Introduction to Quartus II

Adviser: An-Yu Wu
Speaker: 張龍豪
Date: 12/25/2002
Outline

- Introduction
- Design Entry
- Project Compilation
- Timing analysis
- Function simulation
- Device Programming
- Conclusion
Introduction to Quartus II

- Operate in a self-contained environment

### Design Entry
- Graphic Design Entry
- Megafunctions
- Text Design Entry
- Hierarchical Design Entry

### Design Compilation
- Logic Synthesis and Fitting

### Verification & Programming
- Timing simulation
- Functional simulation
- Timing analysis
- Device Programming
Quartus II Design Methodology

1. Design Specification
2. Design Entry
3. Project Compilation
4. Simulation
5. Timing analysis
6. Device programming
7. In-System Verification
8. System Production
9. Design Modification
Outline

- Introduction
- Design Entry
- Project Compilation
- Timing analysis
- Function simulation
- Device Programming
- Conclusion
Design Entry Process

- Project Setup/Management
- Multiple design entry methods
  - MAX+PLUS II
    - Graphic design entry
    - Text design entry
      - AHDL, VHDL, Verilog
  - 3rd party EDA tools
    - EDIF, OrCAD schematics
  - Add flexibility and optimization to the Design entry process by:
    - mixing and matching design files
    - using and Megafunctions to accelerate design entry
Project Setup/Management

- What is a Project?
  - A design file
  - A project is:
    - checked for design entry errors
    - compiled
    - simulated (functional or with timing)
    - analyzed for timing
    - used to generate programming file

- Projects can be archived
New Project Wizard

Select **File => New Project Wizard**

- Working Directory
- Project name
- Top-level design name

Save as *.QAR (QuARtus)

Click finish
New Project Wizard

Add design files
- Graphic (.BDF, .GDF)
- AHDL/VHDL/Verilog
- EDIF

Note:
- All files in the project directory do not need to be added

Add user library pathnames and files
New Project Wizard

Review result and click on finish
Block Diagram/Schematic File Editor

- Block diagram entry is mainly for top-down design methodology
- Schematic file entry is the traditional schematic design entry
- User can enter blocks, primitives, and megafUNCTIONS from Quartus II-provided or user libraries
- Provides “smart” block connection and mapping
Block Editor – Create a New File

- Select **File** => **New**
- Select **Block Diagram / Schematic File**
- Save as *.BDF (Block Design File)
- Add file to current project

Click OK
Block Design File

Block and symbol editors
Block Editor – Enter Symbols

- Schematic file entry
- Click the **symbol tool** button

Symbol library

Preview the symbol
Block Editor – Use Megafuction

- Click MegaWizard Plug-In Manager
Create a New Symbol

- Custom megafunction variations are based on Altera-provided megafunctions.
- Select the Megafunction item that you want to create.
- Modify the item by your own.
Create a New Symbol (cont.)

Input cell name

Click Finish
Create a New Symbol (cont.)

Click OK
Block Editor – Draw Block

- Text Design Entry
- Click the **Block Tool** button
- Click and drag on the block diagram and you can see the Block symbol

![Diagram of Block Editor with tool options]
Block Editor – Specify I/Os

- Right click on the block symbol, choose **Block Property**
- Select I/Os, input your I/O name and type

Click OK
Block Editor – Make connections

- Wire (single bit line)
- Bus (Multiple bits)
- Conduit (Multiple bits)

Connects blocks to any other objects
Block Editor – Mapper properties

- Map the block I/Os when the I/O names are different between the blocks.
- Label the connector
  - Right-click on the connector and choose Properties
Block Editor – Mapper properties (cont.)

- Double-click on the mapper
- Set the I/O on block and connector signal

Double-click on the mapper
Block Editor – Make connections

Now, the I/Os are connected
Block Editor – Add I/O pins

- Click the **symbol tool** button

Click OK
Block Diagram

Arrange the Blocks, Primitives, and Megafunction
Block Editor – Generate Design File

- Right-click on block symbol
- Select **Create Design file from Selected Block**

![Create Design File dialog box]

- Input file name
- Click OK
These lines are necessary for Quartus II to update the source code.

```verilog
// Module Declaration
module hvalues

// {{ALTERA_ARGS_BEGIN}} DO NOT REMOVE THIS LINE!

// Port Declaration

// {{ALTERA_ARGS_END}} DO NOT REMOVE THIS LINE!

// {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
input [1:0] sel;
output [2:0] h;

// {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!

reg [2:0] h;

always @(sel)
case (sel)
    2'b 00 : h = 3'b 111;
    2'b 01 : h = 3'b 101;
    2'b 10 : h = 3'b 011;
    2'b 11 : h = 3'b 001;
endcase

endmodule
```
Block Editor – Generate Design File

➢ Select File => New

Click OK
Outline

- Introduction
- Design Entry
- Project Compilation
- Timing analysis
- Function simulation
- Device Programming
- Conclusion
Compiler Settings

- Compiler control information
  - Level of compilation
  - Device & Device options
  - Pin assignments
  - Synthesis & Fitting
  - Timing analysis

- Accessed via the Processing Menu

- Information stored in a Compiler Settings File (*.CSF)
Compiler Settings – General

- Select **processing** => Compile Mode
- Select **processing** => Compiler Settings...
Compiler Settings – Chips & Devices

- Family: APEX 20K
- Assign Pins... button highlighted
- Available devices:
  - EP20K100QQC8-1
- Package: PQFP
- Pin count: 208
- Speed grade: 1
- Core voltage: 2.5 V
Pin assignment – all I/Os

- Select a pin number
- Invoke Node Finder to find pin name or type

Available I/Os

Click
Pin assignment – Add I/Os

- Select Filter: Pins: all and click Start
- Add to assignment list

Click OK
Compiler Settings – Mode

Select the compilation mode.

Changes apply to Compiler settings 'filtref'

Compilation level

- Netlist extraction and synthesis only, including estimated timing
- Full compilation, including programming file generation and actual timing data

Compilation speed/disk usage tradeoff

- Normal compilation/less disk space.
- Smart compilation/more disk space. The Compiler will save extra data to make future re-compilations run faster.

Preserve fewer node names to save disk space
(This option is available for both Normal and Smart compilation)
Compiler Settings – Synthesis & Fitting

Compiler Settings

General | Chips & Devices | Mode | Synthesis & Fitting | Verification

Specify options for logic synthesis and fitting. These options apply to all logic within the entity that is the current compilation locus. Note: The availability of some options depends on the current device family and filter.

Changes apply to Compiler settings "tiltiiel"

- Timing-driven compilation
  - [ ] Optimize timing
  - [ ] Optimize I/O cell register placement for timing

- Incremental synthesis
  - [ ] Save a node-level netlist into a persistent source file (VHDL Quartus Mapping File)

File name: ...

[OK] [Cancel] [Apply]
Compiler Settings – Verification

Specify options for automatic timing analysis and simulation. These verification processes run automatically after compilation.

Changes apply to Compiler settings 'fitref'.

- Run timing analyses
- Run batch simulation with these Simulator settings:

  <None>
Compiling in Quartus II

Select Processing => Start Compilation

Status Bar

Compilation Result

Compiler Report

Compiler Messages
The Compiler Report

- Contains all information on how a design was implemented in the targeted device
  - Device Summary Statistics
  - Compiler Settings
  - Floorplan Views
  - Device Resources Used
  - State Machines Implemented
  - Equations
  - Timing Analysis Results
  - CPU Resources
- This a read-only window
Outline

- Introduction
- Design Entry
- Project Compilation
- Timing analysis
- Function simulation
- Device Programming
- Conclusion
Reporting Timing Results

- Timing information is part of the Compilation Report
  - Summary Timing Analyses
  - \( f_{max} \) (not include delays to/from pins) or \( f_{max} \) (include delays to/from pins)
  - Register-to-Register Table
  - tsu (Input Setup Time)
  - th (Input Hold Time)
  - tco (Clock to Out Delays)
  - tpd (Pin to Pin Delays)

- All timing results are reported here
Outline

- Introduction
- Design Entry
- Project Compilation
- Timing analysis
- Function simulation
- Device Programming
- Conclusion
Simulations

- Run Functional Simulation
  - Fast compilation
  - Logical model only, no logic synthesis
  - All nodes are retained and can be simulated
  - Outputs are updated without delay

- Run Timing Simulation
  - Slower compilation
  - Timing model: logical & delay model
  - Nodes may be synthesized away
  - Outputs are updated after delay
Create Waveform file

- Create a waveform for simulation
- Select File => New => Other Files
- Select Vector Waveform File
- Save as *.VWF (Vector Waveform File)

Click OK
Setup end time

- Specify maximum length of simulation end time
  - Select **Time** => **End time**
Add I/O pins – select

- Select View => Auxiliary Windows => Node Finder
- Select Filter: Pins: all and Click Start
- Select all pin and drop them into the Name in waveform file
Add I/O pins

- All pins are in the Name list
- With initial values or Hi-Z
Creating a Clock

Right-click on the waveform name and choose **Value** => **Clock**
Creating Counting Pattern

- Right-click on the waveform name and choose **Value** => **Count Value**
Input value

- All input pins have input values
Simulator Settings – General

- Select **Processing** => **Simulate Mode**
- Select **Processing** => **Simulation Settings**
Simulator Settings – Time/Vectors

Specify the time period you want to simulate. You can optionally provide vector stimuli for simulation in a Vector Waveform File (.vwf), a text-based Vector File (.vec), or a Power Input File (.pif). (You can also enter vector stimuli from the Tcl Console Window)

Note: If you select a Power Input File for the source of vector stimuli, the options to automatically add pins to simulation output waveforms and check outputs are not available.

Changes apply to Simulator settings "filter"

- Simulation period:
  - Run simulation until all vector stimuli are used
  - End simulation at:

- Vectors:
  - Source of vector stimuli:
    - [Selection]
  - Automatically add pins to simulation output waveforms
  - Check outputs (automatically display a comparison of expected vs. actual outputs in the simulation report)
Simulator Settings – Mode

Refer to PP. 36
Simulator Settings – Options

Select options for simulation. The setup/hold and glitch options are available only for a simulation that includes Complex-generated timing information. Note: the availability of some options depends on the current device family.

Changes apply to Simulator settings Title:

- Simulation coverage reporting
- Setup and hold time violation detection
- Glitch detection
  - Glitch interval: 1 ns
- Estimate power consumption

offCore Transactor Model File Name:
Simulator Result

- Select **Processing** => **Start Compilation**

![Simulation Waveforms](image)

- Master time bar handle
- New time bar handle
Outline

- Introduction
- Design Entry
- Project Compilation
- Timing analysis
- Function simulation
- Device Programming
- Conclusion
APEX 20K Configuration Overview

- APEX 20K devices are SRAM-based
  - Must be reconfigured with every power-up
- Can configure the APEX 20K device via configuration device or download cable
- APEX 20K family supports configuration through its JTAG ports
ByteBlaster

Figure 1. ByteBlaster Parallel Port Download Cable
Configuration

- Select **Processing** => open programmer
- Save as *.CDF (Chain Description File)
Configuration

- Select Setup => Add => **Hardware Setup** dialog box, select **ByteBlasterMV**

![Programming Hardware Setup dialog box](image)

Click Add
Configuration

- Select **Add File** => *.SOF (SRAM Object File)
- Check on Program/Configure.

Click Start
Conclusion

- Use Hierarchical Design methodology
- Use Hierarchy Display for fast access to design file at any level
- Compile top-level design without any pin assignments first to determine if the design actually fits in the targeted device
- Use Compiler Report to study design implementation and resource usage
- Use Functional Simulation to verify proper operation