Final Year Project

High-level Synthesis Design Space Exploration

Supervisor: Dr. Benjamin Carrion Schafer
Student: LU Shanqi 13116151D
Content

• Part1: Introduction
  – High-level Synthesis

• Part2: Methodology
  – Tool CyberWorkBench
  – Algorithm

• Part3: Graphical User Interface
  – Functions
  – Plotting widget

• Part4: Results
  – Comparison between brute force and simulated annealing
High-level Synthesis

• A process of converting behavior descriptions to hardware implementation
• From SystemC to hardware
• Tools: CyberWorkBench, Xilinx Vivado....
Higher Abstract Level

• Advantages
  – shorter marketing cycle
  – Increasing reusability of programming codes

• Disadvantages
  – Numerous synthesis options are available, hence hard to find optimal designs
Design Space Exploration

- An exploration process aiming to find optimal designs among uncountable candidates in high level synthesis
- Multi-objective optimization problem

- One design pareto dominate another.
- One design is not inferior to another design in all objectives, additionally, there is at least one better objective.
Objectives of this project

• To automate the process of design space exploration
• To develop a heuristic to accelerate design space exploration
• To develop a graphical user interface (GUI) to plot results dynamically
Tool: CyberWorkBench

- Parse description language (e.g. SystemC)
  - Special pragmas can be recognized by parser
- Generate constraint files (e.g. Functional Units constraint file)
- Synthesize
  - Results are store in a *.CSV file
Tool: CyberWorkBench

- Pragma insertion
- Key word: Cyber

```c
/*Cyber unroll_times = all*/
for(int i=0; i<3; i++)
{
    array[i] = i;
}
```

- Change high-level synthesis options (also called attributes) by inserting such kind of pragmas

```c
array[0] = 0;
array[1] = 1;
array[2] = 2;
```
Tool: CyberWorkBench

variable x1

function

values

inline

goto

variable x2

array

values

RAM

REG

ROM

variable x3

for loop

values

unroll all

unroll 2

unroll 0

Prart2: Methodology
Problem clarification

- Treat high-level synthesis process as a black box function
  \[(Area, \text{latency}) = f(x_1, x_2, x_3, \ldots)\]
- Multiple input variables: \(x_1, x_2, x_3, \ldots\)
- Two objectives: area, latency
- Find more pareto optimal designs in shorter time
Problem clarification

\[
(Area, latency) = f(x_1, x_2, x_3 \ldots)
\]

high-level synthesis process

synthesis results
design curve
Program structure

1. Input C file
2. Examining and inserting pragmas
3. Generating a header file which defines preprocessor macros
4. Synthesizing
5. Displaying results dynamically
6. Showing final results including optimal designs and time information
Algorithms

• Brute Force (exhaustion method)
  – a generate-and-test algorithm to check all possible candidates that satisfy specification of a problem

• Simulated Annealing
  – probabilistic heuristic
  – other heuristics (e.g. genetic algorithm)
Simulated Annealing

• Step1: Generate an initial design randomly as the base design and synthesize it. Synthesized result is used to calculate first GCF as base state of system. Set an initial temperature for the system.

\[ GCF = \partial A + \beta L \]

- design1: array=RAM, loop=unroll all, function=inline, ...

• Global cost function is defined as:
Simulated Annealing

• Step 2: Generate a new design from base design by randomly modifying one attribute's value.

- Initial design:
  - \( x_1 \): array=RAM
  - \( x_2 \): loop=unroll all
  - \( x_3 \): function=inline ...

- New design:
  - \( x_1 \): array=RAM
  - \( x_2 \): loop=unroll 0
  - \( x_3 \): function=inline ...

• Step 3: Compare new GCF and previous GCF. Then, determine whether to accept the new design. Probability to accept a worse design:

\[
P = e^{-\frac{\Delta GCF}{k}}
\]
Simulated Annealing

• Step 4: If 5 better designs are consecutively generated, reduce current temperature by 10%. Change parameters in GCF for every 8 designs.

• If no exit condition is met, iterate from step 2 to step 4. Exit iteration if one exit condition is met.
Simulated Annealing

- Exit conditions:
  
  i. Current temperature is less than threshold.
  
  ii. Consecutively, more than 5 new designs are worse than previous design.
  
  iii. Cannot generate new designs by changing one of the attributes.
  
  iv. Synthesized designs are more than 70% of all designs.
Simulated Annealing

• Why SA?

\[(Area, \text{latency}) = f(x_1, x_2, x_3, \ldots)\]

• Some input sequence might lead to a good result. e.g.

array=RAM \quad \text{loop=unroll all} \quad \rightarrow \quad \text{small area and small latency}

• Change one attribute at a time while maintaining other attribute combinations
Graphical User Interface for Design Space Exploration
Part 3: Graphical User Interface
Qt Framework

• A cross-platform framework for developing applications
• Signals and Slots: Communication mechanism between different parts of the program
• QCustomPlot: An online open source widget for plotting
• Multithread programming: To prevent GUI freezing
• Model/View Programming: To modify data outside current program (used in file list)
Functions of the developed GUI

• ComboBox
• Selecting files or options
Functions of the developed GUI

- Selecting other benchmarks
- Selecting other libraries
- Filtering out files with wrong extensions
Functions of the developed GUI

• Automatic attribute insertion
• To examine input SystemC file and insert pragma automatically based on syntax.

```c
sc_uint<8> in_data_read[9];
sc_uint<16> coeff_read[9];
```
Functions of the developed GUI

- To show selection information such as which technology library has been selected.
- To show design information dynamically while running, like that total number of designs, current number of designs and synthesis results.

![GUI images](image-url)
Functions of the developed GUI

• Embedded editor
• To edit synthesis command
• Standard shortcuts for editor like copy, cut and paste can work
Plotting widget
Plotting widget

- Zoom out and zoom in using mouse wheel
Plotting widget functions

- Interactivity
- Mouse clicking
Plotting widget functions

• File list

“sa” stands for simulated annealing

attribute combination
Timer

- Elapsed time

- Remaining time
Progress Bar

• To show progress
• Relatively accurate for brute force
• Not useful if simulated annealing method is used
Conflict handling

- Conflicts between widgets in program sometimes appear and render program's crashing
- Mechanisms to check conflicts and pop up warning messages
Change Coordinate

• Change to area versus throughput

\[ \text{Throughput} = \frac{\text{Output}_\text{port}_\text{number} \times \frac{1}{\text{CP}_\text{delay} \times \text{Latency}}} \]

![Graph showing area versus latency and throughput](image)

change to
Show optimal designs

- Click button
## Synthesis Results

- Three benchmarks

<table>
<thead>
<tr>
<th>Bench</th>
<th>Type</th>
<th>#lines</th>
<th>Explorable operations</th>
<th>Brute</th>
<th>SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>fir</td>
<td>C</td>
<td>86</td>
<td>array(2), loop(2), function(1)</td>
<td>340s</td>
<td>55s</td>
</tr>
<tr>
<td>qsort</td>
<td>C</td>
<td>119</td>
<td>array(1), loop(2), function(3)</td>
<td>843s</td>
<td>64s</td>
</tr>
<tr>
<td>adpcm_encoder</td>
<td>C</td>
<td>179</td>
<td>array(1), for(1), function(2)</td>
<td>130s</td>
<td>54s</td>
</tr>
</tbody>
</table>
Brute Force Results

**fir**

- Area vs. Latency

**qsort**

- Area vs. Latency

**adpcm_encoder**

- Area vs. Latency
Simulated Annealing Results

### Part 4: Results

**fir**

![Graph for fir](image1)

**qsort**

![Graph for qsort](image2)

**adpcm_encoder**

![Graph for adpcm_encoder](image3)
Comparison

- Time Comparison
- Averagely, running time of simulated annealing algorithm for these three benchmarks is 21.8 percent of brute force algorithm, which means SA algorithm’s speed is 4.59 times of BF algorithm’s speed.

<table>
<thead>
<tr>
<th>Bench</th>
<th>Brute force</th>
<th>Simulated annealing</th>
<th>SA versus BF</th>
</tr>
</thead>
<tbody>
<tr>
<td>fir</td>
<td>340s</td>
<td>55s</td>
<td>16.2%</td>
</tr>
<tr>
<td>qsort</td>
<td>843s</td>
<td>64s</td>
<td>7.6%</td>
</tr>
<tr>
<td>adpcm_enconder</td>
<td>130s</td>
<td>54s</td>
<td>41.5%</td>
</tr>
</tbody>
</table>

\[
average = \frac{16.2 + 7.6 + 41.5}{3} = 21.8\%
\]

\[
\frac{1}{21.8\%} = 4.59
\]
Comparison

• Qualitative comparison
• In these experiments, brute force method has gone through all designs. Therefore, pareto dominance for brute force is 100%
• It has been found that simulated annealing could find 66% pareto dominated points.

<table>
<thead>
<tr>
<th>Bench</th>
<th>Brute force</th>
<th>Simulated annealing</th>
<th>SA versus BF</th>
</tr>
</thead>
<tbody>
<tr>
<td>fir</td>
<td>5</td>
<td>2</td>
<td>40%</td>
</tr>
<tr>
<td>qsort</td>
<td>6</td>
<td>5</td>
<td>83%</td>
</tr>
<tr>
<td>adpcm_encoder</td>
<td>4</td>
<td>3</td>
<td>75%</td>
</tr>
</tbody>
</table>

\[
\text{average} = \frac{40 + 83 + 75}{3} = 66\%
\]
Conclusion

Achievements

• This project has completed the main goals. It has developed one heuristic (simulated annealing) for design space exploration that achieved around four times faster speed than brute force method.

• A graphical user interface that could display synthesis results dynamically was developed.

Limitations and future works

• Other heuristics can be developed and compared to simulated annealing

• Number of tested benchmarks is not enough

• Automatic attribute insertion could only support special language formats
Q&A