3D Integrated Circuits Layout Optimization Game

Katarzyna Grzesiak-Kopeć(✉), Leszek Nowak, and Maciej Ogorzałek

Department of Information Technologies,
Jagiellonian University in Krakow, Krakow, Poland
{katarzyna.grzesiak-kopec,leszek.nowak,maciej.ogorzalek}@uj.edu.pl

Abstract. This paper is devoted to the original approach to block-level 3D IC layout design. The circuit components are modeled as autonomous mobile agents that explore their virtual world in order to find a globally near-optimal layout solution. The search space is defined by geometry features, wire connections, goals and constraints of the design task. The approach is illustrated by the example application to one of the MCNC benchmark circuits and implemented using Godot.

Keywords: Floorplaning · Machine learning · Steering behaviors · Optimization · Computer game

1 Introduction

The problem of a valid 3D layout generation can be found in many different domains, starting from practical and scientific purposes, through virtual reality modeling, ending at urban planning or crisis management. Design solutions that fulfill requirements and meet constraints promote minimizing the materials and energy consumption while optimizing the functional properties. The spatial arrangement of components also plays a crucial role in integrated circuit design. The chip design involves myriad conditions related to chip area minimization, thermal hot spots reduction or wire length optimization, which makes it especially challenging. An original approach to block-level 3D IC layout design has been proposed in [7,8], where the intelligent framework architecture uses a simple shape grammar to generate topologically feasible solutions. These proposal solutions are further optimized with a use of the extremal optimization. In this paper, an alternative concept of a computer game like visual 3D optimal layout design is proposed. It is inspired by swarm intelligence algorithms and steering behaviors for autonomous agents in animation and computer games. The components are treated as autonomous agents that navigate around their world in order to find a globally near-optimal solution. Combinations of steering behaviors are used to achieve both goals and constraints of a specific layout design task. The approach is illustrated by the example application to one of the MCNC benchmark circuits [12] and implemented using Godot which is an advanced, feature-packed, multi-platform 2D and 3D open source game engine [6].

L. Rutkowski et al. (Eds.): ICAISC 2017, Part II, LNAI 10246, pp. 444–453, 2017.
DOI: 10.1007/978-3-319-59060-8_40

2 Related Work

The most critical phase in integrated circuits design is floorplanning. It is a kind of a packing task where all circuit components have to be arranged according to given design rules. The circuit components are rectangular modules that cannot overlap. The minimum bounding box of a packing is called the chip [14]. The problem have been effectively solved in 2D spaces, but the proposed algorithms are not easily transformed to introduce the third dimension. The 3D chips allow for the smaller footprint, higher packing density, lower interconnect power consumption and heterogeneous technology chip support [4]. However, the today's 3D IC technology has some important limitations. A truly 3D chip fabrication is actually impossible. All the circuit components are distributed among restricted number of device layers and the height of the inter-layers is fixed (see example configurations in Fig. 1).

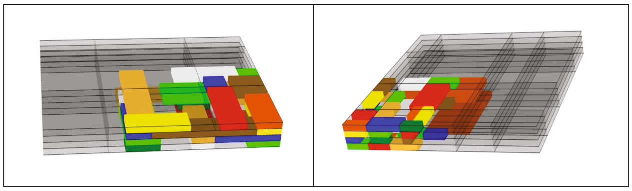

Fig. 1. Quasi-3D ICs configurations example.

The 3D integrated circuits placement problem is known to be NP-hard. Common techniques for global placements are: partitioning-based algorithms, analytic techniques and stochastic ones [10]. Recursive partitioning are constructive techniques that recursively cut the layout into smaller parts. The most common partitioning algorithms are the Kernighan-Lin [11] and the Fiduccia-Mattheyses algorithm [5]. Analytic techniques use either quadratic objective functions or sophisticated nonlinear calculations [16]. The most popular stochastic based placement uses simulated annealing [2]. The generic approach to 3D layout design proposed in [7] generates plausible solutions with a use of a simple shape grammar supervised by an intelligent derivation controller. Design knowledge is fed into system in a form of predicates. The floorplan generation procedure takes into account the current technological limitations and divides a chip into layers. Hence, the obtained design results are quasi-3D ones.

The seminal research into steering behavior by Craig Reynolds [17] modeled the movement patterns of flocks, and since then has been studied from many different perspectives, like swarm robotics [9], crowd simulation [19] or artificial life [15]. In the 3D layout design task, the components should navigate around their world to find a globally near-optimal solution. The use of behavioral animation in generating virtual worlds is still the subject of many different research projects. A number of them use hierarchical schemes for organizing complex control [3].

3 Autonomous Agent

In our approach agents corresponds to physical components of the design, which should be optimally arranged in the search space. The term *autonomous agent* may be used in many different contexts. In this paper, by an agent we understand a computer system situated in a world shared by other entities, which is capable of autonomous actions that lead it to satisfy its design task [20]. It is not only reactive, perceives its environment and responds to changes that occur in it, but exhibits goal-driven behavior as well. It also interacts with other agents. Having all this features, an agent can be recognized as intelligent [21]. It is a real agent in a virtual world embodied in a physical manifestation.

Also the term of behavior has many different interpretations. Being inspired by swarm intelligence heuristics we have decided to solve a layout design task applying various motion behaviors. Combining stochastic approach and motion behaviors may provide an effective mechanism for screening large and discontinuous spaces. Thus after [17], the agent's behavior may be divide into a hierarchy of three layers: *action selection*, *steering*, and *locomotion*. The *action selection* layer involves actions strategy, goals and planning. In the *steering* level, the goal is decomposed into a series of simple subgoals that correspond to some steering behaviors and an agent path is determined. Finally, the *locomotion* layer is responsible for an actual movement.

3.1 Steering Behaviors

Steering behaviors allow autonomous agents to navigate around their environment in a life-like or any imaginative manner. They are usually defined in such a way to be largely independent of the agent's means of locomotion and have a similar structure. They take as an input the kinematic of the agent that is moving and some target information [13]. They can be divided into simple and combined behaviors presented in Table 1.

Table 1. Simple and combined steering behaviors.

Simple behaviors	Combined behaviors
Seek & flee	Pursuit & evade
Arrive	Wander, obstacle avoidance, path following, ...
Align	Flocking behavior: separation, cohesion and alignment

Simple behaviors are applicable to single agents. *Seek* steers the agent towards a specified target. It calculates the direction to the target in the global coordinate system and heads toward it as fast as possible (maximal speed). If no other behavior appears, the agent eventually pass through the target and then turn back to approach again. *Flee* is the opposite of seek. The agent turns away from the target and tries to get as far from it as possible. *Arrive* is a kind of a seek behavior that slows the agent down as it approaches the target and makes it stop there. *Align* is responsible for the agent heading. It turns the agent to reach the target orientation.

Combined behaviors are applicable not only to single agents but to groups of agents as well. *Pursuit* and *evade* derive from seek and flee behaviors respectively and used when a target is moving. *Wander* is a kind of random life-like steering that enables to move agent around the world when no target is specified. It acts as a delegated seek behavior. *Obstacle avoidance* allows an agent to maneuver in a cluttered environment by dodging around obstacles. It casts one or more rays out in its motion direction. If the collision with an obstacle occurs, a new target is calculated in such a way to avoid it. Then a moving agent simply seek on the new target. *Path following* enables an agent to steer along a predetermined path within the specified radius of the spine. It applies a seek behavior to steer toward a predicted future position. The most common group steering behavior is *flocking* [17].

In many computer games, simple steering behaviors can achieve a satisfying movement realization. Some decision making algorithms determine where the agent should move and the seek behavior is applied to perform it. However, in order to reach its goal safety and avoid collisions, an autonomous agent usually needs more than one steering behavior. There are two general methods of combining steering behaviors: *blending* and *arbitration*. Both of them, take a group of steering behaviors and generate a single overall steering output.

Blending uses a set of weights or priorities to combine the results of all the steering behaviors. There are no constraints on the blending weights. The final steering output achieved from the weighted sum may even go far beyond the moving capabilities of the agent, so it is simply trimmed according to the maximum possible value. There is no simple answer how to determine the right coefficients values. Even though there are different research projects trying to automate the tuning of model parameters using evolutionary strategies [1], as in most parametrized systems, they are greatly dependent on the system architect experience and her/his inspired lucky guess or a good trial and error. To be more efficient, weights or priorities may change over time in response to the state of the working environment.

One of the best known combined blended behavior is *flocking*. *Flocking* is a kind of coordinated motion inspired by animals groups such as bird flocks and fish schools [17]. It blends three steering behaviors, namely *separation*, *cohesion* and *alignment*. Separation moves an agent away from agents that are too close. Cohesion works in a quite opposite way and moves an agent toward the center of mass of the flock. Alignment lets all the agents to move in the same direction and at the same velocity. In some cases, using equal blending weights for all of these three behaviors may be sufficient. However, usually separation is more important than cohesion, which is more important than alignment. While blending, it is also possible to use priorities groups of behaviors. Each group contains behaviors with regular blending weights and is considered according to a given priority order.

Arbitration uses different schemes to select a current steering behavior. There are no restrictions imposed on the arbitration, which would enforce to return only one simple steering behavior instead of a combined one. In fact, blending and arbitration are often mixed together to get more realistic implementations.

Both blending and arbitration combine steering behaviors in an independent manner. Yet, in order to obtain more realistic model, some cooperation among different behaviors is required. Being aware of its context, a steering behavior increases its complexity and is more difficult to handle. Thus, collaborative steering behaviors implementations use more sophisticated decision making algorithms like state machines, decision trees, or a steering pipeline.

4 3D ICs Layout Design Constraints and Goals

While investigating the 3D ICs layout design problem, various constraints and goals were identified [7]. They are summarized in Tables 2 and 3 respectively. First of all, all the circuit components should be placed in a specified chip (AREA) without overlapping (NO INTERSECTION). To minimize the chip bounding box, a plausible layout has to be consistent (GLUE). The total wirelength of a chip is minimal if the connected components are as close to each other as possible (ADJACENT). Thermal management requests separating selected modules to minimize a hot spot problem (NEIGHBOR) and also may require to settle the most heating components in the outermost layers (LAYER).

Table 2. 3D ICs floorplanning constraints

Constraint	Description
AREA	A component must be placed in a specified area
NO INTERSECTION	A component does not intersect other components
GLUE	A solution is consistent
NEIGHBOR	A component is in the specified neighborhood range
ADJACENT	Neighboring faces of adjoining components are of the same type
LAYER	A component is in a boundary (intra) layer

Constraints are either true or false, while objectives can be achieved to some extent. Instead of rejecting imperfect solutions, the search procedure should change its direction toward better ones. The main goal is a chip (packing) minimization (MINIMAL SPACE). Some components are preferred to be placed as close to the boundary as possible (POSITION) and some require aligning (SPATIAL RELATION).

Table 3. 3D ICs floorplanning goals

Goal	Description
MINIMAL SPACE	Evaluates the area occupied by a current design in relation to the expected minimal area
POSITION	Evaluates whether components are generated in the expected positions (e.g. boundary)
SPATIAL RELATION	Evaluates whether components are arranged in an expected way (e.g. aligned vertically)

5 Game

In this paper, we propose an original approach to the 3D integrated circuits layout optimization problem that goes far beyond the current technological and manufacturing limitations. The circuit components are autonomous mobile agents situated in a search space defined by their geometry features, wire connections, goals and constraints. To verify the proposed method, a dedicated game has been developed using the open source Godot game engine [6]. All the examples presented in this paper are generated with a use of the original software. The approach is illustrated by the example application to one of the MCNC benchmark circuits [12], namely *apte.yal* which is composed of 9 components all connected to one another (Fig. 2).

5.1 Circuit Components

All circuit components are cuboids with specified geometry features and wire connections. They are modeled with a use of Godot *RigidType* nodes. This kind

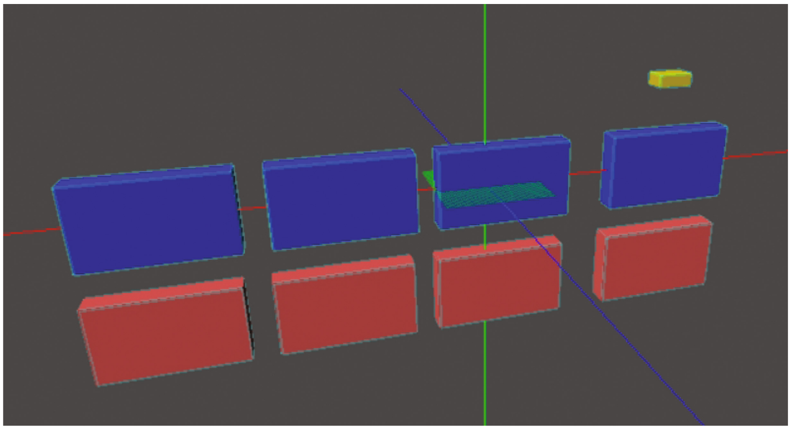

Fig. 2. Godot: *apte.yal* components.

of body has mass, friction, bounce and simulates Newtonian physics. Its motion may be affected by gravity and other entities. Its current position is generated by the simulation of linear and angular velocity from the former one. In order to meet the NO INTERSECTION constraint, each component has also appropriate *CollisionShape* assigned. It also knows his circuit connections (connected components), called *neighbors*.

5.2 Behavioral Animation

The game make use of the flocking behavior and gravity. The optimization procedure is actually driven by the physics engine implemented in Godot. The main challenge is the appropriate assignment of steering forces. Its general algorithmic scheme is very simple and proceeds as follows:

```
while(!stop){
  for each net{
    calculate the center of mass
    for each component in net{
      seek toward the center of mass
    }
  }
 }
```

Before any movement begins, all the constraints and goals must be defined by the means of steering behaviors (see summary in Table 4). The whole process starts with a random positioning of chip components in a 3D search space. Then the game moving algorithm is applied where the main goal is the MINIMAL SPACE one.

Table 4. The constraints and goals mapping to steering behaviors.

Constraint/Goal	Steering behavior
AREA	Cohesion
NO INTERSECTION	Ceparation
GLUE	Cohesion and gravity
NEIGHBOR	Cohesion and separation, respectively
ADJACENT	Faces cohesion and separation, respectively
LAYER	Faces cohesion and separation, respectively
MINIMAL SPACE	Cohesion and gravity
POSITION	Cohesion and separation, respectively
SPATIAL RELATION	Alignment

The whole game may be divided in two logical stages. The aim of the first stage is to find minimal local arrangements of connected neighbors (NEIGHBOR constraint). During this stage, the gravity force is completely neglected and the main steering behavior that let this stage accomplish is *cohesion*. The components are moved toward the center of mass of the neighborhood until they collide. When no further move is possible, components try to rotate in an obtained position to minimize the neighborhood volume. Just like in the simulated annealing algorithm, better configurations are always accepted while worst are accepted with a certain probability. In the same time, while moving closer toward its connected neighbors, components are affected by repulsive forces from components that are either not directly connected to them or should not be placed close to each other (*separation steering behavior*). In order to incorporate the SPATIAL ARRANGEMENT goal, the *alignement* steering behavior is applied. The example configuration of components obtained in such a way is presented in Fig. 3. At the first glance it seems to be far away from the optimal one, but only few more actions are required to improve it.

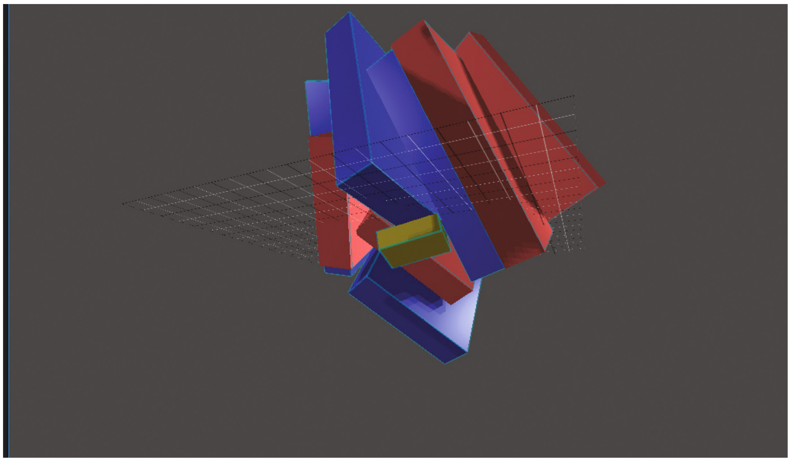

Fig. 3. The example configuration of components obtained in the first stage of the game.

The first stage is finished by the game player (designer). She/he turns on the gravity and the second stage starts. Now, the aim of the game is to squeeze the intermediate solution. To better understand the proposed approach, let's imagine the process of collecting spilled deck of cards from the table. First, we grab all the cards and try to hold them. After that, we lower the cards on the table without dropping them from the hand and they are aligned in one dimension. Then, we make a 90-degree turn and repeat the procedure to aligned them in a second dimension. The same process is applied to a components configuration. While keeping the attraction forces among neighbors, the configuration is affected by gravity. It falls down into a 90-degree V-shape (virtual table). A 90-degree V-like shape eliminates the need of turning the configuration and repeating the falling

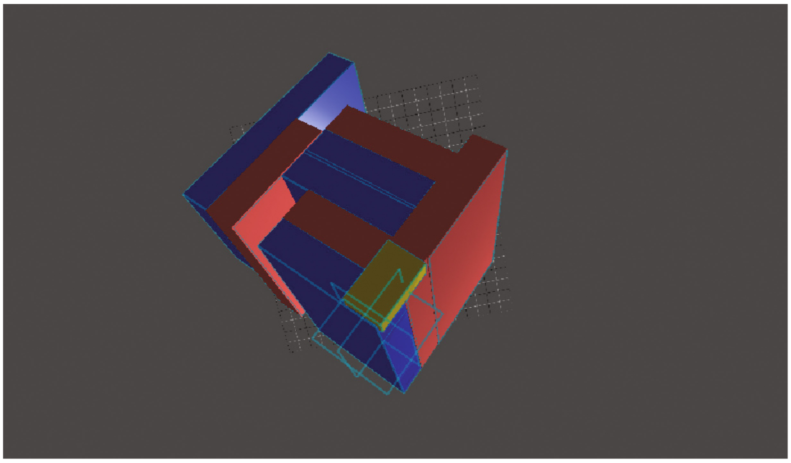

Fig. 4. The example configuration of components obtained in the second stage of the game.

procedure. After reaching the virtual ground, the chip is much more compact (see Fig. 4).

6 Conclusions and Future Prospects

The presented approach is a part of ongoing research on building a flexible software architecture framework which will enable solving the 3D integrated circuits layout problem. The task is not only up-to-date but very challenging one as well. The market electronic design automation (EDA) tools are dedicated solutions adjusted to present technology limitations. Most of them are not fully 3D aware but rather adapt 2D algorithms (2.5D IC design flow) [18]. Treating circuit components as an autonomous agents that are governed by the laws of Newtonian physics and navigate around their virtual reality, is a completely new approach to solving this problem. Both goals and constraints are described by the means of steering behaviors. Even though the final outcome of the research is still hard to predict, taking into account the preliminary results and practical applications of autonomous agents in complex and dynamic environments like a crowd simulation, we strongly belief that it is worth pursuing.

References

1. Berseth, G., Kapadia, M., Haworth, B., Faloutsos, P.: SteerFit: automated parameter fitting for steering algorithms. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2014), Eurographics Association, Aire-la-Ville, Switzerland, pp. 113–122 (2015)

2. Chen, T.C., Chang, Y.W.: Modern floorplanning based on B*-tree and fast simulated annealing. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **25**(4), 637–650 (2006)
3. Donikian, S., Rutten, E.: Reactivity, concurrency, data-flow and hierarchical preemption for behavior animation. In: Veltkamp, R.C., Blake, E.H. (eds.) Programming Paradigms in Graphics. Eurographics Collection. Springer, Vienna (1995)
4. Dong, X., Xie, Y.: System-level cost analysis and design exploration for three-dimensional integrated circuits (3D ICs). In: Proceedings of the 2009 Asia and South Pacific Design Automation Conference (ASP-DAC 2009), pp. 234–241 , IEEE Press, Piscataway, NJ, USA, (2009)
5. Fiduccia, C.M., Mattheyses, R.M.: A Linear-time heuristic for improving network partitions. In: DAC, pp. 175–181 (1982)
6. Godot: An advanced, feature-packed, multi-platform 2D and 3D open source game engine (2016). https://godotengine.org/. Accessed Dec 2016
7. Grzesiak-Kopeć, K., Ogorzałek, M.: Computer-aided 3D ICs layout design. Comput. Aided Des. Appl. **11**(3), 318–325 (2014)
8. Grzesiak-Kopeć, K., Oramus, P., Ogorzałek, M.: Using shape grammars and extremal optimization in 3D IC layout design. Microelectron. Eng. **148**, 80–84 (2015)
9. Joselli, M., Passos, E.B., Zamith, M., Clua, E., Montenegro, A., Feijó, B.: A neighborhood grid data structure for massive 3D crowd simulation on GPU. In: 2009 VIII Brazilian Symposium on Games and Digital Entertainment, pp. 121–131 (2009)
10. Kahng, A.B., Lienig, J., Markov, I.L., Hu, J.: VLSI Physical Design: From Graph Partitioning to Timing Closure. Springer Publishing Company Inc., Heidelberg (2011)
11. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell Syst. Tech. J. **49**(2), 291–307 (1970)
12. MCNC: The MCNC set of benchmark circuits (2015). http://lyle.smu.edu/~manikas/Benchmarks/MCNC_Benchmark_Netlists.html. Accessed June 2015
13. Millington, I., Funge, J.: Artificial Intelligence for Games, 2nd edn. Morgan Kaufmann Publishers Inc., San Francisco (2009)
14. Murata, H., Fujiyoshi, K., Nakatake, S.: VLSI module placement based on rectangle-packing by the sequence-pair. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **15**(12), 1518–1524 (1996)
15. Nathan, A., Barbosa, V.C.: V-like formations in flocks of artificial birds. Artif. life **14**(2), 179–188 (2008)
16. Obermeier, B., Johannes, F.M.: Temperature-aware global placement. In: Proceedings of the 2004 Asia and South Pacific Design Automation Conference (ASP-DAC 2004), pp. 143–148, IEEE Press, Piscataway, NJ, USA, (2004)
17. Reynolds, C.: Steering behaviors for autonomous characters. In: Game Developers Conference, pp. 763–782 (1999)
18. Rhines, W.: 3D IC design challenges. In: GSA Memory Conference, San Jose, CA (2011)
19. Thalmann, D., Musse, S.R.: Crowd Simulation, 2nd edn. Springer, Heidelberg (2013)
20. Wooldridge, M.J., Jennings, N.R.: Intelligent agents: theory and practice. Knowl. Eng. Rev. **10**(2), 115–152 (1995)
21. Wooldridge, M.J.: Intelligent Agents, Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, Cambridge (1999)
22. Zhang, H.: The optimality of naive bayes. In: FLAIRS Conference (2004)