Layout driven FPGA packing algorithm for performance optimization

Linfeng Mo[1], **Chang Wu**[1], **Lei He**[1,2], **and Gengsheng Chen**[1a)]
[1] *State Key Laboratory of ASIC and System, Fudan University*
[2] *Electrical Engineering Department, University of California at Los Angeles*
a) *gschen@fudan.edu.cn*

Abstract: FPGA is a 2D array of configurable logic blocks. Packing is to pack logic elements into device specific configurable logic blocks for subsequent placement. The traditional fixed delay model of inter and intra cluster delays used in packing does not represent post-placement delays and often leads to sub-optimal solutions. This paper presents a new layout driven packing algorithm, named LDPack, based on a novel pre-packing placement for performance optimization. Our results show that after placement and routing LDPack outperforms Xilinx ISE MAP with 8% reduction in area and 5.22% smaller critical path delay, at the cost of 18% more runtime in average.

Keywords: FPGA, packing, placement, layout
Classification: Integrated circuits

References

[1] A. Marquardt, *et al.*: "Using cluster-based logic blocks and timing-driven packing to improve FPGA speed and density," ACM/SIGDA International Symposium on Field Programmable Gate Arrays (1999) 37 (DOI: 10.1145/296399.296426).
[2] G. Chen and J. Cong: "Simultaneous timing driven clustering and placement for FPGAs," International Conference on Field Programmable Logic and Applications (2004) 158 (DOI: 10.1007/978-3-540-30117-2_18).
[3] D. T. Chen, *et al.*: "Improving timing-driven FPGA packing with physical information," 2007 International Conference on Field Programmable Logic and Applications (2007) 117 (DOI: 10.1109/FPL.2007.4380635).
[4] W. Sui, *et al.*: "Physical information driven packing method in FPGA," JCIS-2008 Proc. (2008) (DOI: 10.2991/jcis.2008.20).
[5] G. Karypis, *et al.*: "Multilevel hypergraph partitioning: Applications in VLSI domain," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **7** (1999) 69 (DOI: 10.1109/92.748202).
[6] Y. W. Chang, *et al.*: "Essential issues in analytical placement algorithms," IPSJ Transactions on System LSI Design Methodology **2** (2009) 145 (DOI: 10.2197/ipsjtsldm.2.145).
[7] W. C. Naylor, *et al.*: U.S. Patent 6301693 (2001).
[8] T. C. Chen, *et al.*: "NTUplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. **27** (2008) 1228 (DOI: 10.1109/TCAD.2008.923063).

© IEICE 2017
DOI: 10.1587/elex.14.20170419
Received April 22, 2017
Accepted May 2, 2017
Publicized May 22, 2017
Copyedited June 10, 2017

[9] S. Y. Chen and Y. W. Chang: "Routing-architecture-aware analytical placement for heterogeneous FPGAs," 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC) (2015) 1 (DOI: 10.1145/2744769.2744903).

[10] K. Vorwerk and A. Kennings: "An improved multi-level framework for force-directed placement," Design, Automation and Test in Europe (2005) 902 (DOI: 10.1109/DATE.2005.59).

1 Introduction

FPGA (Field Programmable Gate Array) design flow normally consists of synthesis, packing, placement and routing. An application circuit is first synthesized into a netlist of look-up tables (LUTs), flip-flops and other logic elements. Packing is then performed to form configurable logic blocks, which is also called clusters in this paper. Normally it is to group LUTs and flip-flops together. During the placement stage, clusters are placed into legal physical positions of the FPGA device. The last routing step routes all wires between clusters with available routing resources. Packing plays an important role by determining the structures of clustered netlists and has a big impact on post-routing results.

T-VPack algorithm [1] is a well-known research work on FPGA packing. It aims to reduce critical path delay through minimizing the number of external connections on the critical paths. However, it uses fixed inter and intra cluster delays which does not represent post-placement delays and often leads to sub-optimal solutions.

In [2], Chen and Cong proposed a method to move BLEs (Basic Logic Elements composed of one LUT and one flip-flop) during placement to solve the fundamental problem of T-VPack's fixed delay model. However, due to the complex constraints on internal connections, set and reset lines, carry chains, and clock domains, especially for the complex modern FPGA architectures, the DRC (Design Rule Checking) checking becomes a runtime bottleneck [3]. [3, 4] proposed two other packing algorithms incorporating layout information into the cost function to guide cluster generation. They outperform T-VPack in both wire-length and critical path delay. However, they only use the distance information as one factor of the cost function for greedy packing. This is not good enough since it does not solve the problem of fixed inter-cluster delays directly.

In this paper, we propose a new layout driven packing algorithm LDPack, to obtain a packing solution with smaller post-routing critical path delay. Instead of the fixed inter-cluster delays in T-VPack, we use a pre-packing placement to estimate the inter-cluster delays and to guide a timing driven packing. Since the pre-packing placement needs to deal with a heterogeneous netlist with clusters and basic logic elements like LUTs and flip-flops, we propose a novel multi-density optimization based analytical placement. Notice that all the previous FPGA analytical algorithms deal with packed netlist of all clusters, our method is a very useful approach to solve heterogeneous netlist placement problem. Our experimental results show that LDPack outperforms one industrial standard tool Xilinx ISE MAP with 8% reduction in area and 5.22% smaller critical path delay after

© IEICE 2017
DOI: 10.1587/elex.14.20170419
Received April 22, 2017
Accepted May 2, 2017
Publicized May 22, 2017
Copyedited June 10, 2017

placement and routing, at a cost of 18% more runtime in average.

The rest of this paper is organized as follows. Section 2 introduces the previous work on the packing algorithm for FPGA. Section 3 presents the details of LDPack, and the experimental results are shown in Sections 4. Finally, Section 5 concludes the paper.

2 Background

As a well-known timing driven packing algorithm, T-VPack [1] groups connected LUT and flip-flop pairs to form BLEs, and packs those BLEs into clusters with bounded area. To perform the packing process, T-VPack selects a BLE as a seed and puts it into a blank cluster. Then, T-VPack grows the cluster by absorbing BLEs with the highest attraction factor. Both the seed selection and attraction factor computation are based on timing criticality and edge connections. T-VPack performs this process until all BLEs are packed into clusters.

To compute timing criticality, T-VPack assumes a delay model of fixed intra and inter cluster delays. However, after the placement of the clusters, the inter-cluster delays can vary significantly from several tens picoseconds to several nanoseconds. This fixed delay model is quite inaccurate and usually leads to sub-optimal packing solutions after placement.

There are some existing layout driven packing algorithms trying to solve this problem. In [4], the physical information is used for packing unrelated clusters and BLEs. The hMetis algorithm [5] is used to recursively bi-partition the synthesized netlist to a partitioning tree. The depth-first or breadth-first search orders of the BLEs in the partitioning tree are regarded as their coordinate values for x or y. Then the Manhattan distance between BLEs can be computed and used to guide packing in [4]. Compared to T-VPack, their results show a maximum reduction of 8.58% in wirelength.

DPack [3] uses a similar packing process as T-VPack. Besides T-VPack's cost function based on timing and connections, DPack also considers Manhattan distance information based on a placement derived by a min-cut partitioning. The results in [3] show that DPack outperforms T-VPack in wirelength and critical path delay.

However, above mentioned layout driven packing algorithms only use the distance information as a factor of their cost functions for greedy packing, which does not represent the post-placement delays. In the following, we will introduce a new layout driven packing algorithm LDPack by performing fast global placement to drive a packing process.

3 Layout driven packing (LDPack)

Fig. 1 shows the flowchart of our LDPack algorithm. The fixed packing is first performed to pack logic elements with dedicated connections into clusters to form a heterogeneous netlist with clusters and unpacked logic elements. Pre-packing placement is then to place the heterogeneous netlist. To solve the analytical placement problem on the heterogeneous netlist, we propose a multi-density optimization method which will be discussed in detail in section 3.2. At last,

© IEICE 2017
DOI: 10.1587/elex.14.20170419
Received April 22, 2017
Accepted May 2, 2017
Publicized May 22, 2017
Copyedited June 10, 2017

layout driven packing is performed to group clusters and unpacked logic elements into clusters to form a final packing solution for final placement and routing.

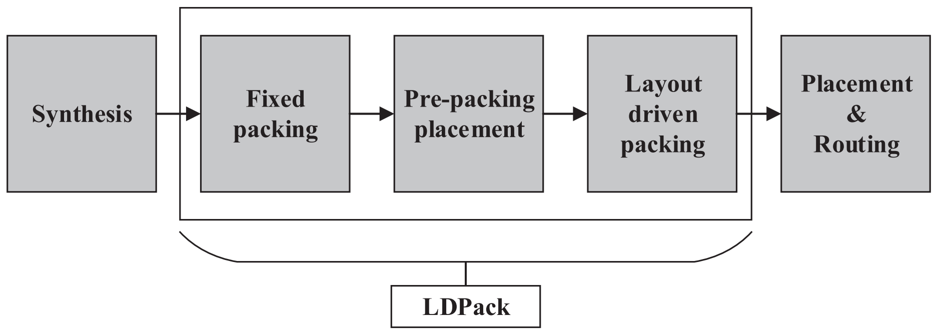

Fig. 1. Flowchart of LDPack

3.1 Fixed packing

Fixed packing is to group logic elements (i.e. LUTs, flip-flops, and etc.) with dedicated connections into clusters to avoid them being separated in subsequent steps. For example, Xilinx FPGAs have dedicated connections of MUXCYs (or CARRY4 in their Virtex-7 devices) to form fast adder chains. Obviously, we need to group those MUXCYs together with surrounding logics (like XORCYs and LUTs). They also have dedicated connections of a pair of LUTs with a special mux to form wide multiplexers. The pair of LUTs and the special mux also need to be packed together. Clearly, this kind of packing based on dedicated connections is a design rule based on target FPGA devices. We perform such fixed packing at the beginning of LDPack because it can reduce solution spaces of subsequent steps and improve runtime without affecting the final quality.

The clusters generated in the fixed packing are likely to have different sizes. For those unsaturated clusters, we allow subsequent steps to further pack logic elements into them. Notice that not all the logic elements are packed in the fixed packing step. Those LUTs and flip-flops without dedicated connections will be packed subsequently.

3.2 Pre-packing placement

Pre-packing placement is to obtain layout information for use in LDPack. We choose the analytical placement [6] engine for our pre-packing placement for its shorter runtime and better quality. Since the netlist after fixed packing is a mixture of logic elements and clusters, a multi-density optimization method is proposed in our analytical placement. Notice that previous analytical placement methods handle fully clustered netlist only. The main difficulty of heterogeneous netlist placement is that the elements have variable sizes and can be merged to form larger elements.

3.2.1 Analytical placement

In the remaining of this paper, we use cells to represent elements for placement which can be partial clusters, basic logic elements or large macros. The analytical placement mathematically addresses the placement problem as an objective func-

© IEICE 2017
DOI: 10.1587/elex.14.20170419
Received April 22, 2017
Accepted May 2, 2017
Publicized May 22, 2017
Copyedited June 10, 2017

tion with a set of constraints, and then optimizes the objective through analytical approaches.

In the circuit, each cell contains several input pins and output pins, and nets connect cells through these pins. In the analytical placement, the circuit is modeled by a hypergraph $H = (V, L)$. $V = \{v_1, v_2, \ldots, v_n\}$ is the set of cells to be placed, and $L = \{l_1, l_2, \ldots, l_n\}$ is the set of the nets that connect cells. For convenience, each net is represented by a non-empty subset of V it connects. Therefore, we say a cell belongs to a net if it is connected with the net. Obviously, a cell can belong to multiple nets. Let x_i and y_i be the x and y coordinates of the center of cell v_i. The whole chip is divided into uniform non-overlapping bin grids. Each bin grid has a fixed capacity to hold cells. The global placement problem can be formulated as a constrained minimization problem as follows:

$$\begin{aligned} &\min \quad W(\mathbf{x}, \mathbf{y}) \\ &\text{s.t.} \quad D_b(\mathbf{x}, \mathbf{y}) \le M_b, \quad \text{for each bin } b \end{aligned} \tag{1}$$

where $W(\mathbf{x}, \mathbf{y})$ is the wirelength function of the whole circuit, $D_b(\mathbf{x}, \mathbf{y})$ is the density function of bin b, and M_b is the capacity of bin b. The wirelength function $W(\mathbf{x}, \mathbf{y})$ is defined as the total half-perimeter wirelength (HPWL):

$$\begin{aligned} W(\mathbf{x}, \mathbf{y}) &= \sum_{l \in L} \big(\max_{v_i, v_j \in l} |x_i - x_j| + \max_{v_i, v_j \in l} |y_i - y_j|\big) \\ &= \sum_{l \in L} \big(\max_{v_i \in l} x_i + \max_{v_i \in l}(-x_i) + \max_{v_i \in l} y_i + \max_{v_i \in l}(-y_i)\big). \end{aligned} \tag{2}$$

Since $W(\mathbf{x}, \mathbf{y})$ is not differentiable, we apply the log-sum-exp (LSE) model [7] to approximate and smooth the max function where:

$$\max_i x_i = \lim_{\gamma \to 0} \Big(\gamma \log \sum_i \exp(x_i/\gamma)\Big). \tag{3}$$

With the LSE model, Eq. (2) can be approximated as follows:

$$\begin{aligned} \hat{W}(\mathbf{x}, \mathbf{y}) = \gamma \sum_{l \in L} \Big(&\log \sum_{v_k \in l} \exp(x_k/\gamma) + \log \sum_{v_k \in l} \exp(-x_k/\gamma) \\ &+ \log \sum_{v_k \in l} \exp(y_k/\gamma) + \log \sum_{v_k \in l} \exp(-y_k/\gamma)\Big). \end{aligned} \tag{4}$$

When γ approaches zero, the LSE wirelength $\hat{W}(\mathbf{x}, \mathbf{y})$ is close to the HPWL. Due to the computer precision, γ can only be set to a reasonably small value to avoid any arithmetic overflow during the implementation.

The density function $D_b(\mathbf{x}, \mathbf{y})$ represents the total overlap area of cells in bin b and can be expressed as:

$$D_b(\mathbf{x}, \mathbf{y}) = \sum_{v \in V} P_x(b, v) P_y(b, v) \tag{5}$$

where P_x and P_y are the overlap functions of bin b and cell v along the x and y directions. P_x is defined by

$$P_x(b, v) = \begin{cases} w_v, & 0 \le d_x \le \dfrac{w_b - w_v}{2} \\ \dfrac{w_b + w_v}{2} - d_x, & \dfrac{w_b - w_v}{2} \le d_x \le \dfrac{w_b + w_v}{2} \\ 0, & \dfrac{w_b + w_v}{2} \le d_x \end{cases} \tag{6}$$

© IEICE 2017
DOI: 10.1587/elex.14.20170419
Received April 22, 2017
Accepted May 2, 2017
Publicized May 22, 2017
Copyedited June 10, 2017

where w_b is the bin width, w_v is the cell width, and d_x is the center-to-center distance of the cell v and the bin b in the x direction. In Eq. (6), we assume that w_b is greater than w_v. Because the overlap function is not differentiable, we use the bell-shaped function [8] to smooth it. With the bell-shaped function, we get a smoothed density function $\hat{D}_b(\mathbf{x}, \mathbf{y})$.

We use the quadratic penalty method [8] to solve Eq. (1) as formulated in Eq. (7):

$$\min \quad \hat{W}(\mathbf{x}, \mathbf{y}) + \lambda \sum_{b} \max(\hat{D}_b(\mathbf{x}, \mathbf{y}) - M_b, 0)^2 \tag{7}$$

where λ is the weight of the density cost. We refer readers to [8] for how to solve Eq. (7) in iterative mode.

3.2.2 Placement of IP blocks and large macro blocks

As in [9], we separate IP (intellectual property) blocks and large macros (like BRAMs and DSPs) from regular elements and clusters in our placement stage. We first use the multilevel placement technique in [9] to pre-place those IPs and macros. The advantage of this approach is that IPs and macros can be placed much faster due to their limited candidate locations. We refer readers to [9] for details.

3.2.3 Placement of regular elements and clusters

Since the previous analytical placement only handles clustered netlist, it only needs one density function (cluster density) to represent the clusters' distribution. However, in our placement, we need to handle unpacked logic elements together with clusters, where one uniform density function is not enough to indicate all cells' distribution.

Here we propose a multi-density optimization based analytical placement. Instead of one density function used in previous algorithms, we propose two density functions of LUT and flip-flop, and combine them together to form our placement optimization problem. We call this multi-density placement. The LUT density function is defined as

$$D_b^{LUT}(\mathbf{x}, \mathbf{y}) = D_{b,LUT}^{LUT}(\mathbf{x}, \mathbf{y}) + D_{b,cluster}^{LUT}(\mathbf{x}, \mathbf{y}) \tag{8}$$

where $D_{b,LUT}^{LUT}(\mathbf{x}, \mathbf{y})$ is the LUT density function of unpacked LUTs, and $D_{b,cluster}^{LUT}(\mathbf{x}, \mathbf{y})$ is the LUT density function of pre-packed clusters, which is calculated based on the number of LUTs in those clusters. $D_{b,LUT}^{LUT}(\mathbf{x}, \mathbf{y})$ and $D_{b,cluster}^{LUT}(\mathbf{x}, \mathbf{y})$ are computed by Eq. (5). In the computation of $D_{b,cluster}^{LUT}(\mathbf{x}, \mathbf{y})$, the width and height of one cluster are defined as follows:

$$\begin{aligned} w_{cluster}^{LUT} &= \sqrt{N_{LUT}}w_{LUT} \\ h_{cluster}^{LUT} &= \sqrt{N_{LUT}}h_{LUT} \end{aligned} \tag{9}$$

where N_{LUT} is the number of LUTs in the cluster, w_{LUT} is the width of LUT, and h_{LUT} is the height of LUT. In this way, we ensure that the area of one cluster which contains N LUTs is N times the area of one LUT.

We define the size of flip-flops same as LUTs. The flip-flop density function and the size of clusters for flip-flop density computation are defined in a similar

© IEICE 2017
DOI: 10.1587/elex.14.20170419
Received April 22, 2017
Accepted May 2, 2017
Publicized May 22, 2017
Copyedited June 10, 2017

way. Like Eq. (9), we give the definition of the width and height of one cluster for flip-flop density computation below:

$$\begin{aligned} w_{cluster}^{FF} &= \sqrt{N_{FF}}w_{FF} \\ h_{cluster}^{FF} &= \sqrt{N_{FF}}h_{FF} \end{aligned} \tag{10}$$

where N_{FF} is the number of flip-flops in the cluster, w_{FF} is the width of flip-flop, and h_{FF} is the height of flip-flop.

Therefore, our placement problem is formulated as

$$\begin{aligned} \min\ \hat{W}(\mathbf{x},\mathbf{y}) + \lambda\Bigg(&\sum_b \max(\hat{D}_b^{LUT}(\mathbf{x},\mathbf{y}) - M_b^{LUT}, 0)^2 \\ &+ \sum_b \max(\hat{D}_b^{FF}(\mathbf{x},\mathbf{y}) - M_b^{FF}, 0)^2\Bigg) \end{aligned} \tag{11}$$

where λ is the weight of the density cost, M_b^{LUT} and M_b^{FF} are the maximum allowable area of LUT and flip-flop in bin b respectively. By solving Eq. (11), we are able to spread the unpacked logic elements together with clusters evenly.

Previous research work [9] states that the HPWL function Eq. (2) cannot model routed wirelength and delays well since the inter-cell delays are highly non-linear, discrete, and non-monotone with respect to the distance. Thus, we apply the routing-architecture-aware cost $R(\mathbf{x},\mathbf{y})$ in [9] to the objective function of pre-packing placement as

$$\begin{aligned} \min\ \hat{W}(\mathbf{x},\mathbf{y}) + \mu R(\mathbf{x},\mathbf{y}) + \lambda\Bigg(&\sum_b \max(\hat{D}_b^{LUT}(\mathbf{x},\mathbf{y}) - M_b^{LUT}, 0)^2 \\ &+ \sum_b \max(\hat{D}_b^{FF}(\mathbf{x},\mathbf{y}) - M_b^{FF}, 0)^2\Bigg) \end{aligned} \tag{12}$$

where μ is the weight of the routing-architecture-aware cost. $R(\mathbf{x},\mathbf{y})$ in [9] is as follows:

$$R(\mathbf{x},\mathbf{y}) = \sum_{net\ l}\left(\sum_{v_k \in l \setminus \{v_s\}} I_d(x_k - x_s, y_k - y_s)\right). \tag{13}$$

In Eq. (13), v_s is the cell driving net l, v_k is a cell driven by net l, (x_s, y_s) is the position of v_s, (x_k, y_k) is the position of v_k, and $I_d(x, y)$ is the inter-cell delay between two connected cells where x and y are the horizontal distance and vertical distance respectively. $I_d(x, y)$ is a discrete function whose domain is a two-dimensional grid due to the regularity of legal positions on FPGAs. We refer readers to [9] for the details of the computation, interpolation and smoothing of the inter-cell delays.

3.3 Layout driven packing

After fixed packing and pre-packing placement, we pack cells into clusters with the concept of Hybrid First Choice Clustering [10].

At first, layout driven packing computes attractions from Eq. (14) for every two connected cells that can be packed in one cluster. The pairs of cells are sorted from highest attraction to the lowest attraction in a packing list. Then layout driven packing chooses the pair of cells with the highest attraction to pack into one cluster. Layout driven packing repeatedly packs remaining pairs of cells until no more

© IEICE 2017
DOI: 10.1587/elex.14.20170419
Received April 22, 2017
Accepted May 2, 2017
Publicized May 22, 2017
Copyedited June 10, 2017

packing pairs can be done. Finally, unrelated packing is performed to further reduce area for cells with or without connections when the area is greater than the area of target device.

The attraction between two connected cells v_i and v_j is defined as follows:

$$Attraction(v_i, v_j) = \alpha \cdot criticality(v_i, v_j) + \beta \frac{|Nets(v_i) \cap Nets(v_j)|}{|Nets(v_i) \cup Nets(v_j)|} + (1 - \alpha - \beta)\left(1 - \frac{1}{1 + \exp(1 - dis(v_i, v_j))}\right) \tag{14}$$

where $criticality(v_i, v_j)$ is the timing criticality between v_i and v_j, $Nets(v_i)$ is the set of nets that connects to v_i, $dis(v_i, v_j)$ is the Manhattan distance between v_i and v_j, α and β are trade-off factors with their values located in [0,1].

To compute the timing criticality, we first explain the concept of edge. In the circuit, each cell contains several input pins and output pins, and nets connect cells through these pins. Each net connects one output pin and several input pins where we use edge to represent the connection between the output pin and one input pin. One edge e_k is represented by a pair of pins (p_s, p_t) where p_s is the output pin and p_t is the input pin. Since one cell contains multiple pins, there may be multiple edges connecting two cells. We define the edge set $E(v_i, v_j)$ between two cells v_i and v_j as follows:

$$E(v_i, v_j) = \{e_k | e_k = (p_s, p_t), p_s \in v_i, p_t \in v_j\} \cup \{e_k | e_k = (p_s, p_t), p_s \in v_j, p_t \in v_i\} \tag{15}$$

$criticality(v_i, v_j)$ can then be computed as follows:

$$criticality(v_i, v_j) = \sum_{e_k \in E(v_i, v_j)} \max(0, 1 - slack(e_k)) \tag{16}$$

where $slack(e_k)$ is the slack of edge e_k. If $E(v_i, v_j)$ is empty, $criticality(v_i, v_j)$ is 0. Slack [1] is defined as the amount of delay which can be added to an edge without increasing the delay of the entire circuit. $slack(e_k)$ is computed as follows:

$$slack(e_k) = T_{required}(p_t) - T_{arrival}(p_t) \tag{17}$$

where p_t is the input pin driven by e_k, $T_{required}(p_t)$ and $T_{arrival}(p_t)$ are the required time and arrival time of signal at p_t respectively. $T_{arrival}$ and $T_{required}$ can be obtained by two breadth-first traversals of the circuit. The first traversal propagates signal forward from both circuit input pins and flip-flop output pins to compute the signal delay for each pin to obtain $T_{arrival}$. The second traversal propagates signal backward from both circuit output pins and flip-flop input pins. $T_{required}$ for each pin is computed as the longest path delay (the maximum value of $T_{arrival}$) minus the signal delay. The signal delay is the sum of the delays of cells and edges in the propagation path. The delays of cells are provided by device manuals and the edge delays are computed by $I_d(x, y)$ in Eq. (13).

The pseudo code of LDPack algorithm is shown in Fig. 2.

4 Experimental results

LDPack is implemented in the C++ programming language. Our experiments are performed on a Linux server with Intel Xeon E5-2643 3.5 GHz CPU and 500 GB

© IEICE 2017
DOI: 10.1587/elex.14.20170419
Received April 22, 2017
Accepted May 2, 2017
Publicized May 22, 2017
Copyedited June 10, 2017

```
LDPack algorithm
Fixed packing;
Pre-packing placement;
Compute attractions for every two connected cells that can be packed in one cluster;
Sort cell pairs in descending order of attractions;
while attraction list is not empty do
    Get the pair (c1, c2) with the highest attraction from the attraction list;
    Form a new cluster for c1 and c2 if they are not clustered yet, or merge the two clusters of c1 and c2 if they are clustered already;
    Remove this pair (c1, c2) from the attraction list;
end do
Perform unrelated packing if area is too large;
```

Fig. 2. Pseudo code of LDPack

memory. We choose Xilinx Virtex-4 as our target FPGA and compare LDPack with industrial standard tool MAP program of ISE 13.1. Notice that, however, our algorithm is suitable for all FPGA devices like Xilinx Virtex-7 and Altera FPGAs. To work on Virtex-4 instead of newer devices is because we only have a Virtex-4 test board to verify our test designs. The benchmark circuits are all from the opencores. Table I shows the statistics of the used benchmark circuits.

Table I. Statistics of benchmark circuits

Circuits	#LUTs	#FFs	#DSPs &RAMs	#Other cells	#Nets	#I/Os
apbtoaes128	3535	924	0	554	5411	79
dmx_rx	2737	4161	0	2038	9080	82
dmx_tx	4994	4330	0	3989	13433	83
ethmac	2977	2341	4	1166	7045	211
othellogame	6178	999	8	1463	9613	16
reed_solomon_decoder	5515	2809	11	1678	10586	21
sd_card_controller	1901	1613	0	973	4960	207
sha1	2199	908	0	493	3812	74
sha256	3178	1107	0	1084	5666	74
sha512	6493	2170	0	1913	10968	76
tiny_tate_bilinear_pairing_151	3179	1326	7	252	5798	15

The benchmark circuits are first synthesized by ISE XST tool as the input of both LDPack and ISE MAP. ISE MAP is run on timing-driven mode that will run packing and placement together, followed by ISE routing. The experimental results of LDPack are obtained by running ISE PAR after LDPack.

Table II shows the values of parameters used in this paper.

Table II. Values of parameters

γ	μ	λ	w_b	h_b	w_{LUT}	h_{LUT}	w_{FF}	h_{FF}	M_b^{LUT}	M_b^{FF}	α	β
0.1	25	2^{30}	1	1	0.35	0.35	0.35	0.35	0.95	0.95	0.8	0.15

We set bin's size as one configurable logic block, which contains 8 LUTs and 8 flip-flops. The bin's width and height are both set to 1, and the LUT's size and flip-flop's size are both set to be 0.35×0.35 so that the area of one bin equals the area of 8 LUTs or flip-flops.

© IEICE 2017
DOI: 10.1587/elex.14.20170419
Received April 22, 2017
Accepted May 2, 2017
Publicized May 22, 2017
Copyedited June 10, 2017

In Eq. (14), we test α from 0.9 to 0.5, and β from 0.05 to 0.45 with the step of 0.05. $(\alpha + \beta)$ is kept below 1 to ensure that the weight of distance cost is positive. Timing cost is given the biggest weight to be the leading optimization goal. We find that the timing results are best when α and β are set to 0.8 and 0.15 respectively.

The experimental results are shown in Table III, where the column Area lists the number of slices after routing, column "Critical Path Delay" lists the post-routing critical path delay in nano seconds, and column Runtime is the total CPU time of packing, placement and routing in seconds.

Table III. Comparison of post-routing results between ISE MAP and LDPack

Circuits	Area (#slice)		Critical Path Delay (ns)		Runtime (sec)	
	ISE MAP	LDPack	ISE MAP	LDPack	ISE flow	LDPack flow
apbtoaes128	2104	1858	7.6	7.2	144	155
dmx_rx	4988	4858	5.3	5.1	92	121
dmx_tx	5816	4840	5.2	5.0	90	148
ethmac	2831	2781	5.3	5.1	71	73
othellogame	3388	3257	13.3	12.7	126	155
reed_solomon_decoder	3555	3096	5.3	4.8	145	134
sd_card_controller	2024	1830	3.4	3.3	31	46
sha1	1566	1288	9.2	8.7	64	68
sha256	1963	1858	10.6	10.2	38	42
sha512	3684	3623	12.1	11.4	251	249
tiny_tate_bilinear_pairing_151	1867	1751	5.1	4.7	66	75
Average	1.00	0.92	1.00	0.95	1.00	1.18

From Table III, it can be seen that the LDPack achieves an average reduction of 8% area compared to ISE MAP. LDPack also outperforms ISE MAP with 5.22% smaller critical path delay, while the total runtime is 18% longer.

5 Conclusions

In this paper, we propose a new layout driven packing algorithm LDPack. We propose a new multi-density optimization based analytical placement to deal with the heterogeneous netlist of clusters together with basic logic elements. With the obtained timing and distance information, we pack all the logic elements and clusters into clusters based on the concept of Hybrid First Choice Clustering. Our experimental results show that LDPack outperforms ISE MAP with 8% reduction in area and 5.22% smaller critical path delay after placement and routing, at the cost of 18% more runtime in average. In the future, we will work on reducing the runtime.

Acknowledgments

The authors wish to acknowledge the funding support by Shanghai Fudan Microelectronics Group Co., Ltd.

© IEICE 2017
DOI: 10.1587/elex.14.20170419
Received April 22, 2017
Accepted May 2, 2017
Publicized May 22, 2017
Copyedited June 10, 2017