Course Outline

◆ Introduction to PLD
◆ Design Flow & Altera Tools
◆ Getting Started
◆ Graphic Design Entry
◆ Text Editor Design Entry
◆ Waveform Design Entry
◆ Design Implementation
◆ Project Verification
  • Functional Simulation
  • Timing Analysis
  • Timing Simulation
◆ Device Programming
◆ Summary & Getting Help
Introduction to PLD

PLD : Programmable Logic Device
SPLD : Small/Simple Programmable Logic Device
CPLD : Complex Programmable Logic Device
FPGA : Field Programmable Gate Array
Main Features

- Field-programmable
- Reprogrammable
- In-circuit design verification
- Rapid prototyping
- Fast time-to-market
- No IC-test & NRE cost
- H/W emulation instead of S/W simulation
- Good software
- ...
Programmability

◆ Why programmable? Why reprogrammable?
  • Logic is implemented by programming the “configuration memory”
  • Various configuration memory technologies
    – One-Time Programmable: anti-fuse, EPROM
    – Reprogrammable: EPROM, EEPROM, Flash & SRAM
Programmable Combinational Logic

Product Term-based Building Block
* 2-level logic
* High fan-in

Look-up Table-based Building Block
* 4 to 5 inputs, fine grain architecture
* ROM-like
Programmable Register

* Typical register controls: clock, enable, preset/clear, ...
Programmable Interconnect

Typical routing resources: switching elements, local/global lines, clock buffers...
Programmable I/O

Typical I/O controls: direction, I/O registers, 3-state, slew rate, ...
Field-Programmability

Why field-programmable?

- You can verify your designs at any time by configuring the FPGA/CPLD devices on board via the download cable or hardware programmer.
Rapid Prototyping

◆ Reduce system prototyping time:
  • You can see the “real” things
    – In-circuit design verification
  • Quick delivery instead of IC manufacture
  • No test development, no re-spin potential (i.e. no NRE cost)
  • Satisfied for educational purposes

◆ Fast time-to-market

0. Design, simulation, & compilation
1. Downloading configuration bitstream
2. Entering input data
3. Obtaining output data
4. Analysis

FPGA/CPLD is on the board!
Software Environment

◆ Various design entries and interfaces
  • HDL: Verilog, VHDL, ABEL, ...
  • Graphic: Viewlogic, OrCAD, Cadence, ...

◆ Primitives & macrofunctions provided
  • Primitive gates, arithmetic modules, flip-flops, counters, I/O elements, ...

◆ Constraint-driven compilation/implementation
  • Logic fitting, partition, placement & routing (P&R)

◆ Simulation netlist generation
  • Functional simulation & timing simulation netlist extraction

◆ Programmer/download program
# FPGA/CPLD Benefits

<table>
<thead>
<tr>
<th></th>
<th>Full-Custom ICs</th>
<th>Cell-Based ICs</th>
<th>Gate Arrays</th>
<th>High-Density PLDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>✅  ✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Integration Density</td>
<td>✅  ✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>High-Volume device cost</td>
<td>✅  ✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Low-volume device cost</td>
<td>✅</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time to Market</td>
<td>✅</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Risk Reduction</td>
<td>✅</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Future Modification</td>
<td>✅</td>
<td></td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Development Tool</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>Educational Purpose</td>
<td>✅  ✅</td>
<td></td>
<td>✅</td>
<td></td>
</tr>
</tbody>
</table>

**Good**

**Excellent**
Altera & CIC

◆ Altera
  • One of the world leaders in high-performance & high-density PLDs & associated CAE tools
  • Supports university program in Taiwan via CIC

◆ From CIC, you can apply:
  • Altera software - it's free for educational purpose!
    - PC : MAX+PLUS II (full design environment)
    - PC : QUARTUS II (full design environment)
  • Altera hardware -
  • University Program Design Laboratory Package (since 9709):
    - UP1/UP2 Education Board
    - ByteBlaster download cable
    - Student Edition Software
  • Of course, CIC is responsible for technical supports
  • WWW: http://www.cic.edu.tw/chip_design/design_intr/altera/
MAX+PLUS II Can …
(An Introduction)

◆ Operate in a self-contained environment

Design Entry
- Standard EDA Design Entry
  - Cadence
  - Mentor Graphics
  - OrCAD
  - Synopsys
  - Viewlogic
  - Synplify
  - Exemplar
  - Others
  - Graphic Design Entry
  - Text Design Entry (AHDL, VHDL, Verilog HDL)
- High-Level Design Entry
  - MegaCore Functions
  - LPM Functions
  - ANPP Megagates
  - Waveform Design Entry
  - Hierarchical Design Entry
  - Floorplan Editing

Design Compilation
- MAX+PLUS II Compiler
  - Design Rule Checking
  - Logic Synthesis & Fitting
  - Multi-Device Partitioning
  - Automatic Error Location
  - Timing-Driven Compilation
  - OpenCore Evaluation

Verification & Programming
- Standard EDA Verification
  - Cadence
  - Mentor Graphics
  - OrCAD
  - Synopsys
  - Viewlogic
  - Others
  - Timing Simulation
  - Functional Simulation
  - Multi-Device Simulation
  - Timing Analysis
  - Device Programming
  - OpenCore Evaluation
MAX+PLUS II Software Products

◆ Fixed-Node Subscription Products
  • Windows 95/98 and Windows NT Operating System, require hardware protection key for node identification
  • FIXEDPC full featured MAX+PLUS II software with VHDL/Verilog

◆ Floating-Node Subscription Products
  • Licensed Using Windows NT and UNIX Servers
  • FLOATPC for Windows 95/98 and Windows NT clients only.
  • FLOATNET for Windows 95/98/NT and UNIX clients.

◆ MAX+PLUS II BASELINE Software
  • entry-level version of the MAX+PLUS II software which is free of charge.
MAX+PLUS II

◆ **Supported Platforms**
  - PC
  - UNIX Platform
    - Sun SPARCstation
    - HP 9000 Series 700/800 workstation
    - IBM RISC System /6000 workstation

*Please read the `README` file with every release of MAX+plus II

◆ **Network licensing supported on both PC and Unix**
Design Flow & Altera Tools

◆ FPGA/CPLD Design Flow
  • Design Ideas
  • Detailed Design
  • Functional Simulation
  • Synthesis & Implementation
  • Timing Simulation
  • Device Programming

◆ Altera MAX+PLUS II Development Software
  • Design Entry
  • Project Processing
  • Project Verification
  • Device Programming
FPGA/CPLD Design Flow

- Design Ideas
- Detailed Design
- Functional Simulation
- Implementation (P&R)
- Device Programming
- Timing Simulation

$t_{pd} = 22.1\text{ns}$
$f_{max} = 47.1\text{MHz}$
Design Ideas

◆ What are the main design considerations?
  • Design feasibility?
  • Design spec?
  • Cost?
  • FPGA/CPLD or ASIC?
  • Which FPGA/CPLD vendor?
  • Which device family?
  • Development time?
Detailed Design

◆ Choose the design entry method
  • Schematic
    – Gate level design
    – Intuitive & easy to debug
  • HDL (Hardware Description Language), e.g. Verilog & VHDL
    – Descriptive & portable
    – Easy to modify
  • Mixed HDL & schematic

◆ Manage the design hierarchy
  • Design partitioning
    – Chip partitioning
    – Logic partitioning
  • Use vendor-supplied libraries or parameterized libraries to reduce design time
  • Create & manage user-created libraries (circuits)
Functional Simulation

◆ Preparation for simulation
  • Generate simulation patterns
    – Waveform entry
    – HDL testbench
  • Generate simulation netlist

◆ Functional simulation
  • To verify the functionality of your design only

◆ Simulation results
  • Waveform display
  • Text output

◆ Challenge
  • Sufficient & efficient test patterns
Design Implementation

◆ Implementation flow
  • Netlist merging, flattening, data base building
  • Design rule checking
  • Logic optimization
  • Block mapping & placement
  • Net routing
  • Configuration bitstream generation

◆ Implementation results
  • Design error or warnings
  • Device utilization
  • Timing reports

◆ Challenge
  • How to reach high performance & high utilization implementation?
Timing Analysis & Simulation

◆ **Timing analysis**
  - Timing analysis is static, i.e., independent of input & output patterns
  - To examine the timing constraints
  - To show the detailed timing paths
  - Can find the critical path

◆ **Timing simulation**
  - To verify both the functionality & timing of the design

$t_{pd} = 22.1\, \text{ns}$
$f_{max} = 47.1\, \text{MHz}$
Device Programming

◆ Choose the appropriate configuration scheme
  • SRAM-based FPGA/CPLD devices
    – Downloading the bitstream via a download cable
    – Programming onto a non-volatile memory device & attaching it on the circuit board
  • OTP, EPROM, EEPROM or Flash-based FPGA/CPLD devices
    – Using hardware programmer
    – ISP

◆ Finish the board design

◆ Program the device

◆ Challenge
  • Board design
  • System considerations
Altera Design Flow

◆ Operate seamlessly with other EDA tools
MAX+PLUS II
Altera’s Fully-Integrated Development System

Design Entry
- MAX+PLUS II Text Editor
- MAX+PLUS II Waveform Editor
- MAX+PLUS II Floorplan Editor
- MAX+PLUS II Graphic Editor
- MAX+PLUS II Symbol Editor

Project Verification
- MAX+PLUS II Simulator
- MAX+PLUS II Timing Analyzer
- MAX+PLUS II Waveform Editor

Project Processing
- MAX+PLUS II Compiler
- CNF Extractor
- Database Builder
- Logic Synthesizer
- SNF Extractor
- Partitioner
- Fitter
- Netlist Writer
- Design Doctor
- Assembler

Device Programming
- MAX+PLUS II Programmer

Message Processor & Hierarchy Display
Design Entry

◆ MAX+PLUS II design entry tools
  • Graphic Editor & Symbol Editor
    – For schematic designs
  • Text Editor
    – For AHDL and VHDL designs
    – However, VHDL is not covered by this course
  • Waveform Editor
  • Floorplan Editor
  • Hierarchy Display
MAX+PLUS II Design Entry

```
SUEDesign time_cnt
|
  enable, clk : INPUT;
  time[7..0] : OUTPUT;
}

VARIABLE
  count[7..0] : DFF;
BEGIN
  count[0], clk = clk;
  time[0] = count[0];
```

Project Processing

◆ MAX+PLUS II tools for project processing (implementation)
  • MAX+PLUS II Compiler
  • MAX+PLUS II Floorplan Editor
    – For pin, logic cell location assignments
  • Message Processor
    – For error detection & location
MAX+PLUS II Project Processing
Project Verification

MAX+PLUS II tools for project verification

- MAX+PLUS II Simulator
- MAX+PLUS II Waveform Editor
- MAX+PLUS II Timing Analyzer
MAX+PLUS II Project Verification
Device Programming

◆ MAX+PLUS tool for device programming
  • MAX+PLUS II Programmer
MAX+PLUS II Features

◆ MAX+PLUS II, Altera’s fully integrated design environment
  • Schematic, text (AHDL), waveform design entry & hierarchy display
  • Floorplan editing
  • DRC, logic synthesis & fitting, timing-driven compilation
  • Multi-device partitioning
  • Automatic error location
  • Functional simulation, timing simulation, and multi-device simulation
  • Timing analysis
  • Programming file generation & device programming
  • EDA interface: industry-standard library support, EDA design entry & output formats (EDIF, Verilog & VHDL)
  • On-line help
Getting Started

- System Requirements
- Installing MAX+PLUS II
- Starting MAX+PLUS II
- Entering Authorization Codes
- MAX+PLUS II Manager Window
- MAX+PLUS II Project
- Hierarchy Display
System Requirements

◆ The minimum system requirements
  • Pentium- or 486-based PC
  • Microsoft Windows NT 3.51 or 4.0, Windows 95, or Windows version 3.1x with Win32s support
  • Microsoft Windows-compatible graphics card & monitor
  • Microsoft Window-compatible 2- or 3-button mouse
  • CD-ROM drive
  • Parallel port

◆ Memory & disk space requirement
  • Go to the read.me file for specific information about disk space & memory requirements in the current version of MAX+PLUS II
    – At least 64MB physical RAM is recommended
    – Memory requirement depends on the selected device and the design complexity
Installing MAX+PLUS II

◆ To install MAX+PLUS II from CD-ROM
  • Insert MAX+PLUS II CD-ROM into the CD-ROM drive. The installation program is located at:
    <CD-ROM drive>:\pc\maxplus2\install.exe
  • Follow the directions provided on-screen
  • Window 3.1x users:
    – Installation program will install Win32s files if they are not already present

◆ Additional Windows NT installation steps
  • You must install Sentinel driver after running the install program
    – To detect the key-pro
  • (Optional) ByteBlaster and Altera LP6 Logic Programmer Card drivers
    – Required only for ByteBlaster or LP6 users
Starting MAX+PLUS II

◆ To start MAX+PLUS II...
  • Double click on the MAX+PLUS II icon
Entering the Authorization Code

◆ When starting MAX+PLUS II for the first time
◆ Options -> license setup
  - You must enter an authorization code obtained from CIC
  - You can use all most MAX+PLUS II features after enter the correct auth-code
MAX+PLUS II Operating Environment

◆ MAX+PLUS II Manager
  • Start-up window

- Toolbar provides shortcuts for commonly used functions
- Status bar provides a brief description of selected menu command and toolbar button
- Project Directory and Project name
- Help menu gives you access to on-line help
- MAX+PLUS II menu gives you access to all MAX+PLUS II functions
MAX+PLUS II Menu

To invoke MAX+PLUS II applications
File Menu

[Image of file menu for Altera Max+Plus II software]
Assign Menu

To specify project assignments & options
Options Menu

To setup user preferences
Help Menu

[Image of MAX+Plus II Help Menu]

- MAX+plus II Table of Contents
- MAX+plus II Manager Help
  - AHDL
  - VHDL
  - MegawonionLPM
  - CMOS-Style MacroFunctions
  - Primitives
  - Devices & Adapters
- Messages
- Glossary
- README
  - New Features in This Release
  - How to Use MAX+plus II Help
- How to Use Help
- About MAX+plus II...
MAX+PLUS II Help Contents

◆ On-line help

- All of the information necessary to enter, compile, and verify a design and to program an Altera device is available in MAX+PLUS II Help.
- Help also provides introductions to all MAX+PLUS II applications, design guidelines, pin and logic cell numbers for each Altera device package.
MAX+PLUS II
Design Methodology
Design Entry

- Design Specification
- Design Entry
  - Command-Line Mode
- Design Compilation
- Functional Verification
- Timing Verification
- Device Programming
- In-System Verification
- System Production
- Design Modification
Design Entry Process

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

◆ To specify a project

  Menu: File -> Project -> Name... (To specify an existing or new design file)
  Menu: File -> Project -> Set Project to Current File (To specify the current design file)
Set Up A New Project

- Every design must have a project name
- Project name must match design file name

![Set up a new project in MAX+PlusII](image)
Design Entry Files

Top-level design files can be .gdf, .tdf, .vhd, .v, .sch, or .edf

- MAX+PLUS II Symbol Editor
- MAX+PLUS II Graphic Editor
- MAX+PLUS II Text Editor
- MAX+PLUS II Floorplan Editor

- OrCAD
- Synopsys, Synplicity, Mentor Graphics, etc...

- Verilog
- VHDL
- AHDL
- Waveform
- Schematic

Generated within MAX+PLUS II
Imported from other EDA tools

© CIC Altera Max+PlusII  Y.T.Chou/Steven
Hierarchy Display

◆ MAX+PLUS II Hierarchy Display
  • The Hierarchy Display shows a hierarchy tree that represents the current hierarchy and allows you to open and close files in the hierarchy.
  • The hierarchy tree branches show a filename and file icon for each subdesign in the hierarchy, and it also shows ancillary files associated with the current hierarchy.
  • To get a better perspective on your project, you can zoom in and out to different display scales or switch between vertical or horizontal orientation.
  • To invoke Hierarchy Display

Menu: MAX+PLUS II -> Hierarchy Display

```
chiptrip.gdf  ---+--- time_cnt:4.tdf
             |        |         |
             +--- tick_cnt:10.gdf  +--- 8count:8.tdf  +--- f8count:sub.gdf
                     |        |         |
                     +--- speed_ch:2.wdf  +--- auto_max:1.tdf
```
Hierarchy Display Window
Graphic Design Entry

◆ MAX+PLUS II Graphic Editor & Symbol Editor
◆ Basic Knowledge
  • Naming Rules
  • User Libraries & System Libraries
◆ Creating Graphic Design Files
◆ Examples
Graphic Design Entry Process

- **Add resource libraries to search list as needed**
- **Draw schematic**
  - Enter design components (symbols)
  - Connect components with net (wires)
  - Add labels to key nets signal
    - Must label all busses, primary inputs, outputs, bidir

  *Note: MAX+PLUS II DOES NOT AUTO SAVE*
- **Save and check the design**
  - The file extension is .gdf
  - Correct any errors with the aid of Message Processor
- **Create symbol or include file for sub-design**
Resource Libraries

- **prim (Altera primitives)**
  - Basic logic building blocks

- **mf (Macrofunction)**
  - 7400 family logic

- **mega_lpm (LPsMs)**
  - Library of Parameterized Modules (LPMs)
  - Megafuctions are high level function module
    - busmux, ram elements, fifo’s, etc...
Value added Libraries

◆ MegaCores IP models you can try before purchase (download from www.altera.com)
  – UARTs, FFT, PCI etc…

◆ AMPP (Altera Megafuction Partners Program)
  • Partners providing PCI, DSP, µControllers, etc…

Note: For the latest information on MegaCores or Megafuctions, refer to Altera’s web site www.altera.com
Add User Resource Libraries

◆ Access user created libraries
  - Add user library directories
  - Set priorities

Select the library directory then click on Add

Library search priority can be changed.

The Project directory has the highest priority, followed by the User Libraries, then by the Altera Libraries.
Open New File & Enter Symbols

- Open a new .gdf file in Graphic Editor
- *Double click* in Graphic file to enter symbol
Graphic Editor Window

- Selection tool
- Text tool
- Orthogonal line tool
- Diagonal line tool
- Arc tool
- Circle tool

Zoom functions

Rubberbanding functions

INPUT symbols

Node

Symbol

Bus

OUTPUT symbol
Making Connections

◆ **Wire**
  - Single bit line

◆ **Bus**
  - Multi-bit line

◆ **Signal name**
  - Matching name
  - Attached to wire

Wire to Bus Connection

Bus - Bus signal names required for LPM module buses

Drawing tool shortcuts
Graphic Editor Options

- **Font, Text Size**
  - Text Control
- **Line Style**
  - Select Wire or Bus
- **Display Assignments**
  - Turns display on or off
- **Guideline Control**
  - Controls grid lines
- **Rubber-banding**
  - Wires move with symbols
Generate Symbols and Include Files

- Create symbol for higher-level schematic capture
- Create include file for AHDL or Verilog function prototype
Symbol Editor

◆ Symbols can be modified with the Symbol Editor
Pin/Node Naming

◆ Pin/node name
  • A pin name is enclosed within a pin primitive symbol; a node name is a text block that is associated with a node line (wire).

◆ Pin/node naming rules
  • It can contain up to 32 name characters
  • It may not contain blank spaces. Leading or trailing spaces are ignored.
  • It must be unique, i.e., no two pins may have the same name in the same design file at the same hierarchy level.
  • Any node that is connected to a bus line must be named
  • Node names that are bits of a dual-range bus must be expressed in the format <name>[<width>][<size>] or <name><width>_<size>. If you name a single node in this format, it will be interpreted as part of a dual-range bus if another single-range or dual-range bus in the file uses the same <name>. 
Bus Naming

◆ Single-range bus name
  • Example: D[3..0] = D3,D2,D1,D0
  • The bus identifier can contain up to 32 name characters; the bus width can contain a maximum of 256 bits. The bus width is a string that defines the number of bits (i.e., nodes) in a bus and uses the form [<MSB>..<LSB>]. The name of a single node within the bus can be specified with the identifier followed by the bit number, either with or without brackets.

◆ Dual-range bus name
  • Example: D[3..0][1..0] = D3_1,D3_0,D2_1,D2_0,D1_1,D1_0
  • A dual-range bus name uses two bracket-enclosed ranges [:]: the bus width and the bus size. Bus widths and sizes can together define a maximum of 256 bits.

◆ Sequential bus name
  • Example: A[31..0],B,C[3..0]
  • A sequential bus name consists of a series of node names and/or bus names, separated by commas (,). The first node or bus bit in the series is the MSB, the last node in the series is the LSB.
Using Buffer Primitives - (1)

◆ **Buffer primitives**
  - Including: CARRY, CASCADE, EXP, GLOBAL, LCELL, OPNDRN, SOFT, TRI
  - All buffer primitives except TRI and OPNDRN allow you to control the logic synthesis process. In most circumstances, you do not need to use these buffers.

◆ **GLOBAL primitive**
  - To indicate that a signal must use a global clock, clear, preset or output enable signal, instead of signals generated with internal logic or driven by ordinary I/O pins
  - A NOT gate may be inserted between the input pin and GLOBAL

◆ **TRI primitive**
  - A active-high tri-state buffer

◆ **OPNDRN primitive**
  - An open-drain buffer, equivalent to a TRI primitive whose output enable input is fed by an signal, but whose primary input is fed by a GND primitive
  - Only supported for the FLEX 10K and MAX 7000S device families
Using Buffer Primitives - (2)

◆ **LCELL primitive**
  
  • The **LCELL** buffer allocates a logic cell for the project. An LCELL buffer always consumes one logic cell. It’s not removed from a project during logic synthesis.
  
  • Although **LCELL** primitives can be used to create an intentional delay or asynchronous pulse
    
    – However, race conditions can occur and create an unreliable circuit because the delay of these elements varies with temperature, power supply voltage and device fabrication process

◆ **SOFT primitive**
  
  • The **SOFT** buffer specifies that a logic cell may be needed in the project
  
  • During project processing, MAX+PLUS II Compiler examines the logic feeding the primitive and determines whether a logic cell is needed. If it’s needed, the **SOFT** buffer is converted into an **LCELL**; if not, the **SOFT** buffer is removed
More on LPM Libraries

◆ **Library of Parameterized Modules**
  - Standard Library of basic and functional elements
  - Based on EDIF standard

◆ **Advantage of LPMs**
  - Portability of design
  - Architecture independence

◆ **MAX+PLUS II and LPMs**
  - LPM can be used in graphical design and HDL designs
  - LPM can be customized via the Megawizard feature
Standard LPM without Megawizard
Using MegaWizard Plug-In Manager

◆ Click on the MegaWizard Plug-In Manager Button

Double click in Graphic Editor

Click on the MegaWizard Plug-In Manager
Accessing the MegaWizard

Select MegaWizard Plug-In Manager

The MegaWizard Plug-In Manager helps you create or modify design files that contain custom variations of megafunctions.

Which action do you want to perform?
- Create a new custom megafunction variation
- Edit an existing custom megafunction variation

Copyright (c) Altera Corporation 1988-1998. All rights reserved.
New vs Existing Megafunction

Choose between a new custom megafunction variation or an existing megafunction variation

New Custom Megafunction

Edit Existing Custom Megafunction
Available Megafuncitons & Output File

Select a function from the available megafunctions.

Select a type of output file:
- AHDL
- VHDL
- Verilog HDL

Select a directory and a output file name:
- c:\altera\mplus\tap_mux

Note: To compile a project successfully in the MAX+PLUS II software, your design files must be in the project directory or a user library that you specify with the User Libraries command (Options menu).

Your current user library directories are:
Customizing the Megafunction

MegaWizard Plug-In Manager - LPM_MUX [page 3 of 4]

- How many 'data' inputs do you want?
  - 4

- How wide should the 'data' input and the 'result' output buses be?
  - 8

- Do you want to pipeline the multiplexer?
  - No

- Create an asynchronous Clear input

Options:
- Cancel
- < Back
- Next >
- Finish
Files generated by the MegaWizard

Design file implemented in the language you selected (.tdf, .vhd, or .v)

INC an AHDL include file

CMP a VHDL component declaration file

SYM a Graphic design symbol file
Entering Customized Megafuntion

Customized megafunction appears the same way as other symbols in the Enter symbol window.

Double click in Graphic Editor.
Make Changes to Customized Megafunction

Double Click symbol will bring you back to the MegaWizard Plug-in Manager.

After the changes, MegaWizard will over-write the source file (tdf, vhd, v), inc file and cmp file for you.

Remember to update the symbol in your graphic editor.
How to Use System Functions?

◆ To get help...

- You can find the detailed description for each primitive, macrofunction, and megafunction in MAX+PLUS II on-line help.
Entering Symbols

◆ Enter a symbol
  Menu: Symbol -> Enter Symbol...
  (or by double clicking on the empty workspace)

◆ Move/cut/copy/paste symbols
  • You can move, cut, copy or paste symbols in the same way as you did in another Windows-based software
    - Move: click & drag (mouse)
    - Cut: Ctrl-X
    - Copy: Ctrl-C or Ctrl-Click & drag
    - Paste: Ctrl-V
    - Undo: Ctrl-Z

◆ Commands regarding the symbol
  • Just click the right mouse button on the symbol
Entering I/O Symbol

◆ **I/O symbols**
  - Input pin/port: enter a **INPUT** symbol
  - Output pin/port: enter a **OUTPUT** symbol
  - Bidirectional pin/port: enter a **BIDIR** symbol

◆ **Name the I/O pins/ports**
  - Double click on the "**PIN_NAME**" field of the I/O symbol

◆ **Pin default value**
  - The values assigned to unconnected INPUT and BIDIR primitives when the symbol that represents the current GDF file is used in a higher-level design file
  - Default is VCC
  - Double click on the "**VCC**" field to set the default value
Save & Check the Design

- Save & check the design file with .gdf extension
- Correct any errors with the aid of Message Processor
Message Processor

- Lists all Info, Warning and Error messages
  - Info messages are general information
  - Warning messages are possible problems
  - Error messages indicate Compiler is unable to complete compilation process
- Provides help on the messages
- Locates source of message in design file

Images:
- A screenshot of the Message Processor window showing an error message "Can't find design file 'not'" with options to go to next or previous message, locate in design file, and help on message.
- Labels: Messages, Information about message, Locate source in design file.
Example: Multiplier

◆ Design a multiplier with \texttt{LPM\_MULT}
  - The easiest way to create a multiplier is to use the \texttt{LPM\_MULT} function
    - Can be unsigned or signed
    - Can be pipelined
    - Also can create a MAC(Multiplier-Accumulator) circuit

```
INPUT_A_IS_CONSTANT="NO"
INPUT_B_IS_CONSTANT="NO"
LPM_PIPELINE=3
LPM_REPRESENTATION="UNSIGNED"
LPM_WIDTH=A8
LPM_WIDTHP=(LPM\_WIDTH+A8)
LPM_WIDTHS=LPM\_WIDTH
```

![Multiplier Diagram](image)
Example: Multiplexer

◆ Design a multiplexer with LPM_MUX
  
  • Use WIRE primitive to rename a bus or node
  • LPM_MUX data input is a dual range bus
Example: RAM

◆ Design RAM circuit with LPM
  - Use `LPM_RAM_IO` to design RAM with a single input & output port
  - Use `LPM_RAM_DQ` to design RAM with separate input & output ports
Example: Sequencer

◆ Design a sequencer with **LPM_COUNTER & LPM_ROM**

- ROM data is specified in a Memory Initialization File (.mif) or a Intel-Hex File (.hex)
- This example only sequences through 19 states so the modulus of lpm_counter is set to 19. It uses a small section of an EAB (19 out of 256-address locations)
Example: Bidirectional Pin

- **Use TRI & BIDIR pin symbol**
  - If the TRI symbol feeds to a output or bidirectional pin, it will be implemented as tri-state buffer in the I/O cell.
Example: Tri-State Buses - (1)

**Tri-state emulation**

- Altera devices do not have internal tri-state buses
- MAX+PLUS II can *emulate* tri-state buses by using multiplexers and by routing the bidirectional line outside of the device and then back in through another pin.

*MAX+PLUS II will automatically convert it into a multiplexer. If the tri-state buffers feed a pin, a tri-state buffer will be available after the multiplexer.*
Example: Tri-State Buses

Tri-state buses for bidirectional communication

- When tri-state buses are used to multiplex signals, MAX+PLUS II will convert the logic to a combinatorial multiplexer.

- When tri-state buses are used for bidirectional communication, you can route this bidirectional line outside of the device, which uses the tri-states present at the I/O pins, or you can convert the tri-state bus into a multiplexer.
Example: Tri-State Buses

Rout this bidirectional line outside of the device

Tri-state emulation
Text Design Entry

- **Set up a new project**
  - Same as Graphic Design Entry

- **Enter text description**
  - AHDL
  - VHDL
  - Verilog

- **Save & check the design**
  - Similar to Graphic Design Entry
  - The file extension is .tdf, .vhd, .v
AHDL

- Altera Hardware Description Language
- High-level hardware behavior description language
- Uses Boolean equations, arithmetic operators, truth tables, conditional statements, etc.
- Especially well-suited for large or complex state machines
- Text Editor has AHDL Template and Syntax Color
- Refer to the Appendix for more info on AHDL
VHDL

- VHSIC Hardware Description Language
- 1987 and 1993 IEEE 1074 standard
- High-level hardware behavior description language
- Especially well-suited for large or complex designs
- Text Editor has VHDL Template and Syntax Color
Verilog

- Hardware Description Language
- 1993 Verilog IEEE 1364 standard
- High-level hardware behavior description language
- Especially well-suited for large or complex designs
- Text Editor has Verilog Template and Syntax Color
MAX+PLUS II Text Editor

Features of MAX+PLUS II Text Editor

- AHDL templates & examples
- AHDL context-sensitive help
- Syntax coloring
- Error location
- Resource & device assignments
HDL Templates

HDL templates make design easier

- You can insert HDL template into your TDF, then replace placeholder variables in the templates with your own identifiers and expressions

Menu: Templates -> HDL Template...
Inserting HDL Template

```vhdl
SUBDESIGN _design_name_
{
  _input_name, _input_name  : INPUT = constant_value;
  _output_name, _output_name : OUTPUT;
  _bidir_name, _bidir_name  : BIDIR;
  _state_machine_name       : MACHINE INPUT;
  _state_machine_name       : MACHINE OUTPUT;
}
```
Using Syntax Coloring

◆ Syntax Coloring command
  • To improve TDF readability & accuracy
    Menu: Options -> Syntax Coloring

◆ To customize the color palette
  Menu: Options -> Color Palette...
  • The HDL-relative options:
    – Comments
    – Illegal Characters
    – Megafuctions/Macrofunctions
    – Reserved Identifiers
    – Reserved Keywords
    – Strings
    – Text

![Color Palette](image-url)
Text Editor with Syntax Coloring

CONSTANT NORTH = $"00";  % Create descriptive name for numbers
CONSTANT EAST = $"01";   % for use elsewhere in file
CONSTANT WEST = $"10";
CONSTANT SOUTH = $"11";
SUBDESIGN auto_max
{
    dic[1..0], accel, clk, reset : INPUT;  % File input:
speed_too_fast, at_altera, get_ticket : OUTPUT; % File output
}

VARIABLE
street_map : MACHINE
OF BITS (q2,q1,q0)
WITH STATES {
yc,                 % Your company
mpld,              % Marigold Park Lane Drive
epld,              % East Pacific Lane Drive
gdf,               % Great Delta Freeway
cnf,               % Capitol North First
upt,               % Regal Park Terrace
epm,               % East Pacific Main
acy,               % Your own street name here!
Creating Text Design Files

◆ Open a new design file
  Menu: File -> New... -> Text Editor file (.tdf)

◆ Save as a TDF file
  Menu: File -> Save As...

◆ Set project to the current TDF file
  Menu: File -> Project... -> Set Project to Current File

◆ Edit the TDF
  • Turn on syntax coloring option
  • Use AHDL Template & on-line help if necessary
  • Follow the AHDL style guide mentioned in MAX+PLUS II Help

◆ Save the file & check for basic errors
  Menu: File -> Project -> Project Save & Check
Example: Decoder

◆ Design a decoder with...
  • If-Then statements
  • Case statements
  • Table statements
  • LPM function: LPM_DECODE

```vhdl
SUBDESIGN decoder
(
  code[1..0] : INPUT;
  out[3..0]  : OUTPUT;
)
BEGIN
  CASE code[] IS
    WHEN 0 => out[] = B"0001";
    WHEN 1 => out[] = B"0010";
    WHEN 2 => out[] = B"0100";
    WHEN 3 => out[] = B"1000";
  END CASE;
END;
```

```vhdl
SUBDESIGN priority
(
  low, middle, high   : INPUT;
  highest_level[1..0] : OUTPUT;
)
BEGIN
  IF high THEN
    highest_level[] = 3;
  ELSIF middle THEN
    highest_level[] = 2;
  ELSIF low THEN
    highest_level[] = 1;
  ELSE
    highest_level[] = 0;
  END IF;
END;
```
Example: Counter

◆ Create a counter with DFF/DFFE or LPM_COUNTER

```
SUBDESIGN ahdlcnt
(
    clk, load, ena, clr, d[15..0] : INPUT;
    q[15..0]           : OUTPUT;
)
VARIABLE
    count[15..0] : DFF;
BEGIN
    count[].clk = clk;
    count[].clrn = !clr;
    IF load THEN
        count[].d = d[];
    ELSIF ena THEN
        count[].d = count[].q + 1;
    ELSE
        count[].d = count[].q;
    END IF;
    q[] = count[];
END;
```

```
INCLUDE "lpm_counter.inc"
SUBDESIGN lpm_cnt
(
    clk, load, ena, clr, d[15..0] : INPUT;
    q[15..0]           : OUTPUT;
)
VARIABLE
    my_cntr: lpm_counter WITH (LPM_WIDTH=16);
BEGIN
    my_cntr.clock = clk;
    my_cntr.aload = load;
    my_cntr.cnt_en = ena;
    my_cntr.aclr = clr;
    my_cntr.data[] = d[];
    q[] = my_cntr.q[];
END;
```
Example: Multiplier

◆ Design a multiplier with \texttt{LPM\_MULT}

\begin{verbatim}
CONSTANT WIDTH = 4;
INCLUDE "lpm_mult.inc";

SUBDESIGN tmul3t
  (
    a[WIDTH-1..0] : INPUT;
    b[WIDTH-1..0] : INPUT;
    out[2*WIDTH-1..0] : OUTPUT;
  )

VARIABLE
  mult : lpm_mult WITH (LPM\_REPRESENTATION="SIGNED",
      LPM\_WIDTHA=WIDTH, LPM\_WIDTHB=WIDTH,
      LPM\_WIDTHS=WIDTH, LPM\_WIDTHP=WIDTH*2);

BEGIN
  mult.dataa[] = a[];
  mult.datab[] = b[];
  out[] = mult.result[];
END;
\end{verbatim}
Example: Multiplexer

◆ Design a multiplexer with LPM_MUX

FUNCTION lpm_mux (data[LPM_SIZE-1..0][LPM_WIDTH-1..0], sel[LPM_WIDTHS-1..0])
    WITH (LPM_WIDTH, LPM_SIZE, LPM_WIDTHS, CASCADE_CHAIN)
    RETURNS (result[LPM_WIDTH-1..0]);

SUBDESIGN mux
(
    a[3..0], b[3..0], c[3..0], d[3..0]    : INPUT;
    select[1..0]                        : INPUT;
    result[3..0]                        : OUTPUT;
)
BEGIN
    result[3..0] = lpm_mux (a[3..0], b[3..0], c[3..0], d[3..0], select[1..0])
        WITH (LPM_WIDTH=4, LPM_SIZE=4, LPM_WIDTHS=2);
END;
Example: RAM

◆ Design RAM circuit with LPM

```
INCLUDE "lpm_ram_dq.inc";

SUBDESIGN ram_dq
  (clk : INPUT;
   we : INPUT;
   ram_data[31..0] : INPUT;
   ram_add[7..0] : INPUT;
   data_out[31..0] : OUTPUT;
  )
BEGIN

  data_out[31..0] = lpm_ram_dq (ram_data[31..0], ram_add[7..0], we, clk, clk)
  WITH (LPM_WIDTH=32, LPM_WIDTHAD=8);

END;
```
**Example: Tri-State Buses**

◆ **Design tri-state buses with TRI**

```vhdl
SUBDESIGN tribus
(
  ina[7..0], inb[7..0], inc[7..0], oe_a, oe_b, oe_c, clock : INPUT;
  out[7..0] : OUTPUT;
)

VARIABLE
  flip[7..0] : DFF;
  tri_a[7..0], tri_b[7..0], tri_c[7..0] : TRI;
  mid[7..0] : TRI_STATE_NODE;

BEGIN
  -- Declare the data inputs to the tri-state buses
  tri_a[] = ina[];  tri_b[] = inb[];  tri_c[] = inc[];
  -- Declare the output enable inputs to the tri-state buses
  tri_a[].oe = oe_a;  tri_b[].oe = oe_b;  tri_c[].oe = oe_c;
  -- Connect the outputs of the tri-state buses together
  mid[] = tri_a[];  mid[] = tri_b[];  mid[] = tri_c[];
  -- Feed the output pins
  flip[].d = mid[];  flip[].clk = clock;  out[] = flip[].q;
END;
```
Example: Moore State Machine

Moore state machine
- The outputs of a state machine depend only on the state

```vhdl
SUBDESIGN moore1
(
    clk : INPUT;
    reset : INPUT;
    y : INPUT;
    z : OUTPUT;
)
VARIABLE
ss: MACHINE OF BITS (z)
    WITH STATES (s0 = 0, s1 = 1, s2 = 1, s3 = 0);
% current_state =
BEGIN
    ss.clk = clk;
    ss.reset = reset;
    TABLE
        ss, y => ss;
        s0, 0 => s0;
        s0, 1 => S2;
        s1, 0 => s0;
        s1, 1 => s2;
        s2, 0 => s2;
        s2, 1 => s3;
        s3, 0 => s3;
        s3, 1 => s1;
    END TABLE;
END;
```
Example: Mealy State Machine

**Mealy state machine**
- A state machine with asynchronous output(s)

```plaintext
SUBDESIGN mealy
(
  clk : INPUT;
  reset : INPUT;
  y : INPUT;
  z : OUTPUT;
)
VARIABLE
  ss: MACHINE WITH STATES (s0, s1, s2, s3);
BEGIN
  ssclk = clk;
  ssreset = reset;
TABLE
  ss, y => z, ss;
  s0, 0 => 0, s0;
  s0, 1 => 1, s1;
  s1, 0 => 1, s1;
  s1, 1 => 0, s2;
  s2, 0 => 0, s2;
  s2, 1 => 1, s3;
  s3, 0 => 0, s3;
  s3, 1 => 1, s0;
END TABLE;
END;
```
Waveform Design Entry

◆ MAX+PLUS II Waveform Editor
◆ Creating Waveform Files
◆ Examples
◆ Design Entry Summary
MAX+PLUS II Waveform Editor

◆ Features of MAX+PLUS II Waveform Editor
  • To serve 2 roles:
    – As a design entry tool: to create Altera waveform design files (*.wdf)
    – As a tool for entering test vectors & viewing simulation results: simulation channel files (*.scf)

◆ For design entry
  • Waveform design entry is best suited for circuits with well-defined sequential inputs & outputs, such as state machines, counters, and registers

◆ For design verification
  • Waveform Editor is a simulation pattern editor/viewer
  • Waveform Editor is fully integrated with MAX+PLUS II Simulator & Programmer to provide full project verification flow
MAX+PLUS II
Waveform Design Environment

Zoom functions
Waveform values overwriting functions
Pop-up menu (clicking mouse B2/B3)
File Menu
Node Menu

To enter the node or group information
Edit Menu

To edit the waveform value
Creating a New Waveform File

◆ Open a new design file
  Menu: File -> New... -> Waveform Editor file (.wdf or .scf)

◆ Save as a WDF / SCF file
  Menu: File -> Save As... ->

◆ Set project to current file (for WDF file only)
  Menu: File -> Project... -> Set Project to Current File
Setting Waveform Editor Options

◆ Set the grid size & show the grid
  Menu: Options -> Grid Size...
  Menu: Options -> Show Grid
  • Setting appropriate grid size is helpful for waveform repeating & overwriting count value operations

◆ Specify the end time
  Menu: File -> End Time...

◆ Regarding the grid size & interval...
  • In a WDF, the grid size & interval are arbitrary. The time scale indicates only a sequential order of operations, not a specific response time.
  • In a SCF, the grid size & interval are important for timing simulation. MAX+PLUS II Simulator reflects the real-world timing according to your SCF and the specific device. Setup & Hold time violation will occur if you enter impractical simulation patterns.
Entering Nodes

◆ Insert the node or group for WDF file

Menu: Node -> Insert Node... (or double click on the node name field)

- You can specify the node name, I/O type, node type & default value
  - Registered & machine node type must specify a clock signal and optionally specify reset or preset signal (active high)
  - You can specify machine values with the state names instead of logic values
Entering Nodes from SNF

◆ Enter the node or group for SCF file

Menu: Node -> Enter Nodes from SNF...

- SNF: Simulation Netlist File
  - Generated by MAX+PLUS II Compiler (discussed later)
  - After compilation, you can list the nodes and help you to create the SCF file
Editing Waveforms - (1)

◆ Edit the waveforms
  - First select the interval to edit
    - Sometimes you may specify new grid size for easy selection
  - To create clock-like waveform
    Menu: Edit -> Overwrite -> Clock...
  - To edit the state machine node values
    Menu: Edit -> Overwrite State Name...

![Overwrite Clock](image1)

![Overwrite State Name](image2)
Editing Waveforms - (2)

◆ Edit the waveforms
  • To edit the node values
    Menu: Edit -> Overwrite -> 0 / 1 / X / Z / Invert / Count Value / Group Value
  • To stretch / compress the selected signal
    Menu: Edit -> Grow or Shrink...

  - To align node values or state names to grid if necessary
    Menu: Edit -> Align to Grid
Saving & Checking the Design

- **Save the WDF/SCF file**
  
  Menu: *File -> Save*

- **Check basic errors for the WDF file**
  
  Menu: *File -> Project -> Project Save & Check*
Waveform File Formats

◆ MAX+PLUS II file formats
  • Binary format: WDF & SCF files
  • ASCII format (Altera vector file format): TBL & VEC files
    – TBL: an ASCII-format table file that records all logic level transitions for nodes and groups in the current SCF or WDF
    – VEC: an ASCII text file used as the input for simulation, functional testing, or waveform design entry
    – Refer to MAX+PLUS II Help for detailed information about vector file format

◆ To create a table file (*.tbl)
  Menu: File -> Create Table File...

◆ To import a vector file (*.vec)
  Menu: File -> Import Vector File...
WDF Design Guidelines

◆ When design a WDF file...
  - WDFs cannot be at intermediate levels of a hierarchy
  - Include all possible combinations of input values
  - Align all logic level and state name transition
  - Assume a 0ns propagation delay for all logic
  - Assume a 0.1ns setup time and 0ns hold time for state machine node
  - For clarity, Altera recommends that you draw inputs that affect registers only on falling clock edges
  - If a function is cyclical, show the last set of conditions looping back to the first by repeating the first time-slice at the end of the cycle
Example: Decoder

◆ When design a decoder...
  • Use “Overwrite Count Value” to help create all possible combinations of decoder input values, and then manually edit the output waveforms

<table>
<thead>
<tr>
<th>Name:</th>
<th>Type:</th>
<th>50.0ns</th>
<th>100.0ns</th>
<th>150.0ns</th>
<th>200.0ns</th>
<th>250.0ns</th>
<th>300.0ns</th>
<th>350.0ns</th>
<th>400ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>in[3..0]</td>
<td>INPUT</td>
<td>F</td>
<td>E</td>
<td>D</td>
<td>C</td>
<td>B</td>
<td>A</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>decout[15..0]</td>
<td>COMB</td>
<td>8000</td>
<td>4000</td>
<td>2000</td>
<td>1000</td>
<td>0800</td>
<td>0400</td>
<td>0200</td>
<td>0100</td>
</tr>
</tbody>
</table>
Example: Counter

◆ When design a counter
  • Use “Overwrite Count Value” command to create a regular counter waveform
Example: State Machine

◆ When design a state machine
  - Use “Overwrite State Name” to help create a state machine output
    - You can specify machine values with the state names instead of logic values
  - Make sure all possible combinations of inputs and states are included
Design Entry Summary

MAX+PLUS II

Symbol Editor
Waveform Editor
Graphic Editor
Text Editor

User

3rd Party EDA Tools

Design Files
Support Files

.gdf
.tdf
.wdf
.vdf
.vhd
.sch
.edf

.sym
.inc
.lmf
Design Implementation

- MAX+PLUS II Compiler
- Preparing for Compilation
- Compiling the Project
- Analyzing the Compilation Results
- Floorplan Editor
- Appendix: Interfacing with 3rd-Party Tools
To invoke MAX+PLUS II Compiler
Menu: MAX+PLUS II -> Compiler
MAX+PLUS II Compiler

◆ Process all design files associated with the project
  • Files can be created with MAX+PLUS II or 3rd party EDA Tools
◆ Checks for syntax errors and common design pitfalls
◆ Performs logic synthesis and place & route
  • According to assignments in .acf file
◆ Generates files for simulation and timing analysis
  • Files can be used by MAX+PLUS II or 3rd party EDA Tools
◆ Generates files for programming targeted devices
Compiler Input and Output Files

3rd Party EDA
Design Files (.edf, .sch)
Mapping Files (.imf)

MAX+PLUS II
Compiler

MAX+PLUS II Design Files (.gdf, .tdf, .vhd, .v, .wdf)
Assignments (.acf)

3rd Party EDA
Simulation/Timing Files (.edo, vo, vho, sdo)

MAX+PLUS II Compiler

Compiler Netlist Extractor (includes all netlist readers)
Functional, Timing, or Linked SNF Extractor
EDIF, VHDL & Verilog Netlist Writers
Database Builder
Logic Synthesizer
Partitioner
Fitter
Design Doctor
Assembler

Functional SNF Files (.snf)
Timing SNF Files (.snf)
Programming Files (.pof, .sof, .jed)
Compiler Input Files

◆ **Design files**
  - MAX+PLUS II
    - Graphics file (.gdf), AHDL file (.tdf), VHDL file (.vhd), Verilog (.v), Waveform file (.wdf)
  - 3rd Party EDA Tools
    - EDIF file (.edf)
      - Select Vendor in EDIF Netlist Reader Settings
      - Library Mapping File (.lmf) required for vendors not listed
    - OrCAD file (.sch)

◆ **Assignment and Configuration File (.acf)**
  - Controls the Compiler’s synthesis and place & route operations
  - Automatically generated when user enter assignments
  - Automatically updated when user changes assignments or back-annotates project
Compiler Output Files

◆ Design verification files
  • MAX+PLUS II
    – Simulation Netlist File (.snf)
  • 3rd Party EDA Tools
    – VHDL netlist file (.vho)
    – EDIF netlist file (.edo)
    – Verilog netlist file (.vo)
    – Standard Delay Format SDF file (.sdo)

◆ Programming files
  • Programmer Object file (.pof)
  • SRAM Object file (.sof)
  • JEDEC file (.jed)
For EDIF Netlist Input

For EDIF input, the EDIF Reader Settings need to be selected.
VHDL Netlist Reader Settings

Select VHDL version and Library names
For EDIF Netlist Output

Need to select vendor in EDIF Netlist Writer settings.

For EDIF output, the EDIF Netlist Writer needs to be selected.
Verilog Netlist Writer & Writer Settings

Select Verilog Netlist Writer to output Verilog file for Verilog simulator

Select Verilog Netlist Writer Settings to adjust Verilog outputs
VHDL Netlist Writer & Writer Settings

Select VHDL Netlist Writer Settings to adjust the VHDL output.

Select VHDL Netlist Writer to output VHDL file for VHDL simulator.
Imported Design

◆ **Top-level Design: can be read in directly**
  - EDIF Netlist files
  - OrCAD schematics
    • Refer to MAX+PLUS II Read Me file for the version of 3rd Parties tools it interface with

◆ **Lower-level modules**
  - EDIF, OrCAD schematics files
    • Create symbols or files to instantiate component
  - Other proprietary files
    • JEDEC, ABEL, PALASM
    • Conversion utilities exist in Altera ftp site
Compiler Modules - (1)

Compiler Netlist Extractor
- The Compiler module that converts each design file in a project (or each cell of an EDIF input file) into a separate binary CNF (Compiler Netlist File)
- The Compiler Netlist Extractor also creates a single HIF that documents the hierarchical connections between design files
- This module contains a built-in EDIF Netlist Reader, VHDL Netlist Reader, and XNF Netlist Reader for use with MAX+PLUS II.
- During netlist extraction, this module checks each design file for problems such as duplicate node names, missing inputs and outputs, and outputs that are tied together.
- If the project has been compiled before, the Compiler Netlist Extractor creates new CNFs and a HIF only for those files that have changed since the last compilation, unless Total Recompile (File menu) is turned on
Compiler Modules - (2)

◆ Database Builder

- The Compiler module that builds a single, fully flattened project database that integrates all the design files in a project hierarchy
- As it creates the database, the Database Builder examines the logical completeness and consistency of the project, and checks for boundary connectivity and syntactical errors (e.g., a node without a source or destination)
Compiler Modules  - (3)

◆ Logic Synthesizer
  - The Compiler module that synthesizes the logic in a project's design files.
  - The Logic Synthesizer calculates Boolean equations for each input to a primitive and minimizes the logic according to your specifications.
  - The Logic Synthesizer also synthesizes equations for flip-flops to implement state registers of state machines.
  - As part of the logic minimization and optimization process, logic and nodes in the project may be changed or removed.
  - Throughout logic synthesis, the Logic Synthesizer detects and reports errors such as illegal combinatorial feedback and tri-state buffer outputs wired together ("wired ORs").

◆ Design Doctor Utility
  - The Compiler utility that checks each design file in a project for poor design practices that may cause reliability problems when the project is implemented in one or more devices.
Compiler Modules - (4)

Partitioner

- The Compiler module that partitions the logic in a project among multiple devices from the same device family
- Partitioning occurs if you have created two or more chips in the project's design files or if the project cannot fit into a single device
- This module splits the database updated by the Logic Synthesizer into different parts that correspond to each device
- A project is partitioned along logic cell boundaries, with a minimum number of pins used for inter-device communication
Compiler Modules - (5)

◆ Fitter

- The Compiler module that fits the logic of a project into one or more devices
- Using the database updated by the Partitioner, the Fitter matches the logic requirements of the project with the available resources of one or more devices
- It assigns each logic function to the best logic cell location and selects appropriate interconnection paths and pin assignments
- The Fitter module generates a “fit file”(*.fit) that documents pin, buried logic cell, chip, clique, and device assignments made by the Fitter module in the last successful compilation
- Regardless of whether a fit is achieved, the Fitter generates a report file(*.rpt) that shows how the project is implemented in one or more devices
Compiler Modules - (6)

◆ SNF(Simulation Netlist File) Extractor

• Functional SNF Extractor
  – The Compiler module that creates a functional SNF containing the logic information required for functional simulation.
  – Since the functional SNF is created before logic synthesis, partitioning, and fitting are performed, it includes all nodes in the original design files for the project

• Timing SNF Extractor
  – The Compiler module that creates a timing SNF containing the logic and timing information required for timing simulation, delay prediction, and timing analysis
  – The timing SNF describes a project as a whole. Neither timing simulation nor functional testing is available for individual devices in a multi-device project.

• Linked SNF Extractor
  – The Compiler module that creates a linked SNF containing timing and/or functional information for several projects
  – A linked SNF of a super-project combines the timing and/or functional information for each project, allowing you to perform a board-level simulation
Compiler Modules - (7)

◆ Netlist Writer

• EDIF Netlist Writer
  – The Compiler module that creates one or more EDIF output files (*.edo). It can also generate one or more optional SDF output files (*.sdo).
  – EDIF output Files contain the logic and timing information for the optimized project and can be used with industry-standard simulators. An EDIF Output File is generated for each device in a project.

• Verilog Netlist Writer
  – The Compiler module that creates one or more Verilog output files (*.vo). It can also generate one or more optional SDF output files.

• VHDL Netlist Writer
  – The Compiler module that creates one or more VHDL output files (*.vho). It can also generate one or more optional VITAL-compliant SDF output files.
Assembler

- The Compiler module that creates one or more programming files for programming or configuring the device(s) for a project
- The assembler generates one or more device programming files
  - POFs and JEDEC Files are always generated
  - SOFs, Hex Files, and TTFs are also generated if the project uses FLEX devices
  - You can generate additional device programming files for use in other programming environment. For example, you can create SBF and RBF to configure FLEX devices.
- File format:
  - POF: Programming Object File
  - SOF: SRAM Object File
  - TTF: Tabular Text File
  - HEX: Intel-format Hexadecimal File
  - SBF: Serial Bitstream File
  - RBF: Raw Binary File
Compiling a Project

- Select functional compilation or timing compilation
- Assignments
- Run the compilation
- Consult the report file (.rpt) or the Floorplan Editor for device utilization summaries and synthesis and place & route results
The Functional Compilation Process

- Compiler Netlist Extractor builds the .cnf netlist file and checks for syntax errors
- Database Builder constructs the node name database
- Functional SNF Extractor build .snf file for functional simulation
The Timing Compilation Process

- Compiler Netlist Extractor and Database Builder build netlist database and check for syntax errors
- Logic Synthesizer performs logic synthesis/minimization
- Design Doctor checks for design violations
- Partitioner and Fitter executes place & route algorithm and builds the .rpt file on device implementation
- Timing SNF Extractor builds .snf file for simulation and timing analysis
- Assembler builds files for programming the device
Compiler Processing Options

◆ Functional
• Compilation generates file for Functional Simulation
  – Functional SNF file (.snf)

◆ Timing
• Compilation generates user selectable files for
  – Timing Simulation and Timing Analysis
    • Timing SNF file (.snf)
  – 3rd party EDA Simulation
    • Verilog file (.vo)
    • VHDL file (.vho)
    • SDF file (.sdo)
  – Device Programming
    • Altera Programmer file (e.g. .pof, .sof)
Compilation Process Settings - (6)

- Customize the report file settings

Menu: Processing -> Report File Settings...
Compilation Process Settings - (7)

◆ “Smart Recompile” & “Total Recompile”
  • The first time the Compiler processes a project, all design files of that project are compiled
  • Use “Smart Recompile” feature to create an expanded project database that helps to accelerate subsequent compilations
    - Allow you to change physical device resource assignments without rebuilding the database & resynthesizing the project
  • Use “Total Recompile” feature to force the Compile to regenerate database & resynthesize the project
    Menu: Processing -> Smart Recompile
    Menu: Processing -> Total Recompile
Assign Menu

- You can specify optional sections to be included in the report file (*.rpt), which is created by the Fitter when a project is compiled
  - All sections are included by default
Assignments Control

◆ Device FIT
  • MAX+PLUS II default settings are designed for maximum fit-ability
  • Almost all assignments affect fitting

◆ Device Utilization
  • Circuit design
  • Logic assignment

◆ Performance
  • Circuit design
  • Logic assignments
  • Logic placements
Assignments

◆ Most common Assignments
  • Device assignments
  • Pin assignments

◆ Other assignments
  • Logic options
  • architectural features
  • Location assignments
    – Lab, Row, Column, LC
  • Clique
  • timing assignments
  • Device Option assignments
Making Device Assignment

◆ Select Device
  - Specific device
  - Auto
    - MAX+PLUS II chooses smallest and fastest device the design fits into
Making Pin Assignment

◆ Highlight node in graphic or text source file
  • Assign > Pin/Location/Chip

◆ Floorplan Editor can also be used (discussed later)

Highlight node and choose Assign Pin/Location/Chip

Node name automatically entered in the Node Name field

Choose pin or LCELL location then click on Add to enter assignment
(Note: You must choose a specific device prior to this step)
Logic Synthesis Style

◆ The most common way toward adjusting these assignments is to apply the predefined Logic Synthesis Style toward the different portion of your design:
  • Normal
  • Fast
  • WYSIWYG

◆ Each of the Logic Synthesis Styles is a collection of both logic synthesis options and individual architectural settings
Global Project Logic Synthesis Style

◆ Choose Assign then Global Project Logic Synthesis
◆ Select from predefined synthesis style
  • NORMAL (default), FAST or WYSIWYG
◆ Or create user tailored settings
Assign Logic Synthesis Style Locally

Select Logic Synthesis Style
Individual Logic Option Assignment

- **Provides controls to turn individual** architectural features and synthesis algorithms **on or off**
  - Gray or Default (default): set by higher level or global setting
  - Check or Auto: enable feature
  - Blank or Ignore: disable feature

![Screen shot of Individual Logic Options window with options for Slow Slew Rate, XOR Synthesis, Turbo Bit, Hierarchical Synthesis, Implement as Output of Logic Cell, Insert Additional Logic Cell, Increase Input Delay, Parallel Expanders, Ignore SOFT Buffers, Use LPM for AHDL Operators, Implement in EAB, Fast I/O, Global Signal, Disable Fast Feedback Path, Minimization, Cascade Chain, Carry Chain, Carry Chain, Max. Auto Length, OK, Cancel, Use Default, Advanced Options.](image)
Location assignments

Select Pin/Location/Chip ...

Select Location
Clique Assignments

Clique assignments tell the compiler to place the nodes with the same clique assignment close together inside the device.
Timing Requirements Assignments

FLEX devices only

◆ Specifies desired speed performance
◆ Use after performing timing analysis to improve specific timing path
◆ Localized control
  • Highlight node, pin or logic block
  • Choose Assign then Timing Requirements
  • Assign desired tpd, tco, tsu, fmax values
◆ Global control
  • Choose Assign then Global Project Timing Requirements
  • Assign desired tpd, tco, tsu, fmax values
Assignment Recommendation

◆ Start with device and pin assignments. Beware, your pin assignments might affect performance. Ideally, you should let MAX+PLUS II choose the pin assignments. If you have pin assignments, you might want to compile your design once without your pin assignments to see if they affect your performance.

◆ Compile design. Check device utilization and performance.

◆ If you need to adjust device utilization or performance try the other assignments. Try the synthesis style assignments first.

◆ Assignments can only be made to “hard” nodes or lower-level designs that contains hard nodes. Hard nodes are objects that translate directly into objects in silicon e.g. Flip-flops, LCELLs and I/O pins
Ignore or Clear Assignments

Ignore Project Assignments

Clear Project Assignments
Global Project Device Options

Global Project Device Options Window contains options related to the operation of the device rather than options that affect the logic synthesis and place & route of the design.

For example,

FLEX Device
• configuration scheme
• multi-volt I/O

MAX Device
• Enable JTAG support
• security bit
More Compiler Processing Options

- **Design Doctor**
  - Checks for common design errors

- **Fitter Settings**
  - Set place & route options

- **Smart Recompile**
  - Faster compilation time

- **Total Recompile**
  - Recompile every file
Compile the Design

- Start Button starts compilation process
- Messages are displayed by the Message Processor
  - Info
  - Warning
  - Error
The Report File

- Project summary
  - Device assignments
  - Error summary
  - Device pin-out diagram (useful for PCB layout)

- Resource utilization
  - Pin
  - LCELL
  - Equations

- Compiler resources
  - Compilation time
  - Memory usage

Open report file by double clicking on the rapt icon
Checking the Messages

◆ Check the messages in Message Processor
  • In Message Processor window, choose the message and click the HELP on Message to understand the meaning of the message, its cause and the possible solutions (suggested actions)

◆ Error location
  • In Message Processor window, choose the message and click the Locate button to locate the source of the message in the original design files
  • You can turn on Locate in Floorplan Editor and click Local All button to find the corresponding nodes in the Floorplan Editor
Help on Message

Node <node> has assignments but doesn't exist or is a primitive array

CAUSE: You assigned resources to a node that does not exist. The node either never existed in the project's design files, or was removed during logic synthesis and minimization. Or, you may have entered a resource assignment on a primitive that is connected to one or more busses to form a primitive array. This message may also occur if you assigned resources to a macroFunction that does not contain any hard logic functions when you used pre-version 5.0 synthesis.

ACTION: Check the design files for accuracy. Remove unused resource assignments and make sure you have not entered assignments on symbols in a primitive array. Refer to Guidelines for Working with Assignments for more information on how to enter assignments for a primitive array.

See also:
- Deleting an Assignment
- MAX+PLUS II Messages
- Message Format
- Partitioning a Project
Checking the Reports

◆ Check the report file
  • Use Text Editor or double click the Report File icon.
  • Device summary, project compilation messages, file hierarchy, resource usage, routing resources, logic cell interconnections, ...
Viewing Report File

** DEVICE SUMMARY **

<table>
<thead>
<tr>
<th>Chip/Device</th>
<th>Input</th>
<th>Output</th>
<th>Bidir</th>
<th>LCS</th>
<th>% Utilized</th>
</tr>
</thead>
<tbody>
<tr>
<td>chiptrip EPF6282ALC84-2</td>
<td>6</td>
<td>13</td>
<td>0</td>
<td>80</td>
<td>38 %</td>
</tr>
</tbody>
</table>

User Pins:
6
13
0

S
Project Information d:\altera0\max0\tutorial\chiptrip.rpt

** PROJECT COMPILATION MESSAGES **

Warnings: Ignored all pin assignments as requested in the Ignore Project Assignments...
Pin-out file (.pin)

◆ An ASCII file that contains the pin out of your device. It is created as a pin-out file for a board layout tool.

N.C. = Not Connected.
VCCINT = Dedicated power pin, which MUST be connected to VCC (5.0 volts).
VCCIO = Dedicated power pin, which MUST be connected to VCC (5.0 volts).
GNDINT = Dedicated ground pin or unused dedicated input, which MUST be connected to GND.
GNDIO = Dedicated ground pin, which MUST be connected to GND.
RESERVED = Unused I/O pin, which MUST be left unconnected.

CHIP "filter" ASSIGNED TO AN EPF10K10QC208-3
TCK : 1
CONF_DONE : 2
nCEO : 3
TDO : 4
VCCIO : 5
VCCINT : 6
N.C. : 7
N.C. : 8
N.C. : 9
x7 : 10
Floorplan Editor

- Graphical user interface for viewing/creating resource assignments
  - Pins
  - Logic cells
  - Cliques
  - Logic options
- Drag-and-drop capability for assigning pins/logic cells
- Graphical view of current assignments as well as last compilation results
- LAB view or external chip view
Floorplan Views

Assign-> Back-Annotate Project

Current Assignment Floorplan (Editable View)

Last Compilation Floorplan (Read-Only)
Floorplan Editor (Read Only)

- Last Compilation Floorplan Full Screen LAB View with Report File Equation Viewer
Floorplan Editor (Read Only)

◆ Last Compilation Floorplan Device View

- Pin number
- Pin name
- Color Legend definition
Floorplan Editor (Editable)

- Current Assignment view has drag and drop capability (Note: Auto Device can not be used)

Click on Node, hold left mouse, drag to location
Floorplan Editor (Editable)
Project Compilation
Recommendations

- Use assignments after design analysis to improve fitting or performance
- Use the Report File to find specific information on the design
- Use the Floorplan Editor to see results of Assignments
Routing Statistics

Routing Statistics

Information on Selected Node/Pin/LAB

Name is: **Multiple Items**
Number is: LC1_B7  LAB is: B7
Row is: 8  Column is: 7
Logic Cell Fan-In: 5  Logic Cell Carry-Out: Yes
Logic Cell Fan-Out: 9  Logic Cell Cascade-Out: No

Cell Total Shared Expanders Used:
Embedded Cell Depth (Bits):
LAB Total Shared Expanders Used:
LAB External Interconnect Used: 6/24 (25%)
Column Interconnect Channels Used: 3/16 (19%)
Full Row Interconnect Channels Used: 128/168 (76%)
Half Row Interconnect Channels Used:
Logic Cell Inputs Borrowed from LC1:

Most Congested Areas in Current Chip

Most Congested LAB (or EAB) is: A20
LAB (or EAB) External Interconnect Used: 15/24 (62%)
Most Congested Row is: 8
Row Interconnect Channels Used: 128/168 (76%)
Most Congested Column is: 6
Column Interconnect Channels Used: 10/16 (62%)
Floorplan Editor Utilities Menu

◆ To find text, node, ...
  • “Find Text” command: to search the current chip for the first occurrence of the specified text
  • “Find Node” command: to find one or more nodes or other logic function(s) in the design file or in the floorplan

◆ To help running timing analysis
  • You can specify source and destination nodes in the floorplan to run timing analysis
Assigning Logic to Physical Resources

◆ Use Floorplan Editor to assign logic to physical resources
  • You can assign logic to a device, to any row or column within a device, or to a specific LAB, pin, logic cell, or I/O cell in Floorplan Editor very easily
  • To toggle between current assignment & last compilation floorplan
    Menu: Layout -> Current Assignments Floorplan
    Menu: Layout -> Last Compilation Floorplan

◆ Back-annotate the floorplan for subsequent compilation
  • If necessary, you can back-annotate the floorplan to ACF(Assignment & Configuration File) and it is useful for retaining the current resource and device assignments for future compilations
  Menu: Assign -> Back-Annnotate Project...
Current Pin Assignment Floorplan
Current LAB Assignment Floorplan

Anywhere on Device

Anywhere on this Column

Row A

Anywhere on this Row

Any Col

Col 1

Col 2

Col 3

Col 4

Col 5

Col 6

Col 7

Col 8

Col 9

Any Row

AOD
Project Compilation Summary

Design Files
- .wdf
- .gdf
- .tdf
- .v
- .vhd
- .sch
- .edf

MAX+PLUS II Compiler
- Compiler Netlist Extractor (includes all netlist readers)
- Database Builder
- Logic Synthesizer
- Functional, Timing, or Linked SNF Extractor
- Partitioner
- Fitter
- EDIF, VHDL & Verilog Netlist Writers
- Design Doctor
- Assembler

Simulation/Timing Files
- .snf

Programming Files
- .pof

Report Files
- .rpt

3rd Party EDA Files
- .sdo
- .edo
- .vo
- .vho
Project Verification

- Project Verification Methodology
- MAX+PLUS II Simulator
- Functional Simulation
- Timing Simulation
- Timing Analysis
Project Verification Methodology

MAX+PLUS II Timing Analyzer

MAX+PLUS II Simulator

MAX+PLUS II Waveform Editor

.snf  .mif  .hex  .cm  .scf  .vec

.tao  .hst  .sif  .log  .tbl  .tbl
MAX+PLUS II Simulator

- MAX+PLUS II Compiler
- MAX+PLUS II Waveform Editor
- MAX+PLUS II Text Editor
- .snf
- .scf
- .vec
- MAX+PLUS II Waveform Editor
- .scf
MAX+PLUS II Simulation

◆ Create Simulation Stimulus
  • Waveform
  • Vector

◆ 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
Simulation Waveform

◆ Stimulus Waveform
  • Waveform Editor File (.scf)
  • Control
    – Clock: Use built-in clock generator
    – Others: Hand drawn with overwrite/copy/paste/repeat
  • Data
    – Counting patterns: Use built-in binary or gray code generator
    – Others: Enter with overwrite/copy/paste/repeat

◆ Reference Compare waveform
  • Waveform Editor File (.scf)
  • Draw or save previous simulation result as reference waveform
  • Use with Compare after new simulation run to verify output
Create Waveform Simulation Stimulus

- Open Waveform Editor
- Select Enter Nodes from SNF... from Node menu
- Enter Nodes into Selected Nodes & Groups field

Select Node
Enter Node into Selected Nodes & Groups field
Simulator Environment
Grid Control

◆ Snap to Grid
  • On: waveforms drawn increments of grid size
  • Off: waveforms can be drawn to any size
Draw Stimulus Waveform

- Highlight portion of waveform to change
- Overwrite with desired value (Group value or single bit)
Create Clock Waveform

- Snap to Grid On: Clock Period is twice the grid size
- Snap to Grid Off: Clock Period can be any value
Create Counting Pattern

- Make sure your counting frequency matches your clock frequency

Specify counting frequency

Specify counting pattern

Highlight waveform

Pattern shortcut

Overwrite a single selected node or group waveform with a specified count sequence
Grouping Signals and Set Radix

- Highlight waveforms to be grouped
  - MSB must be the top waveform
- Enter Group Name and set Radix
Save the Waveform Stimulus File

- Save the waveform stimulus file with .scf extension
- MAX+PLUS II will use Project name as default file name
Create Vector Simulation Stimulus

- Open **Text Editor**
- Type in vector stimulus
  - **Clock**
    - % units default to ns %
    - START 0 ;
    - STOP 1000 ;
    - INTERVAL 100 ;
    - INPUTS CLOCK ;
    - PATTERN
      - 0 1 ; % CLOCK ticks every 100 ns %
  - **Pattern**
    - INPUTS A B ;
    - PATTERN
      - 0>  0 0
      - 220>  1 0
      - 320>  1 1
      - 570>  0 1
      - 720>  1 1;
  - **Output**
    - OUTPUTS Y1 Y0 ;
    - PATTERN % check output at every Clock pulse %
      - = X X
      - = 0 0
      - = 0 1
      - = 1 0
      - = 1 1;
Save the Vector Stimulus File

- Save the vector stimulus file with .vec extension
  - You must change the .vec extension since MAX+PLUS II defaults to .tdf extension for text files

Change the extension to .vec
Select Simulation Stimulus File

- Defaults to .scf file
- For vector input stimulus, set Vector Files Input to .vec file

Set to .vec file
Specify Length of Simulation

- Specify maximum length of simulation time with End Time
Run Functional Simulation

- Click on Start then Open SCF to see result
MAX+PLUS II Functional Simulation

◆ Use to verify operation of design

◆ Advantage over Timing Simulation
  • Fast compilation
  • All nodes are retained and can be simulated
  • Outputs are updated without delay
    – Most of the time, this makes figuring out cause and effect much easier

◆ Disadvantages
  • Logical model only, no logic synthesis
  • No delays in simulation
    – Oscillations, glitches and other timing related errors do not show up
Run Timing Simulation

- Click on Start then Open SCF to see result

Output change after timing delay
MAX+PLUS II Timing Simulation

- Used to debug timing related errors
- Advantages over Functional Simulation
  - Simulation of full synthesis result
  - Outputs change after timing delay
    - Detection of oscillations, glitches and other timing related errors are possible
- Disadvantages
  - Longer compilation time
  - Combinatorial logic nodes cannot be simulated
    - Node may be transformed or removed
  - Only “Hard” nodes can be simulated
  - Timing delays make debugging more difficult because cause and effect relationships are harder to locate
Comparing Different Simulations

Compare Two Simulation Files

◆ Open first channel file
◆ Choose Compare under File menu
◆ Select the name of the second channel file with the Compare dialog box
◆ Waveforms from the first channel file are drawn in black. Waveforms from the second channel file are drawn in red on top of the black waveforms. Deviations of second channel file can easier be spotted.
Project Simulation
Recommendations

- Use built-in clock generator to create clock
- Use built-in count generator to create test pattern
- Use Functional Simulation to verify proper operation
- Use Timing Simulation to examine signal delay effects
- Use Compare function to verify output
- Use the dynamic link (Find Node in Design File) to go to source file to make any necessary corrections
Simulation Input & Output Files

◆ Specify simulation input and output files
  • You can specify SCF or VEC file as the source of simulation input vectors
    Menu: File -> Inputs/Outputs...
    – VEC file will be converted into SCF file by Simulator
    – You can specify a history(*.hst) or log(*.log) file to record simulation commands and outputs
  • During and after simulation, the simulation results are written to the SCF file, you can create another ASCII-format table file
    Menu: File -> Create Table File...
    – TBL file format is a subset of VEC file format
    – A TBL file can be specified as a vector input file for another simulation
Memory Initialization

◆ Give memory initialization values for functional simulation

• To generate memory initialization values in Simulator
  Menu: Initialize -> Initialize Memory...

• You can save the data in the Initialize Memory dialog box to a Hexadecimal File (*.hex) or Memory Initialization File (*.mif) for future use
  Menu: Initialize -> Initialize Memory... -> Export File...
  – An MIF is used as an input file for memory initialization in the Compiler and Simulator. You can also use a Hexadecimal File (.hex) to provide memory initialization data.

• You can load the memory initialization data for a memory block that is saved in a HEX or MIF file
  Menu: Initialize -> Initialize Memory... -> Import File...

Initialize Menu
Initialize Memory Window
Memory Initialization File Formats

```
WIDTH = 16;
DEPTH = 256;
ADDRESS_RADIX = HEX;
DATA_RADIX = HEX;
CONTENT BEGIN
  0 : 0000;  :020000000000fe
  1 : 0000;  :020001000000fd
  2 : 0000;  :020002000000fc
  3 : 0000;  :020003000000fb
ADDRESS_RADIX = HEX;
DATA_RADIX = HEX;
CONTENT BEGIN
  0 : 0000;  :020004000000fa
  1 : 0000;  :020005000000f9
0 : 0000;  :020006000000f8
1 : 0000;  :020007000000f7
2 : 0000;  :020008000000f6
3 : 0000;  :020009000000f5
4 : 0000;  :02000a000000f4
5 : 0000;  :02000b000000f3
6 : 0000;  :02000c000000f2
7 : 0000;  :02000d000000f1
8 : ffff;  :02000e000000f0
9 : ffff;  :02000f000000ff
a : ffff;  :0200ff000000ff
b : ffff;  :00000001ff
```

**MIF file example**

```
... ff : 0000;
END;
```

**HEX file example**
MIF File Format

To edit a MIF file...

- MIF file is an ASCII text file that specifies the initial content of a memory block
  - You can create an MIF in the MAX+PLUS II Text Editor or any ASCII text editor
  - You can also very easily generate an MIF by exporting data from the Simulator's Initialize Memory dialog box

- Example:

```
DEPTH = 32;  % Memory depth and width are required%
WIDTH = 14;  % Enter a decimal number%
ADDRESS_RADIX = HEX;  % Address and value radixes are optional%
DATA_RADIX = HEX;  % Enter BIN, DEC, OCT or HEX(default)%

-- Specify values for addresses, which can be single address or range
CONTENT
BEGIN
  [0..F] : 3FFF;  % Range--Every address from 0 to F = 3FFF%
  6 : F;  % Single address--Address 6 = F%
  8 : F E 5;  % Range starting from specific address%
END;
```
Notes for Compiling & Simulating RAM / ROM

◆ Remember: MAX+PLUS II Compiler uses MIF or HEX file(s) to create ROM or RAM initialization circuit in FLEX 10K EAB
  • Specify the LPM_FILE parameter to a MIF or HEX file for each RAM and ROM block
    – Memory initialization file is optional for RAM
    – Using MIF files is recommended because its file format is simple

◆ If the memory initial file does not exist when MAX+PLUS II Compiler is generating functional SNF file, you must initialize the memory by using Initialize Memory command before starting the functional simulation
  • MAX+PLUS II Compiler reports an warning when it can’t read the memory initialization file when processing Functional SNF Extractor
  • However, the memory initialization file must exist when MAX+PLUS II processes Timing SNF Extractor
Notes for Compiling & Simulating RAM / ROM

If you do not have MIF or HEX files, do the following:

- Run MAX+PLUS II Compiler to generate a functional SNF file first
- Then invoke MAX+PLUS II Simulator, use Memory Initialization command to create memory content for each ROM or RAM block
- Export memory content to a MIF or HEX file
  - And now, you can perform functional simulation for your project
- Invoke MAX+PLUS II Compiler again, turn on “Timing SNF Extractor” and start complete compilation for FLEX 10K devices
Node/Group Initialization

◆ Specify initial logic levels for nodes/groups
  • You can change the initial logic levels of registered nodes/groups in the SNF file for the project before you begin simulation
  
  Menu: Initialize -> Initialize Nodes/Groups...
  - You can also specify an initial state name for a group that represents a state machine.
  • By default, all register outputs are initialized to 0 and pin inputs are initialized to the first logic level provided in the current SCF
Saving Initialization Values

◆ Save the initialization values to SIF file
  • You can save current initialized node and group logic levels and memory values to a Simulation Initialization File (*.sif)
    
    Menu: Initialize -> Save Initialization As...
  • To retrieve initialized node, group, and memory values stored in a SIF file
    
    Menu: Initialize -> Restore Initialization...
  • To reset initial node, group, and memory values to the values stored in the SNF file
    
    Menu: Initialize -> Reset to Initial SNF Values
      – All register outputs are initialized to 0, and pin inputs are initialized to the first logic level provided in the current SCF file
Creating Breakpoints

- **Specify simulation breakpoints**
  - You can create one or more breakpoints, each of which consists of one or more node value, group value, and time conditions
  - **Menu:** Options -> Breakpoints...
  - Specify breakpoint conditions
    - `.TIME` variable in Node/Group list represents the simulation time
    - Operator: `=, !=, >, <, >=, <=, >->` (transition)
    - A breakpoint can consist one or more conditions and must be given a unique name
Monitoring Options

◆ **Setup time & hold time**
  - You can instruct the Simulator to monitor all simulated nodes and groups for setup time and hold time violations
    - It's not available in functional simulation mode
    - In timing simulation linked simulation mode, setup and hold time violations are determined by the architecture of the device(s) being simulated

◆ **Glitch**
  - You can instruct the Simulator to monitor the logic levels of all simulated nodes and groups for glitches or spikes, i.e., two or more logic level changes that occur within a period less than or equal to the specified time
    - It's not available in functional simulation mode

◆ **Oscillation**
  - The Simulator can monitor all simulated nodes and groups for logic levels that do not stabilize within the specified time period after the most recent input vector has been applied
    - In functional simulation mode, oscillation option is always on and check only for nil-period oscillation
Project Simulation Summary

◆ Two types of simulation
  • Functional simulation
    – No logic synthesis
    – No delay model
    – All nodes can be simulated
  • Timing simulation
    – Logic synthesis
    – Delay model
    – Only hard nodes can be simulated

◆ Two types of stimulus file
  • Waveform
  • Vector

◆ Simulation result is stored in .scf file
MAX+PLUS II Timing Analyzer

- MAX+PLUS II Compiler
- .snf
- Delay Matrix
- Setup/Hold Matrix
- Registered Performance

- MAX+PLUS II Floorplan Editor
- MAX+PLUS II Graphic Editor
- MAX+PLUS II Text Editor
Project Timing Analysis

• Timing Analyzer is a static timing analyzer
• Three forms of timing analysis
  • Registered Performance calculates fastest possible internal clock frequency
  • Delay Matrix calculates combinatorial delays
  • Setup/Hold Matrix calculates setup & hold times for device flip-flops
• Source of delay path can be located in
  • Design file
  • Floorplan Editor
Timing Analysis Source & Destination

- **Specify source/destination nodes for timing analysis**
  - The Timing Analyzer provides default timing tagging for source and destination nodes for each analysis mode
    - Menu: Node -> Timing Analysis Source...
    - Menu: Node -> Timing Analysis Destination...
  - Besides, you can specify timing analysis source & destination nodes in the Graphic, Waveform or Floorplan Editor (under Utilities Menu)
Registered Performance Analysis

- Calculates maximum internal register frequency
- Used to determine if design meets clock specification

Clock period = tco + delay + tsu + tskew

Note: tskew is added to the clock period if destination clock edge is earlier than source clock edge
Run Registered Performance Analysis

- Click on Start
- Source/Destination, Clock period and Frequency of the longest path are displayed
- Click on List Paths to trace delay path
Tracing Delay Path In Floorplan Editor

- Highlight Path of interest
- Check Locate in Floorplan Editor
- Click on Locate All
- Click on show path to display path
Application of Registered Performance

- Use Registered Performance Analysis to see if design meets clock frequency requirement

- What to do if frequency is less than desired
  - Use List Path to display the worst case delays
  - Use Floorplan Editor to view the entire path
    - Are Logic Cells and pins scattered among different rows?
    - Can the Logic Cells benefit from carry/cascade chains (FLEX) or parallel expanders (MAX)?
  - Use Assignments (Clique, Logic Options, etc…) on the critical path to improve performance
  - If still less than desired, consider pipelining technique or different design implementations where appropriate
Delay Matrix Analysis

- Calculates combinatorial logic delays
- Typically used to evaluate input pin to output pin delay
- Internal point to point delay analysis is possible by setting node source and destination for analysis
Delay Matrix Source and Destination

- Set Source and Destination to be analyzed
Useful Analysis Options

- **Time Restrictions**
  - Show All Path
  - Show Only Longest Paths
  - Show Only Shortest Paths

- **Cell Width**
  - Control matrix display

- **Cut Off I/O Pin Feedback**
  - See next page

- **Cut Off Clear & Preset Paths**
  - No clear or preset delay analysis

- **List Only Longest Path**
  - List Path lists only longest path between two points
Cut Off I/O Pin Feedback

- Used to break bi-directional pin from the analysis
- When on, paths A and B true C false
- When off, path A, B and C are true
Run Delay Matrix Analysis

- Select Delay Matrix Analysis and click on Start button.
- Matrix shows all paths, longest path, or shortest path depending on Time Restrictions option selected.
- Use List Path to analyze the path of delays.
Setup/Hold Matrix Analysis

◆ Setup/Hold Matrix calculates setup & hold times for device flip-flops

◆ Setup
  • \( t_{setup} = t_{data} - t_{clock} + t_{setup} \)

◆ Hold
  • \( thold = t_{clock} - t_{data} + thold \)
Run Setup/Hold Matrix Analysis

- Click on Start button
- Setup/Hold times are displayed with respect to the clocks
Saving Timing Analysis Results

◆ Save the current Timing Analyzer results to a TAO File
  
  • Timing Analyzer can save the information in the current timing analysis display to an ASCII-format Timing Analyzer Output file (*tao)

  Menu: File -> Save Analysis As...
Listing & Locating Delay Paths

◆ To trace delay paths or clock paths in the design file

- After you run a timing analysis, you can list selected signal paths and locate them in the original design file(s) for the project
- Select the matrix cell or clock, click List Paths
- Select one of the delay paths shown in Message Processor, and click Locate to trace the path in the source file(s)
Listing & Locating Paths
Recommended Verification Flow

- **Functional simulation**
  - Perform functional simulation to verify the design functionality

- **Timing Analysis**
  - Perform static timing analysis to check overall performance
  - Find the delay paths

- **Timing simulation**
  - Perform timing simulation to verify real-world design timing & functionality

- **On-board test**
  - Program FPGA/CPLD device(s) and test the function & timing in system
Timing Analysis Recommendations

- Use Timing Analyzer to locate performance bottleneck
- Use Registered Performance Analysis to determine internal clock frequency performance of the design
- Use Show Only Longest Path Time Restrictions in Delay Matrix to get the longest delay time from input pin to output pin
- Use List Path and Locate in Floorplan Editor to view worst case paths
- Use List Path and Locate to trace through path in design file
- Use assignments and recompile to fine-tune performance
Project Timing Analysis Summary

◆ Timing Analyzer is a static timing analyzer
◆ Three modes of Timing Analysis
  • Registered Performance
  • Delay Matrix
  • Setup/Hold Matrix
◆ Provides ability to trace path through Floorplan Editor or design file
Device Programming

◆ Programming Methods
◆ Altera Configuration EPROM Family
◆ Altera Programming Hardware
  • PL-ASAP2 Stand-Alone Programmer
  • BitBlaster Download Cable
  • ByteBlaster Download Cable
◆ FLEX Device Configuration Schemes
◆ MAX+PLUS II Programmer
Altera Provide Method

◆ Altera provide different methods for
  • Program Device
    – MAX family
  • Configure Device
    – FLEX family
MAX Device

◆ Use Altera Stand Alone Programmer (ASAP2)

◆ Through JTAG port with ByteBlaster
  - JTAG for Single Device (MAX or FLEX)
  - JTAG Chain for Multiple Device (MAX & FLEX)
  - JAM for Single/Multiple Device (MAX & FLEX)

◆ 3rd Programmer
  - Data I/O
  - BP MicroSystem
    - http://www.bpmicro.com
FLEX Device

◆ Through JTAG port with ByteBlaster
  • JTAG for Single Device (FLEX or MAX)
  • JTAG Chain for Multiple Device (FLEX & MAX)

◆ Through PS port with ByteBlaster
  • FLEX for Single Device
  • FLEX Chain for Multiple Device

◆ Serial PROM
  • EPC1 (1Mbits, good for 6K/8K/10K)
  • EPC1441 (441Kbits, good for 6K/8K/10K10, 10K20, 10K30)
  • EPC1213 (213Kbits, only for 8K)
  • EPC1064 (64Kbits, only for 8K)
Configuration File Sizes

◆ FLEX device configuration file sizes
  • Each FLEX device has a different size requirement for its configuration data, based on the number of SRAM cells in the device
  • The following table summarizes the configuration file size required for each FLEX device
    - To calculate the amount of data storage space for multi-device configurations, simply add the file sizes for each FLEX device in the design

<table>
<thead>
<tr>
<th>Device</th>
<th>Data Size(bits)</th>
<th>Device</th>
<th>Data Size(bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPF8282A/V</td>
<td>40,000</td>
<td>EPF10K10</td>
<td>115,000</td>
</tr>
<tr>
<td>EPF8452A</td>
<td>64,000</td>
<td>EPF10K20</td>
<td>225,000</td>
</tr>
<tr>
<td>EPF8636A</td>
<td>96,000</td>
<td>EPF10K30</td>
<td>368,000</td>
</tr>
<tr>
<td>EPF8820A</td>
<td>128,000</td>
<td>EPF10K40</td>
<td>488,000</td>
</tr>
<tr>
<td>EPF81188A</td>
<td>192,000</td>
<td>EPF10K50</td>
<td>609,000</td>
</tr>
<tr>
<td>EPF81500A</td>
<td>250,000</td>
<td>EPF10K70</td>
<td>881,000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EPF10K100</td>
<td>1,172,000</td>
</tr>
</tbody>
</table>
Altera Configuration EPROM Family

◆ **Altera’s serial configuration EPROMs for FLEX devices**
  
  - Simple, easy-to-use 4-pin interface to FLEX devices
  - Available in OTP packages: 8-pin PDIP, 20-pin PLCC and 32-pin TQFP
  - Family member
    - **EPC1064**: 65,536 bit device with 5.0-V operation
    - **EPC1064V**: 65,536 bit device with 3.3-V operation
    - **EPC1213**: 212,942 bit device with 5.0-V operation
    - **EPC1**: 1,046,496 bit device with 5.0-V or 3.3-V operation
Configuration EPROM Block Diagram (1)

- EPC1064, EPC1213, or EPC1 in FLEX 8000A mode
Configuration EPROM
Block Diagram (2)

◆ EPC1 in FLEX 10K mode
Altera Programming Hardware

◆ Hardware to program and configure Altera devices

• For MAX 7000/E/S, MAX 9000 and Altera configuration EPROM(EPC-series) devices
  – Altera stand-alone programmer: PL-ASAP2 (PC platform)
  – 3rd-party universal programmer (PC platform)

• For MAX 7000S and MAX 9000 ISP, FLEX devices downloading
  – Altera BitBlaster download cable (RS-232 port)
  – Altera ByteBlaster download cable (parallel port of PC)

• Of course, you can use another 3rd-party universal programmer or download cable to program or configure Altera devices. In this chapter, we discuss Altera programming hardware only.
Altera Stand-Alone Programmer

◆ PL-ASAP2: Altera stand-alone hardware programmer
  
  • The Altera stand-alone programmer, PL-ASAP2, together with the appropriate programming adapters, supports device configuration and programming for Altera devices
    – All MAX devices
    – Altera serial configuration EPROM: EPC1/V, EPC1064/V, EPC1213
  
  • PL-ASAP2 includes an LP6 Logic Programmer card, an MPU and software
    – LP6 card generates programming waveforms and voltages for the MPU
    – MPU(Master Programming Unit) connects to LP6 card via a 25-pin ribbon cable and is used together with an appropriate adapter to program Altera devices
    – Optional FLEX download cable for configuring FLEX devices
Installing LP6 Card, MPU & Adapter

Connect the 25-pin flat ribbon cable to the LP6 card.
BitBlaster Download Cable

◆ Altera BitBlaster serial download cable
  • BitBlaster serial download cable allows PC and workstation users to
    – Program MAX 9000, MAX 7000S in-system via a standard RS-232 port
    – Configure FLEX devices in circuit via a standard RS-232 port
  • BitBlaster provides two download modes
    – Passive Serial(PS) mode: used for configuring all FLEX devices
    – JTAG mode: industry-standard JTAG implementation for programming or configuring FLEX 10K, MAX 9000, and MAX 7000S devices
  • BitBlaster status lights:
    – POWER: indicates a connection to the target system’s power supply
    – DONE: indicates device configuration or programming is complete
    – BUSY: indicates device configuration or programming is in process
    – ERROR: indicates error detection during configuration or programming
Installing BitBlaster

- Status lights
- Receptacle for pin 1
- 25-pin female port
- Baud rate dip switches
- to 10-pin male header on circuit board
ByteBlaster Download Cable

◆ Altera ByteBlaster parallel port download cable
  
  • ByteBlaster serial download cable allows PC users to
    – Program MAX 9000, MAX 7000S in-system via a standard parallel port
    – Configure FLEX devices in circuit via a standard parallel port
  
  • ByteBlaster provides two download modes
    – Passive Serial(PS) mode: used for configuring all FLEX devices
    – JTAG mode: industry-standard JTAG implementation for programming or configuring FLEX 10K, MAX 9000, and MAX 7000S devices

  • ByteBlaster download cable provides a fast and low-cost method for ISP and FLEX device configuration

  • ByteBlaster download cable uses identical 10-pin circuit board connector as the BitBlaster serial download cable
Installing ByteBlaster
BitBlaster & ByteBlaster Plug Connections

Dimensions are shown in inches.

<table>
<thead>
<tr>
<th>Pin</th>
<th>PS Mode</th>
<th>JTAG Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DCLK</td>
<td>TCK</td>
</tr>
<tr>
<td>2</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>3</td>
<td>CONF_DONE</td>
<td>TDO</td>
</tr>
<tr>
<td>4</td>
<td>VCC</td>
<td>VCC</td>
</tr>
<tr>
<td>5</td>
<td>nCONFIG</td>
<td>TMS</td>
</tr>
<tr>
<td>6</td>
<td>N.C.</td>
<td>N.C.</td>
</tr>
<tr>
<td>7</td>
<td>nSTATUS</td>
<td>N.C.</td>
</tr>
<tr>
<td>8</td>
<td>N.C.</td>
<td>N.C.</td>
</tr>
<tr>
<td>9</td>
<td>DATA0</td>
<td>TDI</td>
</tr>
<tr>
<td>10</td>
<td>GND</td>
<td>GND</td>
</tr>
</tbody>
</table>
FLEX Device Configuration Schemes

◆ Passive Serial (PS) configuration with the download cable
  • Single-device configuration
  • Multiple-devices configuration

◆ JTAG configuration with the download cable
  • Available for FLEX 10K and ISP devices only

◆ Serial configuration EPROM configuration
  • FLEX 8000A Active Serial (AS) configuration with serial configuration EPROM
  • FLEX 10K Passive Serial (PS) configuration with EPC1 configuration EPROM

◆ Parallel EPROM configuration
  • FLEX 8000A Active Parallel Up (APU) or Active Parallel Down (APD) configuration
  • Not available for FLEX 10K devices
FLEX 8000A Single-Device PS Configuration with the Download Cable
FLEX 8000A Multiple-Device PS Configuration with the Download Cable
FLEX 10K Single-Device PS Configuration with the Download Cable

PS. nCE pin of FLEX 10K device must connect to GND.
FLEX 10K Multiple-Device PS Configuration with the Download Cable

PS. nCE pin of the lead FLEX 10K device must connect to GND.
JTAG Configuration with the Download Cable

- FLEX 10K Device
  - TCK
  - TDO
  - TMS
  - TDI

10-Pin Male Header (JTAG Mode)

- Pin 1

- MAX 9000 Device
  - TCK
  - TDO
  - TMS
  - TDI

10-Pin Male Header (JTAG Mode)

- Pin 1

- Other JTAG 1149.1-Compliant Device

- Target Altera Device

- Other JTAG 1149.1-Compliant Device
FLEX 8000A Configuration EPROM Configuration (AS Mode)

AS configuration with automatic reconfiguration on error (nStatus pin is connected to OE pin of the configuration EPROM and when “Auto-Restart Configuration on Frame Error” option bit is turned on)

Serial configuration EPROM (e.g. Altera’s EPC1213)
FLEX 8000A Multiple Configuration EPROMs Configuration

```
+---+        +---+        +---+
| VCC |        | VCC |        | VCC |
+---+        +---+        +---+        +---+
| “0” |        | “0” |        | “0” |
| “0” |        | “0” |        | “0” |
| “0” |        | “0” |        | “0” |
| n/S/P |      | n/STATUS |      | n/S/P |
| CONF_DONE | 1 kΩ | n/STATUS | 1 kΩ | CONF_DONE |
| MSEL1 |      | DATA |      | MSEL0 |
| MSEL0 |      | DATA |      | MSEL1 |
| DATA0 |      | DATA |      | DATA0 |
| n/CONFIG |      | DATA |      | n/CONFIG |
| Configuration EPROM | Configuration EPROM |
| n/CS | n/CASC | n/OE | DATA |
| n/CS | n/CASC | n/OE | DATA |
```

© CIC Altera Max+PlusII Y.T.Chou/Steven
FLEX 8000A Multi-Device Configuration EPROM Configuration
FLEX 10K Configuration EPROM Configuration (PS Mode)

Altera-supplied serial EPC1 configuration EPROM (EPC1: 1-Mbit EPROM)
FLEX 10K Multi-Device Configuration EPROM Configuration
FLEX 8000A APU & APD Configuration
MAX+PLUS II Programmer

◆ To program or configure Altera devices

- After the MAX+PLUS II Compiler has processed a project, it generates one or more programming files, which the Programmer uses to program or configure one or more devices
- The MAX+PLUS II Programmer allows you to program, verify, examine, blank-check, configure, and test Altera all MAX and FLEX devices and configuration EPROM
- With the Programmer and programming hardware--the Altera MPU, add-on cards, programming adapters, the FLEX download cable, the BitBlaster, or the ByteBlaster--you can easily create a working device in minutes
Programmer Operations

◆ 6 operations

- **Program**: programs data onto a blank MAX device or configuration EPROM
- **Verify**: verifies contents of a device against current programming data
- **Examine**: examines a device & stores the data in a temporary buffer
- **Blank-Check**: examines a device to ensure it is blank
- **Test**: functionally tests a programmed device
- **Configure**: downloads configuration data into the SRAM of one or more FLEX devices
Starting Programming

- Program or configure the Altera device
  - Setup the hardware
    Menu: Options -> Hardware Setup... -> Auto-Setup
  - Specify the programming file
    Menu: File -> Select Programming File...
  - Program or configure the device: just click on the Program or Configure button
Functional Test on Device

◆ Functionally test the Altera device

- You can use an SCF or VEC file, or test vectors stored in the current programming file, to functionally test actual device outputs against simulation outputs
  - Functional testing is not available for SRAM-based FLEX devices
  - You can only test devices for single-device projects
  - You also cannot test projects that contain bidirectional buses
- After the device is programmed, select simulation input file
  Menu: File -> Inputs/Outputs
  - You can specify an output Programmer Log File (*.plf) to record the Programmer's activities
- Test the device: click on Test button
Converting or Combining Programming Files

To convert or combine programming files

- You can combine and convert one or more SRAM Object Files (*.sof) into one of the following file formats, which support different FLEX device configuration schemes
  - Programmer Object File (*.pof)
  - Raw Binary File (*.rbf)
  - Tabular Text File (*.ttf)
  - Serial Bitstream File (*.sbf)
  - Hexadecimal (Intel-format) File (*.hex)

Menu: File -> Combine Programming Files...
Configuring Multiple FLEX Devices

Configure multiple FLEX device with the download cable

- You can configure multiple FLEX devices in a FLEX chain with the download cable
  - By typing a command at a DOS command prompt to download configuration data from an SBF file through the BitBlaster
    
    **DOS Prompt: copy <design>.sbf COM1: (or COM2:)
    - The SBF file can be created by using “Combine Programming File” command (under File Menu)
  - By creating “multi-device FLEX chain” (under FLEX Menu) and using the Programmer to download configuration data from SOFs through the BitBlaster, ByteBlaster, or FLEX download cable
    - Multi-device FLEX chain: a series of FLEX devices through which configuration data is passed from device to device using the sequential Passive Serial configuration scheme
Creating Multi-Device FLEX Chain

◆ To configure multiple FLEX devices in a FLEX chain
  • You can specify the order and names of SOFs for multiple FLEX devices in a chain
    Menu: FLEX -> Multi-Device FLEX Chain Setup...
  • You can save the FLEX chain settings to an Flex Chain File(*.fcf) or restore the settings from a existing FCF file
    Menu: FLEX -> Save FCF...
    Menu: FLEX -> Restore FCF...
  • To turn on or off multi-device FLEX chain configuration mode
    Menu: FLEX -> Multi-Device FLEX Chain
  • Click Configure button on Programmer to start configuration
Programming Multiple JTAG Devices

◆ Program or configure multiple JTAG devices with the download cable

- You can program or configure one or more MAX 9000, MAX 7000S, FLEX 10K devices, and other devices that support JTAG programming in a JTAG chain using the BitBlaster or ByteBlaster
  - The JTAG chain can contain any combination of Altera and non-Altera devices that comply with the IEEE 1149.1 JTAG specification, including some FLEX 8000 devices
  - By creating “multi-device JTAG chain” (under JTAG Menu) and using the Programmer to download configuration data from SOFs through the BitBlaster or ByteBlaster cable

- Multi-device JTAG chain: a series of devices through which programming and/or configuration data are passed from device to device via the JTAG boundary-scan test circuitry
Creating Multi-Device JTAG Chain

◆ To program multiple devices in a JTAG chain
  • You can select the names and sequence of devices in the JTAG chain, and optional associated programming files
    Menu: JTAG -> Multi-Device JTAG Chain Setup...
  • You can save the JTAG chain settings to a JTAG Chain File (*.jcf) or restore the settings from an existing JCF file
    Menu: JTAG -> Save JCF...
    Menu: JTAG -> Restore JCF...
  • To turn on or off multi-device JTAG chain programming mode
    Menu: JTAG -> Multi-Device JTAG Chain
  • Click Configure or Program button on Programmer to start programming
Details about Device Programming

◆ Please refer to Altera document for details

  • Altera Data Book
  • Altera Data Sheet
    – dsconf_06.pdf: Configuration EPROMs for FLEX Devices
    – dsbit03.pdf: BitBlaster Serial Download Cable
    – dsbyte01.pdf: ByteBlaster Parallel Port Download Cable
  • Altera Application Note & Application Brief
    – an033_03.pdf: Configuring FLEX 8000 Devices
    – an038_03.pdf: Configuring FLEX 8000 Devices
    – an059_01.pdf: Configuring FLEX 10K Devices
    – ab141_01.pdf: In-System Programmability in MAX 9000 Devices
    – ab145_01.pdf: Designing for In-System Programmability in MAX 7000S Devices
    – an039_03.pdf: JTAG Boundary Scan Testing in Altera Devices
Getting Help

◆ CIC technical support: Steven / 周育德
  • Phone: (03)5773693 ext. 148
  • Email: steven@.cic.edu.tw
  • WWW: http://www.cic.edu.tw/chip_design/design_intr/altera/

To buy Altera chips, hardware or demo boards:
  • Contact Galaxy Far East Corp. 茂綸股份有限公司
    周怡慧 小姐
    電話: (02) 2705-7266 # 108
    candye893@gfec.com.tw

◆ Altera technical support on Internet
  • WWW: http://www.altera.com