DSE Introduction

Luigi Nardi
Assistant Professor Lund University
Researcher Stanford University

@PLDI DSE Tutorial

June 15, 2020
Main Papers Behind This Talk

• Goal of Spatial [Koeplinger, et al.]: design of application accelerators
• On reconfigurable architectures FPGAs and CGRAs
• Spatial compiler lowers user programs into synthesizable Chisel [Bachrach, et al.]

The Spatial Compiler

- Goal of Spatial [Koeplerger, et al.]: design of application accelerators
- On reconfigurable architectures FPGAs and CGRAs
- Spatial compiler lowers user programs into synthesizable Chisel [Bachrach, et al.]

• Goal of Spatial [Koeplinger, et al.]: design of application accelerators
• On reconfigurable architectures FPGAs and CGRAs
• Spatial compiler lowers user programs into synthesizable Chisel [Bachrach, et al.]

The Spatial Compiler

- Goal of Spatial [Koeplinger, et al.]: design of application accelerators
- On reconfigurable architectures FPGAs and CGRAs
- Spatial compiler lowers user programs into synthesizable Chisel [Bachrach, et al.]

The Spatial Compiler

Unrolls loops, retimes pipelines, and performs on-chip memory layout. The optimizations are computed based on the analyses of the previous phase.

Legend

Intermediate Representation
Design Parameters

- IR Transformation
- IR Analysis
- Code Generation

- Goal of Spatial [Koeplinger, et al.]: design of application accelerators
- On reconfigurable architectures FPGAs and CGRAs
- Spatial compiler lowers user programs into synthesizable Chisel [Bachrach, et al.]

The Spatial Compiler

• Goal of Spatial [Koeplinger, et al.]: design of application accelerators
• On reconfigurable architectures FPGAs and CGRAs
• Spatial compiler lowers user programs into synthesizable Chisel [Bachrach, et al.]

[Koeplinger, et al.]: "Spatial: a language and compiler for application accelerators." PLDI 2018
The Spatial Compiler

- Goal of Spatial [Koeplinger, et al.]: design of application accelerators
- On reconfigurable architectures FPGAs and CGRAs
- Spatial compiler lowers user programs into synthesizable Chisel [Bachrach, et al.]

The Spatial Compiler

HyperMapper

Legend

- Intermediate Representation
- Design Parameters
- Design Space Exploration
- Code Generation
- IR Analysis
- IR Transformation

User Parameters

Control Inference
Control Scheduling
Access Pattern Analysis
Mem. Banking/Buffering
Area/Runtime Analysis

HyperMapper DSE
Pipeline Unrolling
Pipeline Retiming
Host Resource Allocation
Control Signal Inference
Chisel Code Generation

Modified Parameters

Host Resource Allocation
Control Signal Inference
Search Space - Optimization Knobs

Input
The Spatial compiler automatically provides the following parameters:
• Tile size (ordinal)
• Inner and outer loop pipelining (ordinal)
• Meta-pipe (categorical)
• Unrolling factor (ordinal)
• Memory banking (ordinal)
• Parallelism (categorical)

Output
The compiler evaluation provides:
• Minimize Clock cycles (runtime): objective 1
• Minimize FPGA logic utilization: objective 2
  • Useful for fitting multiple applications on the same FPGA
  • Proxy for energy consumption
• Feasibility constraint:
  • true if design fits in the chip
## Spatial Examples
- Search Spaces -

<table>
<thead>
<tr>
<th>Application</th>
<th># Parameters</th>
<th>Space Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlackScholes</td>
<td>4</td>
<td>7.68 × 10^4</td>
</tr>
<tr>
<td>OuterProduct</td>
<td>5</td>
<td>1.66 × 10^7</td>
</tr>
<tr>
<td>DotProduct</td>
<td>5</td>
<td>1.18 × 10^8</td>
</tr>
<tr>
<td>K-Means</td>
<td>6</td>
<td>1.04 × 10^6</td>
</tr>
<tr>
<td>GEMM</td>
<td>13</td>
<td>2.9 × 10^7</td>
</tr>
<tr>
<td>TPC-H Q6</td>
<td>5</td>
<td>3.54 × 10^9</td>
</tr>
<tr>
<td>GDA</td>
<td>9</td>
<td>2.40 × 10^{11}</td>
</tr>
<tr>
<td>Shallow CNN</td>
<td>7</td>
<td>1.2 × 10^6</td>
</tr>
<tr>
<td>Deep CNN</td>
<td>7</td>
<td>1.2 × 10^6</td>
</tr>
<tr>
<td>MD Grid</td>
<td>10</td>
<td>1.6 × 10^9</td>
</tr>
</tbody>
</table>
• Benchmark: SLAMBench 1.0 runtime response surface is: non-linear, multi-modal and non-smooth
Design Space Exploration (DSE)  
3-parameters and 2-objectives - Pictorial

Input space  
(a.k.a. search or design space)
Design Space Exploration (DSE)
3-parameters and 2-objectives - Pictorial

**Input space**
(a.k.a. search or design space)

**Output space**
(a.k.a. objective space)
Design Space Exploration (DSE)
3-parameters and 2-objectives - Pictorial

Input space
(a.k.a. search or design space)

Output space
(a.k.a. objective space)
Design Space Exploration (DSE)
3-parameters and 2-objectives - Pictorial

Input space
(a.k.a. search or design space)

Output space
(a.k.a. objective space)
Design Space Exploration (DSE)
3-parameters and 2-objectives - Pictorial

Input space
(a.k.a. search or design space)

Output space
(a.k.a. objective space)
Design Space Exploration (DSE)
3-parameters and 2-objectives - Pictorial

**Input space**
(a.k.a. search or design space)

**Output space**
(a.k.a. objective space)
Design Space Exploration (DSE)
3-parameters and 2-objectives - Pictorial
Design Space Exploration (DSE)
3-parameters and 2-objectives - Pictorial
Design Space Exploration (DSE)
3-parameters and 2-objectives - Pictorial

Input space
(a.k.a. search or design space)

Output space
(a.k.a. objective space)

Feasibility region
Practical DSE: Important Features

1. Real, integer, ordinal and categorical variables (RIOC var.)
   
   • Example: tile size is an ordinal, parallelism is a categorical
Practical DSE: Important Features

1. Real, integer, ordinal and categorical variables (RIOC var.)
   - Example: tile size is an ordinal, parallelism is a categorical

2. User prior knowledge in the search (Prior)
   - Example: # threads on a CPU with 4 cores? Gaussian-shaped around 6
Practical DSE: Important Features

1. Real, integer, ordinal and categorical variables (RIOC var.)
   - Example: tile size is an ordinal, parallelism is a categorical

2. User prior knowledge in the search (Prior)
   - Example: # threads on a CPU with 4 cores? Gaussian-shaped around 6

3. Unknown feasibility constraints (Constr.)
   - Example: does a design fit in the FPGA?
Practical DSE: Important Features

1. Real, integer, ordinal and categorical variables (RIOC var.)
   - Example: tile size is an ordinal, parallelism is a categorical

2. User prior knowledge in the search (Prior)
   - Example: # threads on a CPU with 4 cores? Gaussian-shaped around 6

3. Unknown feasibility constraints (Constr.)
   - Example: does a design fit in the FPGA?

4. Multi-objective optimization (Multi)
   - Example: trade-off runtime and area
None of the tools available support all these DSE features

We introduce a new framework dubbed **HyperMapper**

<table>
<thead>
<tr>
<th>Name</th>
<th>Multi</th>
<th>RIOC var.</th>
<th>Constr.</th>
<th>Prior</th>
</tr>
</thead>
<tbody>
<tr>
<td>GpyOpt</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>OpenTuner</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>SURF</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>SMAC</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Spearmint</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Hyperopt</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Hyperband</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>GPflowOpt</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>cBO</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>BOHB</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td><strong>HyperMapper</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

# DSE Solutions

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Manual optimization</td>
<td>Human expert needed</td>
</tr>
</tbody>
</table>
## DSE Solutions

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Manual optimization</td>
<td>Human expert needed</td>
</tr>
<tr>
<td>• Automated but architecture-dependent</td>
<td>Doesn’t generalize</td>
</tr>
</tbody>
</table>
## DSE Solutions

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Manual optimization</td>
<td>Human expert needed</td>
</tr>
<tr>
<td>• Automated but architecture-dependent</td>
<td>Doesn’t generalize</td>
</tr>
<tr>
<td>• Grid search</td>
<td>Unfeasible most of the times</td>
</tr>
</tbody>
</table>
## DSE Solutions

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Manual optimization</td>
<td>Human expert needed</td>
</tr>
<tr>
<td>• Automated but architecture-dependent</td>
<td>Doesn’t generalize</td>
</tr>
<tr>
<td>• Grid search</td>
<td>Unfeasible most of the times</td>
</tr>
<tr>
<td>• Random search</td>
<td>Inefficient</td>
</tr>
</tbody>
</table>
## Approaches

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual optimization</td>
<td>Human expert needed</td>
</tr>
<tr>
<td>Automated but architecture-dependent</td>
<td>Doesn’t generalize</td>
</tr>
<tr>
<td>Grid search</td>
<td>Unfeasible most of the times</td>
</tr>
<tr>
<td>Random search</td>
<td>Inefficient</td>
</tr>
<tr>
<td>Evolutionary Algorithms</td>
<td>High sampling budget</td>
</tr>
</tbody>
</table>

**DSE Solutions**
# DSE Solutions

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Manual optimization</td>
<td>Human expert needed</td>
</tr>
<tr>
<td>• Automated but architecture-dependent</td>
<td>Doesn’t generalize</td>
</tr>
<tr>
<td>• Grid search</td>
<td>Unfeasible most of the times</td>
</tr>
<tr>
<td>• Random search</td>
<td>Inefficient</td>
</tr>
<tr>
<td>• Evolutionary Algorithms</td>
<td>High sampling budget</td>
</tr>
<tr>
<td>• Bayesian Optimization (BO)</td>
<td>Sampling efficient</td>
</tr>
</tbody>
</table>
Diversity of Optimizers in HyperMapper

What type of algorithm to choose for the DSE?
• Rule of thumb: it depends on the sampling budget

Direct algorithms:
• Random sampling
• Multi-start local search
• Evolutionary Algorithms

Model-based algorithms:
• Bayesian optimization

---

# of samples
DSE Mono-objective Formulation
- Formal Problem Setting -

Find a global minimizer of an unknown objective function $f(x)$ under a set of $q$ unknown feasibility constraint functions $c_i(x)$:

$$x^* = \arg\min_{x \in X} f(x)$$

subject to $c_i(x) \leq b_i, \ i = 1, \ldots, q,$

where $X$ is some space of interest.

In addition:
1. The objective function $f(x)$ and the constraints $c_i(x)$:
   - Have no simple closed form (e.g., software)
   - Can be evaluated at any point $x$ but evaluation is expensive (time)
2. Gradients are unavailable (or available at prohibitive cost)
Optimization: Illustration

The model in this example is a Gaussian Process (GP)
Optimization: Illustration

The model in this example is a Gaussian Process (GP)
Optimization: Illustration

The model in this example is a Gaussian Process (GP)
Optimization: Illustration

The model in this example is a Gaussian Process (GP)
The model in this example is a Gaussian Process (GP)
Optimization: Illustration

The model in this example is a Gaussian Process (GP)
Optimization: Illustration

The model in this example is a Gaussian Process (GP)
Optimization: Illustration

The model in this example is a Gaussian Process (GP)
Optimization: Illustration

The model in this example is a Gaussian Process (GP)
Optimization: Illustration

The model in this example is a Gaussian Process (GP)
Optimization: Illustration

The model in this example is a Gaussian Process (GP)
Optimization: Illustration

The model in this example is a Gaussian Process (GP)
Optimization: Illustration

The model in this example is a Gaussian Process (GP)
Optimization: Illustration

The model in this example is a Gaussian Process (GP)
Optimization: Illustration

The model in this example is a Gaussian Process (GP)
Optimization: Illustration

The model in this example is a Gaussian Process (GP)
Optimization: Illustration

The model in this example is a Gaussian Process (GP)
Optimization: Illustration

The model in this example is a Gaussian Process (GP)
Optimization: Illustration

The model in this example is a Gaussian Process (GP)
Optimization: Illustration

The model in this example is a Gaussian Process (GP)
Optimization: Illustration

The model in this example is a Gaussian Process (GP)

This optimization process is known as Bayesian Optimization
The HyperMapper Framework

Input

Random Samples

Search Space

Optimize the model

Supervised Learning

Optimizer (Bayesian Optimization)

Evaluate Predicted Optimum

Probabilistic Model

Optimum (Incumbent)

Output
Spatial Results
- Real-world Applications -

**Shallow CNN**
- Cycles (log) vs. Number of Evaluations

**Deep CNN**
- Cycles (log) vs. Number of Evaluations

**MD Grid**
- Cycles (log) vs. Number of Evaluations

- Green line: Random Sampling
- Red line: HyperMapper
- Black line: Expert Configuration
- Dashed line: Initialization
Spatial Results
- Real-world Applications -

**Shallow CNN**

- Number of Evaluations vs. Cycles (log)
- 2.74x faster than expert

**Deep CNN**

- Number of Evaluations vs. Cycles (log)

**MD Grid**

- Number of Evaluations vs. Cycles (log)

- Random Sampling
- HyperMapper
- Expert Configuration
- Initialization
Spatial Results
- Real-world Applications -

**Shallow CNN**
- 2.74x faster than expert

**Deep CNN**
- 10.4x faster than expert

**MD Grid**
- Random Sampling
- HyperMapper
- Expert Configuration
- Initialization

Luigi Nardi, Ph.D. - Lund/Stanford
**Spatial Results**  
- Real-world Applications -

- **Shallow CNN**
- **Deep CNN**
- **MD Grid**

- **Shallow CNN**
  - Cycles (log): 24,154,953 to 4,197,501
  - Number of Evaluations: 0 to 50

- **Deep CNN**
  - Cycles (log): 24,154,953 to 729,416
  - Number of Evaluations: 0 to 50

- **MD Grid**
  - Cycles (log): 3,269,017 to 22,026
  - Number of Evaluations: 0 to 140

- **2.74x faster than expert**
- **33x faster than RS**
- **10.4x faster than expert**

Legend:
- Random Sampling
- HyperMapper
- Expert Configuration
- Initialization
Summary

- Introduced the problem of Design Space Exploration (DSE)
  - Mono-objective
  - Will follow:
    - Multi-objective, priors, constrained optimization
    - Details of Spatial/DSE
Summary

• Introduced the problem of Design Space Exploration (DSE)
  • Mono-objective
  • Will follow:
    • Multi-objective, priors, constrained optimization
    • Details of Spatial/DSE
  • A general solution to DSE
    • Widely applicable @PLDI
    • Interdisciplinary research work (ML/Math/PL/Systems)
    • Introduced HyperMapper, an umbrella framework for several optimizers:
      • Bayesian optimization (efficient), RS and heuristics
Summary

• Introduced the problem of Design Space Exploration (DSE)
  • Mono-objective
  • Will follow:
    • Multi-objective, priors, constrained optimization
    • Details of Spatial/DSE
• A general solution to DSE
  • Widely applicable @PLDI
  • Interdisciplinary research work (ML/Math/PL/Systems)
  • Introduced HyperMapper, an umbrella framework for several optimizers:
    • Bayesian optimization (efficient), RS and heuristics
• Use-case: real-world application Spatial
Info on HyperMapper

- Join HyperMapper on **Slack**: hypermapper.slack.com
- **Repo**: [https://github.com/luinardi/hypermapper](https://github.com/luinardi/hypermapper)
- **Wiki**: [https://github.com/luinardi/hypermapper/wiki](https://github.com/luinardi/hypermapper/wiki)

Adopters

- Microsoft
- SENTIAN.AI
- Stanford University
- UC San Diego
- TEXAS
- Imperial College London

Database Management Systems
Automated Machine Learning
Hardware and Network Design
FPGAs
Approximate Computing
Computer Vision and Robotics
Etc.