Improving the Gradient Descent Based FPGA-Placement Algorithm

Tobias Thiemann  Timm Bostelmann  Sergei Sawitzki

FH Wedel
University of Applied Sciences
Contact: bos@fh-wedel.de

CENICS 2020
Presenter’s Resume

**Timm Bostelmann** received his engineer’s degree in computer engineering from the FH Wedel (University of Applied Sciences) in 2008. Since then, he is employed at FH Wedel as a research assistant in the field of embedded systems. In addition, he is working towards his PhD degree at the TU Dresden (University of Technology) in the field of reconfigurable architectures.
FPGA Complexity is Rising

Year
Configuration size/Kbit
10^1 10^2 10^3 10^4 10^5 10^6 10^7
XC2018
XC3090
XC4025E
XC4013E
XC4VLX200
XCV3200E
XCV1000
XC2V8000
XC4028EX
XCVU440
XC6VHX565T
XC7V2000T
XCV1000
XCV3200E
XC4VLX200
XC7V2000T
XCVU440
XC6VHX565T
FPGA Complexity is Rising

EDA Tools have to handle this complexity

Synthesis → Technology-Mapping → Placement → Routing → Bitstream-Generation
FPGA Complexity is Rising

EDA Tools have to handle this complexity
Netlist Placement for FPGAs

Netlist

Architecture

I/O-Cell

Logic-Cell
Netlist Placement for FPGAs

Placement — Illegal Positions
Netlist Placement for FPGAs

Placement — Illegal Types

- The cells are in the grid …
- …but the cell types are not compatible.
Netlist Placement for FPGAs

Placement — Legal

- The cells are in the grid …
- … and the cell types are compatible …
Netlist Placement for FPGAs

Placement — Legal

- The cells are in the grid . . .
- . . . and the cell types are compatible . . .
- . . . but the performance will be poor.
The cells are in the grid . . .

. . . and the cell types are compatible . . .

. . . and the performance will be good.
## Problem Description

Select a resource cell on the target FPGA for every cell of the given netlist in a way that:

1. Every cell of the netlist is assigned to a resource cell of the fitting type (e.g. IO, CLB, DSP)
2. No resource cell is occupied by more than one cell of the netlist
3. The cells are arranged in a way that allows the best possible routing

### Established Solutions

- Iterative algorithms like Simulated Annealing
- Constructive algorithms like min-cut (recursive partitioning)
- Analytical placement
Netlist Placement for FPGAs

Problem Description

Select a resource cell on the target FPGA for every cell of the given netlist in a way that:

1. Every cell of the netlist is assigned to a resource cell of the fitting type (e.g. IO, CLB, DSP)
2. No resource cell is occupied by more than one cell of the netlist
3. The cells are arranged in a way that allows the best possible routing

Established Solutions

- Iterative algorithms like Simulated Annealing
- Constructive algorithms like min-cut (recursive partitioning)
- Analytical placement
Previous Work

Fast FPGA-Placement Using a Gradient Descent Based Algorithm

- Achieved similar results to the reference (based on simulated annealing) regarding the bounding-box quality
- Is on average 3.8 times faster than the reference
- Results in a significantly longer critical path
- Is working single threaded
Previous Work

Fast FPGA-Placement Using a Gradient Descent Based Algorithm

- Achieved similar results to the reference (based on simulated annealing) regarding the bounding-box quality
- Is on average 3.8 times faster than the reference
- Results in a significantly longer critical path
- Is working single threaded

This Work

- Different approaches to reduce the length of the critical path are evaluated
- Different approaches to reduce the runtime (including parallelization) are evaluated
- Extensive benchmarking
General Approach

- Measure the quality of the placement with a cost function
- Move all nodes towards the steepest gradient descent
- Legalize the placement
- Repeat optimization and legalization in a loop
General Approach

- Measure the quality of the placement with a cost function
- Move all nodes towards the steepest gradient descent
- Legalize the placement
- Repeat optimization and legalization in a loop

Cost Function

An exponential function over the distance between the position of the node and the bounding-box of the net is chosen as basis of the cost-function:

\[
C_k = \alpha_2 \cdot \sum_{n \in N_k} \left( e^{\alpha_1 \cdot (x_k - \max_x(n))} + e^{\alpha_1 \cdot (\min_x(n) - x_k)} + \\
              e^{\alpha_1 \cdot (y_k - \max_y(n))} + e^{\alpha_1 \cdot (\min_y(n) - y_k)} \right)
\]
Cost-Gradient

Plot of the gradient for the X coordinate of a node, assuming a net with the boundaries $min_x = 1$ and $max_x = 7$:

\[
\begin{align*}
\alpha_1 &= 1 \\
\alpha_1 &= 2 \\
\alpha_1 &= 3 \\
\alpha_1 &= 4
\end{align*}
\]
## Legalization

<table>
<thead>
<tr>
<th>Illegal Placement</th>
<th>Legal Placement</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Illegal Placement" /></td>
<td><img src="image2.png" alt="Legal Placement" /></td>
</tr>
</tbody>
</table>
### Placement Phases

1. **Presorting (5000 iterations)**  
   high step width, weak legalization

2. **Grid placement (1000 iterations)**  
   high step width, stronger legalization

3. **Initial detailed placement (1000 iterations)**  
   reduced step width

4. **Detailed placement (5000 iterations)**  
   reduced optimization step width

5. **Final placement (100 iterations)**  
   no optimization, only legalization
### Evaluated Approaches

1. Utilization of multithreading  
   The algorithm was profiled and a parallelized implementation was derived

2. Improvement of the initial placement  
   The initial placement was generated with a min-cut approach instead of a random initialization

3. Improvement of the critical path  
   A path metric was introduced to favor nodes on long paths

4. Optimization of the parameters  
   The parameters of the algorithm were optimized using an artificial neural network
Benchmarking Setup

- The original gradient algorithm (GPO), the new gradient algorithm (GPN) and simulated annealing (VPR) are compared.
- All measurements are done for twenty common netlists.
- Non deterministic values are averaged over ten measurements.
Benchmarking Setup

- The original gradient algorithm (GPO), the new gradient algorithm (GPN) and simulated annealing (VPR) are compared.
- All measurements are done for twenty common netlists.
- Non deterministic values are averaged over ten measurements.

Measurement Series

1. Bounding-Box Costs
2. Critical Path
3. Runtime
Comparison of the Bounding-Box Costs

Average (bbcost (GPN)) × 100) = 100.57%
Average (bbcost (GPN)) × 100) = 98.73%

Netlists

Bounding-Box Costs
Comparison of the Bounding-Box Costs

Average\left(\frac{\text{bbcost}(\text{GPN})}{\text{bbcost}(\text{VPR})}\times100\%\right) = 100.57\%

Average\left(\frac{\text{bbcost}(\text{GPN})}{\text{bbcost}(\text{GPO})}\times100\%\right) = 98.73\%
Comparison of the Critical Path

- **Average** (\(\text{critpath}(\text{GPN})\)) \(\times 100\%) = 121.14\%\)
- **Average** (\(\text{critpath}(\text{GPO})\)) \(\times 100\%) = 84.00\%\)

Netlists:
- ex5p
- tseng
- apex4
- misex3
- alu4
- diffeq
- dsip
- seq
- apex2
- s298
- des
- bigkey
- frisc
- spla
- elliptic
- ex1010
- pdc
- s38417
- s38584.1
- clma

Critical Path / ns:
- 0
- 50
- 100
- 150
- 200
- 250
- 300
- 350

Graph characteristics:
- VPR
- GPO
- GPN
Comparison of the Critical Path

Average \left( \frac{\text{critpath}(\text{GPN})}{\text{critpath}(\text{VPR})} \times 100 \% \right) = 121.14 \%

Average \left( \frac{\text{critpath}(\text{GPN})}{\text{critpath}(\text{GPO})} \times 100 \% \right) = 84.00 \%
Comparison of the Runtime

Runtime / s

Netlists

e5x
lseng
apex4
misex3
alu4
diffreq
dsp
seq
apex2
s298
des
bigkey
frisc
spla
elliptic
ex1010
pdc
s38417
s38584.1
clma

Average (runtime (GPN) / runtime (VPR)) × 100% = 19.69%

Average (runtime (GPN) / runtime (GPO)) × 100% = 46.39%
Comparison of the Runtime

Average \( \left( \frac{\text{runtime}(GPN)}{\text{runtime}(VPR)} \right) \times 100 \% \) = 19.69 %

Average \( \left( \frac{\text{runtime}(GPN)}{\text{runtime}(GPO)} \right) \times 100 \% \) = 46.39 %
Conclusion

- The new gradient algorithm is about 5 times as fast as VPR and more than two times as fast as the original gradient algorithm.
- The bounding box quality is about equal for all three algorithms.
- That critical path of the new gradient algorithm is about 20% longer compared to VPR and about 16% shorter compared to the original gradient algorithm.
- Extended benchmarking with even larger netlists might underline the scalability of the approach.
Conclusion

- The new gradient algorithm is about 5 times as fast as VPR and more than two times as fast as the original gradient algorithm.
- The bounding box quality is about equal for all three algorithms.
- That critical path of the new gradient algorithm is about 20% longer compared to VPR and about 16% shorter compared to the original gradient algorithm.
- Extended benchmarking with even larger netlists might underline the scalability of the approach.

Thank you for your attention!