

Article

Accelerated Method for Simulating the Solidification Microstructure of Continuous Casting Billets on GPUs

Jingjing Wang ¹, Xiaoyu Liu ^{2,*}, Yuxin Li ¹ and Ruina Mao ¹

¹ School of Information Engineering, Shandong Youth University of Political Science, Jinan 250103, China; wangjingjing@sdyu.edu.cn (J.W.); 220026@sdyu.edu.cn (Y.L.); mrn@sdyu.edu.cn (R.M.)

² School of Information Science and Engineering, Northeastern University, Shenyang 110819, China

* Correspondence: liuxiaoyu@ise.neu.edu.cn

Abstract: Microstructure simulations of continuous casting billets are vital for understanding solidification mechanisms and optimizing process parameters. However, the commonly used CA (Cellular Automaton) model is limited by grid anisotropy, which affects the accuracy of dendrite morphology simulations. While the DCSA (Decentered Square Algorithm) reduces anisotropy, its high computational cost due to the use of fine grids and dynamic liquid/solid interface tracking hinders large-scale applications. To address this, we propose a high-performance CA-DCSA method on GPUs (Graphic Processing Units). The CA-DCSA algorithm is first refactored and implemented on a CPU–GPU heterogeneous architecture for efficient acceleration. Subsequently, key optimizations, including memory access management and warp divergence reduction, are proposed to enhance GPU utilization. Finally, simulated results are validated through industrial experiments, with relative errors of 2.5% (equiaxed crystal ratio) and 2.3% (average secondary dendrite arm spacing) in 65# steel, and 2.1% and 0.7% in 60# steel. The maximum temperature difference in 65# steel is 1.8 °C. Compared to the serial implementation, the GPU-accelerated method achieves a 1430× higher speed using two GPUs. This work has provided a powerful tool for detailed microstructure observation and process parameter optimization in continuous casting billets.

Keywords: GPU-accelerated method; parallel CA-DCSA algorithm; microstructure simulation; dendrite morphology; continuous casting; speedup



Academic Editor: Andrea Di Schino

Received: 23 February 2025

Revised: 23 April 2025

Accepted: 24 April 2025

Published: 25 April 2025

Citation: Wang, J.; Liu, X.; Li, Y.; Mao, R. Accelerated Method for Simulating the Solidification Microstructure of Continuous Casting Billets on GPUs. *Materials* **2025**, *18*, 1955. <https://doi.org/10.3390/ma18091955>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The mechanical properties of castings heavily depend on the microstructure of billets, especially the ECR (Equiaxed Crystal Ratio) and the secondary dendrite arm spacing. Analyzing the solidification microstructure of billets plays a crucial role in optimizing process parameters and enhancing the performance of castings. Although various methods, such as dendrite corrosion, electron probes, and synchrotron radiation in situ observations [1–3], are available for detecting microstructures, their application in large-scale production is restricted due to their high costs and environmental constraints.

In recent decades, several numerical simulation methods have emerged for predicting the solidification microstructures of castings, including CA [4–7], PF (Phase Field) [8–10], and MC (Monte Carlo) [11–13], etc. Of these methods, CA and PF are the most widely adopted. The PF model is primarily used for simulating the local dendrite morphology of a casting due to its requirement of a large grid size. The CA model is known for its simplicity, ease of implementation, and high calculation efficiency, making it a popular choice for simulating the microstructures of continuous casting billets. However, the solidification

structures [14–16] simulated using the CA method have a significant drawback, i.e., dendrites grow exclusively in the direction perpendicular to the solid wall, which contradicts the actual continuous casting process where dendrites grow in different orientations. This limitation stems from mesh anisotropy, which is caused by the fixed grid layout and capture rule. Various methods have been developed to address this issue, including the random zigzag capture rule [17], block cells [18], and DCSA [19,20]. However, the first two methods are limited by the fact that they can only simulate dendrites with the same orientation simultaneously. The CA model coupled with DCSA is effective for simulating dendrites with different orientations simultaneously by dynamically tracking the dendrite tip, and has been successfully used to simulate the growth of columnar and equiaxed dendrites [21–23]. However, existing studies have focused on simulating the growth of single or multiple dendrites, with no research yet on simulating the microstructure of continuous casting billets. The scarcity of research in this area stems from DCSA's requirement of a precise grid size and the computational complexity introduced by dynamic changes in capture positions and conditions. Consequently, simulating the microstructure of continuous casting billets with this approach demands immense computational power, rendering it a time-consuming endeavor or altogether unfeasible on a CPU. Therefore, improving the calculation of efficiency is the key to establishing a CA model with a low grid anisotropy capable of simulating the microstructure of continuous casting billets.

In related research fields, several methods have been proposed to improve the calculation efficiency, including parallel computing with multi-core CPUs [24–27] and adaptive mesh methods [28,29]. However, these methods are based on CPUs, the speed of which is not ideal because the number of cores is limited by the hardware architecture and they incur significant overheads when launching multi-threaded processes [30]. Recently, GPUs have become increasingly popular for accelerating large-scale calculations, owing to their multiple-core architecture with a particularly high number of lightweight cores [31–35]. Moreover, the CA model is particularly suitable for parallel computations using a GPU because of its discrete time and space, and the homogeneous and asynchronous application of transition rules among cells [36]. The use of heterogeneous GPU–CA models has enhanced the performance of computational models in various fields [37–40].

To effectively address the challenges posed by grid anisotropy and computational efficiency in simulating the microstructure of continuous casting billets, this paper proposes a GPU-accelerated CA-DCSA method based on our previous GPU acceleration scheme [16]. In addition to porting the serial program to the heterogeneous architecture with a refactored parallel algorithm, we further optimized the memory access efficiency by changing the memory access mode and minimizing warp divergence. Compared to previous models, the present work can simulate more detailed microstructures with clear dendrite morphologies. Moreover, the simulation results agree well with industrial experimental data in both structure (ECR and the average secondary arm spacing) and temperature distribution. In addition, this accelerated model achieves a considerable increase in speed of $1430\times$ compared to the serial CPU implementation, making it an invaluable tool for optimizing process parameters.

2. Model Description for Solidification in Continuous Casting

Figure 1 illustrates the continuous casting process, wherein the molten steel flows from the tundish to the water-cooled copper mold to form the initial shell.

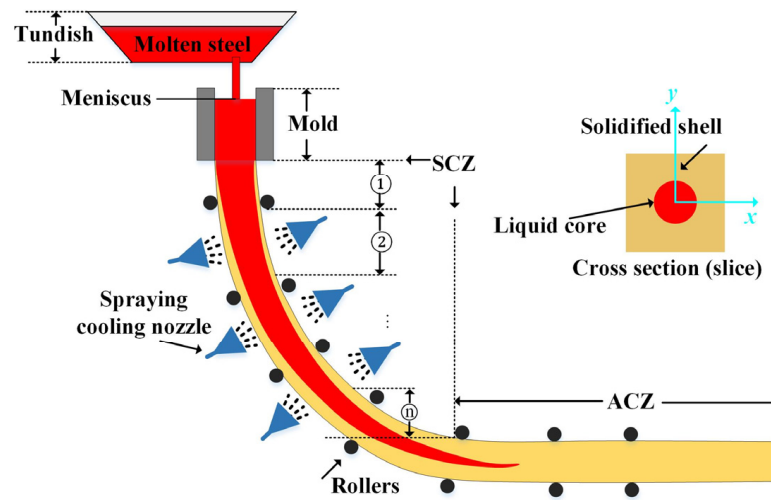


Figure 1. Continuous casting process.

Then, the solidifying shell moves down to the SCZ (Secondary Cooling Zone), where it is cooled by sprayed water or an air mist. Finally, the solid strand enters the ACZ (Air-Cooling Zone) and completes the solidification process gradually. This process involves complex multi-physics coupling, including heat transfer, mass transfer, and phase change, which results in the final solidification microstructure of the continuous casting billet. This paper utilizes the “two-dimensional slice tracking” method to trace the evolution of the microstructure from the meniscus to the end of the withdrawal straightener, where the FVM (Finite Volume Method) is utilized to solve the heat and mass transfer equations, and the CA method is used to describe the phase transition process.

2.1. Heat and Mass Transfer Model

The heat transfer process of continuous casting billets can be described by the heat conduction equation [18,41,42]

$$\rho c_p \frac{\partial T}{\partial t} = \frac{\partial}{\partial x} \left(\lambda \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left(\lambda \frac{\partial T}{\partial y} \right) + q \quad (1)$$

where T is temperature, °C; t is time, s; ρ is mass density, kg/m³; c_p is equivalent specific heat; and λ is thermal conductivity. The Fe-C pseudo-binary phase diagram [43] is adopted to calculate the thermophysical properties. q is the gradient heat flux determined by different boundary conditions which are set as follows:

In the mold-cooling zone, the gradient heat flux boundary condition is given by the modified Savage–Pritchard equation [41,44].

$$-\lambda \frac{\partial T}{\partial \mathbf{n}} = A - B\sqrt{t} \quad (2)$$

where A and B are coefficients that can be adjusted according to the industrial data related to heat transfer conditions (heat flux, surface temperature, cooling water flow rate, and other relevant parameters that influence the heat transfer efficiency), $\mathbf{n}$ represents the normal direction, which is perpendicular to the boundary surface.

In the SCZ, heat is taken away from the billet surface by cooling water and thermal radiation, where the boundary condition can be determined by [34,41]

$$-\lambda \frac{\partial T}{\partial \mathbf{n}} = h_i(T - T_w) + \varepsilon\sigma(T^4 - T_a^4) \quad (3)$$

where h_i is the comprehensive heat transfer coefficient, the subscript i denotes the i th section in SCZ, ε is the emissivity of strand surface, with a value of 0.8; σ is the Stefan-Boltzmann constant, equal to 5.67×10^{-8} (W/m²)K⁴; T_w and T_a are cooling water temperature and ambient temperature, respectively, both equal to 25 °C. There are two kinds of cooling in the SCZ, i.e., water sprays and air-mist sprays, for which the coefficient h_i is defined by Equation (4), describing two conditions [34,41].

$$\begin{cases} h_{i(water)} = \frac{1570w_i^{0.55}[1 - 0.0075(T_w - 273)]}{\alpha_i} \\ h_{i(air-mist)} = \frac{1000w_i}{\alpha_i} \end{cases} \quad (4)$$

where w_i is the water flow density, and α_i is the machine-dependent calibration parameters that can be adjusted to minimize the difference between the simulated and industrial data. The specific application of air-mist cooling and water cooling needs to be determined according to the design of the continuous casting machine and the process' requirements.

In the ACZ, heat is extracted through thermal radiation [34,41,42]

$$-\lambda \frac{\partial T}{\partial n} = \varepsilon \sigma (T^4 - T_a^4) \quad (5)$$

The solute diffusion is governed by [14,23]

$$\frac{\partial C_i}{\partial t} = \frac{\partial}{\partial x} \left(D_i \frac{\partial C_i}{\partial x} \right) + \frac{\partial}{\partial y} \left(D_i \frac{\partial C_i}{\partial y} \right) + C_i(1 - k) \frac{\partial f_s}{\partial t} \quad (6)$$

where D_i means the solute diffusion coefficient calculated by Equation (7), and C_i is the concentration, the subscript i represents two states, i.e., l is liquid and s is solid. The gradient solute flux boundary condition is 0.

$$D_i = \begin{cases} 0 & , f_s = 1 \\ f_s D_s + (1 - f_s) D_l & , f_s < 1 \end{cases} \quad (7)$$

2.2. Phase Transition Model

In the CA method, the calculation domain is uniformly divided into square cells. The nucleation density of a cell with respect to the melt undercooling ΔT is given by Gaussian function [14,23]:

$$\frac{dn}{d(\Delta T)} = \frac{n_{\max}}{\sqrt{2\pi}} \exp \left[-\frac{1}{2} \left(\frac{\Delta T - \Delta T_N}{\Delta T_\sigma} \right)^2 \right] \quad (8)$$

where ΔT_N and ΔT_σ are the average and standard deviation nucleation undercooling, and $n_{\max}$ is the maximum nucleation density. The nucleation probability of a liquid cell is calculated by:

$$P_n = \int_{\Delta T^n}^{\Delta T^{n+1}} \frac{dn}{d(\Delta T')} d(\Delta T') \Delta x \quad (9)$$

where Δx is cell size.

To reduce the mesh anisotropy, the current CA model utilizes the DCSA rule to capture neighbors. If the P_n of the cell labeled as "A" exceeds a randomly generated number (0~1), a nucleation event in cell "A" will be triggered. A square with a random orientation $\theta[-45^\circ, 45^\circ]$ is subsequently placed in the center of cell "A", and its state will be updated

from liquid to interface. Then, the square will grow along the orientation θ with the velocity given by the KGT model [15]:

$$v = \alpha_1 \Delta T^2 + \alpha_2 \Delta T^3 \quad (10)$$

where α_1 and α_2 are growth coefficients [45]. The melt undercooling ΔT consists of thermal, solute, and curvature undercooling, which is calculated by [18,22,23]:

$$\Delta T = T_l - T + m(C_l - C) - \Gamma K f(\varphi, \theta) \quad (11)$$

$$K = \frac{\left(2\partial_x f_s \partial_y f_s \partial_{xy} f_s - \partial_x f_s^2 \partial_{yy} f_s - \partial_y f_s^2 \partial_{xx} f_s\right)}{\left(\partial_x f_s^2 + \partial_y f_s^2\right)^{3/2}} \quad (12)$$

$$f(\varphi, \theta) = 1 - 15\epsilon \cos(4(\theta - \varphi)) \quad (13)$$

$$\varphi = \begin{cases} \cos^{-1}\left(\partial_x f_s / \left(\partial_x f_s^2 + \partial_y f_s^2\right)^{1/2}\right) & , \partial_x f \geq 0 \\ 2\pi - \cos^{-1}\left(-\partial_x f_s / \left(\partial_x f_s^2 + \partial_y f_s^2\right)^{1/2}\right) & , \partial_x f < 0 \end{cases} \quad (14)$$

where C_l and C are actual and initial concentration, wt.%; T and T_l are the melt and liquidus temperatures, and m is the liquidus slope; Γ is the Gibbs–Thomson coefficient; K is the interface curvature; $f(\varphi, \theta)$ is the interface anisotropy function; φ is the interface normal angle; and θ is the preferential growth orientation.

As cell “A” grows, its corners penetrate the neighboring cells and convert them into interface cells, with new squares generated at the capturing points. The square, located in cell “A” will keep expanding until its solid fraction f_s reaches 1 and the state of cell “A” will change to solid. The nucleation, growth, and capture processes will continue until solidification is complete.

2.3. Analysis of Computational Intense

To accurately model the capture process, the DCSA requires a fine grid division. For the billet properties that possess four-fold symmetry, this paper performs dynamic simulation on a quarter section (80 mm × 80 mm) in the top right corner of the billet. In the case of a casting speed of 1.75 m/min, a grid-independent solution can be obtained when the slice is divided into 8000 × 8000 cells with the size of 10 μm. And, the time step dt is set to 0.0005 s to keep the calculation accuracy. As the casting process is 16 m long, there are approximately 1.1×10^6 slices equating to 7.04×10^{13} cells. Furthermore, the dynamic monitoring of the four corners of each square within the interface cells during the intricate three-physics coupling reaction necessitates a substantial increase in computational speed. In a test of 1000 slices on Intel® Xeon® CPU E5-2680 v4 @2.4 GHz, the simulation process took 10.06 h. In this scenario, the whole simulation process would take about 485 days, which is unacceptable and infeasible. Therefore, improving the calculation efficiency is crucial to simulate the microstructure of continuous casting billets with a multi-orientation dendrite morphology.

3. GPU-Accelerated Method for CA-DCSA Model

Here, the GPU-accelerated method consists of two main components: parallelization and optimization. We begin with parallelizing the serial program using the reengineered CA-DCSA algorithm and porting it to a heterogeneous parallel architecture. Then, a series of optimization techniques is devised to enhance memory utilization and prevent divergence, thereby facilitating the effective utilization of GPU resources and enabling high-performance computations.

3.1. Parallelization with Redesigned Algorithm

3.1.1. Porting to the Heterogenous Architecture

The heterogeneous model is implemented on Compute Unified Device Architecture (CUDA), a general-purpose parallel programming model consisting of a CPU code and a GPU code. First, taking full advantage of the computational power of the GPU provided by thousands of processing cores, our implementation strategically assigns the computation-intensive tasks to the GPUs, while allocating memory, initialization, program flow monitoring, and post-processing to the CPU, as shown in Figure 2. The GPU functions as a coprocessor and processes tasks through a parallel thread execution of a set of instructions referred to as a kernel. CUDA has a two-level thread hierarchy decomposed into thread blocks and grids of blocks. Once a kernel is launched, all independent threads in a grid will execute the instructions concurrently. Since the communication between CPU and GPU is very expensive, we designed only two communications, i.e., before and after the computation-intensive task. Moreover, a GPU-friendly SoA structure with all variables stored in a linearized form was utilized to organize the data as it provided the best-coalesced access pattern. In addition, read-only data that pertain to material properties, such as constant parameters and boundary conditions, were cached in constant memory. This type of memory is accessible from all threads within a grid, which can significantly reduce the number of memory transactions and improve overall performance. By caching these data in constant memory, they can be quickly accessed by multiple threads without repeated memory transfers from the global memory, which can lead to a significant reduction in memory latency. After the aforementioned parallelization operation, the program can be ported to the heterogeneous architecture.

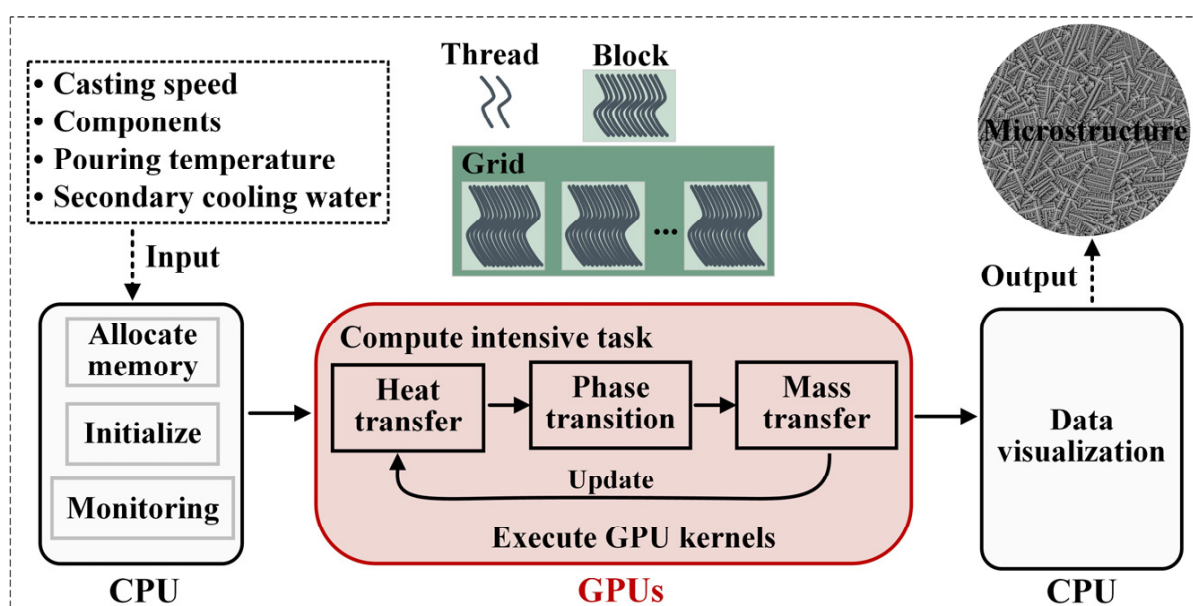


Figure 2. Heterogenous programming framework.

The functions running on a GPU consist of three main parts, as shown in Figure 2, and each part is implemented by several kernel functions; the solution is shown in the following pseudocode:

Compute intensive task

1. Heat transfer (Input: f_s)

getb: boundary conditions

ThermaPro: thermal properties

getT: temperature for Finite Volume

2. Phase transition (Input: T, C_l)

getmT: temperature for cell

calCur: curvature

caldT: undercooling

nucleation: nucleation probability

soliden: capture process

growth: growth velocity

3. Mass transfer (Input: T, f_s)

soluteRedi: solute redistribution in the S/L interface

soluteAdd: load solute discharged by neighbors

soluteDif: solute diffusion in the whole domain

4. Update

This process will be repeated until the solidification is completed.

3.1.2. Redesigning the Capture Process

The simplest way to achieve a high computational efficiency is assigning each thread to a computational unit, built on the foundation of high parallelism. To achieve this, the explicit formulation is utilized for the heat and mass transfer equations. In addition, the data competition that exists in the DCSA capture process when there are multiple neighboring cells that satisfy the capture condition can lead to undefined errors if it cannot be handled reasonably, rendering acceleration ineffective. As shown in Figure 3, at a certain time, cells (3, 4) and (2, 3) have penetrated into (2, 4), and cells (2, 1) and (1, 2) have touched (2, 2). In this case, the threads mapping to cells (3, 4) and (2, 3) will write data onto the memory space of cell (2, 4), and the threads mapping to cell (1, 2) and (2, 1) will write data onto the memory space of cell (2, 2) simultaneously. The *AtomicExch* function provided by CUDA can effectively prevent race conditions among multiple threads, but it comes with the cost of sacrificing computational parallelism and efficiency.

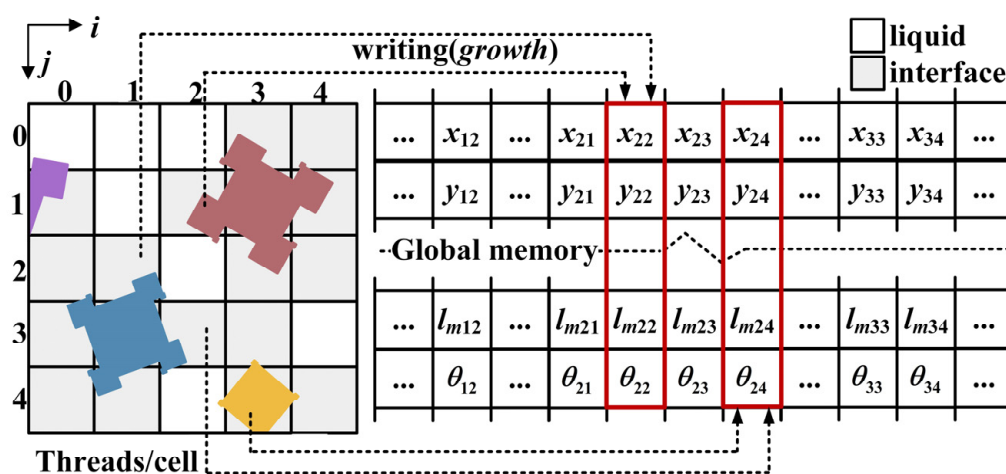


Figure 3. Data competition in DCSA ((x_{ij}, y_{ij}) : the nucleation position, l_{mij} : the maximum half diagonal length, θ_{ij} : preferential orientation, different color represents different orientation).

To eliminate data competition without sacrificing computational parallelism, we redesigned the capture process. As shown in Figure 4, the new parallel CA-DCSA algorithm focuses on preventing threads from modifying data that belong to other threads by dividing the capture process into three primary steps:

- (1) *Growth* kernel: Interface cells grow and update their growth-related parameters in global memory. No neighbor capture occurs at this stage, even if the capture condition is met.
- (2) *Soliden* kernel: Liquid cells read the growth-related parameters from neighboring interface cells, and judge which neighbor satisfies the capture condition based on them. An arbitration mechanism resolves conflicts when multiple neighbors satisfy the capture condition for the same liquid cell. Also, it is crucial to determine which corner touches the current cell because different corners have different capture positions, as shown in Figure 3 where both corners of the square in cell (0, 1) can touch cell (0, 0).
- (3) *Soliden* kernel: Liquid cells update their growth-related parameters based on the arbitration results from the *soliden* kernel.

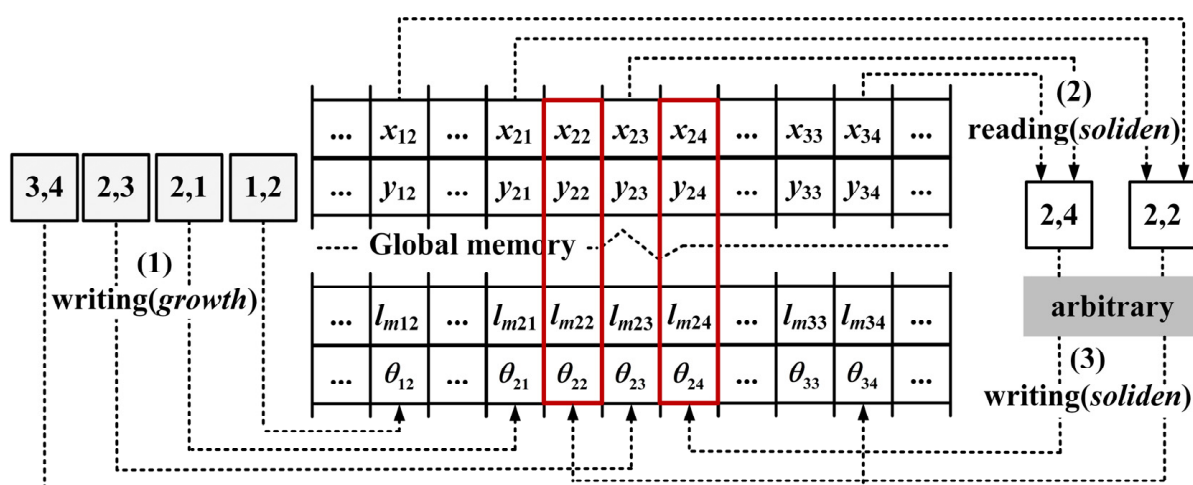


Figure 4. Scheme for the new parallel CA-DCSA algorithm.

In this new algorithm, threads only modify their own data, which can fundamentally avoid data competition without affecting computational parallelism.

3.2. Optimizations

Preliminary analysis shows that the present model suffers from a low memory access efficiency caused by a large number of memory transactions and warp divergence. Thus, to address these two issues, optimizations are performed to maximize GPU utilization and achieve effective acceleration.

3.2.1. Managing Memory Accessing

Memory write operations typically incur a higher latency than read operations. This occurs because write operations require data transfer from processors/registers to memory, while read operations can directly fetch data from memory. In our implementation, a lot of variables need to be updated in the kernel “Update” after each time step, rendering it the most time-consuming part. Most of the updated variables are assigned by the solute redistribution process, which sets nine intermediate variables dC_i ($i = 0, 1, \dots, 8$) for each cell to store the amounts of excess solute being distributed to neighboring cells and the current cell, as shown in the naive algorithm of Algorithm 1 where dC is the total rejected solute by the current cell. After being used in the kernel “SoluteAdd”, these intermediate

variables must be reset to zero to use the next slice, leading to a large number of memory writing transactions. To avoid writing memory frequently, the operation in *soluteAdd* is redesigned, which is noted as the redesigned version in Algorithm 1.

Algorithm 1. Two methods for solute redistribution at S/L interface.

naive version

soluteAdd

$$1. C_l+ = \sum_{i=0}^8 dC_{idxi} / (1 - f_{s idx});$$

Update

$$2. dC^i = 0;$$

$$3. dC = 0;$$

red esigned version

soluteAdd

$$1. C_l+ = \sum_{i=0}^8 dC_{idxi} (dC_{idxi} > 0) / (1 - f_{s idx});$$

Update

$$2. dC = 0;$$

In this new algorithm, we implement a logical expression $dC_{idxi} > 0$: when the expression is evaluated to be true, it returns 1 (preserving the current time step's value); otherwise, it returns 0 (forcing the value to zero). Therefore, the memory writing of $dC_i (i = 0, 1, \dots, 8)$ is replaced by memory reading of dC . Furthermore, we optimize data reuse through instruction reordering, significantly improving memory access efficiency.

3.2.2. Avoiding Warp Divergency

GPUs organize threads into warps, where each warp comprises 32 threads executing in lockstep via an SIMT (Single Instruction Multiple Thread) architecture. However, warp divergence occurs when threads within a warp take different execution paths during conditional operations, severely degrading performance [46]. In this work, conditional operations occur when solving Equations (7) and (14) using an if statement in naive version of Algorithm 2. Thus, in order to remove if conditions, we combine conditions into computational statements in the redesigned version of Algorithm 2.

Algorithm 2. Two methods for solute diffusion coefficient and interface normal angle.

naive version

coefficient of solute diffusion

1. **if** ($f_s == 1$) **then**

2. $D = 0$;

3. **else**

4. $D = f_s D_s + (1 - f_s) D_l$;

5. **endif**

interface normal angle

6. **if** ($\partial_x f_s > 0$) **then**

7. $\text{angle} = \text{acos}(\partial_x f_s / \text{sqrt}(\partial_x f_s^2 + \partial_y f_s^2))$

8. **elseif** ($\partial_x f_s \parallel \partial_y f_s$)

9. $\text{angle} = 2\pi - \text{acos}(-\partial_x f_s / \text{sqrt}(\partial_x f_s^2 + \partial_y f_s^2))$

10. **endif**

Algorithm 2. *Cont.***red esigned version**

coefficient of solute diffusion

$$D = (f_s == 1) \cdot 0 + (f_s < 1) \cdot (f_s D_s + (1 - f_s) D_l);$$

interface normal angle

$$\text{angle} = (\partial_x f_s > 0) \cdot \arccos(\partial_x f_s / \sqrt{\partial_x f_s^2 + \partial_y f_s^2}) + (\partial_x f_s \geq 0 \text{ \&\& } (\partial_x f_s \parallel \partial_y f_s)) \cdot (2\pi - \arccos(-\partial_x f_s / \sqrt{\partial_x f_s^2 + \partial_y f_s^2}))$$

4. Results and Discussion

The GPU-accelerated CA-DCSA method was applied to simulate the microstructure of continuous casting billets from a certain steel plant. The simulation accuracy was validated against the industrial measurement data, including the ECR, the average secondary arm spacing, and the temperature. The relevant caster parameters and billet properties are listed in Table 1, while the chemical compositions of the reference steels are detailed in Table 2. In our case, the SCZ of the continuous caster is divided into four sections. The first two sections use water cooling, while the last two sections employ air-mist cooling.

Table 1. Parameters of caster and thermophysical properties.

Parameters	Values
Billet size (-)	160 mm × 160 mm
Steel grade (-)	65#, 60#
Casting Speed (m/min)	1.75
Effective mold length (m)	0.9
Lengths of SCZ sections (m)	0.37, 1.85, 2.20, 2.32
Liquidus temperatures (°C)	1476 (65#), 1481 (60#)
Solidus temperatures (°C)	1382 (65#), 1383 (60#)

Table 2. Compositions of casting steels (wt.%).

Billets	Components				
	C	Si	Mn	P	S
65#	0.65	0.24	0.59	0.22	0.08
60#	0.60	0.24	0.59	0.23	0.04

4.1. Validation with Industrial Experiment

To validate model accuracy, we simulated billets 65# and 60# using a time step of 5×10^{-4} s and a grid resolution of 10^{-5} m. Given the critical relationship between microstructure and thermal history, temperature validation was prioritized. In order to validate the calculated temperature, a series of temperatures for steel 65# (superheat: 32 °C), were measured using an M9200 infrared imager. The temperature variation in the billet during the continuous casting process is shown in Figure 5, where the measurements are taken at the center of the arc surface. The predicted temperatures at the center of the inner arc surface show a significant reheating phenomenon in the transition cooling zone, which is caused by the differences in the boundary heat flux density due to different boundary conditions. The measured temperatures at 5.3, 7.6, 13.5, and 15.9 m away from the meniscus are 985 °C, 935 °C, 973 °C, and 941 °C, respectively, and the corresponding simulation results are 986.8 °C, 933.9 °C, 974.1 °C, and 940.2 °C. Thus, the calculated temperatures agree well with the measured ones with a maximum deviation of 1.8 °C, which is negligible compared to the temperature of the billet.

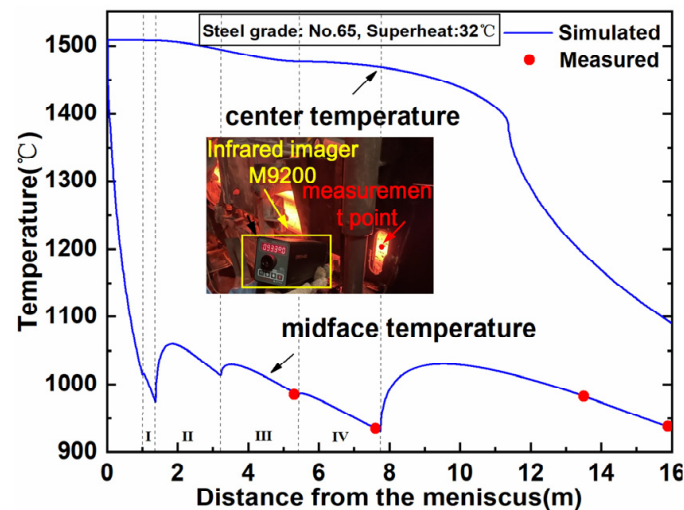


Figure 5. Temperature distribution during continuous casting (I: SCZ 1, II: SCZ 2, III: SCZ 3, IV: SCZ 4).

Moreover, the microstructures of continuous casting billets 65# and 60# were simulated and experimentally validated. To evaluate the method's effectiveness, industrial billet specimens were etched using a picric acid–hydrochloric acid solution, producing low-magnification micrographs (Figure 6b,e). Also, the simulated structures of continuous casting billets from our previous work are displayed as Figure 6c,f, where all dendrites grow perpendicular to the solid wall and we hardly observed a dendrite morphology. Here, the present model can simulate a clear dendrite morphology with not only multi-orientation dendrites but also a secondary dendrite arm, as shown in Figure 6a,d, showing excellent agreement with experimental observations.

To quantitatively analyze the accuracy of this method, we first calculated the ECR at the macroscopic level and the average secondary arm spacing within a selected region at the microscopic level. The structure of the billets has a typical columnar crystal zone and an equiaxed crystal zone, which are distinguished by the average ratio α of the short axis to the long axis [47]. The ECR are calculated using

$$\frac{\text{area of the equiaxed crystal zone}}{\text{total area}} \times 100\%$$

The ECRs of the microstructure are listed in Table 3, where ECRS is the simulated result, ECRE represents the experimental result, and RE is the relative error between them. It can be concluded that the simulated and experimental ECRs match well with the relative errors of 2.5% and 2.1%.

Table 3. The ECR of the microstructure.

65#			60#		
ECRS	ECRE	RE	ECRS	ECRE	RE
45.4%	44.3%	2.5%	48.3%	47.3%	2.1%

Moreover, to enhance the observation of microscopic dendrites, circular regions (radius = 6.46 mm) were selected and enlarged in Figure 6. Regions A1 and A2 are positioned 41mm from the upper boundary and 38.5 mm from the left boundary, B1 and B2 are located at a distance of 30.5 mm from the upper boundary and 30 mm from the left boundary, as depicted in Figure 6. It is clear that the dendrite morphologies in the simulation results are not exactly the same as the dendrites in the industry experiment, and these differences arise from natural randomness in both nucleation processes. Therefore,

our analysis adopts a statistical comparison approach to compare the simulation result to the experiment result.

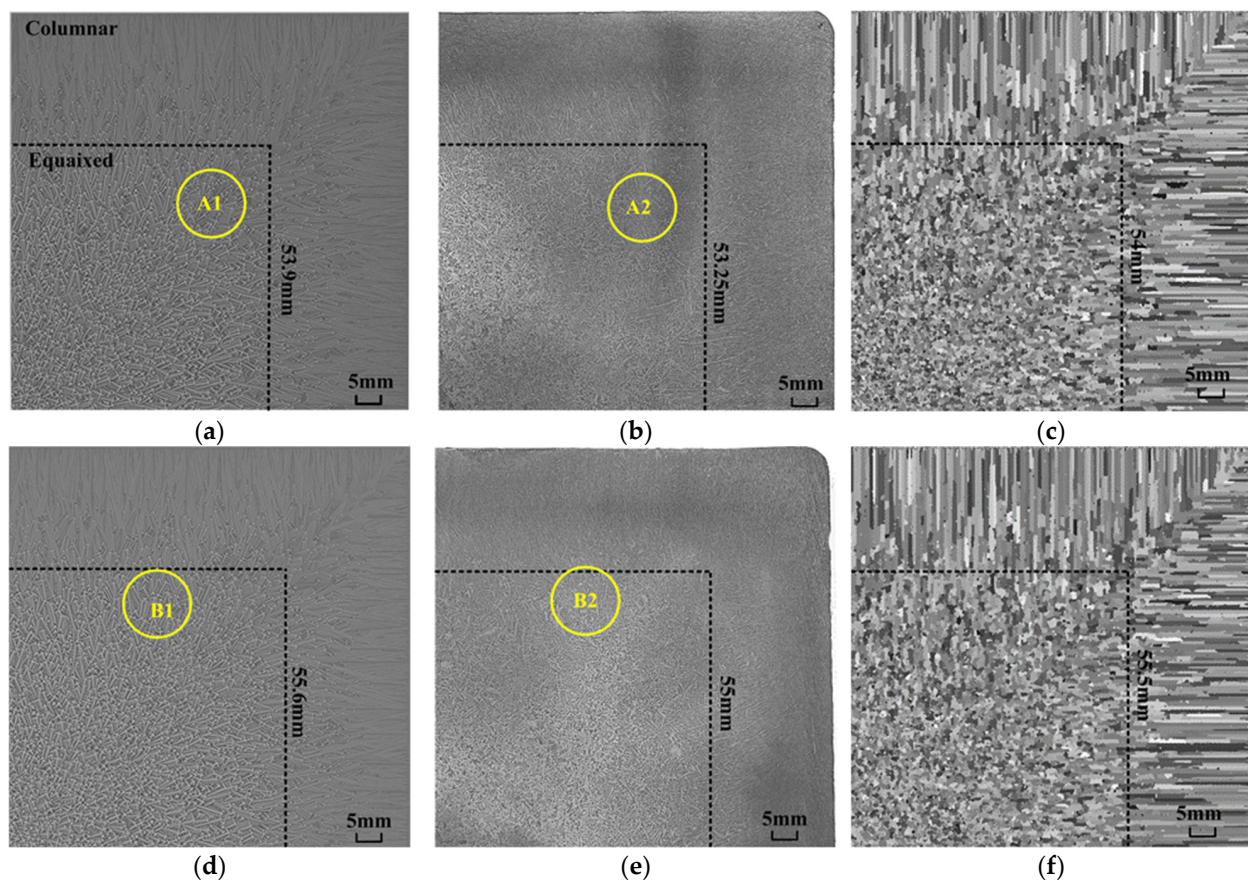


Figure 6. Microstructures of continuous casting billets ((a) simulated results for billet 65#, (b) industrial experiment for 65#, (c) simulated results of billet 65# using previous method [16], (d) simulated results for billet 60#, (e) industrial experiment for 60#, and (f) simulated results for billet 60# billet using previous method [16]).

The experimental dendrite morphology exhibits a lower clarity compared to the simulations, and there are two main reasons for this. Two primary factors contribute to this: (1) experimental artifacts including localized corrosion defects; (2) image acquisition artifacts introduced during optical microscopy. To improve the clarity of the results, negative film processing and other adjustments were applied to the experimental images.

Comparisons will be conducted from the dendrite distribution and the average secondary arm spacing. Firstly, in the actual casting process, as shown in Figure 7b,d, dendrites are observed to be randomly distributed, lacking any discernible trend or directionality. This randomness can be attributed to the significant variability in nucleation positions and growth orientations. The simulated results, i.e., Figure 7a,c, exhibit a similar pattern of dendrite distribution, which matches well with the experiment result. Secondly, the secondary dendrite arm spacings are measured. For the 65# casting billet, the secondary arm spacing of the simulation result ranges from 273.9 μm to 314 μm with an average value of 291.6 μm , while the experiment result ranges from 280.2 μm to 321.4 μm with the average of 298.6 μm . For the 60# casting billet, the secondary arm spacing of simulation result ranges from 217.1 μm to 228 μm with the average value of 223.4 μm , while the experiment result ranges from 216.2 μm to 225.5 μm with an average of 221.8 μm . The relative errors in the average secondary arm spacing between the simulated and experimental values were 2.3% (65#) and 0.7% (60#). These results demonstrate that the proposed model can reproduce the

microstructure of continuous casting billets with detailed dendrite morphologies, which agrees well with the experimental result.

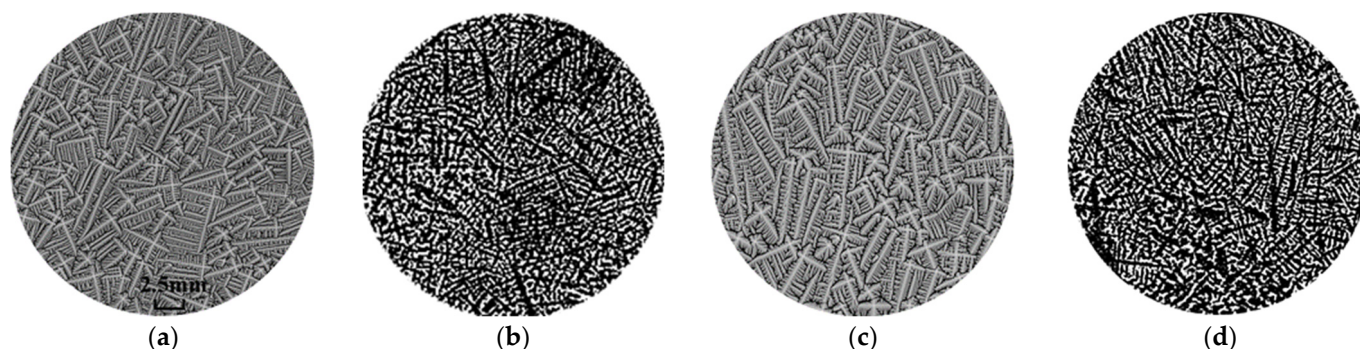


Figure 7. Zoomed-in image of Figure 6 ((a) A1, (b) A2, (c) B1, (d) B2).

4.2. Performance Analysis

In this section, a series of computation experiments are performed to evaluate the effectiveness of the proposed method. It is worth noting that the block size configured when launching the kernel function has significant impact on the computational performance, since the number of threads in a block generally influences the usage of the on-chip resources. So, before comparing the runtime for any case in this section, we manually adjusted the block size to obtain the optimal configuration for each kernel.

To ensure the computational reliability of the GPU algorithms, we first carry out a comparison of the temperatures calculated between the serial CPU algorithm and the GPU algorithms. Here, we have demonstrated that both GPU algorithms have the same calculation accuracy. Figure 7 depicts the temperature deviations at the center and midface of the billet throughout the continuous casting process, where the maximum difference is $3 \times 10^{-4} \text{ }^{\circ}\text{C}$, which can be considered negligible in comparison to the billet's temperature. So, the temperatures calculated using the serial CPU algorithms and GPU algorithms can be considered to be equivalent.

Then, simulations with 1000 time steps on a different number of grids were executed with three different methods, i.e., the serial CPU algorithm, basic GPU algorithm, and optimized GPU algorithm. Note that the serial CPU algorithm is run on the Intel® Xeon® E5-2680v4 and GPU algorithms are implemented on the Tesla P100. Here, we chose the serial CPU algorithm as the baseline, and define the speedup as the ratio of the runtime on the baseline system to the runtime on the GPU system. In this section, the speedup is utilized to evaluate the computational performance of the GPU algorithms. Figure 8 shows the computational performance, including runtime and speedup. Here, CPU-serial, GPU, and GPU* represent the runtime of the serial CPU serial algorithm, basic GPU algorithm, and optimized GPU algorithm, respectively. Meanwhile, S and S* are the speedup of the basic GPU algorithm and optimized GPU algorithm.

As demonstrated in Figure 9, the runtime increases with the increase in the number of grids in all three algorithms. When the number of grids increases from 5^2 ten million to 80^2 ten million, the runtime of the serial CPU algorithm increases steeply from 214 s to 36,229 s, while the runtime of the basic GPU algorithm increases from 0.78 s to 42.74 s, and that of the optimized GPU algorithm increases from 0.55 s to 25.32 s. Moreover, the speedup of both GPU algorithms increases with the increase in the computation domain, and the speedup increases rapidly initially, but the growth rate gradually slows down after turning points “A” and “B”. This is because, as the workload increases, the GPU can more fully utilize its parallel computing capabilities, resulting in a higher speedup. However, as the workload further increases, certain limiting factors, such as memory bandwidth and

computational power limitations, may come into play, gradually restricting the growth rate of the speedup. Moreover, simply porting the serial algorithm to a heterogeneous parallel architecture always makes it impossible to fully utilize the GPU resources due to the different memory architectures between CPUs and GPUs. In the optimized GPU algorithms, we improved the memory accessing efficiency by changing the memory accessing way and avoiding warp divergency (Section 3.2), which achieved a performance improvement of about 70% with an increase in speedup from $848\times$ to $1430\times$. It is fascinating to note that turning point “A” occurs later than “B”. This is because, by increasing memory utilization, the GPU can more effectively utilize its memory bandwidth, reducing the limitations imposed by memory bottlenecks on the growth of speedup. This means that the GPU can handle more data and perform more parallel computations, thereby delaying the decline in the rate of performance improvement. Therefore, by optimizing memory utilization, the occurrence of the turning point can be postponed, allowing the GPU to achieve a better speedup at higher computational workloads.

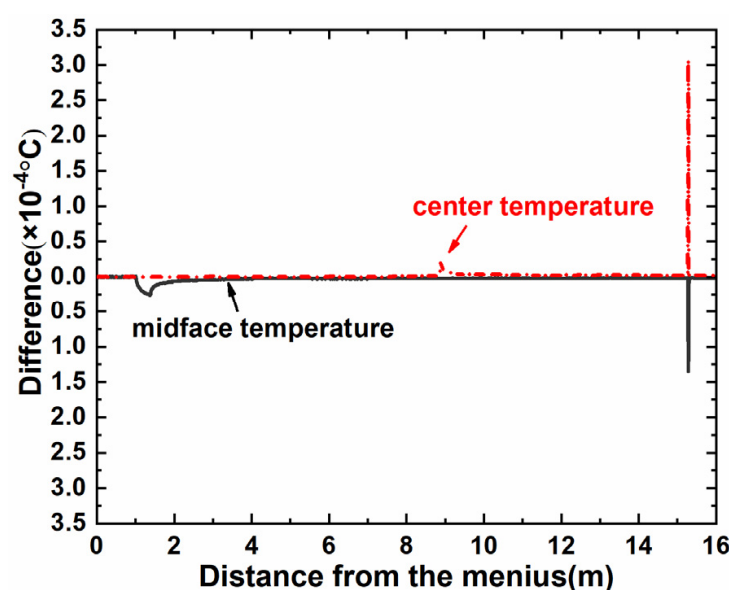


Figure 8. Temperature difference between serial CPU and GPU algorithms.

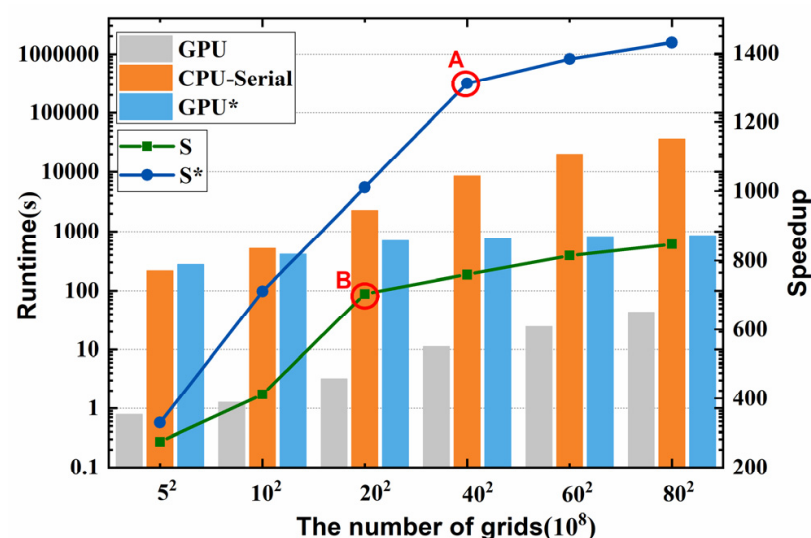


Figure 9. Calculation efficiency of serial CPU and GPU algorithms.

5. Conclusions

The proposed GPU-accelerated CA-DCSA method achieves a breakthrough performance in simulating the dendritic microstructures of continuous casting billets, with three key advancements:

- (1) Computational efficiency: A $1430\times$ speedup over serial CPU implementations enables microstructure simulations within practical timeframes, overcoming previous computational bottlenecks.
- (2) Morphological accuracy: The model resolves crystal zones, dendrite orientations, and secondary dendrite arm spacing with unprecedented clarity, surpassing prior CA methods in geometric fidelity.
- (3) Industrial validation: Experimental validation on steels 65# and 60# demonstrate robust agreement with measurement, with relative errors $< 2.5\%$ for equiaxed crystal ratio, secondary arm spacing, and temperature deviations less than $1.8\text{ }^{\circ}\text{C}$.

The successful implementation of the GPU-accelerated CA-DCSA method represents a significant advancement in microstructure simulation technology, offering valuable insights for optimizing industrial continuous casting parameters. The combination of computational efficiency and simulation accuracy makes it particularly suitable for practical applications in steel production and process development.

Author Contributions: Formal analysis, X.L.; Funding acquisition, X.L.; Investigation, X.L.; Methodology, J.W.; Resources, X.L. and R.M.; Software, J.W.; Validation, Y.L. and R.M.; Writing—original draft, J.W.; Writing—review and editing, J.W.; Writing—review and editing, Y.L. and R.M.; Visualization, Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by National Natural Science Foundation of China (62403111), the Postdoctoral Fellowship Program of CPSF under Grant Number (GZC20240226), the Fundamental Scientific Research Business of Central Universities (N2404026), Doctoral Scientific Research Initiation Foundation of Liaoning Province (2024-BSBA-20), and the Research Startup Fund of Shandong Youth Political College (XXPY24026).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liss, K.D.; Garbe, U.; Li, H.J.; Schambron, T.; Almer, J.D.; Yan, K. In Situ Observation of Dynamic Recrystallization in the Bulk of Zirconium Alloy. *Adv. Eng. Mater.* **2009**, *11*, 637–640. [\[CrossRef\]](#)
2. Wang, T.M.; Wei, J.J.; Wang, X.D.; Yao, M. Progress and Application of Microstructure Simulation of Alloy Solidification. *Acta Metall. Sin.* **2018**, *54*, 193–203. [\[CrossRef\]](#)
3. Domitner, J.; Kharicha, A.; Grasser, M.; Ludwig, A. Reconstruction of Three-Dimensional Dendritic Structures based on the Investigation of Microsegregation Patterns. *Steel Res. Int.* **2010**, *81*, 644–651. [\[CrossRef\]](#)
4. Hashemi, S.; Kalidindi, S.R. A machine learning framework for the temporal evolution of microstructure during static recrystallization of polycrystalline materials simulated by cellular automaton. *Comput. Mater. Sci.* **2021**, *188*, 110132. [\[CrossRef\]](#)
5. Dreelan, D.; Ivankovic, A.; Browne, D.J. Verification of a new cellular automata model of solidification using a case study on the columnar to equiaxed transition previously simulated using front tracking. *Comput. Mater. Sci.* **2022**, *215*, 111773. [\[CrossRef\]](#)
6. Chen, Z.; Jin, Y.; Chen, H.; Hu, S.; Jiang, Y.; Wu, M.; Zhu, B.; Zhang, W.; Li, W. Cellular automata simulation of pitting corrosion of stainless steel in marine environments. *Mater. Today Commun.* **2024**, *41*, 110555. [\[CrossRef\]](#)
7. Liu, A.; Chen, M.-S.; Chen, Q.; Lin, Y.C.; Wang, G.-Q.; Cai, H.-W.; Li, H.-B. A cellular automata model for recrystallization annealing of aged GH4169 superalloy and its application. *Mater. Today Commun.* **2025**, *42*, 111191. [\[CrossRef\]](#)

8. Zhang, J.; Li, X.; Xu, D.; Teng, C.; Wang, H.; Yang, L.; Ju, H.; Xu, H.; Meng, Z.; Ma, Y.; et al. Phase field simulation of the stress-induced α microstructure in Ti–6Al–4 V alloy and its CPFEM properties evaluation. *J. Mater. Sci. Technol.* **2021**, *90*, 168–182. [\[CrossRef\]](#)
9. Riyahi Khorasgani, A.; Steinbach, I.; Camin, B.; Kundin, J. A phase-field study to explore the nature of the morphological instability of Kirkendall voids in complex alloys. *Sci. Rep.* **2024**, *14*, 30489. [\[CrossRef\]](#) [\[PubMed\]](#)
10. Li, C.; Wen, J.; Li, K.; Chen, Q.; Wang, S. Modeling solid air dendrite growth solidification with thermosolutal diffusion using non-isothermal quantitative phase field method. *Int. J. Therm. Sci.* **2024**, *199*, 108929. [\[CrossRef\]](#)
11. Zhang, Z.; Ge, P.; Li, J.Y.; Ren, D.X.; Wu, T. Monte Carlo simulations of solidification and solid-state phase transformation during directed energy deposition additive manufacturing. *Prog. Addit. Manuf.* **2022**, *7*, 671–682. [\[CrossRef\]](#)
12. Zhang, J.; Liu, M.; Qi, J.; Lei, N.; Guo, S.; Li, J.; Xiao, X.; Ouyang, L. Advanced Mg-based materials for energy storage: Fundamental, progresses, challenges and perspectives. *Prog. Mater. Sci.* **2025**, *148*, 101381. [\[CrossRef\]](#)
13. Rodgers, T.M.; Moser, D.; Abdeljawad, F.; Jackson, O.D.U.; Carroll, J.D.; Jared, B.H.; Bolintineanu, D.S.; Mitchell, J.A.; Madison, J.D. Simulation of powder bed metal additive manufacturing microstructures with coupled finite difference-Monte Carlo method. *Addit. Manuf.* **2021**, *41*, 101953. [\[CrossRef\]](#)
14. Luo, S.; Zhu, M.Y.; Louhenkilpi, S. Numerical simulation of solidification structure of high carbon steel in continuous casting using cellular automaton method. *Isij. Int.* **2012**, *52*, 823–830. [\[CrossRef\]](#)
15. Li, J.; Wu, H.T.; Liu, Y.; Sun, Y.H. Solidification structure simulation and casting process optimization of GCr15 bloom alloy. *China Foundry* **2022**, *19*, 63–74. [\[CrossRef\]](#)
16. Wang, J.J.; Meng, H.J.; Yang, J.; Xie, Z. A Fast Method based on GPU for Solidification Structure Simulation of Continuous Casting Billets. *J. Comput. Sci.* **2021**, *48*, 101265. [\[CrossRef\]](#)
17. Wei, L.; Lin, X.; Wang, M.; Huang, W.D. A cellular automaton model for the solidification of a pure substance. *Appl. Phys. A-Mater.* **2011**, *103*, 123–133. [\[CrossRef\]](#)
18. Beltran-Sanchez, L.; Stefanescu, D.M. Growth of solutal dendrites: A cellular automaton model and its quantitative capabilities. *Met. Mater. Trans. A Phys. Met. Mater. Sci.* **2003**, *34*, 367–382. [\[CrossRef\]](#)
19. Wang, W.; Lee, P.D.; McLean, M. A model of solidification microstructures in nickel-based superalloys: Predicting primary dendrite spacing selection. *Acta Mater.* **2003**, *51*, 2971–2987. [\[CrossRef\]](#)
20. Yuan, L.; Lee, P.D. Dendritic solidification under natural and forced convection in binary alloys: 2D versus 3D simulation. *Model. Simul. Mater. Sc.* **2010**, *18*, 055008. [\[CrossRef\]](#)
21. Zhao, Y.; Chen, D.F.; Long, M.J.; Arif, T.T.; Qin, R.S. A Three-Dimensional Cellular Automata Model for Dendrite Growth with Various Crystallographic Orientations During Solidification. *Met. Mater. Trans. B* **2014**, *45*, 719–725. [\[CrossRef\]](#)
22. Chen, R.; Xu, Q.Y.; Liu, B.C. A Modified Cellular Automaton Model for the Quantitative Prediction of Equiaxed and Columnar Dendritic Growth. *J. Mater. Sci. Technol.* **2014**, *30*, 1311–1320. [\[CrossRef\]](#)
23. Wang, W.L.; Luo, S.; Zhu, M.Y. Development of a CA-FVM Model with Weakened Mesh Anisotropy and Application to Fe–C Alloy. *Crystals* **2016**, *6*, 147. [\[CrossRef\]](#)
24. Feng, W.M.; Xu, Q.Y.; Liu, B.C. Microstructure simulation of aluminum alloy using parallel computing technique. *ISIJ Int.* **2002**, *42*, 702–707. [\[CrossRef\]](#)
25. Jelinek, B.; Eshraghi, M.; Felicelli, S.; Peters, J.F. Large-scale parallel lattice Boltzmann-cellular automaton model of two-dimensional dendritic growth. *Comput. Phys. Commun.* **2014**, *185*, 939–947. [\[CrossRef\]](#)
26. Bauer, M.; Hotzer, J.; Jainta, M.; Steinmetz, P.; Berghoff, M.; Schornbaum, F.; Godenschwager, C.; Kostler, H.; Nestler, B.; Rude, U. Massively Parallel Phase-Field Simulations for Ternary Eutectic Directional Solidification. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Austin, TX, USA, 15–20 November 2015. [\[CrossRef\]](#)
27. George, W.L.; Warren, J.A. A parallel 3D dendritic growth simulator using the phase-field method. *J. Comput. Phys.* **2002**, *177*, 264–283. [\[CrossRef\]](#)
28. Wei, L.; Lin, X.; Wang, M.; Huang, W.D. Orientation selection of equiaxed dendritic growth by three-dimensional cellular automaton model. *Physical B* **2012**, *407*, 2471–2475. [\[CrossRef\]](#)
29. Provatas, N.; Greenwood, M.; Athreya, B.; Goldenfeld, N.; Dantzig, J. Multiscale modeling of solidification: Phase-field methods to adaptive mesh refinement. *Int. J. Mod. Phys. B* **2005**, *19*, 4525–4565. [\[CrossRef\]](#)
30. Mattson, T.G.; He, Y.; Koniges, A.E. *The OpenMP Common Core: Making OpenMP Simple Again*; The MIT Press: Cambridge, MA, USA, 2019.
31. Hong, S.H.; Shu, J.I.; Ou, J.; Wang, Y. GPU-enabled microfluidic design automation for concentration gradient generators. *Eng. Comput.* **2022**, *39*, 1637–1652. [\[CrossRef\]](#)
32. Ames, J.; Puleri, D.F.; Balogh, P.; Gounley, J.; Draeger, E.W.; Randles, A. Multi-GPU Immersed Boundary Method Hemodynamics Simulations. *J. Comput. Sci.* **2020**, *44*, 101153. [\[CrossRef\]](#)

33. Fain, B.G.; Dobrovolny, H.M. GPU acceleration and data fitting: Agent-based models of viral infections can now be parameterized in hours. *J. Comput. Sci.* **2022**, *61*, 101662. [[CrossRef](#)]
34. Liu, X.-Y.; Xie, Z.; Yang, J.; Meng, H.-J.; Wu, Z.-Y. A faster than real-time heat transfer model for continuous steel casting. *J. Mater. Res. Technol.* **2022**, *19*, 4220–4232. [[CrossRef](#)]
35. Liu, X.-Y.; Xie, Z.; Yang, J.; Meng, H.-J. Accelerating phase-change heat conduction simulations on GPUs. *Case Stud. Therm. Eng.* **2022**, *39*, 102410. [[CrossRef](#)]
36. Bandini, S.; Mauri, G.; Serra, R. Cellular automaton: From a theoretical parallel computational model to its application to complex system. *Parallel Comput.* **2001**, *27*, 539–553. [[CrossRef](#)]
37. Campos, R.; Amorim, R.; de Oliveira, B.; Rocha, B.; Sundnes, J.; Barra, L.; Lobosco, M.; dos Santos, R. 3D Heart Modeling with Cellular Automata, Mass-Spring System and CUDA. *Parallel Comput. Technol.* **2013**, *7979*, 296–309. [[CrossRef](#)]
38. Ferrando, N.; Gosálvez, M.A.; Cerdá, J.; Gadea, R.; Sato, K. Octree-based, GPU implementation of a continuous cellular automaton for the simulation of complex, evolving surfaces. *Comput. Phys. Commun.* **2011**, *182*, 628–640. [[CrossRef](#)]
39. Blečić, I.; Cecchini, A.; Trunfio, G.A. Cellular automata simulation of urban dynamics through GPGPU. *J. Supercomput.* **2013**, *65*, 614–629. [[CrossRef](#)]
40. Ntinas, V.G.; Moutafis, B.E.; Trunfio, G.A.; Sirakoulis, G.C. Parallel fuzzy cellular automata for data-driven simulation of wildfire spreading. *J. Comput. Sci.* **2017**, *21*, 469–485. [[CrossRef](#)]
41. Yang, J.; Xie, Z.; Meng, H.-J.; Liu, W.H.; Ji, Z.P. Multiple time steps optimization for real-time heat transfer model of continuous casting billets. *Int. J. Heat Mass Transf.* **2014**, *76*, 492–498. [[CrossRef](#)]
42. Zhang, J.; Chen, D.-F.; Zhang, C.-Q.; Wang, S.-G.; Hwang, W.-S. Dynamic spray cooling control model based on the tracking of velocity and superheat for the continuous casting steel. *J. Mater. Process. Technol.* **2016**, *229*, 651–658. [[CrossRef](#)]
43. Xie, Z.; Yang, J. Calculation of Solidification-Related Thermophysical Properties of Steels Based on Fe-C Pseudobinary Phase Diagram. *Steel Res. Int.* **2015**, *86*, 766–774. [[CrossRef](#)]
44. Savage, J.; Pritchard, W.H. The problem of rupture of the billet in the continuous casting of steel. *J. Iron Steel Inst.* **1954**, *178*, 269–277.
45. Liu, D.R.; Wu, S.P. Computer Simulation of Microstructure in Solidification of Aluminium and Silicon Alloy. *J. Harbin Univ. Sci. Technol.* **2002**, *7*, 97–100.
46. Cheng, J. CUDA execution Model. In *Professional CUDA C Programming*; Wrox Press Ltd.: Hoboken, NJ, USA, 2014; pp. 68–133.
47. Biscuola, V.; Martorano, M. Mechanical Blocking Mechanism for the Columnar to Equiaxed Transition. *Met. Mater. Trans. A Phys. Met. Mater. Sci.* **2008**, *39*, 2885–2895. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.