Department of Electrical
and Computer Engineering

The Logic Circuit CAD Process

Introduction

This series of lectures looks at the CAD process using the tool set supplied by Altera. This tool set has its own particular way of implementing the design process, but is representative of tool sets by other vendors.

Design Environment

- The diagram below show the flow of a design from the conception stage through to the design verification and programming.

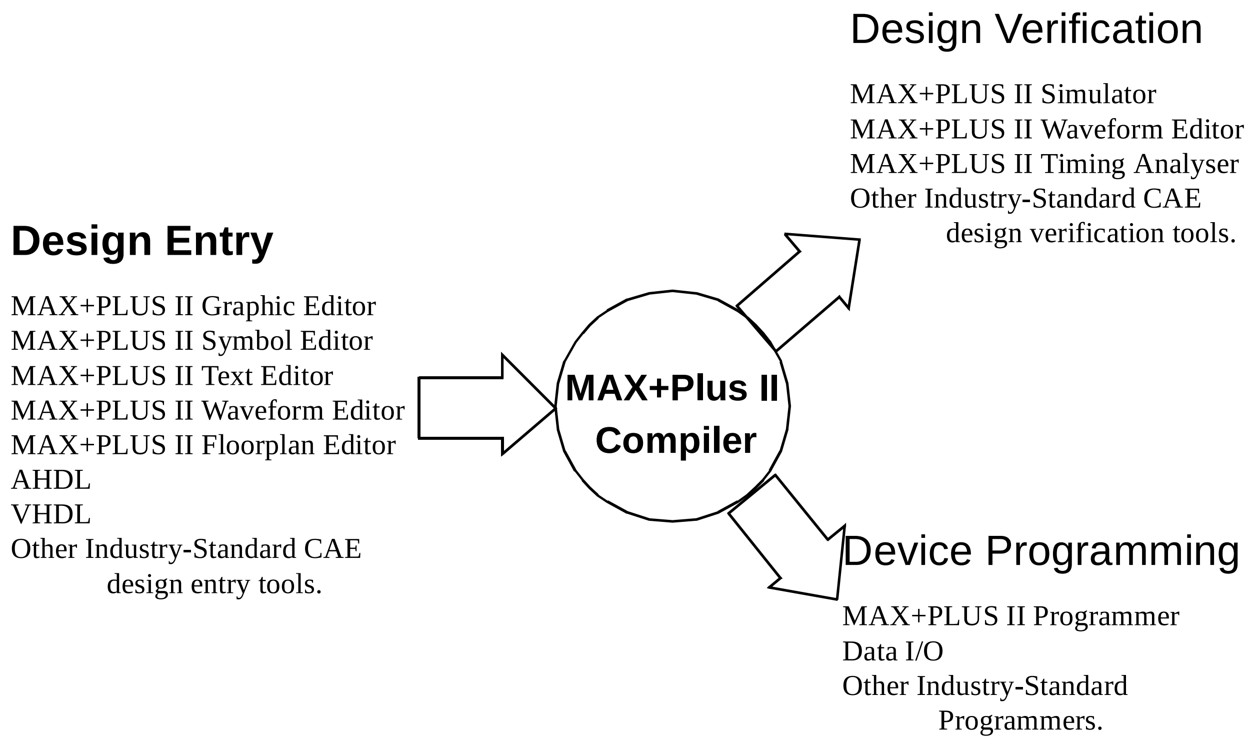

Design Verification
MAX+PLUS II Simulator
MAX+PLUS II Waveform Editor
MAX+PLUS II Timing Analyser
Other Industry-Standard CAE design verification tools.
Design Entry
MAX+PLUS II Graphic Editor
MAX+PLUS II Symbol Editor
MAX+PLUS II Text Editor
MAX+PLUS II Waveform Editor
MAX+PLUS II Floorplan Editor
AHDL
VHDL
Other Industry-Standard CAE design entry tools.
MAX+Plus II Compiler
Device Programming
MAX+PLUS II Programmer
Data I/O
Other Industry-Standard Programmers.

Figure 1 : MAX+PLUS II Design Environment

Design Flow

1. Create a new design file or a hierarchy of multiple design files in any combination of the MAX+PLUS II design editors, i.e. Graphic, Text or Waveform editors.
2. Specify the top-level design file name as the project name.
3. Assign a device family for the project. You can either allow the Compiler to select a device for you or assign a specific device.
4. Open MAX+PLUS II compiler window and choose the **Start** button to compile the project. You can turn on the SNF extractor module to create a netlist file for timing simulation and timing analysis.
5. If project compiles successfully, you can optionally perform a simulation and timing analysis:
   (i) To run a timing analysis, open the MAX+PLUS II Timing Analyser window, select an analysis mode, and choose the **Start** button.
   (ii) To run a simulation, you must first create vector inputs in a Simulator Channel File (.scf) in the Waveform Editor or in a Vector File (.vec) in the Text Editor. Then open the MAX+PLUS II Simulator window and choose the **Start** button.
6. Open the MAX+PLUS II Programmer window and either insert a device into a programmer adapter on the Master Programming Unit (MPU) or connect the BitBlaster to a device that is mounted in-system.
7. Choose the **Program** button to program an EPROM, EEPROM or FLASH-based device, or choose the **Configure** button to configure an SRAM-based device.

Files in the Environment

- Three main sets of files in the design environment:

Design Files

A *design file* is either a graphic, text or waveform file created with the MAX+PLUS II Graphic, Text or Waveform Editor, or with another industry standard schematic or text editor or an EDIF or VHDL netlist writer.

Design files are processed by the compiler. The compiler can compile the following file types:

- Graphic Design Files (**.gdf**).
- Text Design Files (**.tdf**).
- Waveform Design Files (**.wdf**).
- VHDL Design Files (**.vhd**).
- OrCAD Schematic Files (**.sch**).
- EDIF Input Files (**.edf**)
- Xilinx Netlist Format Files (**.xnf**).
- Altera Design Files (**.adf**).
- State Machine Files (**.smf**)

Ancillary Files

These are files that are associated with the MAX+PLUS II project but are not part of the project tree hierarchy. Most files of this type do not contain design logic. some are automatically generated by the design environment, others are user entered, e.g.

Assignment and configuration files (**.acf**), Symbol files (**.sym**), report files (**.rpt**), vector files (**.vec**).

Projects

- A *project* consists of all the files in a design hierarchy, including the ancillary files and the output files.
- Project name is the name of the top-level design file, without the file name extension. The user nominates the project file.
- Each project should be placed in a separate directory of the **\max2plus** directory.

Design Tools

Design Entry

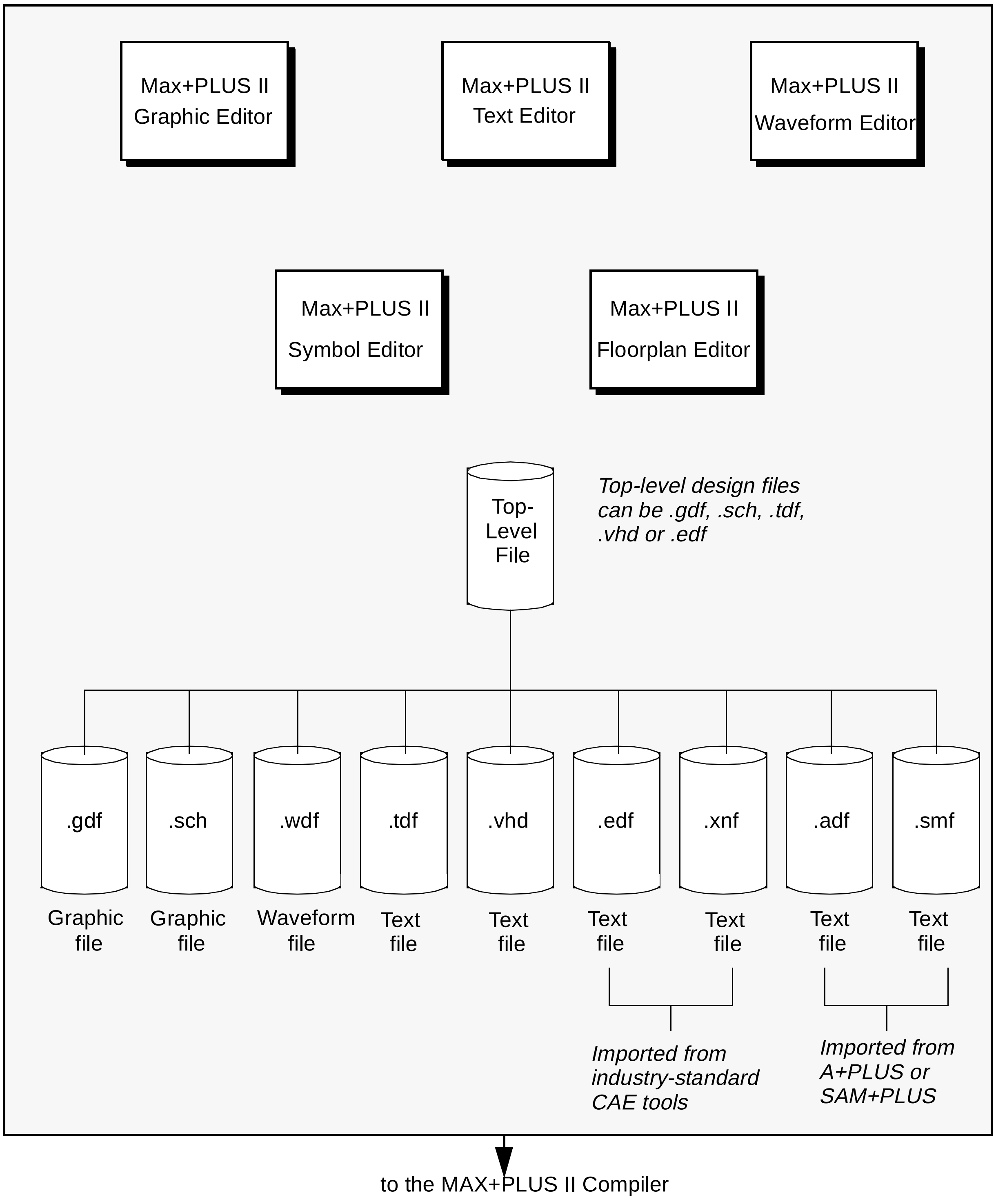

Max+PLUS II Graphic Editor
Max+PLUS II Text Editor
Max+PLUS II Waveform Editor
Max+PLUS II Symbol Editor
Max+PLUS II Floorplan Editor
Top-Level File
Top-level design files can be .gdf, .sch, .tdf, .vhd or .edf
.gdf
.sch
.wdf
.tdf
.vhd
.edf
.xnf
.adf
.smf
Graphic file
Graphic file
Waveform file
Text file
Text file
Text file
Text file
Text file
Text file
Imported from industry-standard CAE tools
Imported from A+PLUS or SAM+PLUS
to the MAX+PLUS II Compiler

Figure 2 : MAX+PLUS II Design Entry Methods

MAX+PLUS II Graphic Editor

- Offers WYSIWYG graphic design environment.
- Allows schematic capture of a design.
- Supports primitive, megafunction and macrofunction libraries, including the Library of Parameterized Modules (LPM).
- Has a symbol generation capability to make building hierarchical designs simple.
- A Graphic Design File (**.gdf**) can include any combination of primitive and macrofunction symbols. Symbols can represent *any type* of design file.

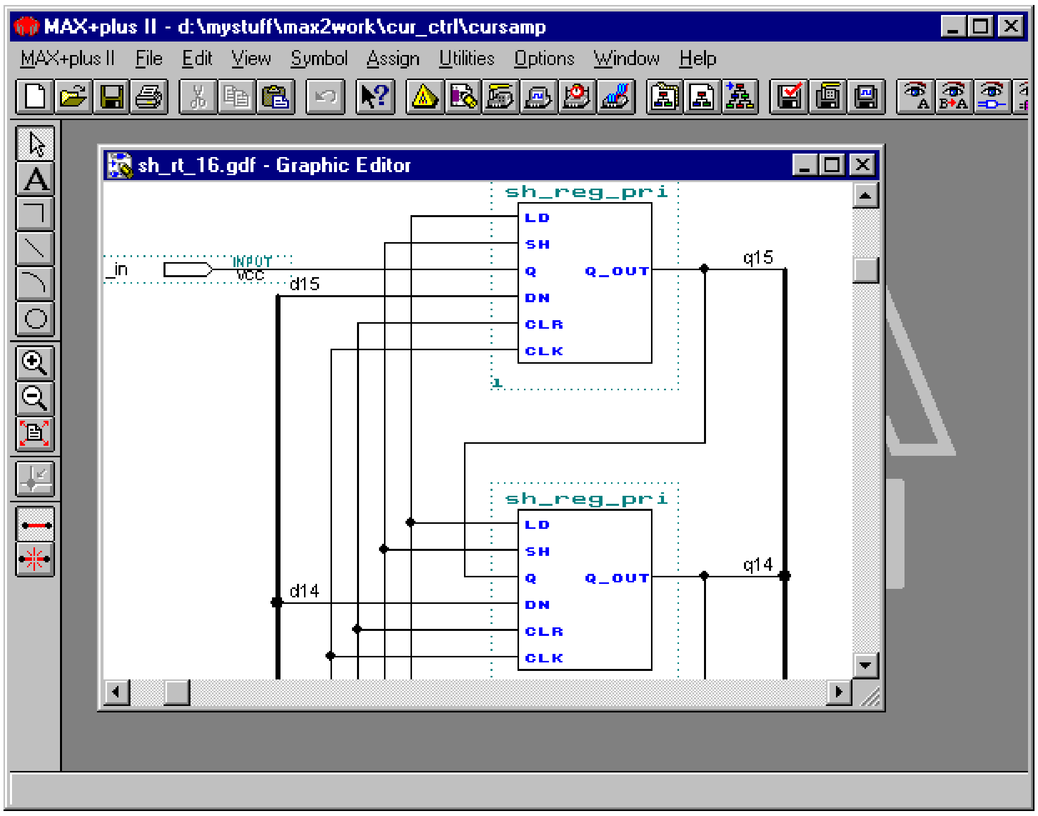

MAX+plus II - d:\mystuff\max2work\cur_ctrl\cursamp
MAX+plus II File Edit View Symbol Assign Utilities Options Window Help
sh_rt_16.gdf - Graphic Editor
sh_reg_pri
LD
SH
Q
DN
CLR
CLK
Q_OUT
_in
INPUT
VCC
d15
q15
d14
q14

Figure 3 : Graphic Editor screen in the MAX+PLUS II Development Environment.

MAX+PLUS II Symbol Editor

- Enables one to view, create, and edit a symbol that represents a logic circuit.
- Symbol files have the same name as the design file it represents, with the extension **.sym**. The **Create Default Symbol** command, available from the File menu of the Graphic, Text and Waveform Editors, creates a symbol from any design file.

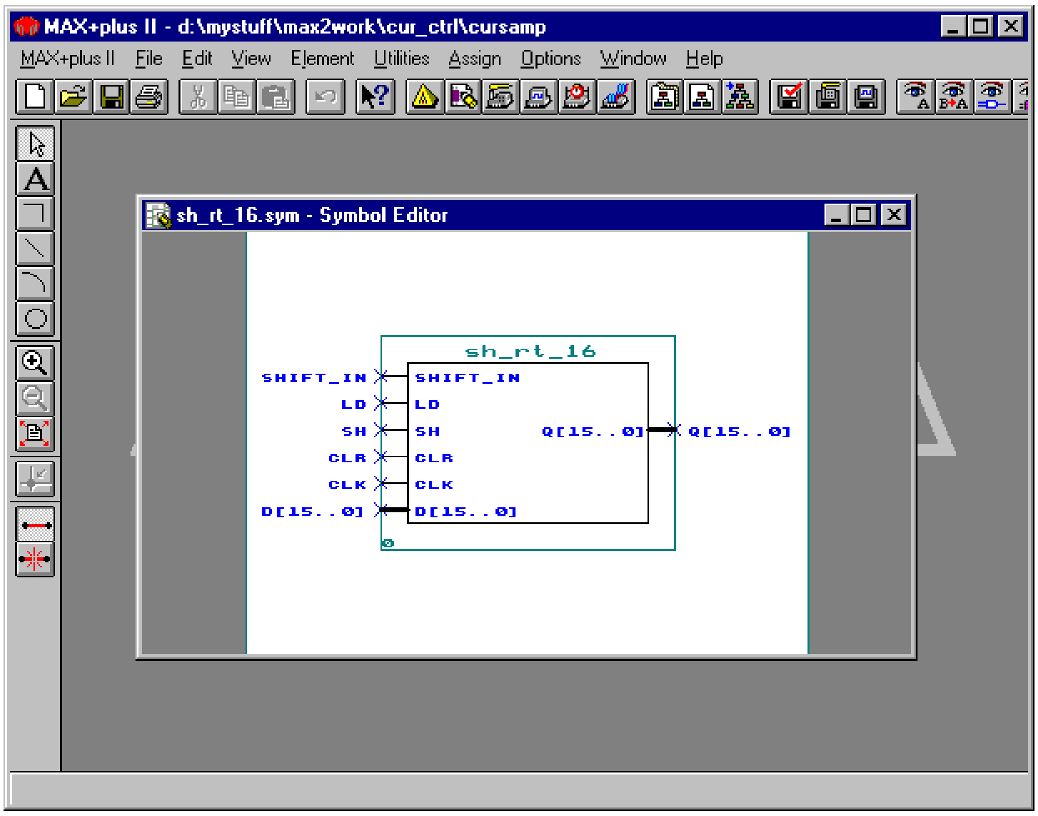
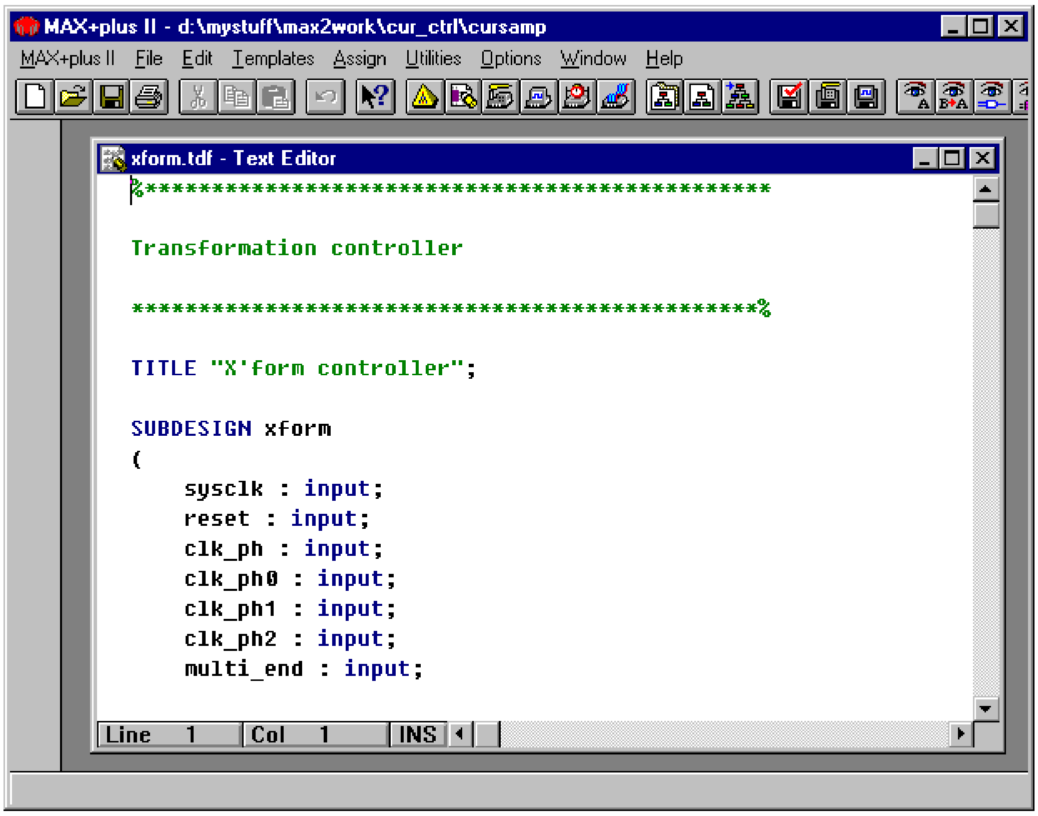


Figure 4 : Symbol Editor screen in the MAX+PLUS II Development Environment.

MAX+PLUS II Text Editor

- A tool for entering Text Design Files in Altera Hardware Description Language (AHDL) (**.tdf**) or Very High Speed Integrated Circuit Hardware Description Language (VHDL) (**.vhd**).

- Can also edit any ascii file as well.
- Offers feature that support the Altera design environment -e.g. syntax highlighting, automatic location of compilation errors, templates for AHDL and VHDL.
- Offers a link to the compiler or simulation.


MAX+plus II - d:\mystuff\max2work\cur_ctrl\cursamp
MAX+plus II File Edit Templates Assign Utilities Options Window Help
xform.tdf - Text Editor
Line 1 Col 1 INS

```
%**************************************************

Transformation controller

**************************************************%

TITLE "X'form controller";

SUBDESIGN xform
(
    sysclk : input;
    reset : input;
    clk_ph : input;
    clk_ph0 : input;
    clk_ph1 : input;
    clk_ph2 : input;
    multi_end : input;
```

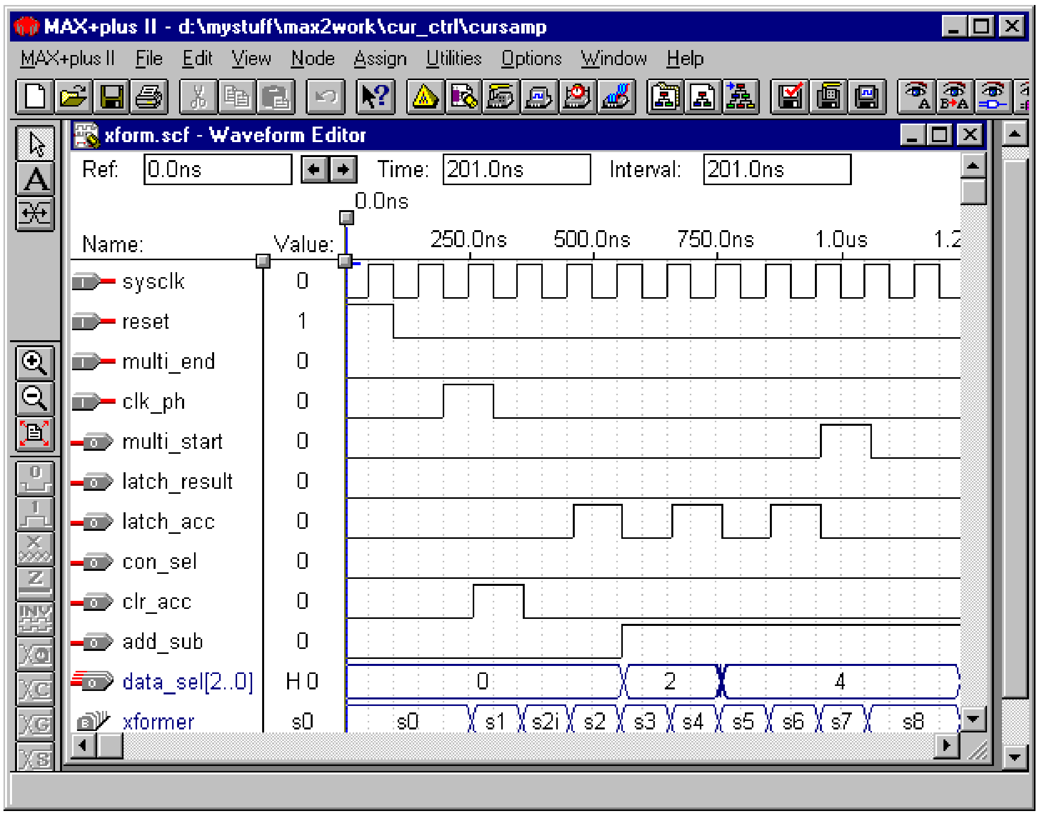
Figure 5 : Text Editor screen in the MAX+PLUS II Development Environment

MAX+PLUS II Waveform Editor

- Serves two roles: a design entry tool, and as a tool for entering test vectors and viewing simulation results.
- Waveform Design Files (**.wdf**) can contain design logic for a specific project. These files offer an alternative to graphic or text entry for a design. One creates a design by specifying the input waveforms and the output waveforms. One can also generate state

machines as well using this method.

- One can also create Simulator Channel Files (**.scf**) that contain input vectors for simulation and functional testing.


MAX+plus II - d:\mystuff\max2work\cur_ctrl\cursamp

xform.scf - Waveform Editor

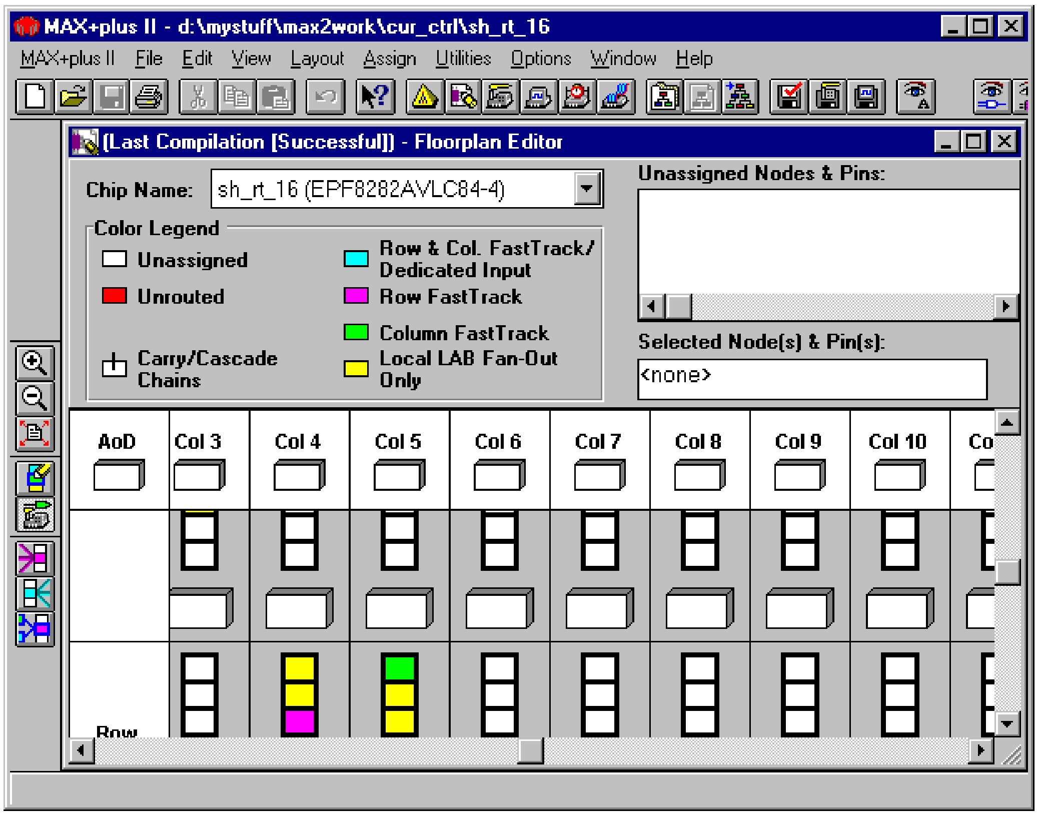
Ref: 0.0ns Time: 201.0ns Interval: 201.0ns

Figure 6 : Waveform Editor screen in the MAX+PLUS II Development Environment

Floorplan Editor

- Allows physical resources to be assigned and to view Compiler partitioning and fitting results.
- Has two displays:
  - (a) The device view shows all pins on the device package and their function.
  - (b) The LAB view shows the interior of the device, including all Logic Array Blocks (LABs) and the individual logic cells within each LAB.

- Provides a list of unassigned node and pin names in your project, the handles to which can be dragged to an individual pin, logic cell, I/O cell, or embedded cell in the Device View or LAB view display. One can also drag and assigned pin or node to the list of unassigned nodes.


MAX+plus II - d:\mystuff\max2work\cur_ctrl\sh_rt_16

[Last Compilation [Successful]] - Floorplan Editor

Chip Name: sh_rt_16 (EPF8282AVLC84-4)

Unassigned Nodes & Pins:

Selected Node(s) & Pin(s): <none>

Color Legend
- ⬜ Unassigned
- 🟥 Unrouted
- Carry/Cascade Chains
- 🟦 Row & Col. FastTrack/ Dedicated Input
- 🟪 Row FastTrack
- 🟩 Column FastTrack
- 🟨 Local LAB Fan-Out Only

Figure 7 : Floorplan Editor screen in the MAX+PLUS II Development Environment

Altera Hardware Description Language (AHDL)

- High level modular language that is completely integrated into the MAX+PLUS system.
- AHDL files can be created with a normal text editor or with the editor included in the MAX+PLUS II system. This has some advantages because of the tight connect of AHDL into the Altera development system.

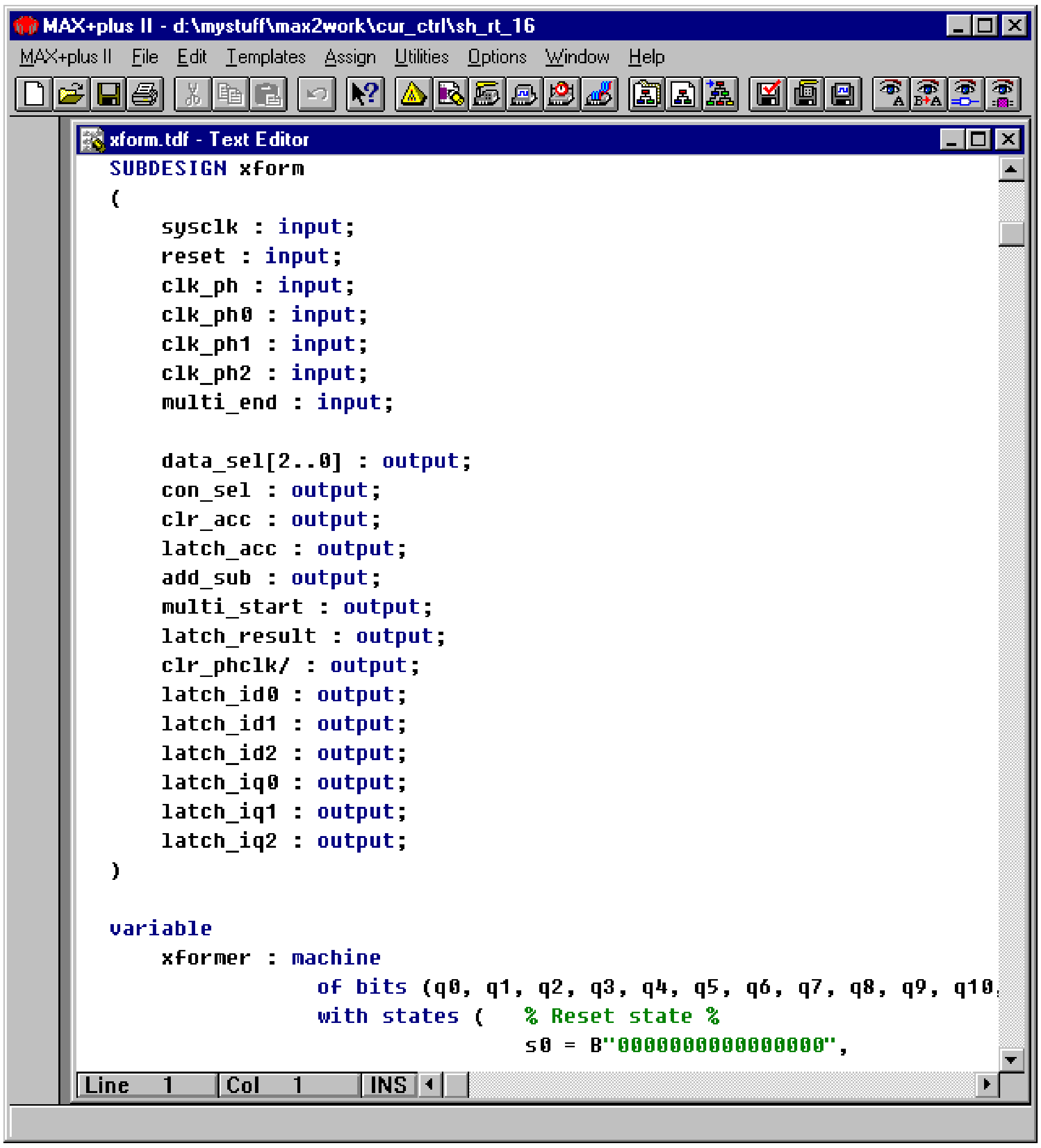
- AHDL consists of a variety of elements and behavioural statements that describe logic.
- AHDL is an ideal tool for describing functions such as state machines, truth tables, boolean equations, conditional logic, and group operations.
- One can use Library Parameterized Modules (LPM) functions to implement logic. LPMs provide gate, arithmetic, and storage components that implement combinational logic such as decoders, multiplexers, and adders, and sequential logic such as registers and counters.
- AHDL is ideal for state machine designs: one can assign state bits and state values, or let the compiler do the work for you. See Figure 8 overleaf

VHDL

- Very High Speed Integrated Circuit (VHSIC) Hardware Description Language is a high level, modular language that is completely integrated into the MAX+PLUS II system.
- VHDL is an industry standard hardware description language that describes inputs and outputs, behaviour, and function of circuits. Defined by IEEE Standard 1076-1987.
- VHDL files (**.vhd**) created by a standard text editor or by the integrated text editor.

Primitives, Megafunctions & Macrofunctions

- Altera provides libraries of logic functions – primitives, megafunctions, and old style (e.g. 74 series TTL) macrofunctions, including functions that are optimized for the architecture of a particular device family.
- All logic functions are copied to subdirectories of **\maxplus2\maxlib** and **\maxplus2\max2vhdl** directories.


MAX+plus II - d:\mystuff\max2work\cur_ctrl\sh_rt_16
MAX+plus II File Edit Templates Assign Utilities Options Window Help
xform.tdf - Text Editor
Line 1 Col 1 INS

```
SUBDESIGN xform
(
    sysclk : input;
    reset : input;
    clk_ph : input;
    clk_ph0 : input;
    clk_ph1 : input;
    clk_ph2 : input;
    multi_end : input;

    data_sel[2..0] : output;
    con_sel : output;
    clr_acc : output;
    latch_acc : output;
    add_sub : output;
    multi_start : output;
    latch_result : output;
    clr_phclk/ : output;
    latch_id0 : output;
    latch_id1 : output;
    latch_id2 : output;
    latch_iq0 : output;
    latch_iq1 : output;
    latch_iq2 : output;
)

variable
    xformer : machine
                of bits (q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10,
                with states (   % Reset state %
                                s0 = B"0000000000000000",
```

Figure 8 : Part of an AHDL program

Primitives

- Examples: buffer, flip-flop, latch, input/output, and logic primitives.
- These are the basic functional blocks used to design circuits.
- If a primitive is connected to a named bus then the compiler will

automatically expand the number of primitives to the number of bus lines saving space and design entry time.

Megafunctions

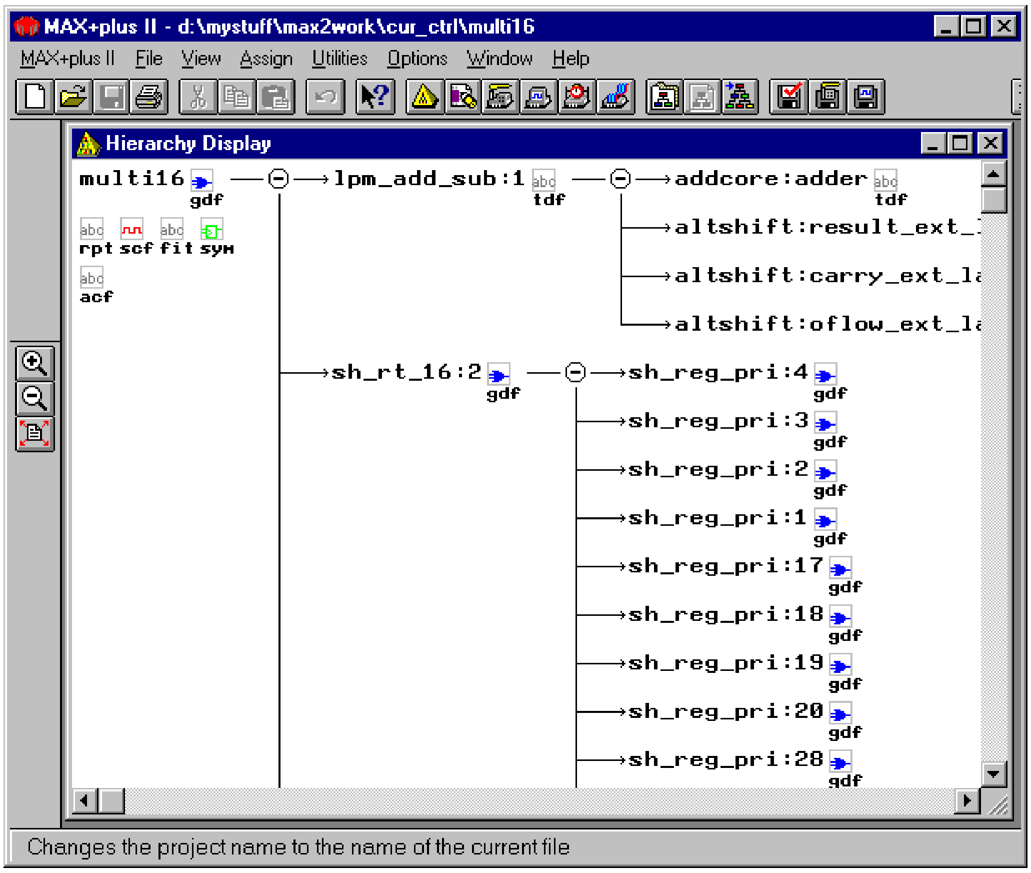
- Complex or high level building blocks that can be used together with primitives and other mega and macrofunctions to create a logic design.
- LPMs are an example of a megafunction that is parameterized – i.e. the user can set the size, behaviour and silicon implementation.
- Megafunctions can be used freely in GDFs and TDFs; non-parameterized megafunctions can be used in VHDL design files.
- The compiler automatically removes any unused gates and flip-flops to ensure optimum design efficiency.

Old Style Macrofunctions

- Similar to megafunctions above
- Not inherently parameterized.
- Altera recommends using LPMs instead – scalable and use the silicon more efficiently.

Project Hierarchy

- Allows the display of the design hierarchy where the lower level files are represented as branches.
- This display makes it easy to move between the different files that make up a design. Also shows the relationship between all the different sections of a design.



Changes the project name to the name of the current file

Figure 9 : Hierarchical display of a logic design

Project Processing

- Compiles a project to produce simulation, timing and programming files. See Figure 11 for a view of the compiler window.

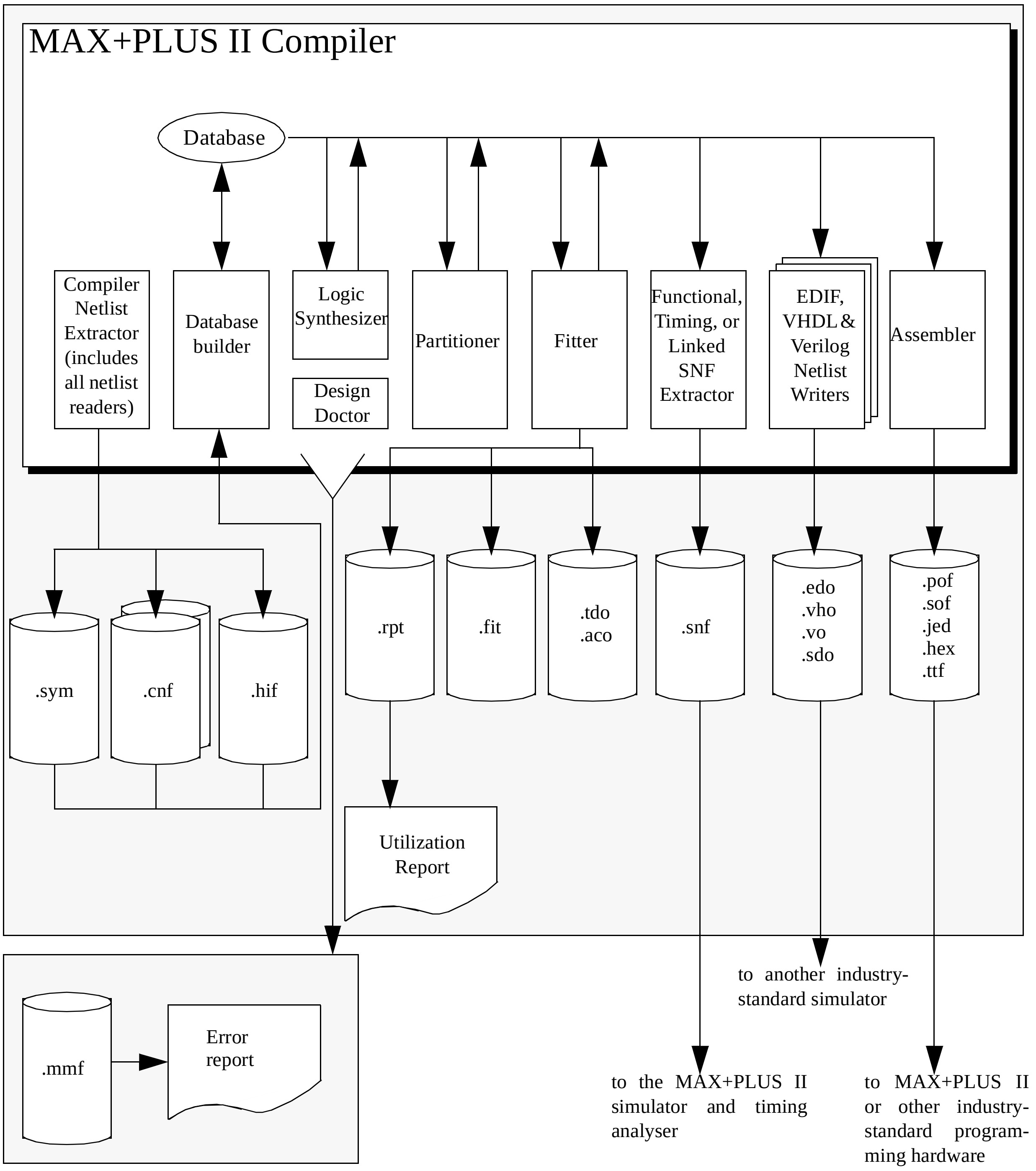
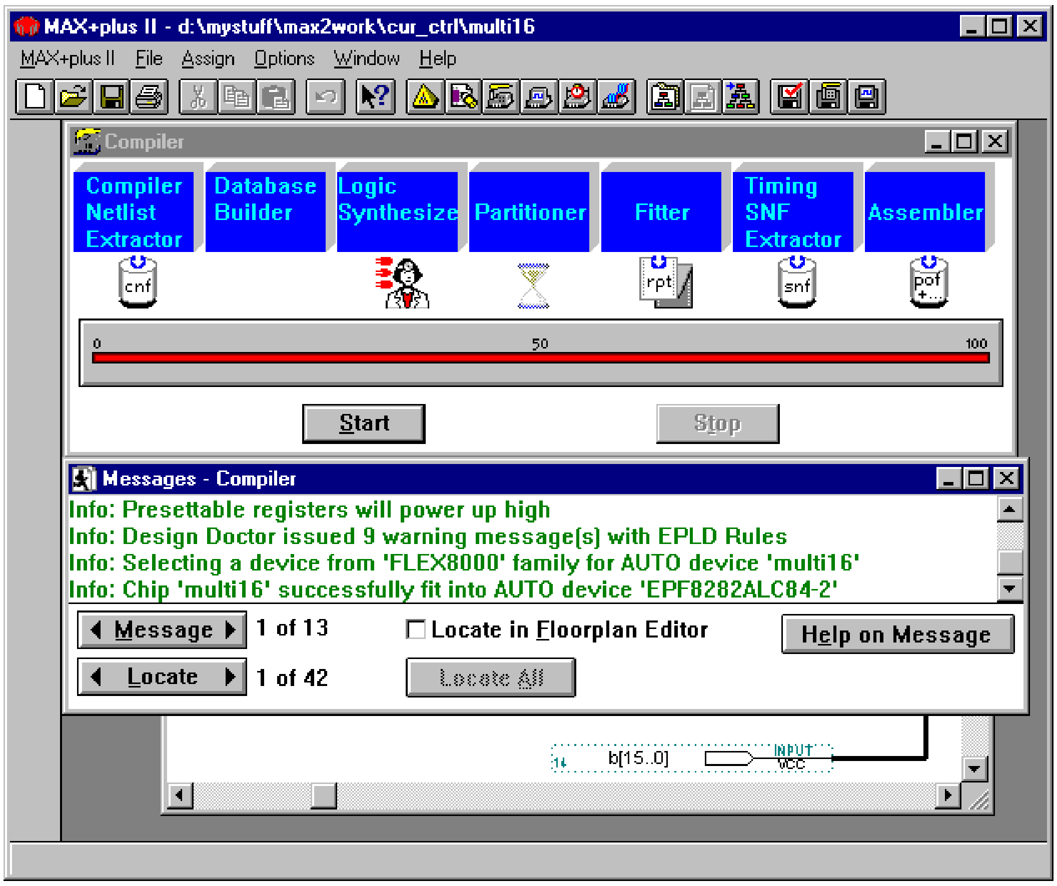

MAX+PLUS II Compiler
Database
Compiler Netlist Extractor (includes all netlist readers)
Database builder
Logic Synthesizer
Design Doctor
Partitioner
Fitter
Functional, Timing, or Linked SNF Extractor
EDIF, VHDL & Verilog Netlist Writers
Assembler
.sym
.cnf
.hif
.rpt
.fit
.tdo .aco
.snf
.edo .vho .vo .sdo
.pof .sof .jed .hex .ttf
Utilization Report
.mmf
Error report
to another industry-standard simulator
to the MAX+PLUS II simulator and timing analyser
to MAX+PLUS II or other industry-standard programming hardware

Figure 10 : Project processing in the Altera Development System



```
Info: Presettable registers will power up high
Info: Design Doctor issued 9 warning message(s) with EPLD Rules
Info: Selecting a device from 'FLEX8000' family for AUTO device 'multi16'
Info: Chip 'multi16' successfully fit into AUTO device 'EPF8282ALC84-2'
```

Figure 11 : Compiler screen after compilation.

- Compiler can accept a number of different file type inputs: .gdf, .tdf, .wdf, .vhd, .edf, .sch, .xnf, .adf, .smf etc.

Compilation Process

Compiler Netlist Extractor

- Converts each design file in the project into one or more binary Compiler Netlist Files (**.cnf**).
- Creates a Hierarchy Interconnect File (**.hif**) that documents the hierarchical connections between the project files (necessary for the Hierarchy Display).
- Generates the Node Database File (**.ndb**) that contains project node names for the resource database.
- Has built in filters that convert EDIF, VHDL and XNF files into Altera's MAX+PLUS II format.

Database Builder

- Uses the HIF file to link the CNFs that describe the project.
- Based on the HIF the Database Builder copies each CNF into a single, fully flattened project database. This database preserves the electrical connectivity of the project.
- As database is built the Database Builder checks for logical completeness and consistency of the project.
- Each success module in the compiler processes and updates the database formed during this stage.

Logic Synthesizer

- Applies a number of algorithms that reduce resource usage and remove redundant logic to ensure that the logic cell structure is used as efficiently as possible for the architecture of the target device family.
- Also applies logic synthesis techniques to help implement user-specified timing and other implementation requirements.

Partitioner

- If a design does not fit into a single device the partitioner divides the database updated by the logic synthesizer into multiple devices of the same type, attempting to use the smallest number of devices.
- Can be fully automatic, partially user controlled, or fully user controlled.

Fitter

- Using the database updated by the partitioner, the fitter matches the requirements of the project with the known resources of one or more devices.
- Assigns each logic function to the best logic cell location and selects appropriate interconnection paths and pin assignments in an attempt to match your resource assignments – i.e., the pin, logic cell, I/O cell, embedded cell, chip, clique, device, timing, and connected pin assignments in the project’s assignment and configuration file (**.acf**) – with the available resources.
- If a fit cannot be found a warning is issued and gives one the option to terminate the compilation, or to ignore some or all of ones assignments.
- A report file (**.rpt**) is generated regardless of whether a fit is found or not. Documents fitting information on project partitioning, input and output pin names, project timing, unused resources for each device in the project.
- A Fit File (**.fit**) documents resource and device assignments for the entire project, as well as routing information. This can be viewed with the floorplan editor regardless of whether a fit was achieved or not.

Functional SNF Extractor

- Creates a functional Simulator Netlist File (**.snf**) required for functional simulation.
- Creates this file before it synthesizes the project, therefore it contains all nodes present in the original design files.

Timing SNF Extractor

- Creates a Timing Netlist File (**.snf**) which contains the timing data for the fully optimized project.
- Used for timing simulation and timing analysis.
- Optionally one can instruct the compiler to generate an optimized SNF containing dynamic models that represent types of combinational logic – can save you time during simulation and timing analysis.

Linked SNF Extractor

- Creates a SNF for a multi-project board level type simulation.
- Can have different devices in the projects.

EDIF, Verilog and VHDL Netlist Writers

- Produces output files that contain functional and operational timing information that allows various industry standard simulators to be used.

Assembler

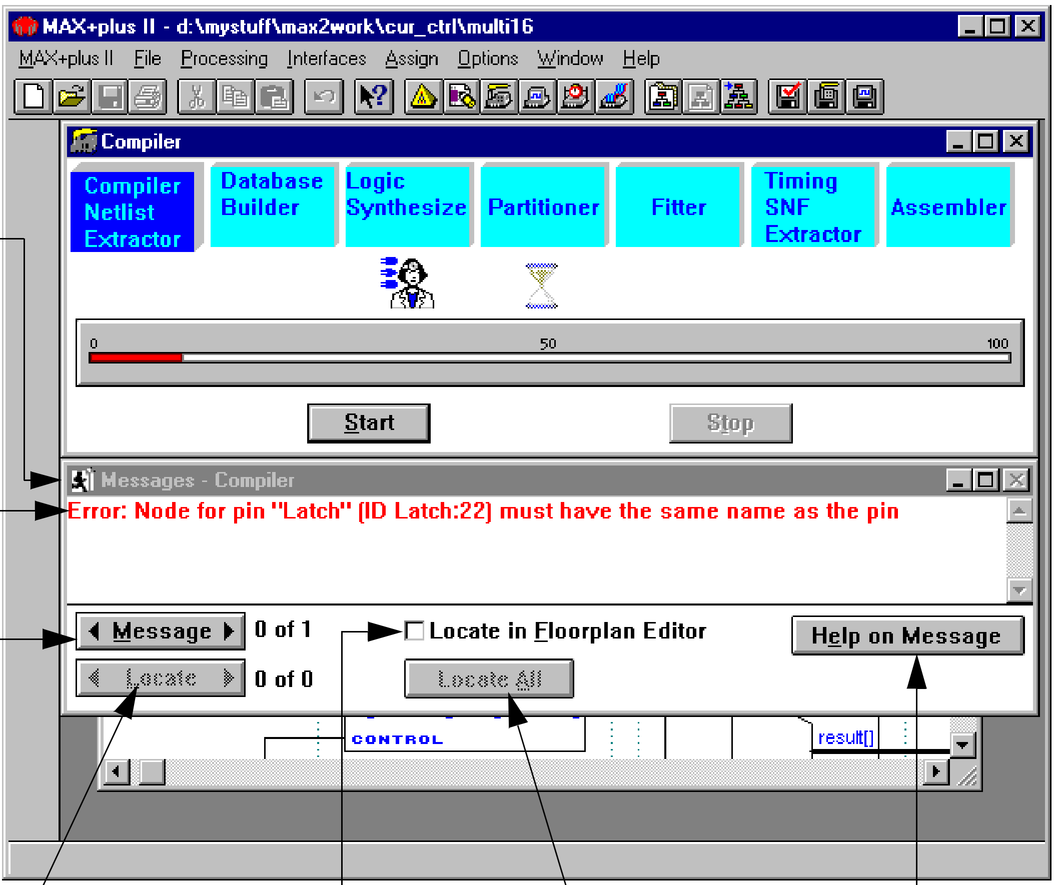
- Converts the Fitters logic cell, pin, and device assignments into a programming image for the device(s) in the format of one or more binary Programming Object Files (**.pof**) or SRAM Object Files (**.sof**) for some devices.
- Can also generate JEDEC (**.jed**), tabular text files (**.ttf**) and hexa-

decimal (Intel-format) files (**.hex**). These are then processed by the MAX+PLUS II programmer and Altera programming hardware, or another industry standard programmer to produce working devices.

Design Doctor Utility

- Checks each design file for logic that may cause system level reliability problems.
- One can choose one of three predefined sets of design rules with different levels of design rule checking, or create a custom set of design rules.

Error Detection and Location



Error: Node for pin "Latch" (ID Latch:22) must have the same name as the pin

Message processor window opens during compilation if a message is generated

Double clicking on a message is a short cut for choosing the locate button

Allows one to scroll through all messages. The total number of messages generated and the number of the currently selected message are displayed.

Automatically highlights the source of an error or warning; if a message has multiple sources, you can locate each source in succession.

Allows you to trace the source(s) of a message in the Floorplan Editor instead of in a design or ancillary file.

Opens the Floorplan Editor window and highlights all source(s) of the selected message simultaneously

Displays information about the cause of the message and how to correct it.

- If a message is caused by a problem in multiple locations, the message processor allows you to find each location.

Department of Electrical
and Computer Engineering

Project Verification

- Three applications are provided in the MAX+PLUS II system to help testing and location of errors: Simulator, Timing Analyser, and the Waveform Editor.

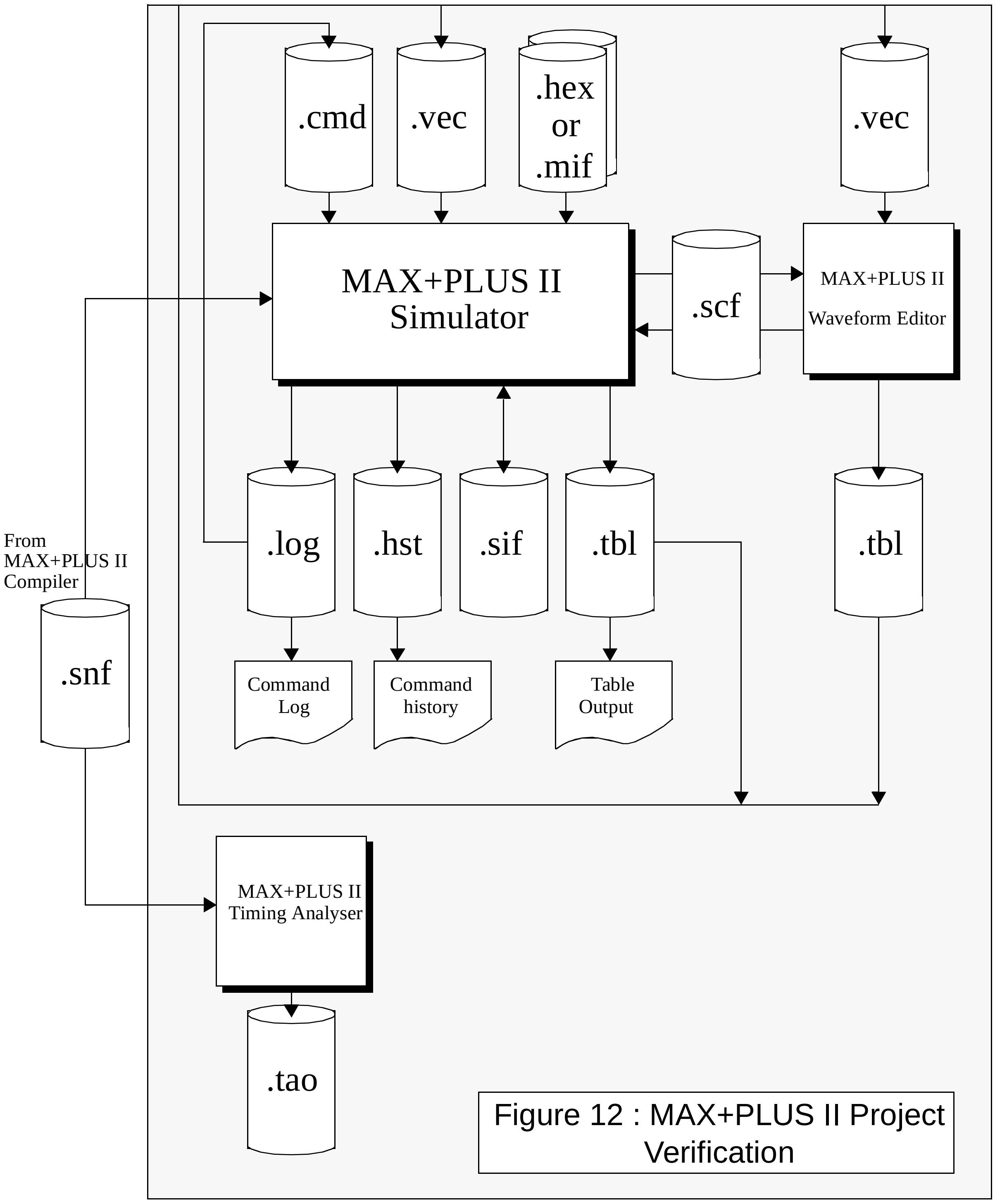


Figure 12 : MAX+PLUS II Project Verification

Simulator

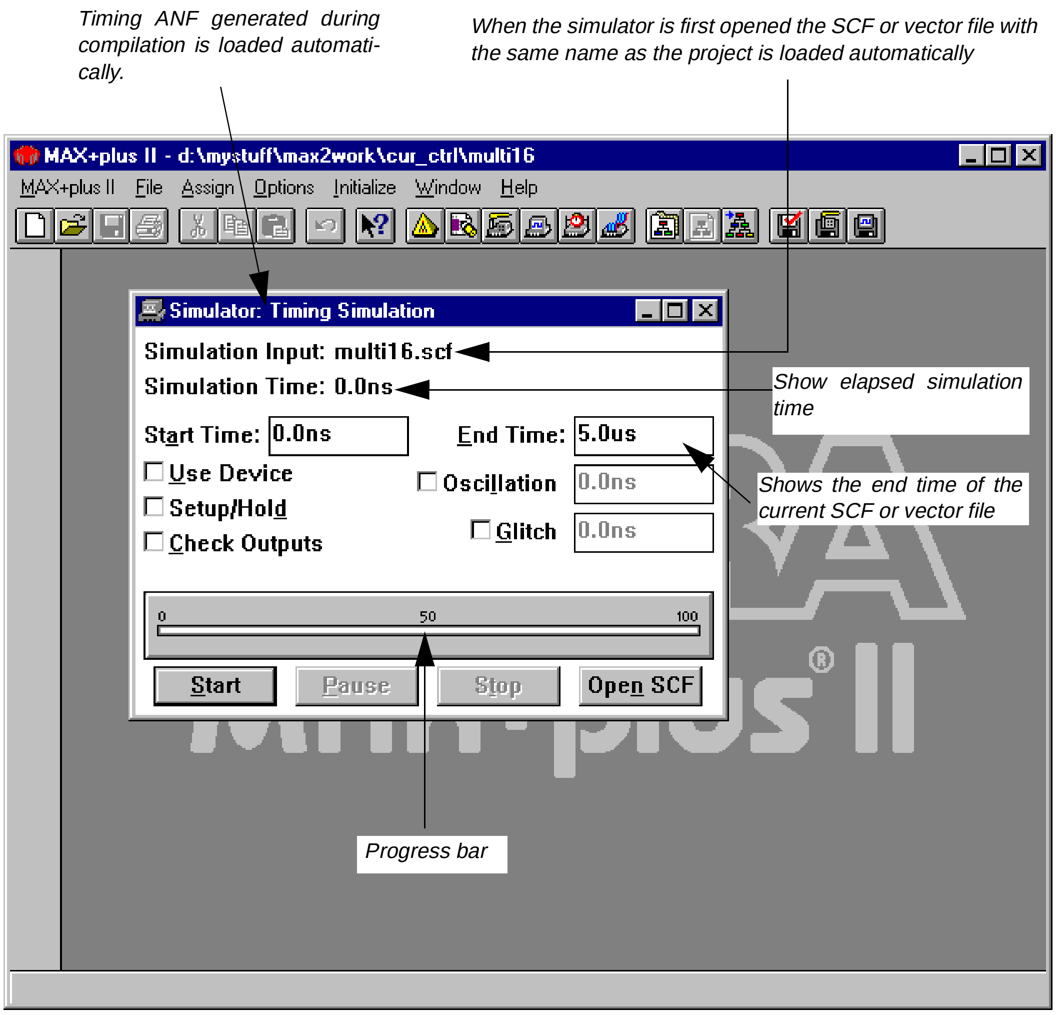
- Tests the logical operation and internal timing of a project, allowing one to model a circuit design before it is programmed into a device.
- To run a simulation one must firstly compile a project to generate a Simulator Netlist File (.snf) for functional, timing, or linked multi-project simulation. This file is loaded automatically when one opens the simulator with a particular project.
- Simulator uses a graphical waveform simulator channel file (**.scf**) or an ascii vector file (**.vec**) as the source of input vectors.

Functional Simulation

- The functional SNF is generated before it synthesizes the project. Therefore the file contains all the nodes in the project.
- During functional simulation the simulator ignores all propagation delays.

Timing Simulation

- When a timing SNF is generated it is generated after the project has been fully synthesized and optimized. Does not contain the nodes eliminated by the logic synthesis process.
- The timing files contain information from the Device Model Files (**.dmf**) which are for a target device nominated.
- Timing simulation can be accelerated by instructing the compiler to generate optimized SNF containing dynamic models that represent various types of combinational logic.
- The simulator can handle multi-project simulation by combining the SNFs for the multiple projects.
- Simulator can look for glitches and oscillations, register setup and hold time violations, setup break points on certain conditions. See Figure 13.


Timing ANF generated during compilation is loaded automatically.
When the simulator is first opened the SCF or vector file with the same name as the project is loaded automatically
Show elapsed simulation time
Shows the end time of the current SCF or vector file
Progress bar
MAX+plus II - d:\mystuff\max2work\cur_ctrl\multi16
MAX+plus II File Assign Options Initialize Window Help
Simulator: Timing Simulation
Start
Pause
Stop
Open SCF

Simulation Input: multi16.scf

Simulation Time: 0.0ns

Start Time: 0.0ns

End Time: 5.0us

☐ Use Device

☐ Setup/Hold

☐ Check Outputs

☐ Oscillation 0.0ns

☐ Glitch 0.0ns

0 50 100

Figure 13 : Simulator screen for the MAX+PLUS II Development System

Waveform Editor

- Used for entering input vectors and for viewing simulation results.
- When used to generate input vectors the default set of nodes and groups using the Simulator Netlist File (**.snf**).
- One can import a vector file (**.vec**) and generate a graphical display of its content.
- See Figure 6 for an example of the waveform editor display.

Timing Analyser

- Used to analyse the timing performance of a project after it has been optimized by the Compiler.
- One can trace all signal paths in the project, determining critical speed paths and paths that limit the project's performance.
- Generates three types of analyses:
  (a) The *Delay Matrix* shows the shortest and longest propagation delay paths between multiple source and destination nodes in a project

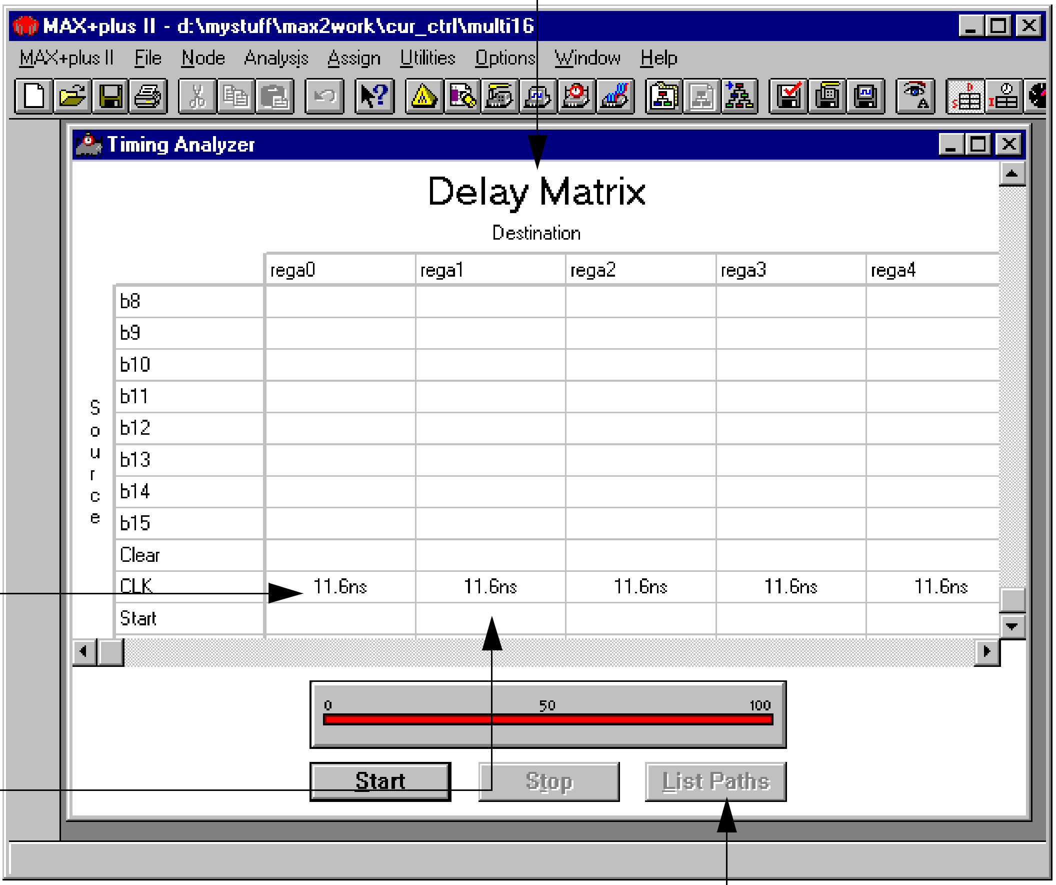
  (b) The *Setup/Hold Matrix* shows the minimum required setup and hold times from input pins to the D, Clock, Latch Enable, address, and Write Enable inputs to flipflops, latches, and asynchronous RAM.

  (c) The Register Performance Display shows the results of a registered performance analysis, including a user-defined number performance-limiting delays, minimum Clock period, and maximum circuit frequency.
- See Figure 14 for a sample screen of the timing analyser.

Delay matrix is one of the three types of analyses performed by the timing analyser.

Only one delay time shows that all the delays are of the same length.

Blank cells indicate that no connection exists between the two nodes.


MAX+plus II - d:\mystuff\max2work\cur_ctrl\multi16
MAX+plus II File Node Analysis Assign Utilities Options Window Help
Timing Analyzer
Delay Matrix
Destination
Source
0
50
100
Start
Stop
List Paths

	rega0	rega1	rega2	rega3	rega4
b8					
b9					
b10					
b11					
b12					
b13					
b14					
b15					
Clear					
CLK	11.6ns	11.6ns	11.6ns	11.6ns	11.6ns
Start					

Opens message processor window and lists all delay times and paths between a pair of nodes.

Figure 14 : Timing Analyser screen for the MAX+PLUS II development system.

Device Programming

- All hardware and software required for programming an verifying Altera devices is provided by Altera.
- Several modes of programming available:

  (a) Altera Master Programming Unit – can also allow functional testing of the programmed part using the vector files used for the software simulation.

  (b) FLEX download cable – can connect any configuration EPROM programming adapter, which is installed on the MPU, to a single target FLEX 8000 or FLEX 10K device in a prototype system.

  (c) BitBlaster serial download is a hardware interface to a standard RS-232 port that provides programming or configuration data to devices mounted on system boards.

- The MAX+PLUS II programmer accepts the following file formats:

  (a) A Programmer Object File (**.pof**) to program Altera Classic, MAX 5000, MAX 7000, and MAX 9000 devices, as well as the Configuration EPROMs used to configure the FLEX 8000 and 10K devices.

  (b) An SRAM Object File (**.sof**) to configure Altera FLEX 8000 or FLEX 10K devices.

  (c) A JEDEC File (**.jed**) to program Altera Classic devices, EPM5016 and EPM5032 devices from the MAX 5000 family, and FLASHlogic devices. (Configuration information for FLASHlogic devices is also stored in JEDEC Files).

  (d) A JTAG Chain File (**.jcf**) – describes the order in which POFs, SOFs, and JEDEC Files for multiple devices are to be programmed or configured in a chain of multiple devices that are connected by JTAG circuitry.

- See Figure 15 for details of the programming environment.

Department of Electrical
and Computer Engineering

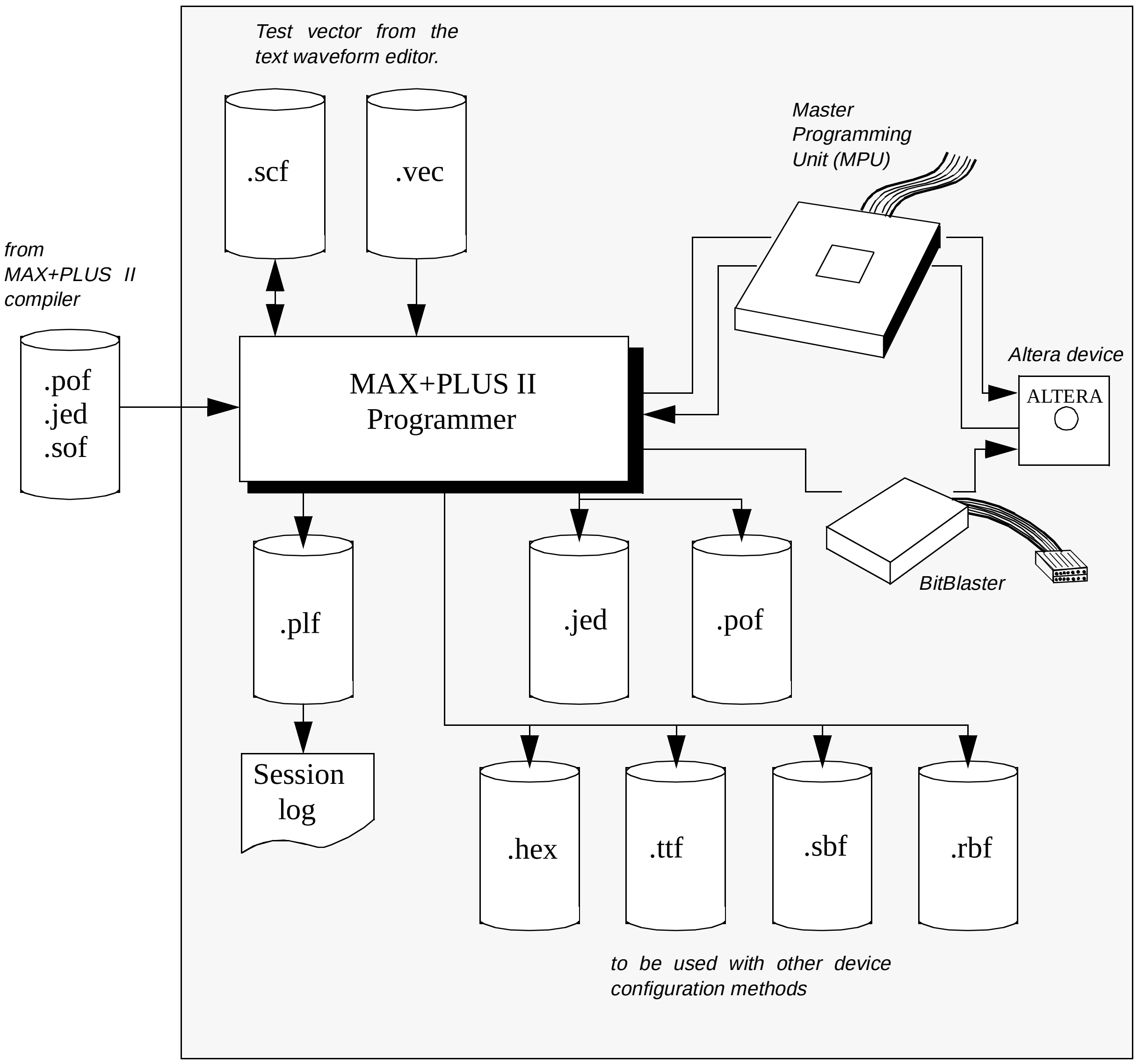

Test vector from the text waveform editor.
.scf
.vec
Master Programming Unit (MPU)
from MAX+PLUS II compiler
.pof
.jed
.sof
MAX+PLUS II Programmer
Altera device
ALTERA
BitBlaster
.plf
.jed
.pof
Session log
.hex
.ttf
.sbf
.rbf
to be used with other device configuration methods

Figure 15 : MAX+PLUS II Device Programming environ-

- The Classic, 5000, 7000 and 9000 devices are programmed by the Master Programming Unit. To erase the devices an EPROM eraser is used.
- The Flex 8000 and 10K devices use external memory to load their configuration at power-up. The configuration information is loaded into internal RAM cells in the device that specify the routing information.

Flex 8000 Configuration

- Flex 8000 (and 10K) use SRAM cells for the storage of configuration information.
- SRAM is loaded each time the circuit powers up.
- After the configuration is loaded the device resets its registers, enables its I/O pins and begins operation.
- The configuration mode and reset mode is known as *command mode*.
- Devices can be reconfigured during operation by forcing the chip back into *command mode* from *user mode* using a dedicated pin.

Configuration Modes

1. Active mode: at system power-up the device supplies the requires signals to carry out the power-up operation.
2. Passive mode: the device acts as a slave to some other device at power-up.

Configuration Schemes

Table 1: Configuration Schemes

Configuration	Acronym	Data Source
Active serial	AS	Altera Configuration EPROM
Active parallel up	APU	Parallel EPROM
Active parallel down	APD	Parallel EPROM
Passive serial	PS	Serial data path
Passive parallel synchronous	PPS	Intelligent host
Passive parallel asynchronous	PPA	Intelligent host

- Each Flex 8000 device has a different size requirement for its configuration data, based on the number of SRAM cells in the device (from 5K bytes for the smallest to 31K bytes for the largest).
- In the serial modes the configuration is moved into the device serially. Altera make special serial EPROMs for this purpose.
- In the parallel modes the data is transferred to the device in parallel and then serialized internally via a shift register.