OPEN-97-034
11 Nov 1997

EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

A model for Intelligent Random Access Memory architecture (IRAM): cellular automata algorithms on the Associative String Processing machine (ASTRA)

Francois Rohrbach
CERN
Genève 23, CH-1211, Switzerland
F.Rohrbach@cern.ch

Géza Ódor
Research Institute for Materials Science
P.O.Box 49 H-1525 Budapest, Hungary
odor@gergo.atki.kfki.hu

György Vesztergombi
Research Institute for Particle Physics
P.O.Box 49 H-1525 Budapest, Hungary
veszter@rmki.kfki.hu

Abstract

In the near future, the computer performance will be completely determined by how long it takes to access memory. There are bottle-necks in memory latency and memory-to processor interface bandwidth. The IRAM initiative could be the answer by putting Processor-In-Memory (PIM).

Starting from the massively parallel processing concept, one reached a similar conclusion. The MPPC (Massively Parallel Processing Collaboration) project and the 8K processor ASTRA machine (Associative String Test bench for Research & Applications) developed at CERN [26] can be regarded as a forerunner of the IRAM concept.

The computing power of the ASTRA machine, regarded as an IRAM with 64 one-bit processors on a 64×64 bit-matrix memory chip machine, has been demonstrated by running statistical physics algorithms: one-dimensional stochastic cellular automata, as a simple model for dynamical phase transitions. As a relevant result for physics, the damage spreading of this model has been investigated.

1 Introduction

Over the last two decades we became witnesses of an information-technological revolution. The performances of basic computing devices (processor, memory, network) has grown exponentially. However the rate of speed growth was different. While the single processor speed has increased 60 % per a year the DRAM access time decreased by only 10 % yearly. This results in an increasing processor-memory unbalance, excluding the possibility of fast growth achieved by the processors even with additional hardware and software complexity (cashes, complicated compilers). It was found that cashes don't work very well (example the alpha 21162 chip with 2 levels of cashes spends 75 % of all CPU time waiting for memory access) and following the trends the situation will be even worse in the future.

A natural solution for this problem suggested by D. Patterson [25] would be the integration of processors into the memory chips (PIM) making intelligent RAMs (IRAM). So the processors can access the memory matrix bits with a huge bandwidth without using costly cashes or complicated compiler algorithms etc.

Ancestors of the IRAM idea can be found among graphics accelerators, in microprocessor+cash chips and in SIMD, MIMD parallel processing architectures. But these efforts were memory limited requiring multiple chips and therefore the inter-chip communication overhead suppressed the advantages compared to the rapidly developing conventional, commercial machines.

One example of such a SIMD architecture is the ASTRA machine, which has been built as the outcome of the Massively Parallel Processing Collaboration [27] as a prototype and test-bed for real applications to be introduced later.

Now the situation has changed:

- The DRAM capacity has reached the 1 Gbit/chip integration, which is enough to store a typical program.
- The processor-memory speed gap has became larger than 2 orders of magnitude, causing expensive contortions of the current systems.

Independently of the knowledge of the current IRAM research projects in 1997 we proposed a very similar architecture, the "bit-matrix machine" [29] motivated by the need to improve our current ASTRA SIMD architecture. In the age of informations graphical images, multimedia processing will play a prominent role. That requires the processing of huge data in a SIMD like fashion. We have shown that our proposed architecture fits very well to diverse applications like matrix algorithms [29, 30] or computer graphics[23].

In this new paper we demonstrate with the help of the ASTRA machine, that the IRAM architecture is applicable for simulations in statistical physics too. We show that new and well accepted simulation methods can be implemented effectively on the ASP by mapping one cell variable per APE. We shall investigate the critical phase transition of different one-dimensional stochastic cellular automata (SCA). In order to make the simulations of the SCA feasible on the ASP we needed fast, on-processor random number generator (RG). Since the memories of APE-s are very limited we implemented a lagged Fibonacci RG with limited (18-bit) resolution. That means that although the RG cycle is large, the accuracy of the probability that we can achieve is 2^{-18}, which is enough, since usually the maximum accuracy what we need is $4 - 5$ digits. The algorithm of this RG will be shown in section 3. To illustrate the way of ASP programming in section 5 we shall show some tricks of the algorithm (thresholding etc.). In section 6 we introduce time dependent simulations to measure survival probability and critical exponents. Two SCA possessing different set of critical exponents (universality classes) have been chosen to test the algorithms and the random generator. The critical exponents, that we have determined with great accuracy are all in accordance with previous results. Finally in section 7 we apply the method to measure damage spreading and investigate the relation of damage spreading transition to the parity conserving phase transition as well. Timing results will be compared with serial and other parallel machine's data.

2 The ASTRA machine

The associative string processing (ASP) ASTRA machine has been built as the outcome of the Massively Parallel Processing Collaboration [27] as a prototype and test-bed for real applications. Existing machines with 4 and 8K processors are real examples of massively parallel machines. The processors (APE-s) have very simple structure: 64-bit memory register, 1-bit adder CPU and a few flag bits for special purposes. The communication is also as simple as possible; basically a couple of string buses connecting the APE-s to the central controller, plus a one-bit inter APE communication channel, but the special associative functions make it a quite flexible architecture. The basic

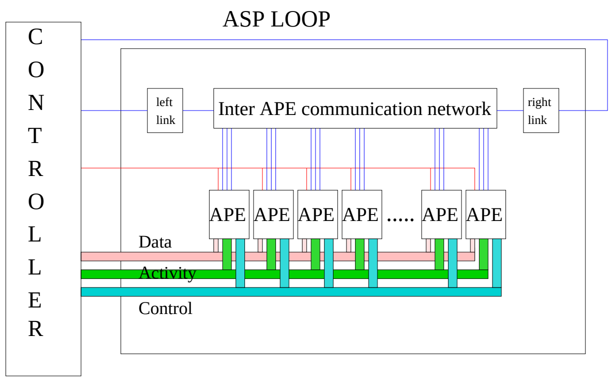

Figure 1. Associative String Processor (ASP) architecture

designing idea was to integrate as many as possible of these simple APE strings on a simple chip, making it an effective tool for processing large quantity of data in real time. This architecture, and the basic operations (convolution type) for which it was designed make it applicable for simulating cellular automata (CA).

As it was mentioned the APE-s are very simple processing elements, built up from a one-bit adder, a 64-bit memory register, where each bit in the registers can be selected in parallel for all APE by the central controller. There is a 6-bit "Activity Register" a Carry, an Activity, and tagging flags in each APE, see Figure 2. The central controller, the "LAC", dictates the program to be executed by each APE, makes the Input/Output data transfer between the 32-bit Data Bus, the 6-bit ternary-logic Activity Bus and its global registers. It is connected to the Inter-APE communications network

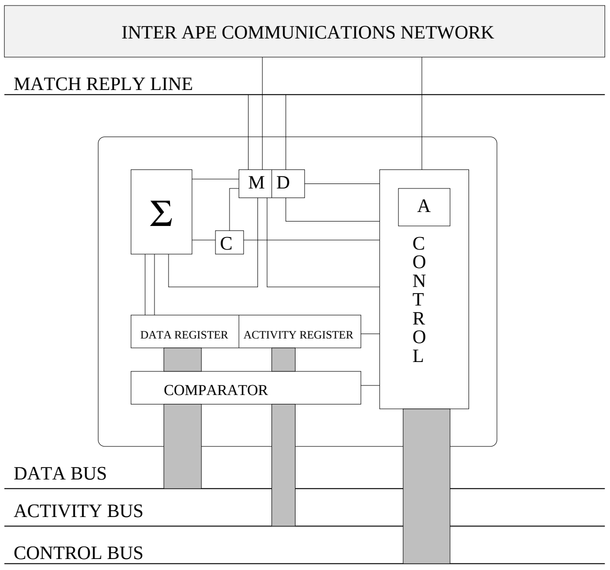

Figure 2. Associative Processing Element

by left and right links of the ASP substring (TagShift operations) and to the global Match Reply line.

The detailed description can be found in [17]. To minimize the data movement the APE-s are content addressable via the Data and the Activity Buses and the comparators with ternary logic (1,0,dont'care).

The flow of instructions roughly starts as follows. Partition of the string, according to some architecture (e.g. subdivision to substrings etc.) fitting the best to our data structure. Partition of the local memories of APE-s simultaneously. Each operation is built of two main parts:

- Selection of operands
  - by selection of bits within the memory registers of all APEs,
  - by content addressing ("Tagging") of APEs (this involves logical function of Data and Activity Register bits)
  - and by an activation of some neighbourhoods of the selected APEs.
- The execution of the logical functions on the operands and storing them to the selected places

The horizontal and vertical repartitioning of the string-memory can be repeated any time during the program flow.

3 The random number generator

Owing to the bit-addition capabilities of APE-s, a natural choice for random generator was a "lagged Fibonacci" generator :

$$x_i = (x_{i-l} \pm x_{i-k} \pm c)\ mod\ 2^b, \qquad (1)$$

where the lags are in practice (l,k) : $(17,5)$; $(55,24)$; $(24,10)$ ..., and the carry c comes from a previous generation step. We choose (l,k) : $(17,5)$, which is used in the Connection Machine 5 for parallel random number generation too. This type of generator has a very long cycle in general [16, 18].

Generating "good" random numbers is not an easy task, one can never be sure of the true randomness of a very long sequence and as digital computers have been developed more and more generators failed on more and more demanding statistical tests [16]. In fact the growing number of scientific papers and disputes aiming to increase the quality of random generators shows that it is far from being solved.

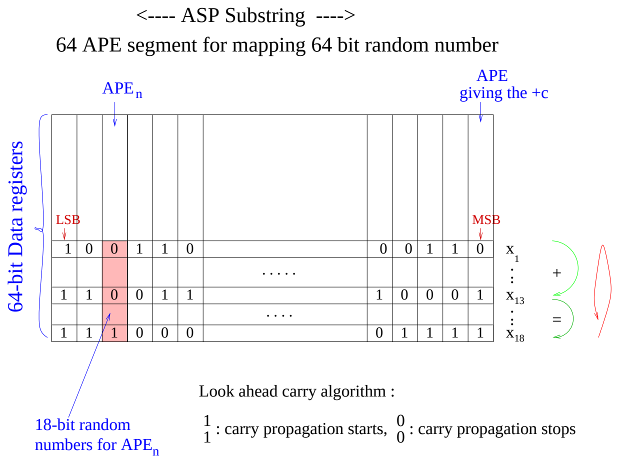
<---- ASP Substring ---->
64 APE segment for mapping 64 bit random number
APE_n
APE giving the +c
64-bit Data registers
LSB
MSB
1 0 0 1 1 0 ..... 0 0 1 1 0
1 1 0 0 1 1 .... 1 0 0 0 1
1 1 1 0 0 0 0 1 1 1 1
x_1
x_13
x_18
+
=
Look ahead carry algorithm :
1/1 : carry propagation starts, 0/0 : carry propagation stops
18-bit random numbers for APE_n

Figure 3. The random number generator algorithm

We have tested our generator by checking the results on CA simulations. The 64-bit long x_i random numbers are loaded along the 64-bit segmented ASP string, so that each APE contains one bit of x_i, $i \in (1..18)$ (see Fig. 3). Therefore the bits of different random numbers make up a random integer word of size 18 in each APE. The advantage of this kind of representation is that it does not take too much register space (since we don't need too high resolution) but still we have a long period (determined by the length (64) of x_i. Furthermore we can execute the additions in bit parallel with the "look-ahead carry algorithm", which avoids carry propagation. To build random words for all APE-s we update the 18 bits sequentially within 18 simple operation cycles.

4 Cellular automata

Physicists became interested in CA, invented by von Neumann, when they discovered, that CA with some external noise (stochastic CA (SCA)) can be regarded as models of statistical physical systems [31]. Since the detailed balance condition [1] is not necessarily satisfied during the evolution, the systems are not converging to a thermodynamic equilibrium. Therefore SCA in general provide a model for non-equilibrium systems. This area of statistical physics is far less known than the equilibrium physics and therefore much interest is paid for it currently. It turned out that this field covers a very rich area of phenomena ranging from physical and chemical reactions to biological and economical models. One can model with it complex systems of many variables, which can be humans (e.g voter models or disease spreading), animals (e.g. races of populations), economical units (e.g. crashes of stock exchange), vehicles (e.g. traffic models), elementary particles and many other things [11].

Our interest now is concentrated on the exploration of phase transitions and critical phenomena. They are connected to many other new topics like self organised criticality, chaotic systems, fractals etc. The advantage here is that even one-dimensional systems may exhibit phase transitions and different critical phenomena. Therefore these low-dimensional "toy models" serve as learning base of more complex systems. Furthermore in many cases the real physical space is low-dimensional.

5 ASP algorithmic tricks

The random number integers should be compared to some threshold value in order to make a decision (see Fig.3). For this purpose the global "SerialShiftRegister" of the controller (LAC) was used. We load the $pt \in [0, 2^{18})$ integer threshold in it at the initialisation, which is the rescaled $p \in [0, 1)$ value. Then we use the test-and-rotate command of the LAC to apply parallel thresholding for all APEs simultaneously from the most significant bit to the least significant bit. The generator works in a constant number of steps, the thresholding requires 18×2 steps.

[1] $P(\{s\})W(\{s\} \rightarrow \{s'\}) = P(\{s'\})W(\{s'\} \rightarrow \{s\})$, where $P(\{s\})$ is the probability of a state, and W is the transition probability

The CA automaton rule is easily parallelizible. We demonstrate it on the stochastic Rule 18 (Fig. 4). This is a rule, according to the state of $s(i,t) \in$

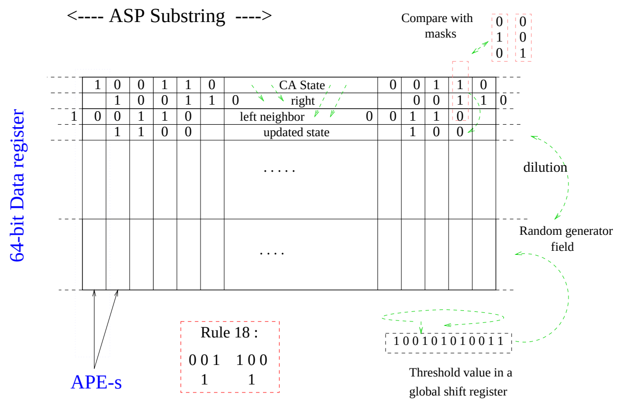

Figure 4. Stochastic Rule 18 CA algorithm on the ASP

$(0, 1)$ variable at time t and space i is determined by $s(i-1, t-1), s(i, t-1)$ and $s(i+1, t-1)$. For the deterministic Rule 18 we get a 1 at time t if just one of the neighbours of $s(i, t-1)$ was 1 and $s(i, t-1)$ was in 0 state. The possible cases look like :

```
t-1:  100 001 000 111 011 110 010 101
 t:    1   1   0   0   0   0   0   0
```

The nearest neighbour communication can be done in parallel for all APEs by the "TagShift" operation within 2 CPU clocks. The rule can be applied in 4 clock units by the "TagByte" and "BitWrite" instructions.

To make the CA random, we dilute the '1'-s with probability p acceptance rule. If the random number is greater than pt the '1' state is put to '0'.

Altogether we can see that updating the SCA state can be done within a few clock cycles independently of the size.

To process the information carried by the states one would need in general global parameters obtained by data concentration (ex.: counting the '1'-s). The calculation of some statistical measure requires $O(\log(\text{system size}))$ time theoretically, limited by the inter APE communications. We have chosen a simpler variable for our demonstration: we measure the time necessary for falling in the absorbing state. Indeed, the presence of existing '1'-s can be checked by the global "Match-Reply" capability of the ASP, and we only have to count the elapsed time steps during a run, equal to the number of updating cycles.

This SCA can have two different type of steady states – separated by a phase transition – by varying the acceptance probability. For small p values the system always evolves to the "absorbing state" built up from 0-s only and from which there is no return according to the rule. For $p > p_c(\sim 0.809)$ a chaotic steady state of finite concentration emerges. At the critical point the survival probability shows a power-law behaviour characterised by some universal critical exponent.

6 Time dependent Monte Carlo simulation

Time dependent Monte Carlo simulation (TDS) suggested by [10] have become a very precise and effective tool in statistical physics. We start the system from a single active state and follow its statistical properties for a few thousand time steps. In this case we don't let the maximum size of the active cluster to over-grow the computer memory, therefore no finite size effects can come in. We have to take into account finite time corrections and statistical errors only. In this case mapping of the cellular space onto processors is not very effective since at the phase transition point to the absorbing state the system is very sparse, plus we started from a single active seed. List oriented algorithms exploit the processor power much better. We have chosen to start our simulations on the ASTRA machine with a one cell - to one APE mapping TDS of the Rule 18 SCA because this is simple and there are well established results in the literature to check the programming and the random number generator. Also this mapping can algorithmically be very effective for slightly more complex time dependent simulations, like damage spreading (see section 7) or persistence measurements (recently, one more critical exponent was proposed [3], the persistence exponent Θ, associated with the probability $p(t) \propto t^{-\Theta}$, that the global order parameter has not changed sign up to time t after a quench to the critical point [19]).

In general one can calculate the following quantities;

- survival probability: $p_s(t)$
- concentration of '1' states: $c(t)$
- average mean square distance of the spreading of '1' states from the center $R^2(t)$

The evolution runs are averaged over N_s independent runs for each different value of p in the vicinity of p_c (but for $R^2(t)$ only over the surviving runs). At the critical point we expect from theory that these quantities to behave in accordance with power law as $t \to \infty$, i.e.

$$p_s(t) \propto t^{-\delta} \ , \tag{2}$$

$$c(t) \propto t^{\eta} \ , \tag{3}$$

$$R^2(t) \propto t^{z} \ . \tag{4}$$

For our test purposes we measured the exponent δ, of the scaling function of the survival probability.

We have chosen for the first test a well known SCA, the stochastic Rule 18 model [1, 28] possessing universality of the Directed Percolation (DP) or Reggeon Field Theory[13, 5, 2]. Since the definition of this model has been given already in section 5, we just present the results here.

We followed the time evolution of this system for $t = 8192$ time steps and averaged over $N_s = 10^5$ independent samples for each p control parameter. To estimate the critical exponents we determined the local slope:

$$-\delta_p(t) = \frac{\ln\left[p_s(t)/p_s(t/m)\right]}{\ln(m)} \tag{5}$$

using $m = 8$. In the case of power-law behaviour we should see a straight line as $1/t \to 0$, when $p = p_c$. The off-critical curves should possess curvature. Curves corresponding to $p > p_c$ should veer upward, curves with $p < p_c$ should veer downward. The Figure 5, shows

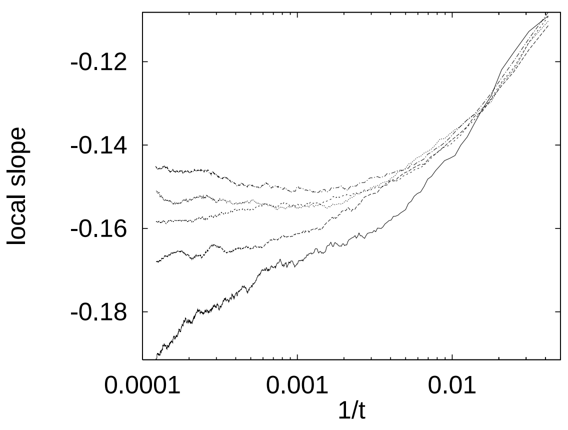

Figure 5. TDS results for the slope $\delta_p(t)$ at the critical point. Parameters corresponding to curves from the bottom to top $p = 0.809, 0.8092, 0.80948, 0.80949, 0.8095$.

our result for this quantity as function of $1/t$. As we can see, the value $p = 0.80948(1)$ results in the most power-law like behaviour therefore the critical point is there in agreement with former results of the Rule 18 SCA [1, 28]. For the critical exponent δ we can read of ~ 0.16. This value agrees with the most precise series expansion estimate for the directed percolation [15] : $\delta_{DP} = 0.1597(3)$.

The other model, which is an exception from the robust class of DP is Grassbergers's B model. This model belongs to the so called "Parity Conserving" (PC) universality [6, 12, 14, 20], because the number of particles is conserved modulo 2 globally. It is realized by the following SCA (we show the configurations at $t-1$ and the probability of getting 1 at time t)

```
t-1: 100 001 101 110 011 111 000 010
 t:   1   1   1   p   p   0   0   0
```

The time evolution pattern for small p is a regular chess-board (with double degeneracy), while for $p > p_c$ it is chaotic. If we consider differences from the alternating ground state pattern as kinks : 00 or 11, the dynamics of these kinks is described by the processes above. These kinks can be regarded either as particles or "Bloch walls of a spin system". They can also be mapped to surface growing problem with a roughening transition.

For this model we also have performed simulations using the ASTRA machine to measure exponent δ and the critical point. As the Figure 6 shows the results are in good agreement with that of Grassberger[6]. The

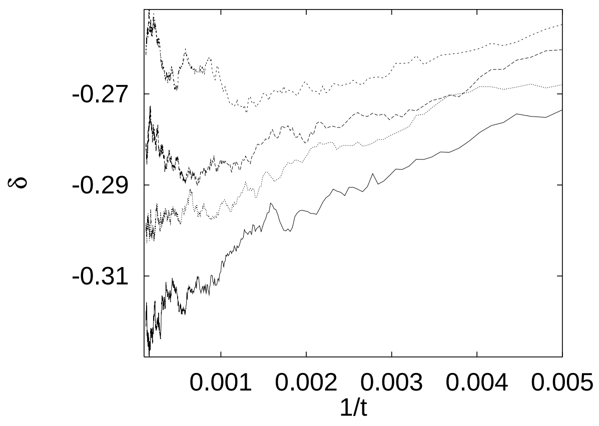

Figure 6. TDS results for $\delta(t)$ for nearly critical point values of $p = 0.535, 0.538, 0.54, 0.545$. The statistical averaging was done over 10000 independent samples

exponent δ is around $0.29(1)$, which agrees well with the PC universality class value $\delta_{PC} = 0.285(2)$[14].

7 Damage spreading calculations

Damage spreading simulations have become an important tool for exploring time-dependent phenomena in statistical physics[7]. By applying it to spin systems or stochastic cellular automata (SCA) one follows the evolution of a single difference (damage) between two identical systems (replicas), driven by the same random sequence. The initial damage may grow or shrink depending on control parameters (temperature etc.) of the model. If there is a phase transition between such regimes one can measure dynamical critical exponents very precisely[8]. Grassberger conjectured that [9] damage spreading transitions separating chaotic and non-chaotic phases always belong to the DP universality class except they coincide with some ordinary phase transition point of different universality class. Examples for this hypothesis are the Ising model (spin model of magnetic medium) with different dynamics and certain SCA. In case of absorbing phase transitions there have been a few exception from the DP universality class up to now. One of them is the PC transition and therefore we thought it would be interesting to investigate the damage spreading properties for models exhibiting this kind of phase transition. One candidate for this would be the Grassberger B SCA, which is chaotic for $p = 1$ and ordered in the $p = 0$ limiting cases, therefore there must be a damage spreading transition in between.

The replica systems are started from random initial conditions in this case and therefore the full capacity of the parallel machine is used now at the beginning. We applied periodic boundary conditions. We followed the extinction of the damage in case of Grassberger's model B. Similarly to the simple TDS case near the transition point the damage survival probability curves veer up or down showing that the initial damage survives or heals in the long time limit. As figure 7 shows the two regimes are separated by a straight line meaning power-like behaviour of the extinction. Therefore the damage spreading critical point is at $p = p_d = 0.632(1)$ not coinciding with the ordinary phase transition point ($p_c = 0.54$) of kinks. This means that the system at the ordinary critical point (p_c) is insensitive to perturbations like random mixing because they die out exponentially in time. The slope of the straight line determines the exponent δ, which is $0.160(1)$ indicating DP universality class in agreement with Grassberger's hypothesis although the parity of the damage variables is conserved modulo 2. This is similar to what was found in PC-conserving systems, when the absorbing state symmetry was broken by external field [24, 21]. Here the absorbing state symmetry is broken too, since it is fluctuating owing to $p_d > p_c$. This means that the PC conservation is not sufficient condition for having non-DP universality. More discussion on this topic is in preparation [22]. In conclusion, the active phase is

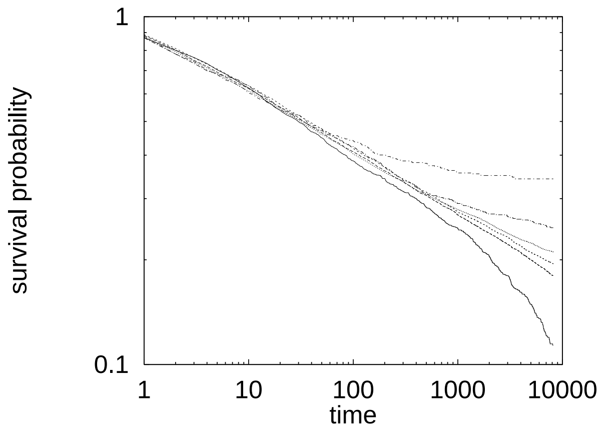

Figure 7. Log-log plots of the survival probability curves for different values of $p = 0.62, 0.63, 0.632, 0.634, 0.64, 0.65$. The statistical averaging was done over 10000 independent samples

divided up by the damage spreading critical point similarly to the case of the Domany-Kinzel SCA [4, 9].

As for the timing, we achieved 2×10^{-7} sec / cell update speed at the critical point. This is about one fifth of the speed that we measured on FUJITSU AP-1000 supercomputer built up from 64 Supersparc processors where we applied task parallelism. That means that each processor runs the same program with different numbers to increase the statistics. That means that the 8K ASTRA was about thirteen times faster than a single Supersparc CPU.

8 Conclusions

Efficient program have been developed and tested for the massively parallel ASTRA machine. The realization of the fast on-processor random generator made it possible to use the architecture for simulation of statistical physical systems. The generator and the algorithm were tested by different cellular automaton simulations. Critical points and exponents were determined with great accuracy and the timing results were compared with sequential and other parallel machines data. Since the hardware elements of the machine represent the late 80-ies technology we can give a performance scaling to show the effectiveness of the architecture. By assuming 200 MHz clock speed instead of the present 20 MHz and 10^6 number of processors contrary the present 8192 we could achieve a speed-up factor of ~ 1000 because of the linear scaling of the algorithm we presented (the first initialisation of the random generator is the only sequential communication along the string). We have performed damage spreading studies resulting in relevant physical results. The algorithms can easily be developed further for other dynamical simulations.

We thank John Lancaster for the support he gave concerning the ASTRA machine.

References

[1] N. Boccara and M. Roger. *Instabilities and Nonequilibrium Structures IV.* Kluwer Academic, The Netherlands, 1993.

[2] J. L. Cardy and R. L. Sugar. *J. Phys. A: Math. Gen.*, 13:L423, 1980.

[3] B. Derrida and et al. *J. Phys.*, A 27, 1994.

[4] E. Domany and W. Kinzel. *Phys. Rev. Lett.*, 53:447, 1984.

[5] P. Grassberger. *Z. Phys. B*, 47:365, 1982.

[6] P. Grassberger. *J. Phys. A : Math. Gen.*, 22:L1103, 1989.

[7] P. Grassberger. *J. Phys. A : Math. Gen.*, 28:L67, 1995.

[8] P. Grassberger. *Physica A*, 214:547, 1995.

[9] P. Grassberger. *J. Stat. Phys*, 79:13, 1995.

[10] P. Grassberger and A. de la Torre. *Ann. of Phys*, 122:373, 1979.

[11] H. Gutowitz. Frequently asked questions about ca. *http://alife.santfe.edu/alife/topics/cas/apps/apps.html*, June 1996.

[12] A. T. H. Takayasu. *Phys. Rev. Lett*, 68:3060, 1992.

[13] H. K. Janssen. *Z. Phys. B*, 42:151, 1981.

[14] I. Jensen. *Phys. Rev. E*, 50:3623, 1994.

[15] I. Jensen and R. Dickman. *J. Stat. Phys.*, 71:89, 1989.

[16] K. Kankaala and et al. *Phys. Rev. E*, 48:R4211, 1993.

[17] R. M. Lea. Asp: a cost effective parallel microcomputer. *IEEE Micro*, October 1988.

[18] M. Luscher. *DESY preprint*, 93(133), 1993.

[19] S. N. Majumdar and et al. *Phys.Rev.Lett.*, 77, 1996.

[20] N. Menyhárd. *J. Phys. A : Math. Gen.*, 27:6139, 1994.

[21] N. Menyhárd and G. Ódor. *J. Phys.*, A 29, 1996.

[22] G. Ódor. in preparation.

[23] G. Ódor and et al. Optimisation of computer graphic algorithms in pim-simd architecture. *preprint*, June 1997.

[24] H. Park and et al. *Park, Phy. Rev.*, E 52, 1995.

[25] D. A. Patterson. Microprocessors in 2020. *Scientific American*, September 1995.

[26] F. Rohrbach. Massively-parallel processing: a need and a powerful tool in sciences for the future, with particular emphasis on associative string processors for high energy physics experiments, 5appc, kuala lumpur. August 1992.

[27] F. Rohrbach. The mppc project : final report. *CERN preprint*, 93(7), 1993.

[28] G. Szabó and G. Ódor. *Phys. Rev. E*, 49:2764, 1994.

[29] G. Vesztergombi and et al. Boolean matrix exponentiation. *Proceedings of the Fifth Euromicro Workshop on Parallel and Distributed Processing. London UK*, pages 461–467, January 1997.

[30] G. Vesztergombi and et al. Scalable matrix multiplication algorithm for iram architecture machine. *preprint*, June 1997.

[31] S. Wolfram. *Rev. Mod. Phys.*, 55:601, 1983.