Efficient Placement Algorithm for the Interposer-Based Multi-FPGAs Systems

RUNXIAO SHI † LAN MA†
TAKAHIRO WATANABE †

Abstract: Interposer-based multi-FPGAs systems are composed of several monolithic FPGAs dies by the connection of silicon interposers. This novel FPGAs allow us to use much more logic resources, but inter-die communication also increases the circuit latency. In this paper, we propose a novel interposer-based FPGAs architecture model, separating monolithic FPGAs die vertically and horizontally at the same time. According to the new FPGAs architecture, during the calculation of cost function, we propose extra wire and delay cost on the netlists crossing the interposer, especially the netlists crossing the intersection points of interposers. Finally, in order to improve the convergence speed of the algorithm, we introduce a temperature-dependent perturbation model based on Cauchy distribution to generate new solutions.

Keywords: Placement, Silicon interposer, Simulated annealing, 2.5D FPGAs

1. Introduction

1.1 Interposer-based Multi-FPGAs

The interposer-based FPGAs are composed of several FPGA dies which are connected by the silicon interposers. The world's first interposer-based FPGAs, the Virtex-7 XC7V2000T [1], was manufactured and put into commercial use by Xilinx Inc in 2012. The general architecture of Virtex-7 XC7V2000T shows in Figure 1. When we place the circuit block on this kind of FPGAs, we need to use three-digit coordinates (x, y, z) to determine its location, where x and y are the traditional coordinates and z represents the index of FPGA dies. Therefore, interposer-based multi-FPGAs has the basic 3-dimensional structure. However, the circuit blocks put on the FPGAs are still two-dimensional. Therefore, we call this kind of FPGAs as 2.5-dimensional FPGAs.

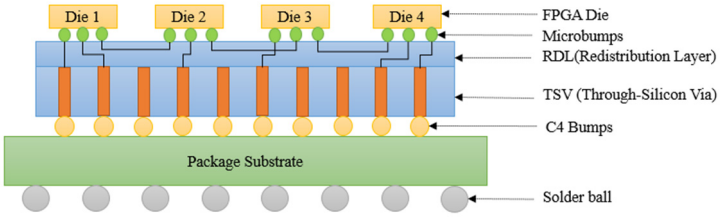


Figure 1 The general architecture of the 2.5D FPGAs.

Recently, the number of transistor on the integrated circuits can no longer grow at the rate predicted by Moore's Law due to manufacturing constraints. However, the introduction of interposer-based FPGAs realizes a "More than Moore" [3] increase on the capacity of logical resource on FPGAs. More logical resources enable us to use FPGAs to handle much more complex tasks such as using FPGAs to design the accelerator for the machine learning.

At the same time, the use of interposer also brought obvious problems. In Figure 1, if a CLB on SLR3 want to communicate with SLR2, the signal needs to go through tracks on SLR3, microbumps under SLR3, super long line within interposer, microbumps under SLR2, tracks on SLR2 and finally arrives at the CLB on SLR2 [4]. This process is much more complex compared with the traditional communication on monolithic FPGAs and increase the time delay dramatically.

1.2 Placement Process

Placement is a very important stage in the FPGAs CAD flow. It determines the location we map the circuit blocks (logical blocks and I/O blocks) on the FPGAs. The placement result directly affects the subsequent routing process and the maximum delay of the circuit. Figure 2 shows an example of placement and routing results. Placement is a classical NP-hard problem. The size of the solution space is exponential to the number of logical blocks on the FPGAs. As the capacity of logic resources on the FPGAs increases, the placement becomes more and more complex. Simulated annealing (SA) algorithm [5] is one of the most popular algorithms to solve the placement problem.

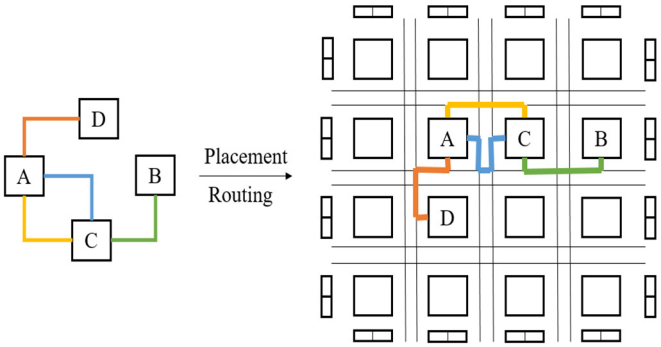


Figure 2 Example of placement and routing results

1.3 Prior Work

Many previous works have addressed various algorithms for the placement on FPGAs. In [5], the authors develops a CAD tool called Versatile Place and Route (VPR) , using simulating annealing algorithm for placement. Since then, simulating annealing-based placement algorithm has become the

† Graduate School of Information, Production and Systems, Waseda University

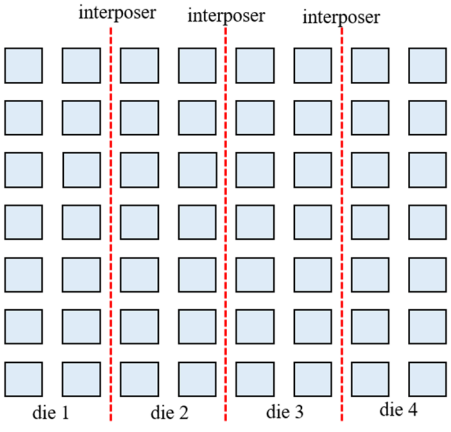
mainstream for its open cost function and high placement quality. Many researches [12-15] have carried out to optimize the traditional simulating annealing-based placement algorithm, attempting to achieve better placement results in shorter CPU time. However, most of them focus on the island style FPGAs and do not take interposers into consideration. [16] proposed a chip-interposer codesign flow for the interposer-based multi-chips systems, which still had some differences between the placement on interposer-based FPGAs. [2, 3] studied the CAD flow of the interposer-based multi-FPGAs systems, simulating the interposer by modifying the routing resources, adding extra time delays and extra wire length when netlists cross the interposer. However, they only separated the monolithic FPGAs in one direction.

In our work, we not only take the interposer into consideration, but also separate the monolithic FPGA simultaneously in both horizontal and vertical directions, which make the structure much more flexible. Also, we modify the traditional simulating annealing-based placement algorithm to improve the convergence speed and apply it on the interposer-based multi-FPGAs systems.

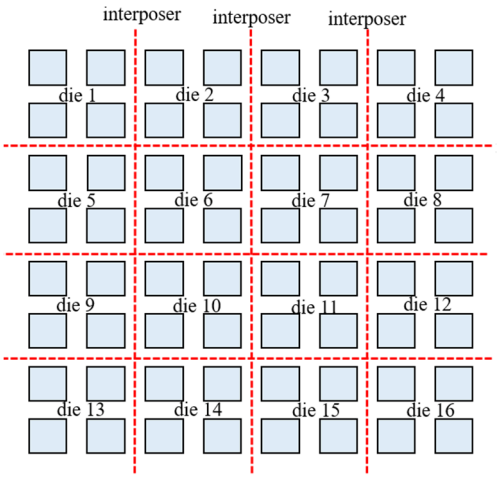
The rest of this paper is organized as follows. Section 2 describes a novel architecture model of interposer-based FPGAs. Then, in the Section 3, we introduce a new perturbation model based on Cauchy distribution to generate new solutions and improve the traditional SA algorithm. According to the new architecture model, new cost functions have been used. In Section 4, we study the effect of architecture perimeters on placement result and demonstrate the superiority of our modified SA algorithm for the placement on interposer-based multi-FPGAs. Finally, section 5 concludes this paper.

2. Novel FPGAs Architecture Model

The Virtex-7 XC7V2000T has 4 dies and 1.95 million logic units [1]. 4 dies aligned in a row (1×4). Therefore, traditional work usually models interposer-based FPGAs by making a vertical cut across the entire large monolithic FPGAs [2] and treating the cutline as the interposer (Figure 3 (a)). In this paper, we arrange the FPGAs dies in n×m 2-D array (n>1, m>1). Therefore, during the modeling process, we simultaneously cut horizontally and vertically (Figure 3 (b)).



(a)



(b)

Figure 3 Architecture Models of interposer-based FPGAs

When the circuit netlists go through the interposer, the time delay between CLBs on the different dies will increase compared with the communication within one FPGA die. Also, the routing resources within the interposer are not as rich as the routing resources on FPGAs dies. Some tracks will be cut when the routing channel crosses the cutline. Several variables are used to describe the FPGA structure and three of them are proposed in [2]: *% wires cut*, *DelayIncrease* and *number of cut*.

% wires cut represents the reduction percentage of routing tracks when the routing channel passes through the interposer. For example, if the *% wires cut* is 80 and there is a channel with 100 tracks, then 20 tracks can cross the interposer and 80 tracks will be cut. Figure 4 shows an example.

DelayIncrease represents the extra timing cost when the netlists go through the interposer. We will introduce the extra time delay described in the following section.

number of cut describes the number of horizontal and vertical cuts we made to separate the monolithic die. In Figure 3 (b), the horizontal *number of cut* is 3 and vertical *cut number is also 3.* After the cut, we have 16 FPGAs dies.

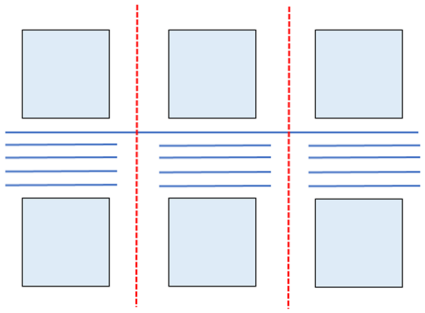


Figure 4 *% wires cut* equal to 80.

3. Placement Algorithm

The placement algorithm used in this paper is based on the simulating annealing (SA) algorithm [6]. SA algorithm simulates the movement of particles during metal annealing process. This process can be divided into two stages, first heating the metal,

then gradually cooling it. In the process of cooling, particle movement gradually transfers from disorder to order, and eventually reaches the minimum internal energy state, which is the most stable state of the system.

For FPGAs placement, the arrangement of particles is a potential placement solution during the layout process. Particle motion is to generate new layout solutions through perturbations. The lowest energy state is the best layout solution. The value of cost function is the internal energy of each potential solution.

We add several modifications to the traditional SA algorithm. First, in order to improve the convergence speed of the algorithm, we introduce a temperature-dependent perturbation model based on Cauchy distribution to generate new solutions. Based on this new model, we use a new acceptance probability to jump out the local optima. We call this modified SA algorithm as the Very Fast Simulating Annealing (VFSA) algorithm. Then, according to the new FPGAs architecture, during the calculation of cost function, we evaluate additional wire cost and timing cost on the circuit netlists crossing the interposer, especially the netlists crossing the intersection of cutlines.

3.1 Perturbation Model

In the placement process, we generate new solutions by changing the position of blocks. The new temperature-dependent perturbation model is based on Cauchy distribution [7] and can be formulated by equation (1) and (2) [8]. The new perturbation model enables our algorithm to perform long-distance location change at high temperature and small modification at low temperature, making the convergence much more directional.

$$x_i^{new} = x_i^{old} + C \times W \tag{1}$$

$$C = T \times \mathrm{sgn}(u - 0.5) \times [(1 + \frac{1}{T})^{|2u-1|} - 1] \tag{2}$$

where x_i is the coordinate of block i, W is the width of whole multi-FPGAs systems, T is the temperature, sgn() is the symbol function, u is a random number between 0 and 1.

3.2 Acceptance Probability

The probability we accept worse solution and allow local deterioration is shown in equation (3), where ΔC represents the cost difference between old solution i and new solution j, h is a positive real number. The calculation of cost function will be shown in the following parts. Refer to [9] and [10] about detailed derivation process of equation (3).

$$P(i \to j) = \begin{cases} 1 & C(i) > C(j) \\ [1 - \frac{(1-h)\Delta C}{T}]^{\frac{1}{(1-h)}} & C(i) \le C(j) \end{cases} \tag{3}$$

3.3 Placer Timing Cost

The cost function in our algorithm is based on [2]. Inter-die communication requires the participation of microbumps and supper long line within the interposer. The signal needs to go through tracks, microbumps, supper long line, microbumps and tracks in turn. Compared with the traditional single-die communication, the delay of the inter-die communication and the required wiring resource are significantly increased. Therefore, the modification of the cost function focuses on adding extra penalty to the netlists crossing the interposer.

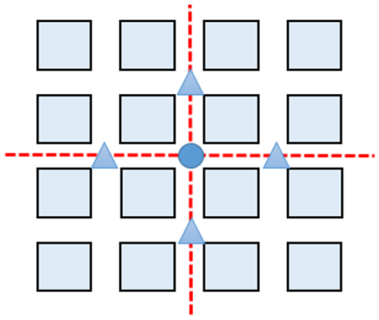


Figure 5 Example of ODP (△)and TDP (○).

As we mentioned before, when a routing channel crossing the interposer, some tracks will be cut. For the intersection point of two interposers, both vertical and horizontal routing resources are reduced. This situation is much worse compared with the reduction in only one direction. The location where wiring resources are reduced is called a defect point, the point where routing resources reduction in one direction is called one-direction defect point (ODP) and the point where routing resources are decreased in both directions is called two-direction defect point (TDP). The example of ODP and TDP is shown in Figure 5.

Figure 6 shows several situations of a bounding box of logical blocks to be placed on the interposer-based multi-FPGAs. Situation A is the best case, since all the blocks in the bounding box are on one FPGA die, inter-die communication will not happen. Situation B is the second and situation C is the worst one. In the placement process, we want to place more netlists under situation A.

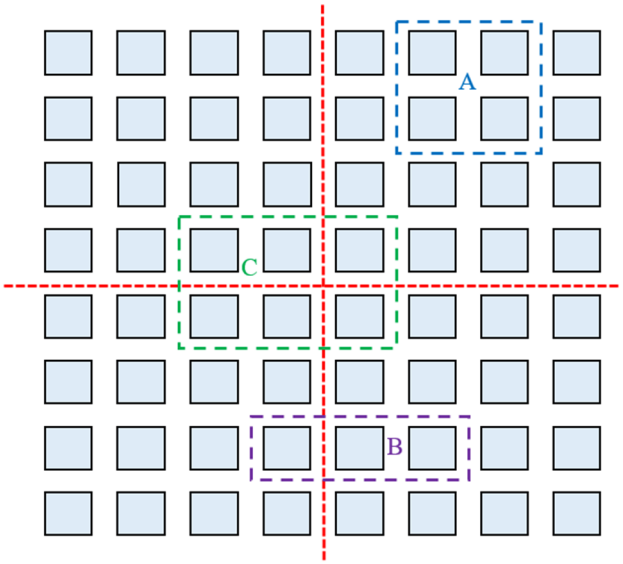


Figure 6 Several Situations for nets on the FPGAs

We calculate the timing cost according to equation (4). The first term of equation (4) is same as [5], which is the traditional timing cost function we use for the monolithic FPGAs placement. The second term InterposerDelay(i, j) is defined by equation (5).

$$Timing\_Cost = \sum_{\forall i,j \subset circuit} [delay(\Delta x_{ij}, \Delta y_{ij}) \times critically(i,j) + InterposerDelay(i,j)] \quad (4)$$

$$InterposerDelay(i,j) = \begin{cases} CrossTime_{ODP}(i,j) \times DelayIncrease_{ODP} & for\ ODP \\ CrossTime_{TDP}(i,j) \times DelayIncrease_{TDP} & for\ TDP \end{cases} \quad (5)$$

In equation (5), *DelayIncrease* of ODP is set to 1ns [11], while *DelayIncrease* of TDP is set to 1.2ns. By equation (5), we can impose more penalties on situation C, making more netlists under situation B and A.

3.4 Placer Wiring Cost

In monolithic FPGAs placement problem, we use half perimeter wire length (HPWL) of the bounding box to estimate the wire length of the current placement result. We use equation (6) to calculate the wiring cost, where the extra wiring cost is added due to the netlists cross the interposers.

$$Wiring\_Cost = \sum_{i=1}^{r} q(n_i) \times [\frac{bb_x(n_i)}{avg\_chanx\_W(n_i)} + \frac{bb_y(n_i)}{avg\_chany\_W(n_i)}] + InterposerWiringCost \quad (6)$$

Since we have separated the monolithic FPGAs vertically and horizontally, we calculate the extra wiring cost in both horizontal and vertical directions. The equation (8) is used to calculate the extra wiring cost caused by crossing interposers.

$$InterposerWiringCost = \sum_{i=1}^{r} [\frac{bb_x(n_i) \times CrossTime_y}{avg\_chanx\_W(n_i)} + \frac{bb_y(n_i) \times CrossTime_x}{avg\_chany\_W(n_i)}] \quad (7)$$

where n_i represents the netlists i, the whole circuit has r netlists, avg_chanx_W(n_i) denotes the average channel width in x direction, $bb_x(n_i)$ is the width of the bounding box, $CrossTime_y$ is the time the netlist crosses the y-directed interposers.

4. Experimental Results

In this section, we evaluate our FPGAs architecture model and the performance of our modified SA algorithm. For the architecture model, we focus on the effect of *% wires cut* and *DelayIncrease* on placement results. Then, we compare the CPU time and placement quality of SA and VFSA algorithms.

All the experiments are run with 8 MCNC beachmarks and the parameters of these beachmarks after packing are shown in Table 1. All the experiments are carried out under a Micro-Star ge60 with Intel i7-4700MQ CPU and 8GB DDR3L RAM.

Table 1 The characteristics of MCNC beachmarks.

Beachmarks	No. of netlist	No. of block	No. of CLB	No. of I/O
or1200	2481	1088	306	779
mkDelayWorker32B	5269	1668	561	1064
stereovision1	11317	1673	1357	278
stereovision0	8665	1832	1478	354
bgm	20669	3949	3649	289
stereovision2	35658	4188	3644	331
LU32PEEng	52121	9298	8882	216
mcml	56923	10896	10638	69

4.1 Architecture Model

To analyze the effect of *% wires cut*, we keep the value of *DelayIncrease* and *number of cut* constant. We set horizontal *number of cut* to 2 and set vertical *number of cut* to 1, which means we have 6 FPGAs dies. *DelayIncrease* for ODP set to 1ns, while *DelayIncrease* for TDP set to 1.2ns. The impact on maximal time delay and number of tracks is studied. We take the average of the above 8 beachmarks and use the data when *% wires cut* is equal to 0 to normalize the subsequent data.

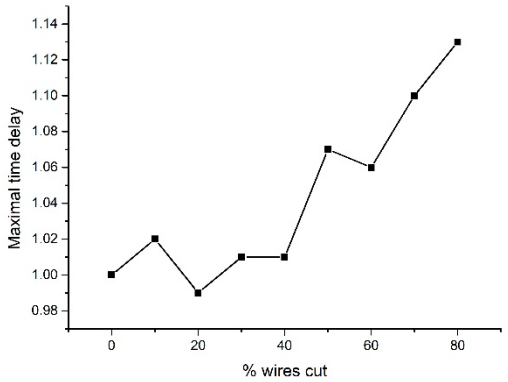


Figure 7 The impact of *% wires cut* on maximal time delay.

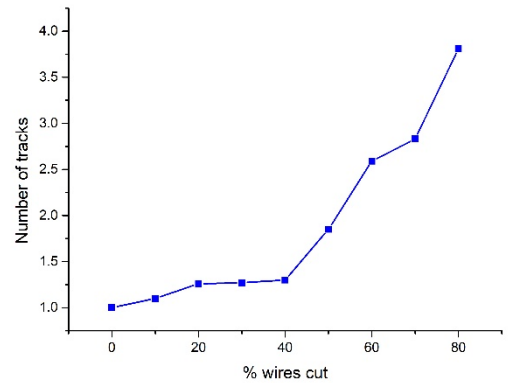


Figure 8 The impact of *% wires cut* on number of tracks.

Figure 7 demonstrates that, when *% wires cut* is less than 40, the impact of *% wires cut* on the maximum time delay is not obvious, and the maximum time delay remains almost unchanged. When *% wires cut* is greater than 40, the maximum time delay begins to increase as *% wires cut* increases, since the routing resources reduction within the interposer cannot be ignored. However, the increase is not significant, at 80% of the wire cut,

the maximal time delay rises, somehow, by about 13%. This is because router usually tries to minimize the delay and wire length of the critical path during the routing circulation, which means critical path has priority to use the tracks within the interposer.

Figure 8 shows the effect of *% wires cut* on the number of tracks. The trendency of the number of track is almost same that of maximal time delay, when *% wires cut* reaches 40, the number of tracks begins to grow. However, the growth rate of the number of tracks is far greater than the growth rate of maximal time delay. The increase of track can provide more routing resources at the interposer. When *% wires cut* reaches 90, some of the large beachmarks cannot be routed due to the large loss of routing wires.

When we explore the impact of *DelayIncrease*, we keep *% wires cut* and *number of cut* unchanged. We set *% wires cut* to 50. From Figure 9 and Figure 10, we can see that the delay setting has a significant effect on the maximum time delay, but has almost no effect on the number of tracks. In Figure 9 and 10, abscissa is the *DelayIncrease* for ODP and the *DelayIncrease* for TDP = *DelayIncrease* for ODP + 0.2ns.

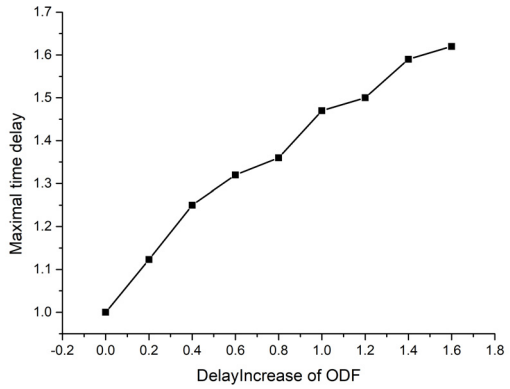


Figure 9 The impact of *DelayIncrease* on maximal time delay.

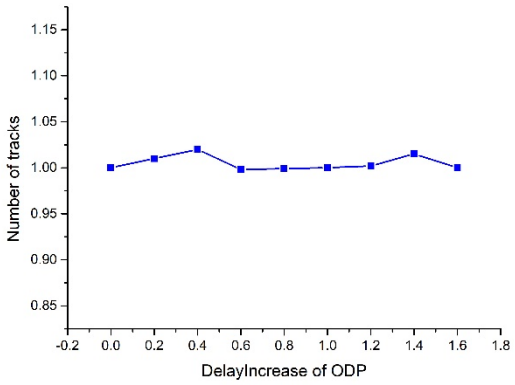


Figure10 The impact of *DelayIncrease* on number of tracks.

4.2 Placement Algorithm

We compare the performance of SA algorithm and VFSA algorithm for the placement on the interposer-based multi-FPGAs systems. From the Table 2, we can see that, although the quality of the layout is slightly degraded (the maximum delay is increased by 3.6%), the CPU time required for placement is significantly reduced by 18.7%. The main reason is that temperature variables are introduced into the perturbation model, which enables algorithm to make global search at high temperature and carry out local optimization at low temperature. The new model makes the convergence of the algorithm much more directional, which saves the computing time.

5. Conclusions

We propose a new interposer-based multi-FPGAs architecture model, cutting the monolithic FPGAs die horizontally and vertically at the same time. Subsequently, based on the new structure model, we modified the placement cost function to increase the wire cost and timing cost of the netlists crossing the interposer, especially the netlists passing through the intersection point of the interposers. In order to improve the convergence rate of the algorithm, we introduce a temperature-dependent perturbation model into the traditional SA algorithm, which greatly improves the layout speed at the slight cost of the layout quality. Finally, we discuss the influence of the architecture parameters (*% wires cut* and *DelayIncrease*) on the placement quality. We evaluate the quality from two different angles: maximal time delay and number of tracks. We find that *% wires cut* has an effect on both maximal time delay and the number of tracks, but the impact on track number is much more significant. *DelayIncrease* has almost no effect on the number of tracks but has a significant effect on the maximal time delay.

Table 2 Comparison between SA and VFSA

Beachmarks	SA			VFSA		
	No. of Swap	CPU time	Max delay	No. of Swap	CPU time	Max delay
or1200	1x	1x	1x	0.53782x	0.76974x	0.96403x
mkDelayWorker32B	1x	1x	1x	0.80252x	0.90673x	1.03773x
stereovision1	1x	1x	1x	0.67402x	0.87994x	1.04518x
stereovision0	1x	1x	1x	0.72611x	0.67149x	1.03549x
bgm	1x	1x	1x	0.53395x	0.74234x	0.99831x
stereovision2	1x	1x	1x	0.70469x	0.86859x	1.10819x
LU32PEEng	1x	1x	1x	0.54884x	0.69296x	1.02748x
mcml	1x	1x	1x	0.87221x	0.97282x	1.07206x
Average	1x	1x	1x	0.67502x	0.81307x	1.03605x

Reference

[1] Saban, Kirk. "Xilinx stacked silicon interconnect technology delivers breakthrough FPGA capacity, bandwidth, and power efficiency." Xilinx, White Paper 1 (2011): wP380.

[2] Hahn Pereira, Andre, and Vaughn Betz. "Cad and routing architecture for interposer-based multi-fpga systems." Proceedings of the 2014 ACM/SIGDA international symposium on Field-programmable gate arrays. ACM, 2014.

[3] Nasiri, Ehsan, et al. "Multiple Dice Working as One: CAD Flows and Routing Architectures for Silicon Interposer FPGAs." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 24.5 (2016): 1821-1834.

[4] Mao, Fubing, et al. "Modular placement for interposer based multi-FPGA systems." Great Lakes Symposium on VLSI, 2016 International. IEEE, 2016.

[5] Betz, Vaughn, and Jonathan Rose. "VPR: A new packing, placement and routing tool for FPGA research." International Workshop on

Field Programmable Logic and Applications. Springer, Berlin, Heidelberg, 1997.
[6] Kirkpatrick, Scott, C. Daniel Gelatt, and Mario P. Vecchi. "Optimization by simulated annealing." science 220.4598 (1983): 671-680.
[7] Ingber, Lester. "Very fast simulated re-annealing."Mathematical and computer modelling 12.8 (1989): 967-973.
[8] Ingber, Lester, and Bruce Rosen. "Genetic algorithms and very fast simulated reannealing: A comparison." Mathematical and computer modelling 16.11 (1992): 87-100.
[9] Tsallis, Constantino. "Possible generalization of Boltzmann-Gibbs statistics." Journal of statistical physics 52.1 (1988): 479-487.
[10] Penna, Thadeu JP. "Traveling salesman problem and Tsallis statistics." Physical Review E 51.1 (1995): R1.
[11] Chaware, Raghunandan, Kumar Nagarajan, and Suresh Ramalingam. "Assembly and reliability challenges in 3D integration of 28nm FPGA die on a large high density 65nm passive interposer." Electronic Components and Technology Conference (ECTC), 2012 IEEE 62nd. IEEE, 2012.
[12] Lin, Mingjie, and John Wawrzynek. "Improving FPGA placement with dynamically adaptive stochastic tunneling."IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 29.12 (2010):1858-1869.
[13] Vorwerk, Kristofer, Andrew Kennings, and Jonathan W. Greene. "Improving simulated annealing-based FPGA placement with directed moves." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 28.2 (2009): 179-192.
[14] Guo, Xiao, et al. "Fast FPGA placement Algorithm using Quantum Genetic Algorithm with Simulated Annealing." ASIC, 2009. ASICON'09. IEEE 8th International Conference on. IEEE, 2009.
[15] An, Matthew, J. Gregory Steffan, and Vaughn Betz. "Speeding up FPGA placement: Parallel algorithms and methods." Field-Programmable Custom Computing Machines (FCCM), 2014 IEEE 22nd Annual International Symposium on. IEEE, 2014.
[16] Ho, Yuan-Kai, and Yao-Wen Chang. "Multiple chip planning for chip-interposer codesign." Design Automation Conference (DAC), 2013 50th ACM/EDAC/IEEE. IEEE, 2013.