Publishing House
AKAPIT

COMPUTER METHODS IN MATERIALS SCIENCE
Informatyka w Technologii Materiałów

Vol. 14, 2014, No. 4

EFFICIENT CELLULAR AUTOMATA MODEL FOR PREDICTION OF DAMAGE OF HOT FORGING TOOLS DUE TO THERMAL FATIGUE

PAWEŁ NOWAK, ŁUKASZ RAUCH*

AGH – University of Science and Technology, Mickiewicza 30, 30-059 Kraków, Poland
**Corresponding author: lrauch@agh.edu.pl*

Abstract

The paper presents design and implementation of the cellular automata model, which predicts damage of forging tools due to fatigue. The transition rules for the model were developed on the basis of known information regarding crack initiation and propagation. The coefficients in the model were determined by the inverse analysis of the thermal fatigue tests performed on the Gleeble 3800 simulator. The model was connected with the finite element (FE) program, which simulates stresses in tools during forging. This multiscale approach is extremely demanding as far as computing times are considered, therefore, an efficient implementation of this model was the main objective of the work.

Cellular Automata (CA) model of fatigue fracture was realized for heterogeneous hardware architectures. The source codes were implemented in form of the CA framework and algorithm on the basis of API (Application Programming Interface), delivered by the OpenCL library. The framework which supports CA computations was divided into library and application parts. The former supplies API to handle kernels of the program in the OpenCL technology dedicated to both CPU and GPU devices. Application part uses API offered by the library to supply and control input parameters for calculations. This part creates also CA space, introduces parameters of cells and sends them to calculations on accelerators. Implementation of the model was described in the paper and an example of calculation for the hot forging die was presented. High speed-up was obtained due to application of four GPU cards, while a good scalability was maintained.

Key words: Cellular Automata, forging tools, fatigue, distributed computing

1. INTRODUCTION

Die forging has been for years and is now an advanced forging technique used in the mass production of critical parts. Low durability of the forming tools is an important limitation of this process. It is estimated that the costs of the tools may amount to as much as 8–15% of the total production costs. Actually, if the time needed to replace the worn out tooling are accounted for, the costs may increase even further. Moreover, tool wear significantly contributes to deterioration in the quality of the produced forgings. Basic information on wear can be found in (Bayer, 2004) and a review of the degradation mechanisms in forging tools was presented by Gronostajski et al. (2014). Manufacturers of die forged products make efforts to reduce their costs and improve the quality of the forgings, e.g. (Behrens et al., 2012). Numerical modelling of the tool wear can be helpful in reaching this goal.

Thus, development of the model, which can predict cracking due to fatigue accounting directly for the material structure, was the main objective of this work. CA method has capabilities which allow reaching this goal. Several examples of successful applications of the CA technique to simulation initiation and propagation of microcracks can be found in the literature. Examples for modelling fracture in steels (Shterenlikht, 2003), creep (Nowak, 2011), fatigue (Shibutani, 1999), deformation of multiphase steels (Perzyński, 2014) confirmed very good pre-

ISSN 1641-8581

dictive capabilities of this method. On the other hand, application of this method to simulation fracture in forging dies requires connection with the Finite Element (FE) method in the macro scale (so called CAFE method). In this approach states of strains and stresses in the macro scale are calculated using FE code and crack initiation and propagation in the micro scale is calculated using CA attached to the FE nodes. The local states of strains and stresses are used as boundary conditions for the CA space. This approach, although very accurate, requires long computing times. Therefore, search for a possibility of decreasing of these times was the second objective of the paper.

2. CA MODEL

2.1. Physical basis

Tool life is defined in several ways. In production it is usually expressed by the number of forgings which can be manufactured with this tool to obtain products of desired quality. According to this definition the average tools durability can change in wide range from 2000 to 20 000 pieces. In tool terms, durability is associated with degradation and so it is defined as the ability to withstand degradation phenomena (Gronostajski et al., 2014). In the present paper the latter definition is considered.

Prediction of the tool life using conventional methods has been reasonably well researched, see for example (Kim & Choi, 2009; Behrens & Schäfer, 2009; Behrens et al., 2012). There is also an extensive knowledge based on physical analysis of macro- and microstructure at various stages of exploitation of tools. Micrographs of cross-sections of the hot forging dies investigated by (eg. Lavtar et al., 2011; Gronostajski et al., 2014) revealed information, which was used in the present work to develop transition rules for the Cellular Automata model. Briefly, a compound layer was observed on the part of the die where pressures were low and sliding was small. On the other hand, this layer was removed on the edge attributed to higher contact pressures and longer sliding lengths. They prevailed in these areas and this led firstly to cracking of the compound layer and then to its spalling (the compound layer is very brittle in comparison to the diffusion layer).

Detailed analysis of micrographs (eg. Lavtar et al., 2011) have shown that initiation of cracks inside the die is not likely. These observations were also confirmed by the thermal fatigue tests performed on the Gleeble 3800 and on the special device described by Gronostajski et al. (2012). The majority of cracks was initiated from the surface (figure 1). Cracks with a length of 50-100 μm were observed at that surface.

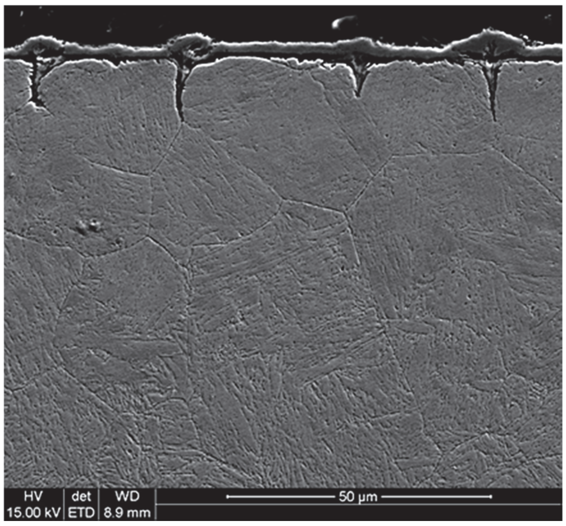

Fig. 1. *Macrograph of the cross section of the tool subjected to 2500 thermal cycles.*

2.2. Description of the model

Cellular automata model for low cycle thermal fatigue of hot forging dies was developed. Two transition rules, one for crack initiation and the second for crack propagation, were defined. Both phenomena occur with certain probability. The magnitude of the stress, which is the internal variable, is the main factor controlling these phenomena. Orientation of this stress with respect to grain structure plays crucial role in the definition of probabilities. For the purpose of defining of the transition rules four possible locations of the cell were assumed:

- at the grain boundary at the surface of the die (figure 2a),
- at the grain boundary inside the die (figure 2b),
- inside the grain at the surface of the die (figure 2c),
- inside the grain inside the die (figure 2d).

These locations are directly related to the probability of nucleation and propagation of cracks.

Critical stress for the crack initiation is the second internal variable defined for each cell. External variables include temperatures which involve thermal strain tensor $\boldsymbol{\varepsilon} = \alpha\Delta T\mathbf{I}$, where: α – coefficient of thermal expansion, ΔT – increment of the temperature, $\mathbf{I}$ – unit matrix.

Two states of the cell are possible: non-active (NA) without cracks and active (A) with cracks. The general equation, which defines the transition rule, is:

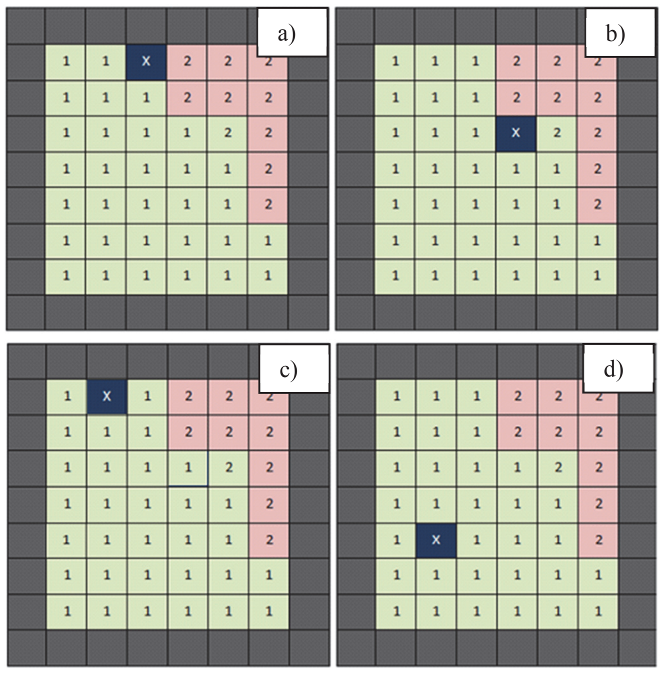

Fig. 2. *Various possible locations of the cell determining nucleation and propagation of fracture.*

$$Y_{i,j}^{t+1} = \begin{cases} if\,(\Lambda) \Rightarrow new\,state \\ else \;\; \Rightarrow Y_{i,j}^{t} \end{cases} \tag{1}$$

where: $Y_{i,j}^{t+1}, Y_{i,j}^{t}$ - state of the (i,j) cell in the current and previous time step, respectively, Λ - logical function which controls the state of the cell and which depends on the state of this cell and its neighbours in the previous time step and on the internal and external variables. Equation (1) means that when the function Λ is true the cell changes its state to a new state, otherwise the state remains unchanged.

Following discussion in section 2.1 it can be concluded that cracks initiate mainly at the surface of the die at grain boundaries, due to diffusion along these boundaries. In the present work the general idea of the transition rule for the crack initiation was based on the Gurson (1977) model, which introduced volume fraction of voids $\xi = \xi_n + \xi_g$. In this model ξ_n refers to initiation and ξ_g refers to growth of cracks. Critical stress for the crack initiation was calculated as:

$$\sigma = P\sigma_0 + \frac{\sigma_m}{1-\xi} \tag{2}$$

where: σ_0 – yield stress, σ_m – average stress, P – material parameter:

The crack initiates when stress σ exceeds certain critical value. Thus the following function Λ in the transition rule for crack initiation was proposed:

$$\Lambda \Rightarrow (Y_{i,j}^{t} = \mathrm{NA}) \wedge (Y_{k,l}^{t} = \mathrm{NA}) \wedge \\ (l_{(0,1)} \le p_n(\sigma, \sigma_{cr}, \beta, S_{i,j}^{t})) \tag{3}$$

where: $Y_{k,l}^{t}$ – state of the cell adjacent to the cell (i,j) in the previous time step, $l_{(0,1)}$ – random number from the interval [0,1], σ_{cr} – critical stress, β – angle between grain border and stress direction, p_n – probability for crack nucleation defined as:

$$p_n\left(\sigma, \sigma_{cr}, \beta, S_{i,j}^{t}\right) = \\ \frac{\arctan\left[\dfrac{m\left(S_{i,j}^{t}\right)x}{1 - m^2\left(S_{i,j}^{t}\right)xq\left(S_{i,j}^{t}\right) + m^2\left(S_{i,j}^{t}\right)q^2\left(S_{i,j}^{t}\right)}\right]}{s\left(S_{i,j}^{t}\right)} \tag{4}$$

where: $S_{i,j}^{t}$ – type of the cell, m, x, q, s – coefficients.

It was assumed that crack can propagate when the cell has a neighbour with the state A. The transition rule for crack propagation is:

$$\Lambda \Rightarrow (Y_{i,j}^{t} = \mathrm{NA}) \wedge (Y_{k,l}^{t} = \mathrm{A}) \wedge \\ \wedge (l_{(0,1)} \le p_{gr}(\sigma, \sigma_{cr}, \beta, S_{i,j}^{t})) \tag{5}$$

where: p_{gr} – probability for crack growth defined as:

$$p_{gr}\left(\sigma, \sigma_{cr}, \beta, S_{i,j}^{t}\right) = p_n(\sigma, \sigma_{cr}, \beta, S_{i,j}^{t}) + \\ + \frac{step_i}{step_{all}} p_{coeff}\left[1 - p_n(\sigma, \sigma_{cr}, \beta, S_{i,j}^{t})\right] \tag{6}$$

where: $step_i$, $step_{all}$ – current step and total number of steps, respectively, p_{coeff} – coefficient.

Coefficients in probabilities p_n and p_{gr} were determined on the basis of thermal fatigue tests performed on the Gleeble 3800 simulator. The values of these coefficients are given in table 1.

Table 1. *Coefficients in equations (4) and (6) obtained on the basis of the thermal fatigue tests*

Parameter	die surface grain boundary	die surface inside the grain	inside the die grain boundary	inside the die inside the grain
m	10.0	5.0	3.0	2.0
q	0.2	0.8	1.2	1.8
s	π	1.4 π	2 π	3 π
p_{coeff}	0.2	0.2	0.2	0.2

2.3. Computing costs

Developed model describes crack initiation and propagation in the micro scale, therefore, application of this model to a practical industrial problem of forging requires calculation of the macro scale parameters. It was done by a connection of the CA model with the FE code, which simulates fields of strains, stresses and temperature in the die. This approach is called CAFE and it has, from the one side, extensive predictive capabilities, but on the other side it requires very long computing times. In the case of fully covered solution where CA model is attached to each integration point in the FE code, the computing costs are extremely high. Computational complexity of FE solution depends on meshing procedure and solver, which is usually proportional to a number of finite elements in the mesh. CA complexity depends also on a number of elements, called cells, which form cellular space being computational domain. In the case of 2D procedure square shape of cellular space requires m^2 cells, where m is width and height of the computing domain. Therefore computational complexity of the homogeneous CA model is defined as O($k \cdot m^2$), where k is a number of floating operations required in one iteration of the CA model multiplied by a number of iterations. That is why distribution and parallelization of calculations is of the highest importance. Possibilities of application of distributed computing were analysed. Since CA calculations in various Gauss points are independent, they can be easily performed on different processors and this problem was not analysed in details. Contrary, possibilities of parallel calculation for a single CA space were investigated. Results of numerical tests and speed-up evaluation are described in the next chapter.

3. PARALLEL CALCULATIONS FOR THE CA MODEL OF CRACKS IN HOT FORGING DIES

CA computing has to face limitations in parallel approach, which are due to the fact that each current state of a CA space requires information from the previous iteration. It means that parallel computing cannot be used efficiently without barriers. All the data from CA space have to be synchronized before the next time step. This involves additional computing costs. The parallelization proposed in this work is focused on multilevel approach, which allows to distribute large computing tasks onto separate computing nodes, composed of many computing devices in heterogeneous architecture. Therefore, at the first level of parallelization CA space is divided into smaller sub-spaces according to the equal division strategy, while on the second level the parallelization takes place regarding available computing devices. The division of a CA space on the second level of parallelization depends strongly on computing capabilities of available devices, their type and number inside the computing node. Generally, in most cases modern processers in computing devices are composed of internal streaming processors, which include computing cores realizing calculations in threads. Streaming processors usually perform calculations according to Single Instruction Multiple Data (SIMD) strategy. Groups of cores receive models implemented in form of computing kernels and CA sub-spaces. All working elements in one group are performed together on one device and then they are synchronized. Description of applied CA platform as well as realization of calculations on heterogeneous computing nodes is presented in this chapter.

3.1. GPGPU platform for CA calculations

The model was implemented on the CA platform using GPGPU technique. The platform enables implementation of algorithms using Application Programming Interface (API) supplied by the OpenCL library. It was done by dividing the platform into two parts: library and application (executable part). The former supplies API to serve core of the program in the OpenCL technology dedicated on the computing accelerators (multicore CPUs, Intel Knights Corner/Landing coprocessors or GPUs). The latter uses API supplied by OpenCL technology. Submitting and control of the input data is the main task of this part. This part reads configuration data with input parameters, sends them to the library and initiates computations. Creation of the CA space and introduction of the states of cells is an additional task of this part.

The platform was programmed in C++, what enables running the program in different operating systems (Linux, Windows or OS X). Platform does not have graphical interface what increases its heterogeneous applicability. However, application of external libraries (STL, Boost, OpenMP, Xerces) creates difficulties in compilation of the project. Multiplatform tool Cross-platform Make (CMake) was used to facilitate this process. CMake does not compile the program itself, but it creates files with compilation rules for a selected environment (e.g. Makefile for Linux or Microsoft Visual Studio files for Windows).

Due to C++ implementation the object-oriented design was applied and the part of the library was divided into the following classes: *Model*, *Space*, *CompCore*, *KernelManager* and *MemoryManager*. This approach allows material engineers to define the algorithms modeling material behavior in micro scale and relieves them of thinking about algorithm efficiency, parallelization or hardware architecture. This functionality is covered by *MemoryManager* class, which responses for organizing device resources and rewriting the memory space for data arrays. These arrays originate from one of the OpenCL constraints allowing the upstreaming of data to the device exclusively in the form of primitive types or arrays of primitive types.

The created models are represented by *Model* class, which aggregates all of the CA space information and boundary conditions. This class constitutes the application programming interface by integrating the following entities: *Cell*, *Space*, *Event* (*Rule*) and *Neighbourhoods*, which are common for most of the CA based algorithms. Model class contains crucial method *int run(const unsigned int iteration,* *size_t global,* *size_t local)*, which is responsible for execution of the implemented CA model. Set of input arguments passed to this procedure contains number of iterations as well as global and local size of thread arrays. The latter parameters are passed further to *clEnqueueNDRangeKernel* method implemented in OpenCL API. Additionally, two predefined classes related to real material features are implemented i.e. *Grain* and *GrainList*. These structures allow to avoid the redundancy of data on different levels of material description (similar solution is presented in other Authors' work (Spytkowski et al., 2009), where CA framework for multicore CPU processors was proposed). The assumed OpenCL standard forces kernels to be implemented in specific language based on C99. Therefore, all material models implemented previously in C++ and OpenCL pseudo code finally have to be converted into C99 kernels. The *DeviceRule* class, which extends the *Rule* class is responsible for that functionality, providing a bridge between a model and a final code for heterogeneous hardware architectures. *KernelManager* class combines all converted codes into a kernel, compile it and finally deploy it on target OpenCL device.

3.2. Many devices in one computing node

The most innovative High Performance Computing (HPC) infrastructures are composed of computing nodes, which include more than one multicore CPU and many additional devices typical for computational purposes like GPUs or Intel Knights Corner/Landing coprocessors. Thus, in this case we have to face multilevel distribution and parallelization of the problem. The possibilities of distribution between computing nodes was described in the previous chapter. The parallelization inside one computing node can be divided into two steps i.e. parallelization between computing devices and parallelization between many cores in one computing device. The Khronos group developing OpenCL library supports a lot of different devices, including Intel and IBM and selected graphical cards from AMD and Nvidia, which make it possible to implement fully parallelized computing engine by using OpenCL. The solution presented in this paper is designed and implemented to support such multilevel parallelization by using *clGetPlatformIDs* and *clGetPlatformInfo* methods. The former procedure gets specific identifiers of all available computing platforms inside one node. The latter procedure generates information about devices in each platform (figure 3).

```
Number of available platforms: 2
Platform names:
        [0] NVIDIA CUDA
        [1] Intel(R) OpenCL
Number of devices available for each type:
  CL_DEVICE_TYPE_CPU: 2
  CL_DEVICE_TYPE_GPU: 4
  CL_DEVICE_TYPE_ACCELERATOR: 0
```

Fig. 3. *Listing of platforms and devices generated by computing engine on multi-devices nodes.*

The list of available devices is related to the configuration of the CA space decomposition (figure 4), which allows to assign specific portion (*space_frag*) of computing domain to computing device. At the beginning of calculations the division of the CA space is made equally. After the first iteration of the model, divided subspaces are regrouped according to obtained execution times to balance the load on each device. Such simplified dynamic load balancing allows to minimize time losses before synchronization.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<config
    run="debug"
    log="console">
    <devices type="local">
        <device address = "local" type="gpu" platform="nvidia" space_frag="0.3"/>
        <device address = "local" type="gpu" platform="nvidia" space_frag="0.3"/>
        <device address = "local" type="cpu" platform="intel" space_frag="0.2"/>
        <device address = "local" type="cpu" platform="intel" space_frag="0.2"/>
    </devices>
</config>
```

Fig. 4. *The configuration of the CA space decomposition for different.*

4. RESULTS

4.1. Modelling

The model described in chapter 2.2 was validated for the semi-industrial tests of die forging after 1000 cycles. The microstructure of hot forging die was submitted to image recognition procedures to extract grain borders. At first high frequency noise was remove from the image and then die wear layer was separated from the rest of material. Afterwards border detection filter was applied followed by grain growth and labelling algorithm according to the approach presented by Rauch and Madej (2010). Then the removed die wear was reconstructed to determine whole microstructure before forging. The image of microstructure is presented in figure 5a.

inside the die microstructure, even though the model does not predict die wear.

4.2. Parallelization

Verification of CA platform efficiency in parallel environment was performed on one computing node equipped with 2x Intel Xeon X5650 CPU and 4x Nvidia Tesla M2090 GPU. At first benchmark CA models were selected and the computing jobs have been configure with specific runtime parameters. In OpenCL software model there is a possibility to setup the parameters of Local and Global Work Sizes (LWS and GWS), meaning the number of threads on one streaming processor and number of all threads running on GPGPU. These parameters are used in *run* method (chapter 3.1) since they influence final efficiency of the CA model. They are

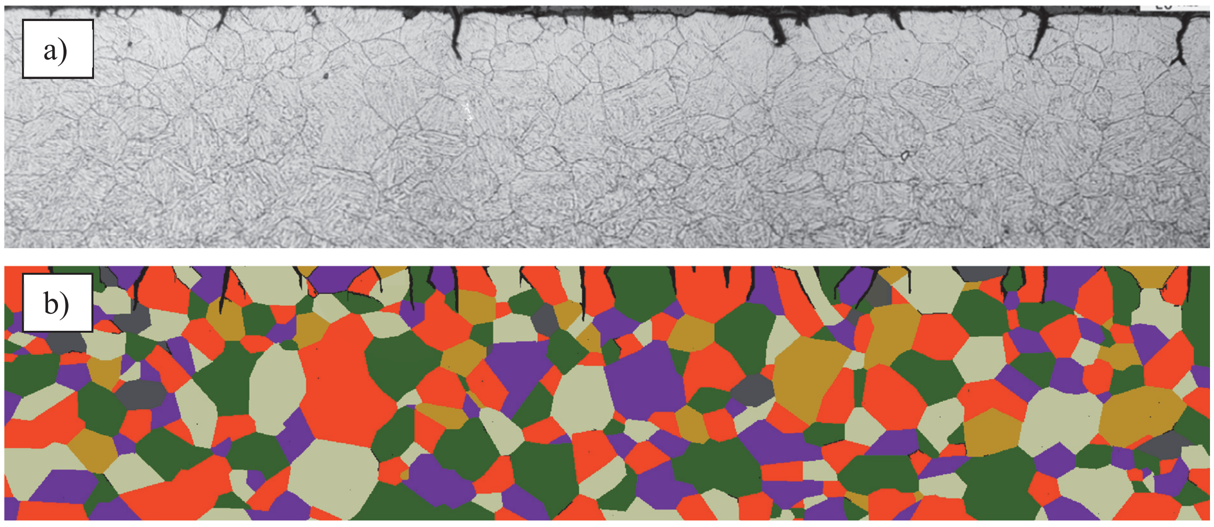

Fig. 5. *Validation of the model prediction (b) with fractures inside real microstructure (a).*

Prepared microstructure was digitalized by transformation of each pixel into one CA cell. Thus, the ready to use CA space was constructed of 2358x500 cells. The prediction of cracking after 1000 cycles obtained from implemented CA model is presented in figure 5b. The qualitative results proved satisfactory forecast of the cracks number and their depth

also used to estimate the speedup of the proposed solution according to the following equation:

$$S_p = \frac{t_{GWS=LWS}}{t_{GWS=p \cdot LWS}} \tag{7}$$

LWS is set on maximum size (512 for GPU used in this work) and GWS is increased geometrically. This allows to obtain speedup measurements, which are presented in figure 6 for GPU and CPU. It can be seen that the best results were obtained for LWS = 512 and GWS = 32768 for GPU. In case of calculations performed on CPU, maximum possible values of LWS and GWS gave the fastest performance of the model. The values determined for GPU and CPU were used further for multi-device analysis.

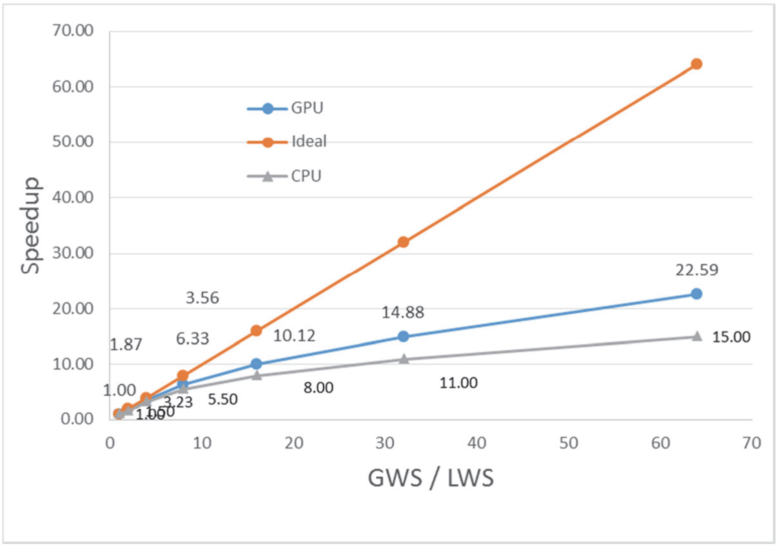

Fig. 6. Speedup obtained for CA model for one GPU (Nvidia Tesla M2090).

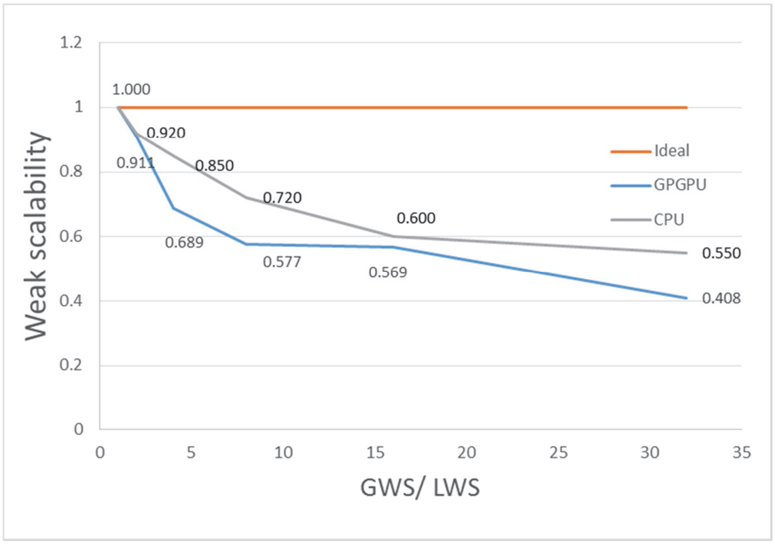

Fig. 7. Weak scalability for one CPU (Intel Xeon X5650) and one GPU (Nvidia Tesla M2090).

The analysis of weak scalability, which is crucial for dynamic load balancing, was performed for the same setting of LWS and GWS as speedup experiments, but with the size of a CA space growing proportionally to the ratio between GWS and LWS. The results of calculations presented in figure 7 showed that in the case of the created CA platform, applied GPU is data intensive. The size of initial CA space was 200x200. In the following experiments it was increased proportionally to the ratio between GWS and LWS. In comparison to results obtained for conventional GPU, CPU has better scalability, however GPU works relatively faster for larger CA spaces than in the case of small data spaces, where GPU memory is not properly fed with the data.

Verification of the CA fatigue cracking model for multi-device computing node was performed for all devices described in previous paragraph simultaneously i.e. 2 CPUs and 4 GPUs. Analyzed CA space (2358 x 500 cells) was divided into six vertical subspaces 393x500. The times obtained after the first iteration showed that GPU was about 6.4 times faster than CPU. Therefore the CA space was passed to the proposed load balancing procedure, which takes into account estimated data scalability, times of calculations on specific devices, communication between host and device as well as obtained speedup. The following new division of the CA space was proposed: 85x500 CA subspaces for each CPU and 547x500 for each GPU. This division allowed to equalize execution times in each iteration of the CA model between many devices, which led to a faster synchronization between CA subspaces. Finally obtained speedup was estimated separately for GPUs and CPUs. For 4 GPU devices the speedup was equal 3.59, while for 2 CPUs the speedup was 1.74. The average usage of GPUs during calculations was equal for all devices and it reached about 93%. The monitoring was performed by using *nvidia-smi* tool.

5. CONCLUSIONS

The paper presents CA model for simulation of fatigue micro cracking of hot forging dies. The model was developed on the basis of the CA approach with transition rules described by using probability functions related to critical stress estimated inside different cells. The parameters of the model were identified with the inverse analysis procedure executed for measurements obtained on Gleeble 3800. Then, the model was verified on real examples of microstructures.

The main achievement of this work is multilevel parallelization of the CA model, which was obtained for computing nodes with heterogeneous hardware architecture (2 CPUs and 4 GPUs). High level of parallelization was reached by using dedicated dynamic load balancing procedures proposed in this work, which allowed to equalize execution times influencing acceleration of synchronization procedure in each iteration. Low level parallelization was obtained for GPUs, where multicore architecture was efficiently utilized during simulations. Applied methodology allowed to create efficient CA based model predicting damages in hot forging tools. Now, the proposed approach can be applied for computationally intensive multiscale simulations, where micro scale models can be distributed separately among multi-device computing nodes in modern HPC infrastructure.

ACKNOWLEDGEMENTS

Financial support of the NCN, project no. 2011/01/D/ST6/02023, is acknowledged.

REFERENCES

Bayer, R.G., 2004, *Mechanical Wear Fundamentals and Testing*, Marcel Dekker Inc., New York.

Behrens, B.-A., Schäfer, F., 2009, Service life predictions for hot bulk forming tools, *Steel Research International*, 80, 887-891.

Behrens, B.-A., Bouguech, A., Hadifi, T., Klassen, A., 2012, Numerical and experimental investigations on the service life estimation for hot-forging dies, *Key Engineering Materials*, 504-506, 163-168.

Gurson, A.L., 1977, Continuum theory of ductile rapture by void nucleation and growth, Part 1. Yield criteria and flow rules for porous ductile media, *Trans ASME, Journal of Engineering Materials and Technology*, 99, 2-15.

Gronostajski, Z., Hawryluk, M., Krawczyk, J., Marciniak, M., 2013, Numerical modelling of the thermal fatigue of steel WCLV used for hot forging dies, *Eksploatacja i Niezawodność – Maintenance and Reliability*, 15, 129-133.

Gronostajski, Z., Kaszuba, M., Hawryluk, M., Zwierzchowski, M., 2014, A review of the degradation mechanisms of the hot forging tools, *Archives of Civil and Mechanical Engineering*, 14, 528-539.

Kim, Y.J., Choi, C.H., 2009, A study on life estimation of hot forging die, International *Journal of Precision Engineering and Manufacturing*, 10, 105-113.

Lavtar, L., Muhic, T., Kugler. G., Tercelj, M., 2011, Analysis of the main types of damage on a pair of industrial dies for hot forging car steering mechanisms, *Engineering Failure Analysis*, 18, 1143-1152.

Nowak, K., 2009, Micro- versus macro-modelling of creep damage, *Computer Methods in Materials Science*, 9, 249-255.

Perzyński, K., Sitko, M., Madej, L., 2014, Numerical modelling of fracture based on coupled cellular automata finite element approach, *Proc. 11th Int. Conf. on Cellular Automata for Research and Industry, ACRI 2014*, Krakow, 22-25.

Rauch L., Madej L., 2010, Application of the automatic image processing in modelling of the deformation mechanisms based on the digital representation of microstructure, *International Journal for Multiscale Computational Engineering*, 8(3), 1-14.

Rauch, L., Madej, L., Spytkowski, P., Golab, R., 2014, Development of the cellular automata framework dedicated for metallic materials microstructure evolution models, *Archives of Civil and Mechanical Engineering*, in print.

Shibutani, Y., 1999, Mesoscopic dynamics on dislocation patterning in fatigued material by Cellular Automata, Materials Science Research International, 5, 258-263.

Shterenlikht, A., 2003, 3D CAFE modeling of transitional ductile – brittle fracture in steels, PhD Thesis, The University of Sheffield.

Spytkowski, P., Klimek T., Rauch L., Madej L., 2009, Implementation of cellular automata framework dedicated to digital material representation, *Computer Methods in Materials Science*, 9 (2), 283-288.

Tercelj, M., Panjan, P., Urankar, I., Fajfar, P., Turk, R., 2006, A newly designed laboratory hot forging test for evaluation of coated tool wear resistance, *Surface Coating Technology*, 200, 3594-3604.

EFEKTYWNY MODEL AUTOMATÓW KOMÓRKOWYCH PRZEWIDUJĄCY ZNISZCZENIE ZMĘCZENIOWE NARZĘDZI DO KUCIA NA GORĄCO

Streszczenie

W pracy przedstawiono implementację modelu automatów komórkowych (ang. Cellular Automata - CA) przewidującego rozwój pękania zmęczeniowego w narzędziach do kucia na gorąco. Reguły przejścia opracowano na bazie dostępnych informacji dotyczących inicjacji i propagacji pęknięć. Współczynniki modelu wyznaczono na podstawie prób zmęczeniowych przeprowadzonych na symulatorze Gleeble 3800. Model CA połączono z programem MES, który symuluje naprężenia podczas kucia. To wieloskalowe podejście jest bardzo wymagające obliczeniowo. Dlatego efektywna równoległa implementacja tego modelu była głównym celem pracy.

Model pękania zmęczeniowego zrealizowano w technice GPGPU, co ułatwiło implementację algorytmów za pomocą udostępnienia API (z j. ang. Application Programming Interface) dostarczającego funkcje z biblioteki OpenCL. Platformę wspomagającą obliczenia CA podzielono na części biblioteczną i aplikacyjną. Biblioteka dostarcza API do obsługi jądra programu w technologii OpenCL dedykowanego na akceleratory obliczeniowe CPU i GPU. Aplikacyjna część wykorzystuje API dostarczone przez bibliotekę i jej zadaniem jest przekazanie i zarządzanie parametrami wejściowymi do obliczeń. Zadaniem aplikacji jest również stworzenie wejściowej przestrzeni automatów komórkowych, wczytanie i ustawienie wartości wewnętrznych komórek CA oraz przekazanie ich do obliczeń na akceleratorach. W pracy opisano implementację modelu oraz przedstawiono przykładowy wynik symulacji pęknięć w matrycy kuźniczej. Dzięki zrównolegleniu algorytmu i wykorzystaniu czterech kart GPU uzyskano wysokie przyspieszenie obliczeń przy dobrej skalowalności.

Received: September 13, 2014
Received in a revised form: Novemebr 09, 2014
Accepted: December 12, 2014