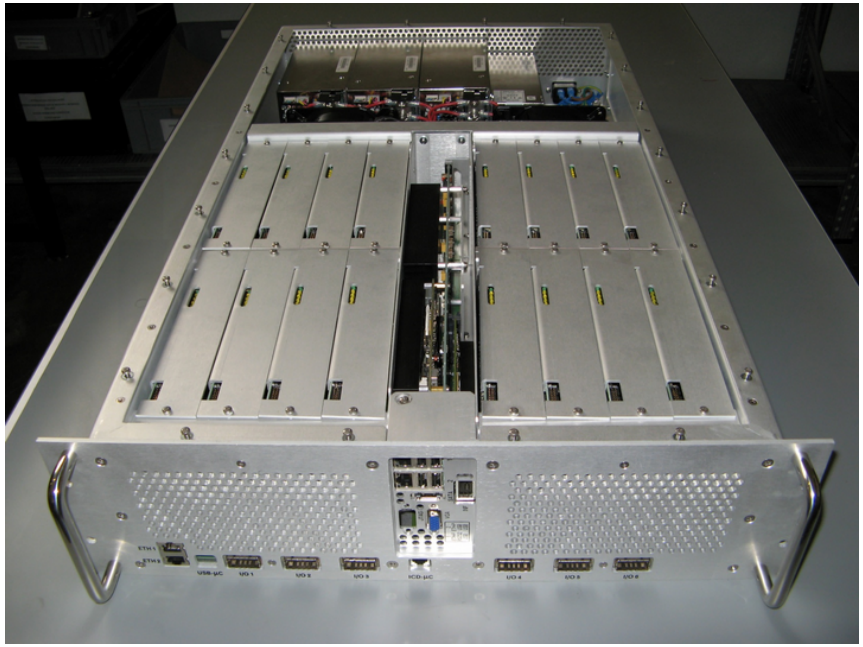


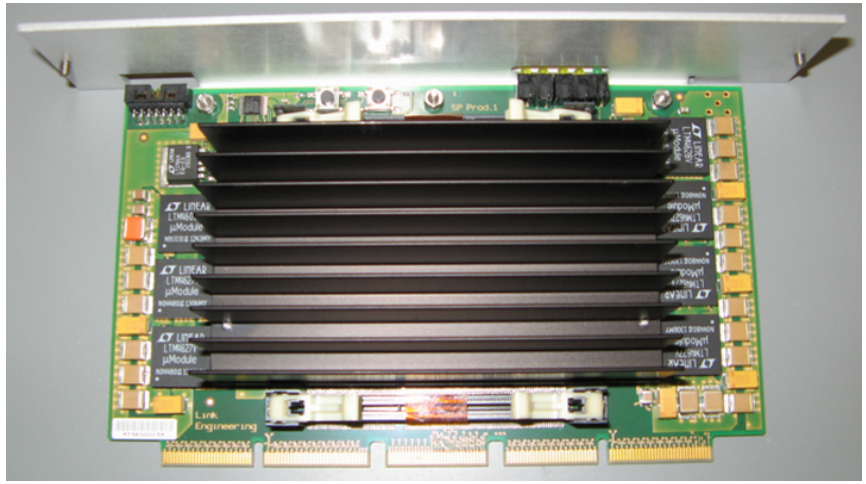
Aurora Machine (2008)





Janus II (2012)

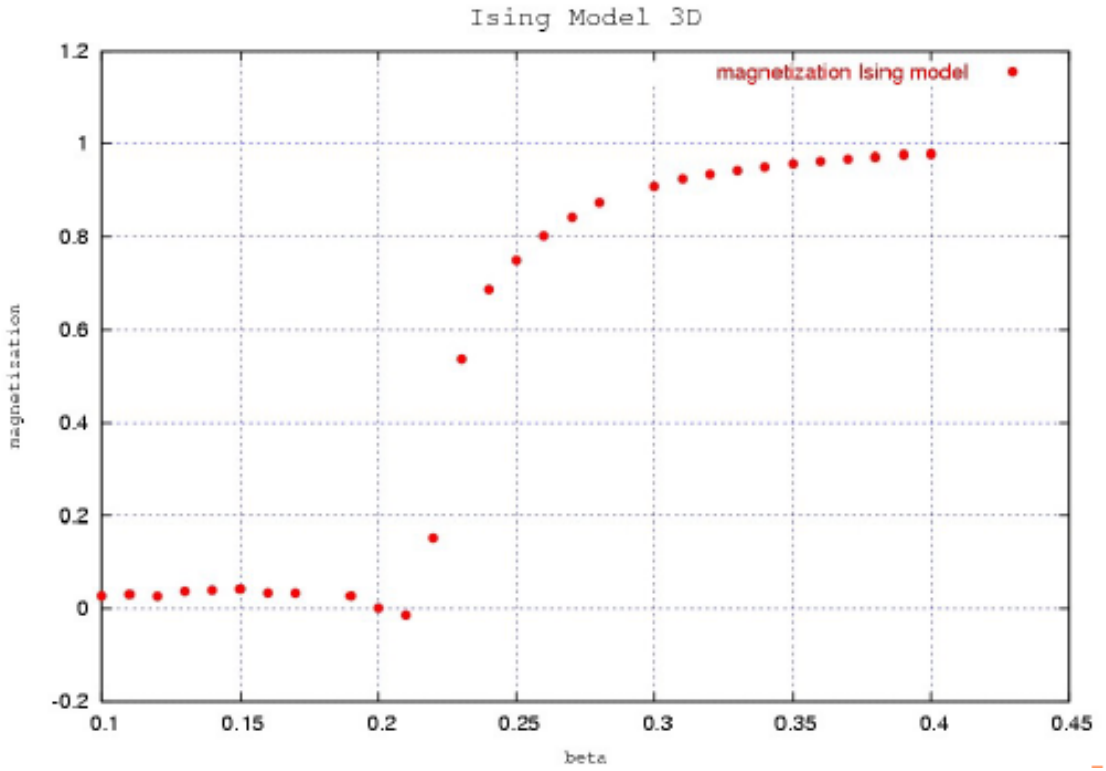




Spin-Glass

The Spin-glass is a statistic model to study some behaviours of complex macroscopic systems like **disordered magnetic materials**.

An apparently trivial generalization of ferromagnet model.



Spin-Glass Models

Ising Model

$$E(\{S\}) = -J \sum_{\langle ij \rangle} s_i \cdot s_j, \qquad J > 0, \;\; s_i, s_j \in \{-1, +1\}$$

Edwards Anderson Model (Binary)

$$E(\{S\}) = \sum_{\langle ij \rangle} J_{ij} \cdot s_i \cdot s_j, \qquad J_{ij}, s_i, s_j \in \{-1, +1\}$$

Edwards Anderson Model (Gaussian)

$$E(\{S\}) = \sum_{\langle ij \rangle} J_{ij} \cdot s_i \cdot s_j, \qquad J_{ij} \in \mathbb{R}, s_i, s_j \in \{-1, +1\}$$

Heisenberg Model

$$E(\{S\}) = \sum_{\langle ij \rangle} J_{ij} \cdot \vec{s_i} \cdot \vec{s_j} \qquad J_{ij} \in \mathbb{R}, s_i, s_j \in \mathbb{R}^3$$

The Edwards-Anderson (EA) Model

The system variables are spins (± 1), arranged in D-dimensional (usually D=3) lattice of size L .

- Spins s_i interacts only with its nearest neighbours
- Pair of spins (s_i, s_j) share a coupling term J_{ij}
- The energy of a configuration $\{S\}$ is computed as:

$$E(\{S\}) = \sum_{\langle ij \rangle} J_{ij} s_i s_j$$

- Each configuration $\{S\}$ has a probability given by the **Boltzmann** factor:

$$P(\{S\}) \propto e^{\frac{-E(\{S\})}{kT}}$$

- Average of macroscopic observable (**magnetization**) are defined as:

$$\langle M \rangle = \sum_{\{S\}} M(\{S\}) P(\{S\}) \quad \text{where} \quad M(\{S\}) = \sum_i s_i$$

Spin Glass Monte Carlo Algorithms

- A lattice size L has 2^{L^3} different configurations (e.g. $L = 80 \Rightarrow 2^{80^3}$)
- pratically impossible to manage to generate all configurations
- not all configurations have the same probability and are equally important.

Monte Carlo algorithms, like the Metropolis and Heatbath, are adopted:

- configurations are generated according to their probability
- observables average are computed as **unweighted** sums of Monte Carlo generated configurations:

$$\langle M \rangle \sim \sum_i M(\{S_i^{MC}\})$$

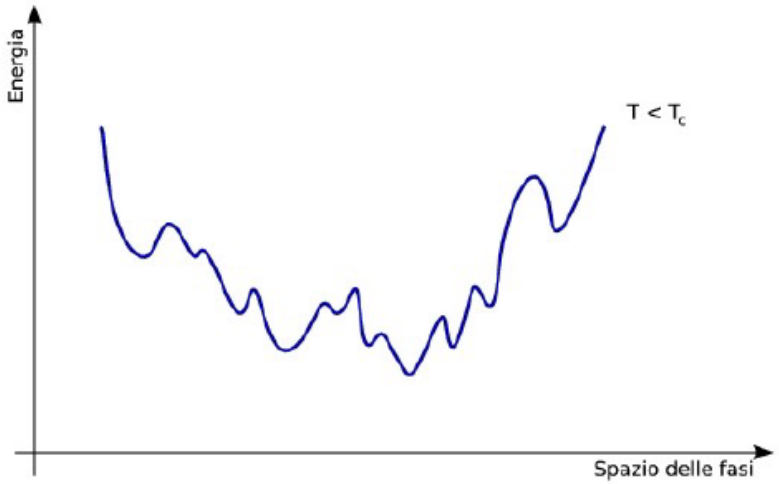
Metropolis Algorithm for EA

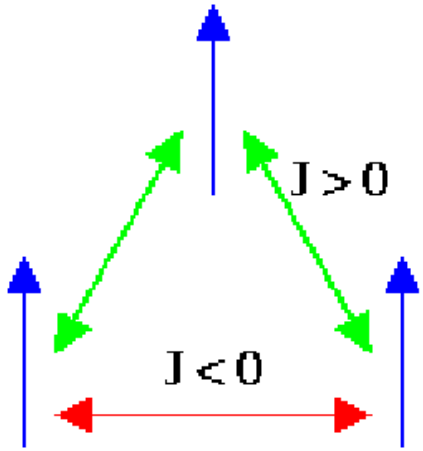
Require: set of $\{S\}$ and $\{J\}$

1: **loop** // loop on Monte Carlo steps
2: **for all** $s_i \in \{S\}$ **do**
3: $s'_i = (s_i == 1)\ ?\ -1\ :\ 1$ // flip tentatively value of s_i
4: $\Delta E = \sum_{\langle ij \rangle}(J_{ij} \cdot s'_i \cdot s_j) - (J_{ij} \cdot s_i \cdot s_j)$ // compute energy change
5: **if** $\Delta E \leq 0$ **then**
6: $s_i = s'_i$ // accept new value of s_i
7: **else**
8: $\rho =$ rnd() // compute a random number $0 \leq \rho \leq 1,\ \rho \in \mathbb{Q}$
9: **if** $\rho < e^{-\beta \Delta E}$ **then** // $\beta = 1/T,\ T =$ Temperature
10: $s_i = s_i$‘ // accept new value of s_i
11: **end if**
12: **end if**
13: **end for**
14: **end loop**

Spin Glass Simulation is Computer Challenging

$$E(\{S\}) = -\sum_{\langle ij \rangle} J_{ij} s_i s_j, \quad s_i, s_j \in \{+1, -1\}, \quad J_{ij} \in \{+1, -1\}$$





Frustation effects make:

- the energy function landscape corrugated
- the approach to the thermal equilibrium a slowly converging process.

Spin-glass is Computer Challenging

To bring a lattice $L = 48 \ldots 128$ to the thermal equilibrium, typical state-of-the-art simulation-campaign steps are:

- simulation of *Hundreds* (*Thousands*) systems, **samples**, with different initial values of spins and couplings,
- for each sample the simulation is repeated *2-4* times with different initial spin-values (coupling values kept fixed), **replicas**.
- Each simulation may requires $10^{12} \ldots 10^{13}$ Monte Carlo update steps.

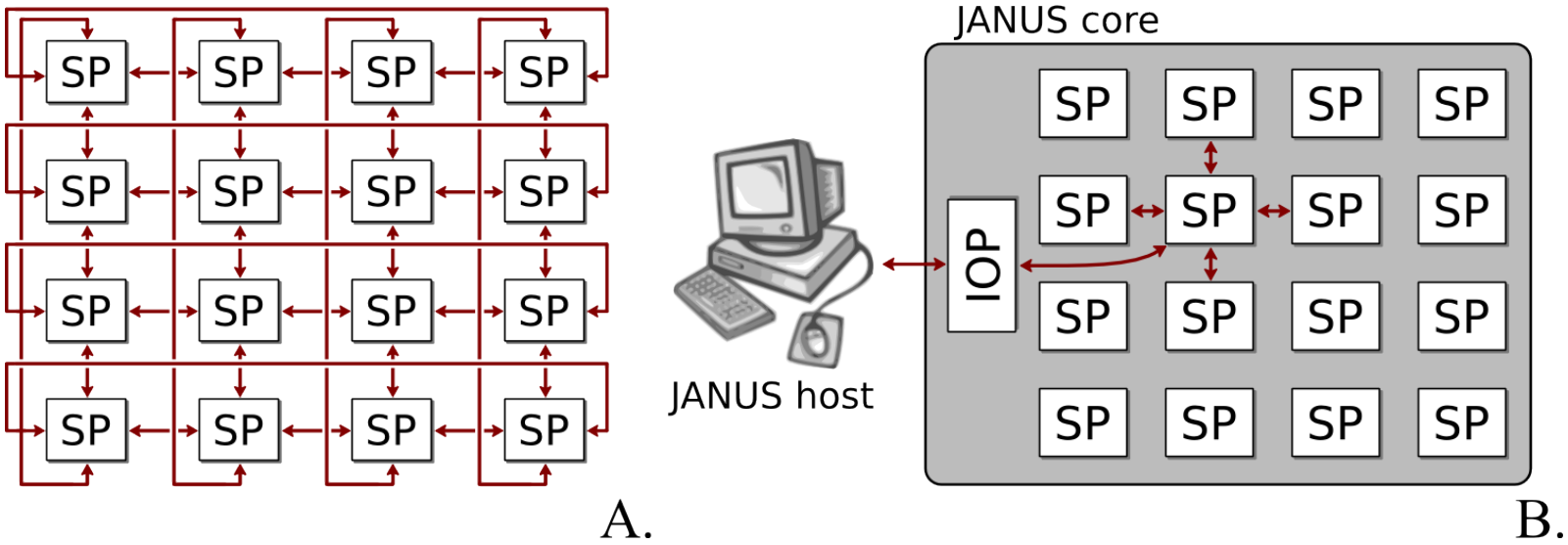
$$80^3 \times 10\ \text{ns} \times 10^{11}\ \text{MC-steps} \approx 16\ \text{years}$$

Exploiting of parallelism is necessary.

The Janus System

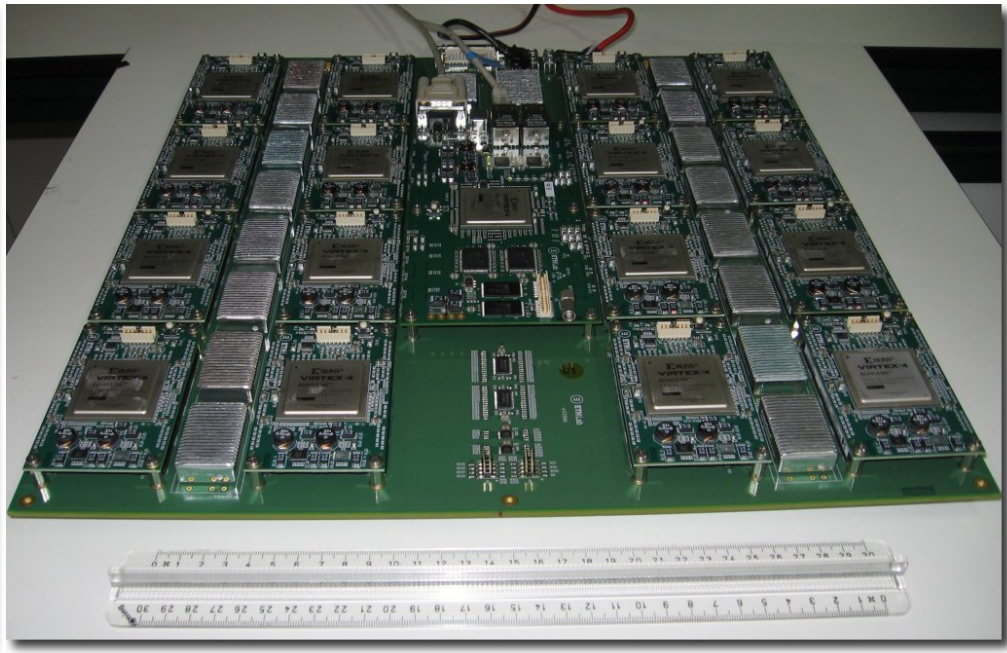
Architecture:

- a cluster of 16 boards
- each board is a 2D toroidal grid of 4×4 FPGA-based *Simulation Processors* (SP)
- data links among nearest neighbours on the grid
- one *Control Processor* (CP) on each board

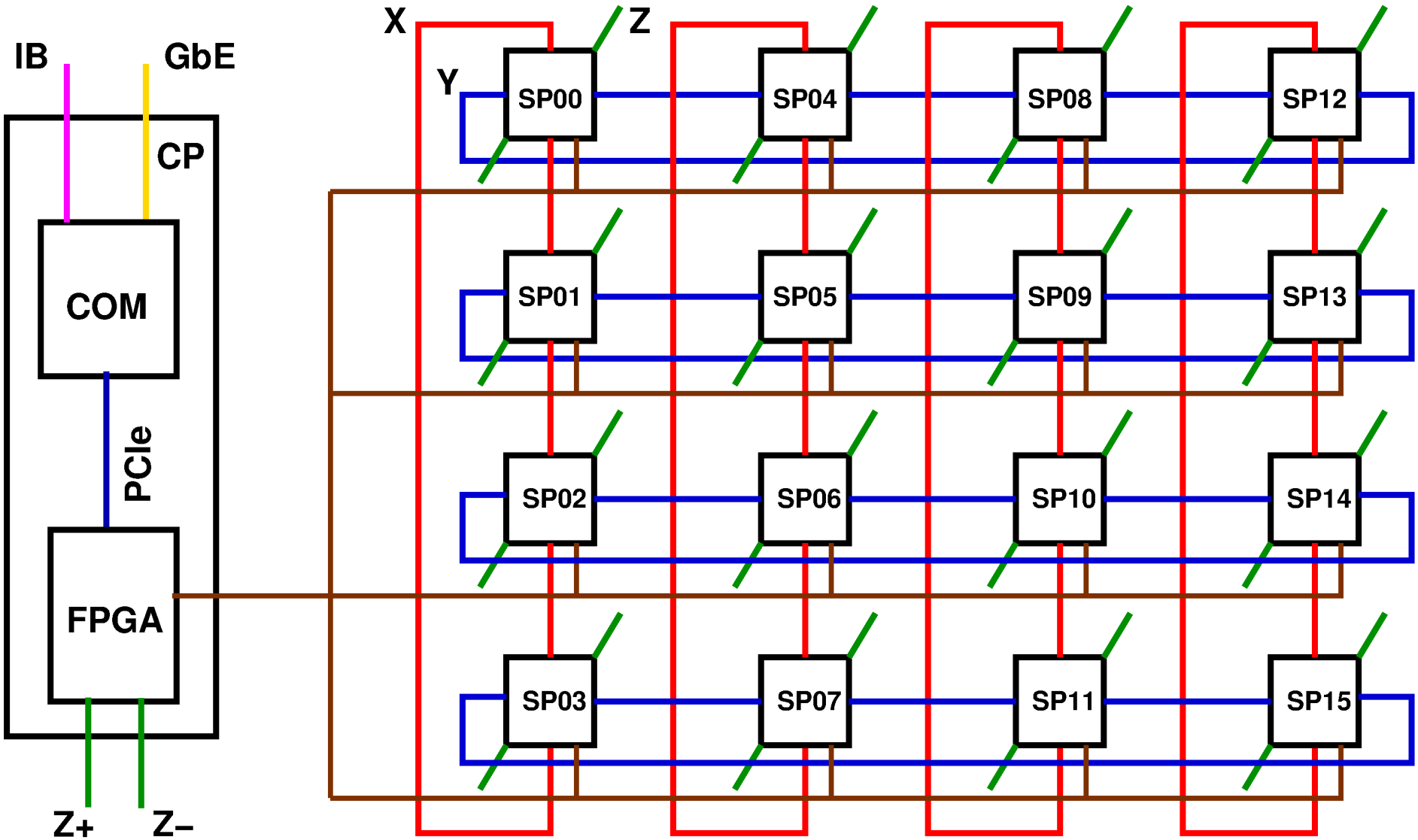


JANUS is a project carried out by BIFI, University of Madrid, Estremadura, Rome and Ferrara, and by Eurotech.

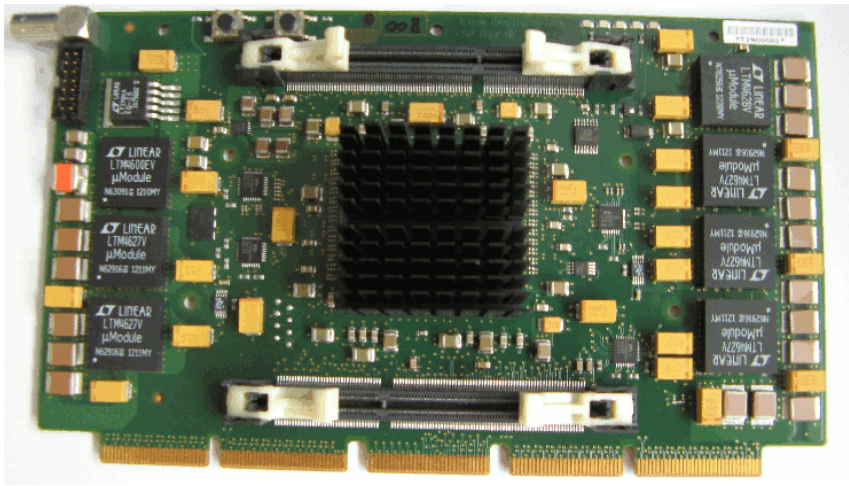
The Janus I System

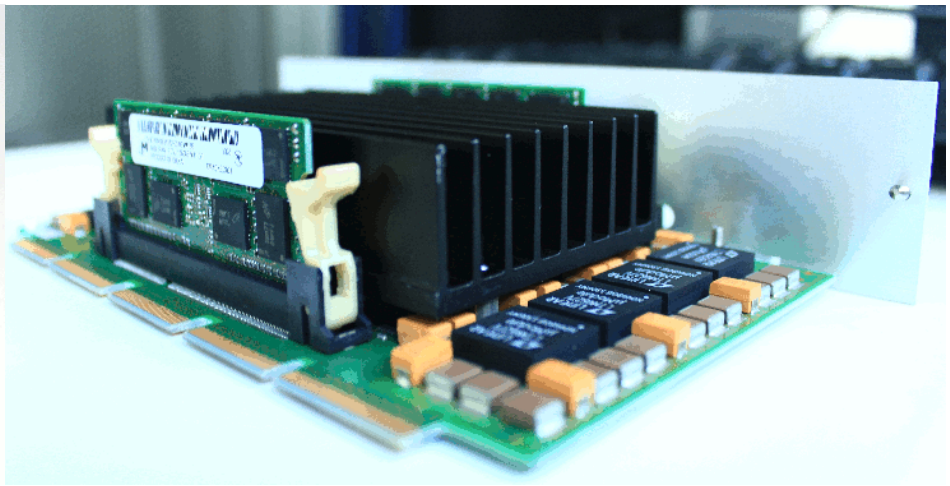


The Janus II System: Architecture



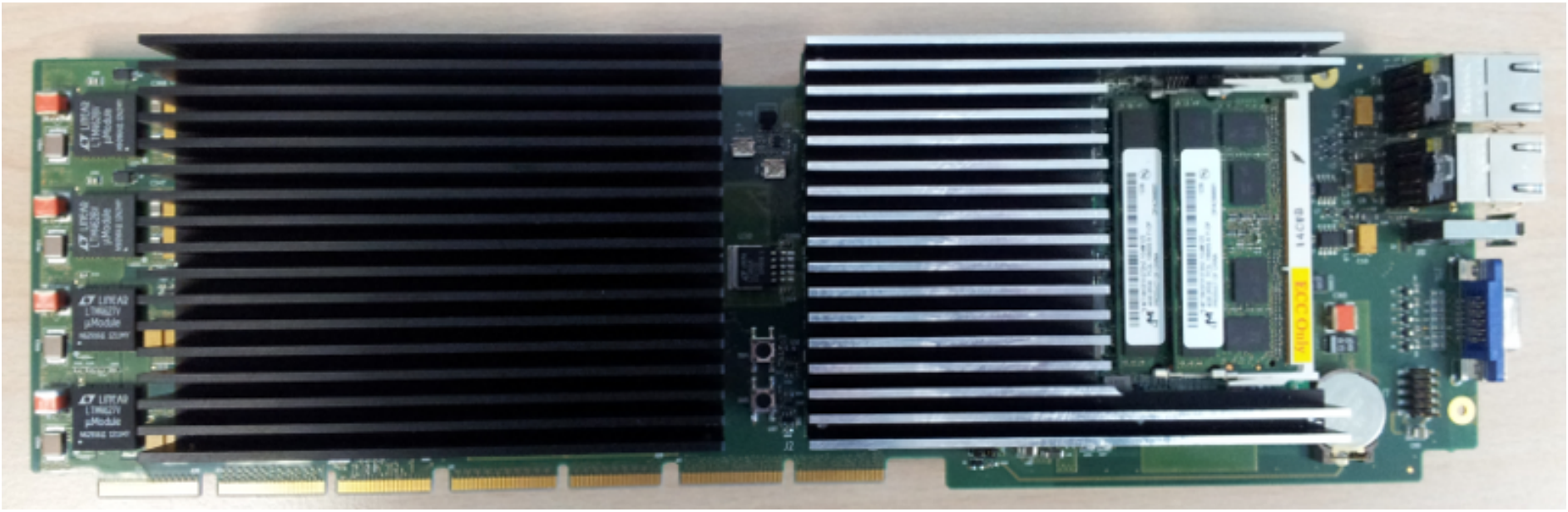
The Janus II System: SP





- Xilinx Virtex-7 XC7VX485T FPGA
  - 485000 logic cells
  - $\sim$ 32 Mbit embedded memory
- two banks of DDR-3 memory of 8 Gbyte

The Janus II System: CP



- *Computer-on-Module* (COM) system
- Intel Core i7 processor running at 2.2 GHz running standard Linux OS
- one input-output FPGA connected on the PCIe bus:
  - configure the FPGAs of SPs
  - manage all input-ouput operations
  - monitor codes execution

Single-Spin Update Algorithm

1. flip the value of the spin $S'_i = \bar{S}_i = -S_i$
2. compute the variation of energy $\Delta E = E'_i - E_i$

$$\begin{aligned} E_i &= -S_i \sum_{\langle j \rangle} J_{ij} S_j \\ E'_i &= -\bar{S}_i \sum_{\langle j \rangle} J_{ij} S_j = S_i \sum_{\langle j \rangle} J_{ij} S_j \\ \Delta E_i &= E'_i - E_i = -E_i - E_i = -2E_i \end{aligned}$$

3. if $\Delta E_i < 0$ accept the new value of spin $S'_i = \bar{S}_i$
4. if $\Delta E_i \geq 0$:
   1. compute a random number ρ ($\rho \in [0 \ldots 1]$)
   2. if $\rho < e^{-\beta \Delta E_i}$ accept the new of spin S
   3. se $\rho \geq e^{-\beta \Delta E_i}$ reject the new value of spin S

where $\beta = 1/T$ and T is the value of the temperature.

The energy E_i associated to the site i takes then all even integer values in the range $[-6, 6]$, and correspondingly:

$$\Delta E_i \in \{-12, -8, -4, +0, +4, +8, +12\}.$$

Random Wheel Generator Engine

The Parisi-Rapuano generator is a popular choise for Spin Glass simulations:

$$\text{WHEEL[K]} = \text{WHEEL[K-24]} + \text{WHEEL[K-55]}$$

$$\rho = \text{WHEEL[K]} \oplus \text{WHEEL[K-61]}$$

- WHEEL is a circular array of 64 32-bit unsigned-integers random values
- ρ is the generated pseudo-random number

Single-Spin Update Engine

Integers numbers are expensive in terms of resources.

- mapping spins and coupling into bit-valued ({0,1}) variables:

$$S_i \to \sigma_i = (1 + S_i)/2 \qquad J_{ij} \to \gamma_{ij} = (1 + J_{ij})/2$$

- then evaluation of contribution to energy at site i from site j

$$\zeta_{ij} = S_i J_{ij} Sj$$

can be computed as

$$\zeta'_{ij} = 2(\sigma_i \oplus \gamma_{ij} \oplus \sigma_j) - 1$$

S_i	J_{ij}	S_j	ζ_{ij}	σ_i	γ_{ij}	σ_j	ζ'_{ij}
-1	-1	-1	-1	0	0	0	-1
-1	-1	1	1	0	0	1	1
-1	1	-1	1	0	1	0	1
-1	1	1	-1	0	1	1	-1
1	-1	-1	1	1	0	0	1
1	-1	1	-1	1	0	1	-1
1	1	-1	-1	1	1	0	-1
1	1	1	1	1	1	1	1

Single-Spin Update Engine

Having spins as bit-variables, the variation of energy ΔE_i at site i can be computed as:

$$
\begin{aligned}
\Delta E_i &= -2E_i = -2(-\sum_{\langle j \rangle} \zeta'_{ij}) = \\
&= -2(-\sum_{\langle j \rangle}(2(\sigma_i \oplus \gamma_{ij} \oplus \sigma_j) - 1) = \\
&= -2(-2\sum_{\langle j \rangle}(\sigma_i \oplus \gamma_{ij} \oplus \sigma_j) - \sum_{\langle j \rangle}(-1)) = \\
&= -2(-2\sum_{\langle j \rangle}(\sigma_i \oplus \gamma_{ij} \oplus \sigma_j) + 6) = \\
&= 4\sum_{\langle j \rangle}(\sigma_i \oplus \gamma_{ij} \oplus \sigma_j) - 12 \\
&= 4\Sigma_i - 12
\end{aligned}
$$

where $\Sigma_i = \sum_{\langle j \rangle} (\sigma_i \oplus \gamma_{ij} \oplus \sigma_j) \in \{0 \ldots 6\}$

Single-Spin Update Engine

Values of Σ_i have a one-to-one correspondence with value of $\Delta E_i = 2E_i$:

E_i	$\Delta E_i = -2E_i$	Σ_i	$\Delta E_i = (4\Sigma_i - 12)$
-6	+12	6	+12
-4	+8	5	+8
-2	+4	4	+4
+0	+0	3	+0
+2	-4	2	-4
+4	-8	1	-8
+6	-12	0	-12

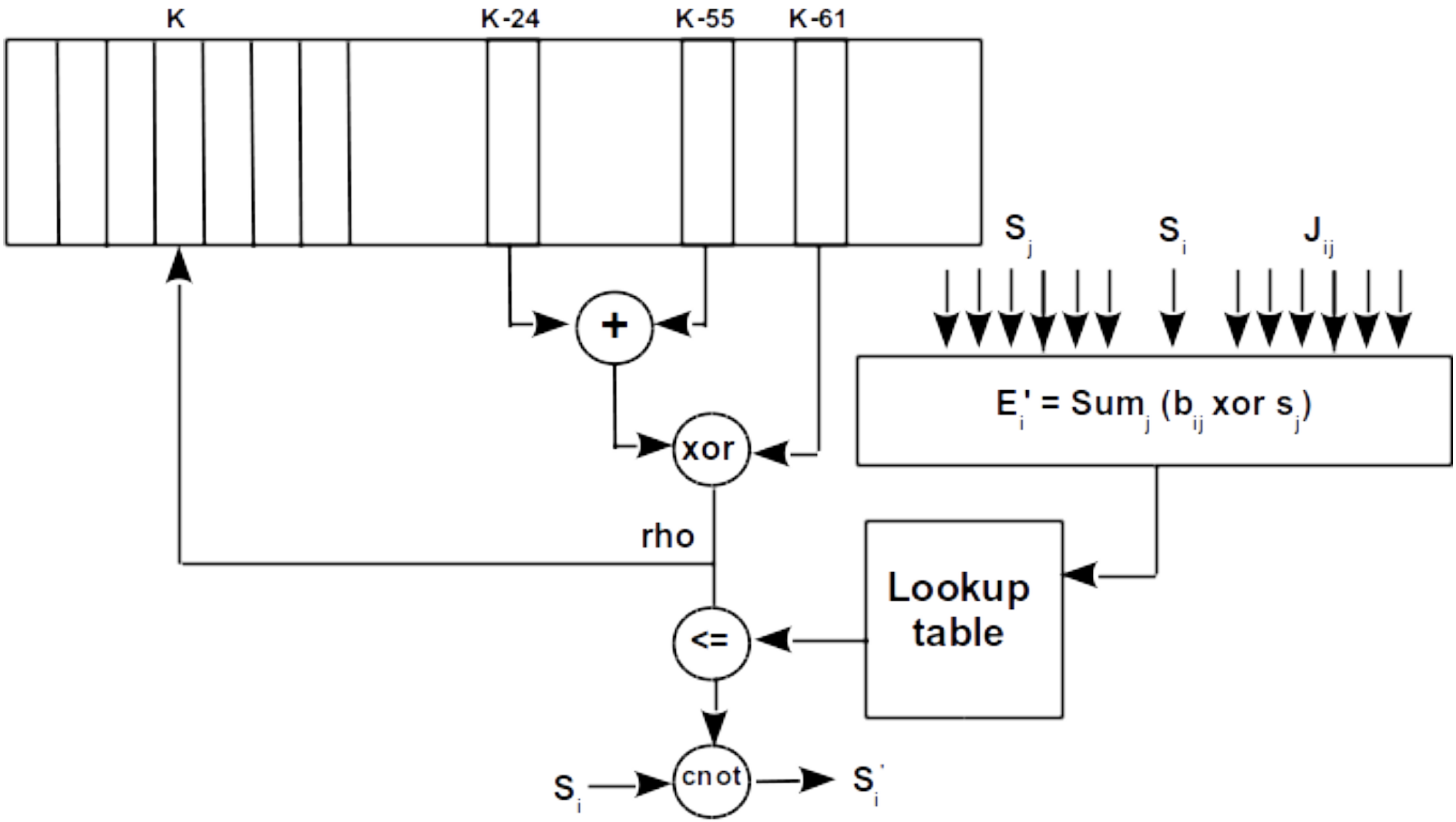
then values $e^{-\beta \Delta E_i}$ cab be pre-loaded on a lookup table indexed by Σ_i.

Since in $\Sigma_i = \sum_{\langle j \rangle} (\sigma_i \oplus \gamma_{ij} \oplus \sigma_j)$ value of σ_i is constant we can use as index

$$\Sigma_i' = \sum_{\langle j \rangle} (\gamma_{ij} \oplus \sigma_j)$$

reducing the number of xor to compute Σ_i from 12 to 6.

Single-Spin Update Engine



Single-Spin Update Engine

- 62.5 MHz on Janus I with 1000 spin-update engine
- 200 MHz on Janus II, with 2000 spin-update engine
- 13 bits read + 1 bit write, total bandwidth of $\approx$ 5 Tbit/s
- on one SP we measure: 16 ps/spin on Janus I, 2 ps/ spin on Janus II
- on 16 SP we have: 1 ps/spin on Janus I, 0.125 ps/spin on Janus II
- 30-35 Watts on Janus I, 25-30 Watts on Janus II
- each update engine requires
  - 6 1-bit XOR
  - 5 3-bit ADD
  - 1 32-bit ADD and 1 32-bit XOR (computation of random)
  - 1 32 bit CMP

  accounting the above ops as 3 32-bit standard operations, we have a performance of $\approx$ 190 GOPS for Janus I, and $\approx$ 1.2 TOPS for Janus II.

Parallel Simulation of Spin Glass on Commodity Processors

Several levels of parallelism can be exploited in Monte Carlo Spin Glass simulations.

- The lattice can be divided in a *checkerboard* scheme: alghorithm is first applied to all **white** spins, and then to all **blacks** (order is irrelevant).
- SIMD instructions can be used to update up to $V \leq L^3/2$ (white or black) spins in parallel (**internal parallelism**).
- The lattice can be divided in several sub-lattices and allocated to different *cores*. Boundaries need to be updated after updating the bulk (**internal parallelism**).
- Several lattices (**samples** or **replicas**) can be simulated in parallel using **multispin**-coding approach (**external parallelism**).

Multispin Encoding (1)

Multispin encoding (for the EA model) allows to simulate several systems in parallel.

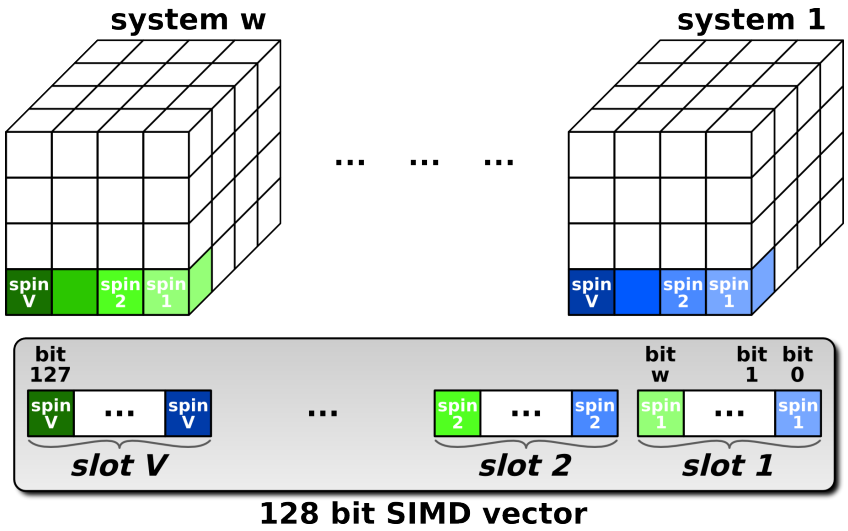
Assuming to run simulation on a k-bit architecture ($k = 32, 64, 128, 256, 512$):

- spins and couplings are represented by binary values $\{0, 1\}$
- a k-bit architectural word hosts k-spins of k different systems
- Metropolis update procedure can be bit-wise coded (no conditional statements, only bit-wise operations)

Require: ρ pseudo-random number
Require: $\psi = \text{int}\,(-(1/4\beta)\log\rho)$, encoded on two bits
Require: $\eta = (\text{ not } X_i)$, encoded on two bits
$c_1 = (\psi[0] \text{ and } \eta[0])$
$c_2 = (\psi[1] \text{ and } \eta[1]) \text{ or } ((\psi[1] \text{ or } \eta[1]) \text{ and } c_1)$
$s_i' = s_i \text{ xor } (c_2 \text{ or not } X_i[2])$ // update value of spin s_i

Multispin Encoding (2)

We enhanced **multispin encoding** approach combining it with SIMD-instructions to exploit both **internal-** and **external-**parallelism.



- the 512-bit SIMD-word is divided in $V = 8 \ldots 512$ slots
- each slot hosts one spin-values of a system
- each slot hosts w spin-values of different lattices.

V = **internal**-parallelism degree, w = **external**-parallelism degree.

Random Number Generation

At each MC-step V (pseudo-)random numbers are needed.
Same random value can be shared among the w lattice-replicas.

WHEEL[K] = WHEEL[K-24] + WHEEL[K-55]

ρ = WHEEL[K] $\oplus$ WHEEL[K-61]

- WHEEL is an array of unsigned integer
- SIMD instructions can be used to generate several random numbers in parallel.

Spin Glass Simulation on MIC

Lattice is split in C (number of cores) sub-lattices of contigous planes, and each one (of $L \times L \times L/C$ sites) is mapped on a different core.

- each core first update all the white spins and then all the blacks
- w/b spins are stored in *half-plane* data-structures (of $L^2/2$ spins)

1: update the boundaries half-plane (indexes (0) and $((L^3/C) - 1)$).
2: **for all** $i \in [1..((L^3/C) - 2)]$ **do**
3: update half-planes (i)
4: **end for**
5: exchange half-plane (0) to the *previous core* and half-plane $((L^3/C) - 1)$ to the *next core*.

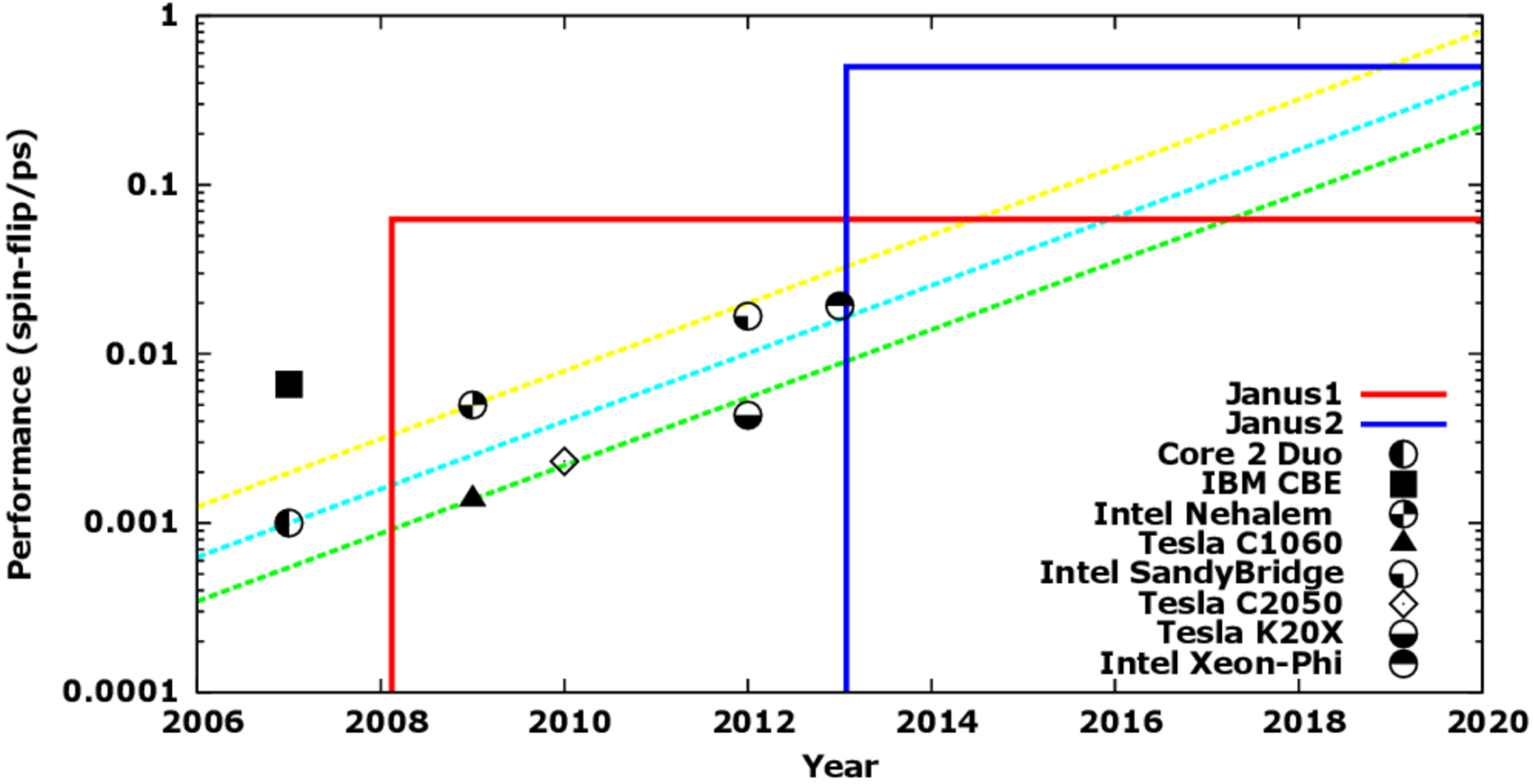
This approach requires only data exchange across the cores.

Results Comparison

System	Core 2 Duo	CBE (16 cores)	JANUS	C1060	NH (8 cores)	C2050 (16 cores)	SB	K20X	Xeon-Phi	*Janus II*
Year	2007	2007	2008	2009	2009	2010	2012	2012	2013	2013
Power (W)	150	220	35	200	220	300	300	300	300	25
SUT (ps/flip)	1000	150	16	720	200	430	60	230	52	2
Energy/flip (nJ/flip)	150	33	0.56	144	244	129	18	69	15.6	0.05

- Spin-update-time (SUT) of EA simulation codes on a 64^3 lattice
- The table also shows rough estimates of the energy needed to perform all the computing steps associated to one spin flip.

Results Comparison



Conclusions

- FPGAs play an important role and are also used for many other applicationsi, e.g. telecommunications, finance, security, . . .
- more recently they have been used also to accelerate also more complex applications:
  - Reverse Time Migration to analyse echos produces by “shot” sources; used by Oil&Gas industry
  - LQCD simulations
  - LBM and CFD simulations (see next talk)
- FPGAs have lot of potential computing power but programmability is the main issue preventing to make them available for users
- the main focus of EuroEXA project is that of using FPGA as accelerators providing high-level programming tools to code applications (see next talk).