HDL Coder 최적화 기법

이웅재 부장
Application Engineering Group
Agenda

- Introduction

- Area Optimizations
  - Resource Sharing
  - MATLAB Optimizations

- Speed Optimizations
  - Pipelining + Retiming
  - MATLAB Optimizations

- Conclusions and Results
HDL Coder: It’s all about the **Workflow**

- **MATLAB**
- **SIMULINK**

**This Session**

- HDL Coder
- VHDL or Verilog

**Devices**

- ASIC
- FPGA Boards
- HW-SW Co-design
HDL Compiler Flow

- **SIMULINK**
- **Simulink Front-end**
- **Data+Control flow**
- **Hierarchical IR**
- **HDL Back-end**
- **VHDL or Verilog**

**Structural Optimizations**
HDL Compiler Flow

MATLAB Front-end

Control flow IR

CFG $\rightarrow$ DFG Conversion

Simulink Front-end

Data+Control flow Hierarchical IR

HDL Back-end

VHDL or Verilog

Structural Optimizations
HDL Optimizations: What, How and Why

The three golden questions:
1. Speed: Does it meet timing?
2. Area: Does it fit on my FPGA?
3. Validation: Does it do the right thing?

FPGA Engineer

HDL optimizations assists the engineer in meeting these constraints
Agenda

- Introduction

- Area Optimizations
  - Resource Sharing
  - MATLAB Optimizations

- Timing Optimizations
  - Pipelining + Retiming
  - MATLAB Optimizations

- Conclusions and Results
Area Optimization Workflow

HDL Optimization Settings

Resource Report

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multipliers</td>
<td>154</td>
</tr>
<tr>
<td>Adders/Subtractors</td>
<td>90</td>
</tr>
<tr>
<td>Registers</td>
<td>852</td>
</tr>
<tr>
<td>RAMs</td>
<td>0</td>
</tr>
<tr>
<td>Multiplexers</td>
<td>2</td>
</tr>
</tbody>
</table>

MATLAB

SIMULINK

Front-end

Data+Control flow
Hierarchical IR

HDL Back-end

VHDL or Verilog

Structural Optimizations
Timing-aware Area Optimization

A “resource” may be:
- Multipliers/Subsystems (Sharing)
- Vector channels (Streaming)
- RAM I/O ports (RAM Mapping)
- Loop Body (Loop Streaming)

Time multiplexing through local multi-rate

154 mult, 90 adders

6 mults, 46 adders
MATLAB to HDL: RAM Mapping

```
persistent myMat
if isempty(myMat)
    myMat = zeros(M, N);
end

a = myMat(x, w) + myMat(x, z);
b = myMat(t, w) + myMat(t, z);
myMat(t, w) = a + b;
```

Memories on FPGAs

- Registers (in slices)
- Block RAMs (and ROMs)
  - Also linear (1-D)
- Distributed RAMs (in slices)

Automatic RAM Scheduling
5 cycles

FPGA RAM interface
MATLAB Loops and Vector Operations

- **Loop Unrolling**
  - Expose concurrency
  - Enabler

- **Loop Streaming**
  - Resource sharing in loops

```
function w = foo(u)
    persistent y;
    for i = 1:3
        for j = 1:5
            y = y + u * k(i,j);
        end
    end
    w = y;
```
Loop Streaming is resource sharing in loops

**Loop Streaming = off**

```matlab
function w = foo(u)
persistent y;
for i = 1:3
    for j = 1:5
        y = y + u * k(i,j);
    end
end
w = y;
```

- Multipliers = 15
- Adders = 15
- ROMs = 15

**Loop Streaming = on**

```
function w = foo(u)
persistent y;
for i = 1:3
    for j = 1:5
        y = y + u * k(i,j);
        w = y;
    end
end
```

- Multipliers = 1
- Adders = 3 (2 for loop counts)
- ROMs = 1
Generic Loop Streaming Architecture

for \( i = 1 : N \)
BODY
end

State Regs
- Persistent variables
- Loop-carried Dependencies

\( u \uparrow^N \)  \( \rightarrow \)  BODY  \( \rightarrow \)  \( y \downarrow^N \)

Loop Counter
Demo Model: Harris-Stephens’ Corner Detection

1. Sobel Edge Filter
2. Calculate Corner Metrics
3. Threshold & Find Local Maxima
4. Corner Metric $M_c$
Demo
Area Optimization
Demo
Area Optimization

<table>
<thead>
<tr>
<th>Multipliers</th>
<th>154</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adders/Subtractors</td>
<td>90</td>
</tr>
<tr>
<td>Registers</td>
<td>928</td>
</tr>
<tr>
<td>RAMs</td>
<td>0</td>
</tr>
<tr>
<td>Multiplexers</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Multipliers</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adders/Subtractors</td>
<td>46</td>
</tr>
<tr>
<td>Registers</td>
<td>559</td>
</tr>
<tr>
<td>RAMs</td>
<td>4</td>
</tr>
<tr>
<td>Multiplexers</td>
<td>304</td>
</tr>
</tbody>
</table>
Agenda

- Introduction

- Area Optimizations
  - Resource Sharing
  - MATLAB Optimizations

- Timing Optimizations
  - Pipelining + Retiming
  - MATLAB Optimizations

- Conclusions and Results
Timing Optimization Workflow

HDL Optimization Settings

- BalanceDelays: inherit
- ConstrainedOutputPipeline: 0
- DistributedPipelining: off
- FlattenHierarchy: inherit
- InputPipeline: 0
- OutputPipeline: 0

Critical Path Back-Annotation

4.3. Annotate Model with Synthesis Result

- Analysis
- Annotate synthesis result back to the Simulink model
- Input Parameters
  - Critical path source: pre-route
  - Critical path number: 1

Front-end

Data+Control flow
Hierarchical IR

Structural Optimizations

HDL Back-end

VHDL or Verilog

Synthesis / P+R
Back-Annotation: Visualize the Critical Path
Functional Equivalence:
Automatic Delay Balancing on parallel paths and through multi-rate
Distributed Pipelining: Global Optimization

Apply retiming on design

- Subsystem granularity
- Cross-hierarchy
  - Hierarchical rebuild or
  - Flatten Hierarchy
- Functional Equivalence
  - Latency-preserving
  - Numerics-preserving
- Visualize and verify pipeline movement in validation model

23 MHz

74 MHz
Demo

Timing Optimization
Demo
Timing Optimization

38 MHz → 62 MHz
Agenda

- Introduction

- Area Optimizations
  - Resource Sharing
  - MATLAB Optimizations

- Timing Optimizations
  - Pipelining + Retiming
  - MATLAB Optimizations

- Conclusions and Results
## Design Space: Corner Detection

<table>
<thead>
<tr>
<th>Design</th>
<th>DSP48 (%)</th>
<th>Slices (%)</th>
<th>Flip-Flops (%)</th>
<th>RAMB16 (%)</th>
<th>Critical Path (ns)</th>
<th>Overclocking</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Model</td>
<td>82</td>
<td>26</td>
<td>11</td>
<td>0</td>
<td>42.11</td>
<td>1</td>
</tr>
<tr>
<td>Design #1</td>
<td>5</td>
<td>30</td>
<td>19</td>
<td>3</td>
<td>26.86</td>
<td>75</td>
</tr>
<tr>
<td>Design #2</td>
<td>82</td>
<td>19</td>
<td>12</td>
<td>0</td>
<td>17.52</td>
<td>1</td>
</tr>
<tr>
<td>Design #3</td>
<td>9</td>
<td>35</td>
<td>26</td>
<td>3</td>
<td>13.41</td>
<td>15</td>
</tr>
<tr>
<td>Design #4</td>
<td>7</td>
<td>42</td>
<td>50</td>
<td>2</td>
<td>28.21</td>
<td>1</td>
</tr>
<tr>
<td>Design #5</td>
<td>30</td>
<td>41</td>
<td>33</td>
<td>3</td>
<td>14.48</td>
<td>3</td>
</tr>
<tr>
<td>Design #6 (M)</td>
<td>5</td>
<td>52</td>
<td>37</td>
<td>3</td>
<td>14.30</td>
<td>75</td>
</tr>
</tbody>
</table>

**3x Frequency Speedup**

**16x Area Reduction**
Conclusions: HDL Optimizations

- Area optimization is a workflow
  - Features: Sharing, Streaming, RAM Mapping, Loop streaming
  - Both MATLAB and Simulink

- Timing optimization is a workflow
  - Features: Pipelining, Retiming, Critical-path Analysis
  - Both MATLAB and Simulink

- Tools for validation and verification

- Design space exploration (DSE)
  - Design check-pointing
  - Explore design space through workflow
HDL Optimization Workflow for DSE

**HDL Optimization Settings**

- **BalanceDelays**: inherit
- **ConstrainedOutputPipeline**: 0
- **DistributedPipelining**: off
- **FlattenHierarchy**: inherit
- **InputPipeline**: 0
- **OutputPipeline**: 0

**Questions?**

**Timing/Area Reports**

<table>
<thead>
<tr>
<th>4.3. Annotate Model with Synthesis Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
</tr>
<tr>
<td>Annotate synthesis result back to the Simulink model</td>
</tr>
<tr>
<td>Input Parameters</td>
</tr>
<tr>
<td>Critical path source: pre-route</td>
</tr>
<tr>
<td>Critical path number: 1</td>
</tr>
</tbody>
</table>

**Front-end**

**Checkpointing**

**Structural Optimizations**

**HDL Back-end**

**Synthesis / P+R**

VHDL or Verilog