Lecture 22
Design Compiler in Depth

Xuan ‘Silvia’ Zhang
Washington University in St. Louis

http://classes.engineering.wustl.edu/ese461/
Timeline

• Class project tasks
  - logic synthesis
  - design optimization and iteration
  - place and route
  - final report

• Project milestones
  - 12/7: in-class presentation
  - 12/12: final report due
  - hard deadlines, not extendable

• Lecture plans
  - 11/30: 30min team meeting
  - 12/5: last lecture, Encounter tips, conclusion
Feedback from Mid-Project Reports

- **Synthesizable Verilog code style**
  - If-else, case: complete default branch
  - DO NOT mix blocking and non-blocking statements
  - proper FSM implementation
  - clocked Always block, reset and sensitivity list

- **Testbench setup**
  - instantiate memory
  - simulation termination
  - endianess (Bitcoin)

- **To-do list**
  - logic synthesis
  - place and route
Figure 1-1 Design Compiler and the Design Flow

- HDL
  - HDL Compiler
    - Design Compiler
      - Timing optimization
      - Datapath optimization
      - Power optimization
        - Area optimization
        - Test synthesis
        - Timing closure
      - Optimized netlist
        - Back-annotation
          - Place & route
            - Timing & power analysis
            - Formal verification
• Develop HDL files
  - Chapter 3, “Preparing design files for synthesis”

• Specify libraries
  - Chapter 4, “Working with libraries”

• Read design
  - Chapter 5, “Working with designs in memory”

• Define design environment
  - Chapter 6, “Defining the design environment”

• Set design constraints
  - Chapter 7, “Defining design constraints”

• Select compile strategy

• Synthesize and optimize the design
  - Chapter 8, “Optimizing the design”
Figure 2-2 Basic Synthesis Flow

Develop HDL files

Specify libraries

Library objects
- link_library
- target_library
- symbol_library
- synthetic_library

Read design

- analyze
- elaborate
- read_file

Define design environment

- set_operating_conditions
- set_wire_load_model
- set_drive
- set_driving_cell
- set_load
- set_fanout_load
- set_min_library

Set design constraints

- Design rule constraints
  - set_max_transition
  - set_max_fanout
  - set_max_capacitance

- Design optimization constraints
  - create_clock
  - set_clock_latency
  - set_propagated_clock
  - set_clock_uncertainty
  - set_clock_transition
  - set_input_delay
  - set_output_delay
  - set_max_area

Select compile strategy

- Top down
- Bottom up

Synthesize and optimize the design

- compile or compile_ultra

Analyze and resolve design problems

- check_design
- report_area
- report_constraint
- report_timing

Save the design database

- write
Organize the Design Data

Figure 3-1  Top-Down Compile Directory Structure

Design

src

.vhd

.v

work

syn

.script

.log

.unmapped

.mapped

.netlist

.v

.vhd

.sim

.synopsys_dc_setup

Figure 3-2  Bottom-Up Compile Directory Structure

Design

src

.vhd

.v

work

syn

.script

.log

.unmapped

.mapped

.netlist

.v

.vhd

.sim

.synopsys_dc_setup

.pass1

.script

.log

.unmapped

.mapped

.netlist

.v

.vhd

.pass2
HDL Coding for Synthesis

- FSM
- Sensitivity list
- Incomplete control statement

*Example 3-3  Incorrect if Statement (Verilog)*

```verilog
if (a == 1) && (b == 1)
  z = 1;
```
HDL Coding for Synthesis

- Value assignments
  - use nonblocking assignments within sequential always
  - use blocking assignments within combinational always

- Constant definition

  Example 3-7  Using Macros and Parameters (Verilog)
  
  // Define global constant in def_macro.v
  `define WIDTH 128

  // Use global constant in reg128.v
  reg regfile[WIDTH-1:0];

  // Define and use local constant in module foo
  module foo (a, b, c);
    parameter WIDTH=128;
    input [WIDTH-1:0] a, b;
    output [WIDTH-1:0] c;

- Guidelines for identifiers, expressions, and functions
Design Terminology and Objects

- Flat vs Hierarchical Designs

![Design Objects Diagram](image)

**Figure 5-1 Design Objects**

- Design
- Port
- Instance or Cell
- Net
- Pin

- Design: {TOP, ENCODER, REGFILE}
- Reference: {ENCODER, REGFILE, INV}
- Instance: {U1, U2, U3, U4}
Designs, Instances, and References

- analyze
- elaborate
- read_file
- link: link_library, search_path

Figure 5-2  Instances and References
Ungroup Hierarchies Automatically

- **compile_ultra**
  - by default, perform delay-based auto-ungrouping

<table>
<thead>
<tr>
<th>Compile flow</th>
<th>Effect on hierarchical pin timing constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ungrouping a hierarchy before optimization by using <code>ungroup</code></td>
<td>Timing constraints placed on hierarchical pins are preserved. In previous releases, timing attributes placed on the hierarchical pins of a cell were not preserved when that cell was ungrouped. If you want your current optimization results to be compatible with previous results, set the <code>ungroup_preserve_constraints</code> variable to <code>false</code>. The default for this variable is <code>true</code>, which specifies that timing constraints will be preserved.</td>
</tr>
<tr>
<td>Ungrouping a hierarchy during optimization by using compile -ungroup_all or set_unigroup followed by compile</td>
<td>Timing constraints placed on hierarchical pins are not preserved. To preserve timing constraints, set the <code>auto_ungroup_preserve_constraints</code> variable to <code>true</code>.</td>
</tr>
<tr>
<td>Automatically ungrouping a hierarchy during optimization, that is, by using the <code>compile_ultra</code> or `compile -auto_ungroup area</td>
<td>delay command`</td>
</tr>
</tbody>
</table>
## Edit Designs

### Table 5-12 Design Editing Tasks and Commands

<table>
<thead>
<tr>
<th>Object</th>
<th>Task</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cells</td>
<td>Create a cell</td>
<td>create_cell</td>
</tr>
<tr>
<td></td>
<td>Delete a cell</td>
<td>remove_cell</td>
</tr>
<tr>
<td>Nets</td>
<td>Create a net</td>
<td>create_net</td>
</tr>
<tr>
<td></td>
<td>Connect a net</td>
<td>connect_net</td>
</tr>
<tr>
<td></td>
<td>Disconnect a net</td>
<td>disconnect_net</td>
</tr>
<tr>
<td></td>
<td>Delete a net</td>
<td>remove_net</td>
</tr>
<tr>
<td>Ports</td>
<td>Create a port</td>
<td>create_port</td>
</tr>
<tr>
<td></td>
<td>Delete a port</td>
<td>remove_port</td>
</tr>
<tr>
<td></td>
<td></td>
<td>remove_unconnected_port</td>
</tr>
<tr>
<td>Pins</td>
<td>Connect pins</td>
<td>connect_pin</td>
</tr>
<tr>
<td>Buses</td>
<td>Create a bus</td>
<td>create_bus</td>
</tr>
<tr>
<td></td>
<td>Delete a bus</td>
<td>remove_bus</td>
</tr>
</tbody>
</table>
dc_shell> get_pins U8/*
{"U8/A", "U8/Z"}
dc_shell> all_connected U8/A
{"n66"}
dc_shell> all_connected U8/Z
{"OUTBUS[10]"}
dc_shell> remove_cell U8
Removing cell 'U8' in design 'top'.
1
dc_shell> create_cell U8 IVP
Creating cell 'U8' in design 'top'.
1
dc_shell> connect_net n66 [get_pins U8/A]
Connecting net 'n66' to pin 'U8/A'.
1
dc_shell> connect_net OUTBUS[10] [get_pins U8/Z]
Connecting net 'OUTBUS[10]' to pin 'U8/Z'.
1
Define the Design Environment

Figure 6-1 Commands Used to Define the Design Environment
**Figure 6-3 Drive Characteristics**

```
dc_shell> current_design top_level_design
dc_shell> set_drive 1.5 (I1 I2)
dc_shell> current_design sub_design1

dc_shell> set_driving_cell -lib_cell IV {I3}

dc_shell> set_driving_cell -lib_cell AN2 -pin Z -from_pin B {I4}
```
Model the System Interface

- Define drive characteristics for input ports
- Define load on input and output ports
- Define fanout loads on output ports
- (Similarly) Set logic constraints on ports
  - define ports as logically equivalent
  - define logically opposite input ports
  - allow assignment of any signal to an input
  - always one or zero
  - unconnected
Define Design Constraints

Figure 7-1  Major Design Compiler Constraints

- **Design Rule Constraints**
  - **Maximums:** Transition Time, Fanout, Capacitance
  - **Cell Degradation**
    - `set_cell_degradation`
  - **Minimum Capacitance**
    - `set_max_transition`
    - `set_max_fanout`
    - `set_max_capacitance`

- **Optimization Constraints**
  - **Area**
    - `set_max_area`
  - **Speed**
    - `create_clock`
    - `set_input_delay`
    - `set_output_delay`
    - `set_max_delay`
    - `set_min_delay`
## Summary of Design Rule Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_max_fanout</td>
<td>Input ports or designs</td>
</tr>
<tr>
<td>set_fanout_load</td>
<td>Output ports</td>
</tr>
<tr>
<td>set_load</td>
<td>Ports or nets</td>
</tr>
<tr>
<td>set_max_transition</td>
<td>Ports or designs</td>
</tr>
<tr>
<td>set_cell_degradation</td>
<td>Input ports</td>
</tr>
<tr>
<td>set_min_capacitance</td>
<td>Input ports</td>
</tr>
</tbody>
</table>
Optimization Constraints

- Timing constraints (performance and speed)
  - input and output delays (synchronous paths)
  - minimum and maximum delay (asynchronous paths)
  - note: set_fix_hold

- Maximum area
  - number of gates
Cost Function Calculation

- Design rule cost function

\[
\sum_{i=1}^{m} \max(d_i, 0) \times w_i
\]

- Max delay cost function

\[
\sum_{i=1}^{m} v_i \times w_i
\]

- Min delay cost function

\[
\sum_{i=1}^{m} v_i
\]
Optimize the Design

- **Architecture optimization**
  - sharing common subexpressions
  - sharing resources
  - reordering operators
  - selecting DesignWare implementations, etc.

- **Logic-level optimization**
  - structuring
  - flattening

- **Gate-level optimization**
  - mapping
  - delay optimization
  - design rule fixing
  - area optimization
Optimization Example

Figure 8-1  Design to Illustrate Compile Strategies

Table 8-1  Design Specifications for Design TOP

<table>
<thead>
<tr>
<th>Specification type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating condition</td>
<td>WCCOM</td>
</tr>
<tr>
<td>Wire load model</td>
<td>&quot;20x20&quot;</td>
</tr>
<tr>
<td>Clock frequency</td>
<td>40 MHz</td>
</tr>
<tr>
<td>Input delay time</td>
<td>3 ns</td>
</tr>
<tr>
<td>Output delay time</td>
<td>2 ns</td>
</tr>
<tr>
<td>Input drive strength</td>
<td>drive_of (IV)</td>
</tr>
<tr>
<td>Output load</td>
<td>1.5 pF</td>
</tr>
</tbody>
</table>
Top-Down Optimization

Example 8-1  Constraints File for Design TOP (defaults.con)

```
set_operating_conditions WCCOM
set_wire_load_model "20x20"
cREATE_CLOCK -period 25 CLK
set_input_delay 3 -clock clk \ 
  [remove_from_collection [all_inputs] [get_ports clk]]
set_output_delay 2 -clock clk [all_outputs]
set_load 1.5 [all_outputs]
set_driving_cell -lib_cell IV [all_inputs]
set_drive 0 clk
```

Example 8-2  Top-Down Compile Script

```
/* read in the entire design */
read_verilog E.v
read_verilog D.v
read_verilog C.v
read_verilog B.v
read_verilog A.v
read_verilog TOP.v
current_design TOP
link

/* apply constraints and attributes */
source defaults.con

/* compile the design */
compile
```
Bottom-Up Compile Script

Example 8-3  Bottom-Up Compile Script

set all_blocks { E D C B A}

# compile each subblock independently
foreach block $all_blocks {
    # read in block
    set block_source "$block.v"
    read_file -format verilog $block_source
    current_design $block
    link
    # apply global attributes and constraints
    source defaults.con
    # apply block attributes and constraints
    set block_script "$block.con"
    source $block_script
    # compile the block
    compile
}

# read in entire compiled design
read_file -format verilog TOP.v
current_design TOP
link
write -hierarchy -format ddc -output first_pass.

# apply top-level constraints
source defaults.con
source top_level.con

# check for violations
report_constraint

# characterize all instances in the design
set all_instances { U1 U2 U2/U3 U2/U4 U2/U5}
characterize -constraint $all_instances

# save characterize information
foreach block $all_instances {
    current_design $block
    set char_block_script "$block.wscr"
    write_script > $char_block_script
}

# recompile each block
foreach block $all_blocks {
    # clear memory
    remove_design -all
    # read in previously characterized subblock
    set block_source "$block.v"
    read_file -format verilog $block_source

    # recompile subblock
    current_design $block
    link
    # apply global attributes and constraints
    source defaults.con
    # apply characterization constraints
    set char_block_script "$block.wscr"
    source $char_block_script
    # apply block attributes and constraints
    set block_script "$block.con"
    source $block_script
    # recompile the block
    compile
}
Mixed Optimization

Figure 8-2  Mixing Compilation Strategies

- Specification has detailed time budgets for first level of hierarchy: A, B, C, and D.
- Top-down compile is used for hierarchy below D.
- Bottom-up compile is used for hierarchy below B.
Suggested Reading

- Partitioning for Synthesis
  - Chapter 3 (3-4)

- Working with Attributes
  - Chapter 5 (5-40)

- Define Design Constraints
  - Chapter 6
    - reporting constraints
    - characterize subdesigns

- Optimize the Design
  - Chapter 7
Questions?

Comments?

Discussion?