599 Menlo Drive, Suite 100
Rocklin, California 95765, USA
Office: (916) 624-8333
Fax: (916) 624-8003

General: info@parallaxinc.com
Technical: support@parallaxinc.com
Web Site: www.parallaxinc.com
Educational: www.stampsinclass.com

Stratix 672 SmartPack - EP1S25 (#60001)
Stratix 672 SmartPack - EP1S10 (#60002)
Cyclone 324 SmartPack - EP1C20 (#60003)
Cyclone 100 FastPack - EP1C3 (#60004)

Introduction

The SmartPack (interchangeable with 'FastPack' in this document) makes it easy for you to utilize Altera's Stratix or Cyclone devices. The SmartPack provides a Stratix or Cyclone chip, regulated power supplies, a simple loading mechanism, clock, reset, and convenient access to I/O pins.

Packing List

Verify that your kit is complete in accordance with the list below:

- SmartPack or FastPack board
- DB9 serial cable (for loading by Parallax's method)
- Altera ByteBlaster cable (for loading by JTAG method, not for FastPack)
- Parallax Stratix SmartPack software diskette
- Quartus II Web Edition software CD (Altera Digital Library)
- 5 VDC, 2.4A Power Supply (5VDC, 1A for FastPack)

Features

- The Stratix or Cyclone device is mounted on the SmartPack with 128 I/Os conveniently available from both top and bottom via .100" headers (the FastPack provides 44 I/O connections, plus differential speaker and 3 pushbuttons).
- The SmartPack regulates both 3.3V for I/O and 1.5V for the FPGA core from the power jack.
- The four-pin Clock socket accepts a 3.3V oscillator whose output is routed to a clock input on the Stratix or Cyclone device.
- Power filtering for all PLLs is provided, as well as dedicated clock inputs piggybacked on I/O pins.
- A reset button provides easy restart of your application.
- The integrated PX Loader provides configuration/boot functions over your PC's serial port using the PX.EXE software.
  - 115.2k/230.4k baud operation with run-length data compression for speedy downloads
  - .RBF files can be sent, as well as memory-content files which use the PXL_.TDF module (more info below).
  - An 8Mbit flash device stores complete device and memory contents for power-up boot.
- The DB9 serial port is under Stratix/Cyclone control when the PX Loader is not busy.
- A JTAG connector is provided as an alternate means to configure the Stratix or Cyclone device (not provided on FastPack).

Setting up the Quartus II Web Edition Software

Altera's Quartus II software is used to compose and compile Stratix and Cyclone designs. This software is currently available in two editions:

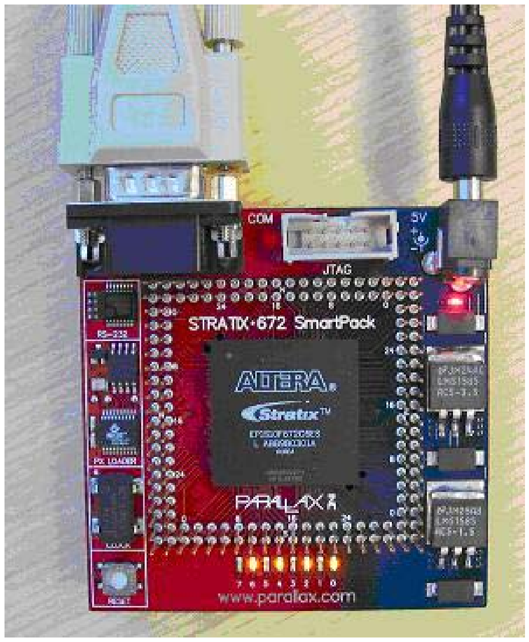
- Quartus II Full Edition This software is available for purchase, only. Details are at http://www.altera.com/products/software/pld/products/q2/qts-index.html. The full Quartus II software is required to use the EP1S25, though at the time of this documentation (April 2003) Altera indicates that the Web Edition (see below) will be revised to program the EP1S25s in the near future.
- Quartus II Web Edition This software is available for download but is also included on the CD-ROM in the kit. On-line documentation is at: http://www.altera.com/products/software/pld/products/quartus2/sof-quarwebmain.html.

If you purchased the Quartus II software you will find complete setup instructions and license agreements in the box, so this setup only pertains to the Quartus II Web Edition software. Both versions of software require that a license be obtained through Altera's web site. The full Quartus II software also requires that a dongle be present on your PC's parallel port.

SmartPack Quick Check

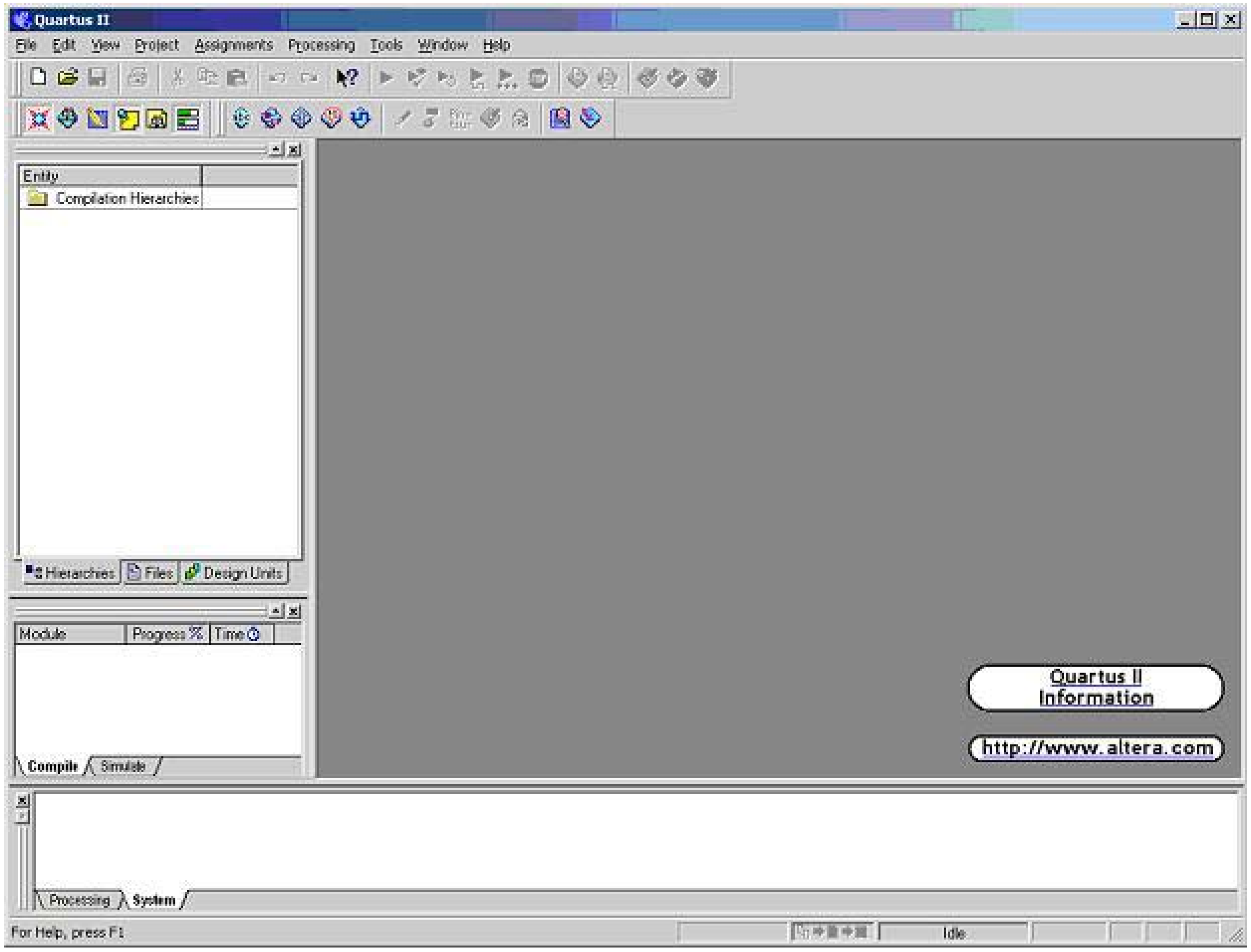
This is a brief overview of how to setup the SmartPack using Parallax's downloader.

1. Provide 5VDC @ 5A into the power jack, center positive. ***
2. Make sure that the Clock socket contains an oscillator.
3. Connect a DB9 cable between the SmartPack and your PC serial port.



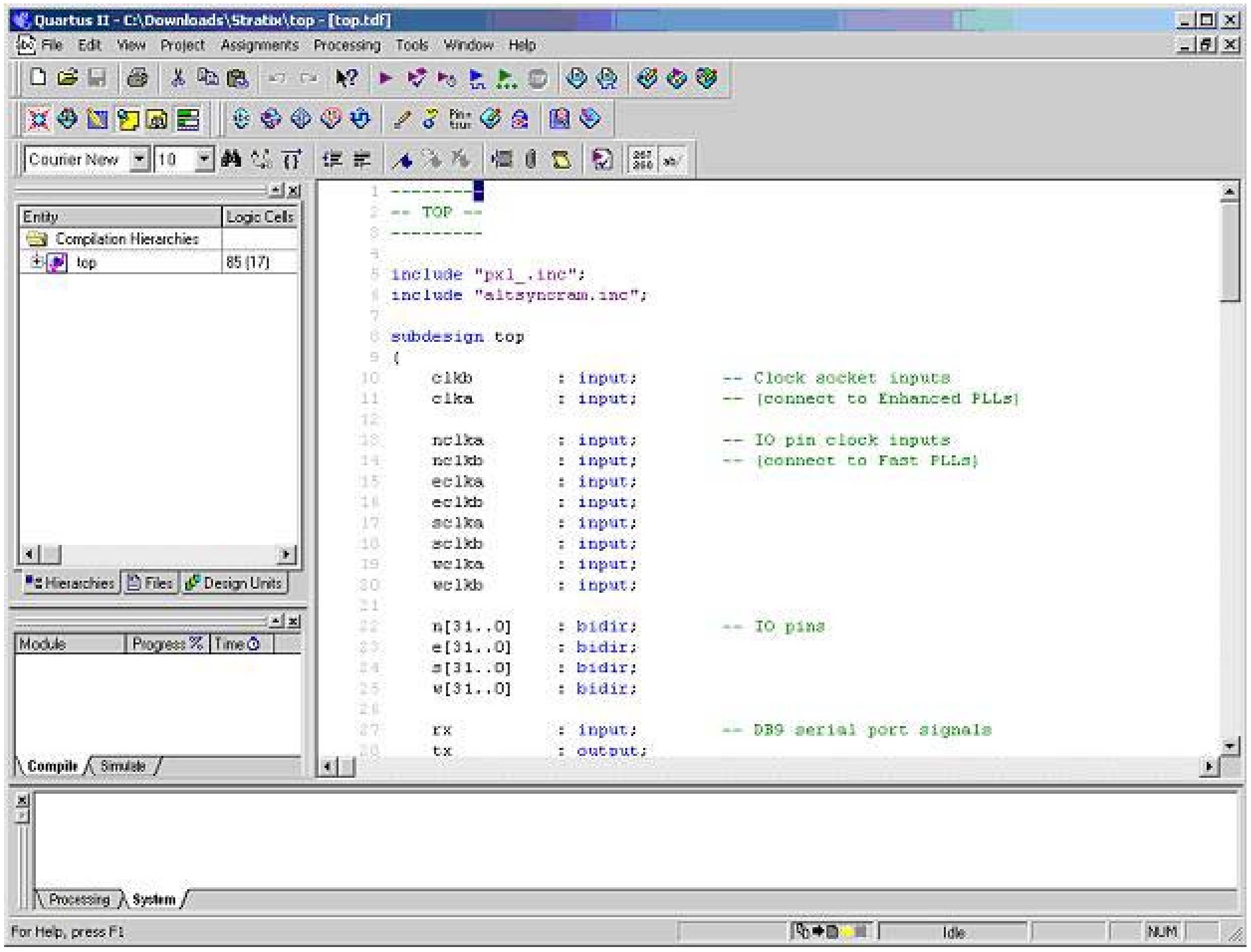
SmartPack Connected to Serial Cable and Power Supply

4. Launch the Quartus II software. If you receive a licensing note then you need to do one of the following: (a) Quartus II software – install the dongle on your PC's parallel port and copy the license file to the appropriate software directory then use the Tools/License Setup to browse and find the file; or (b) Quartus II Web Edition software – copy the license file to the appropriate directory and use the Tools/License Setup menu to find the file (no dongle required).



Quartus II Software Running

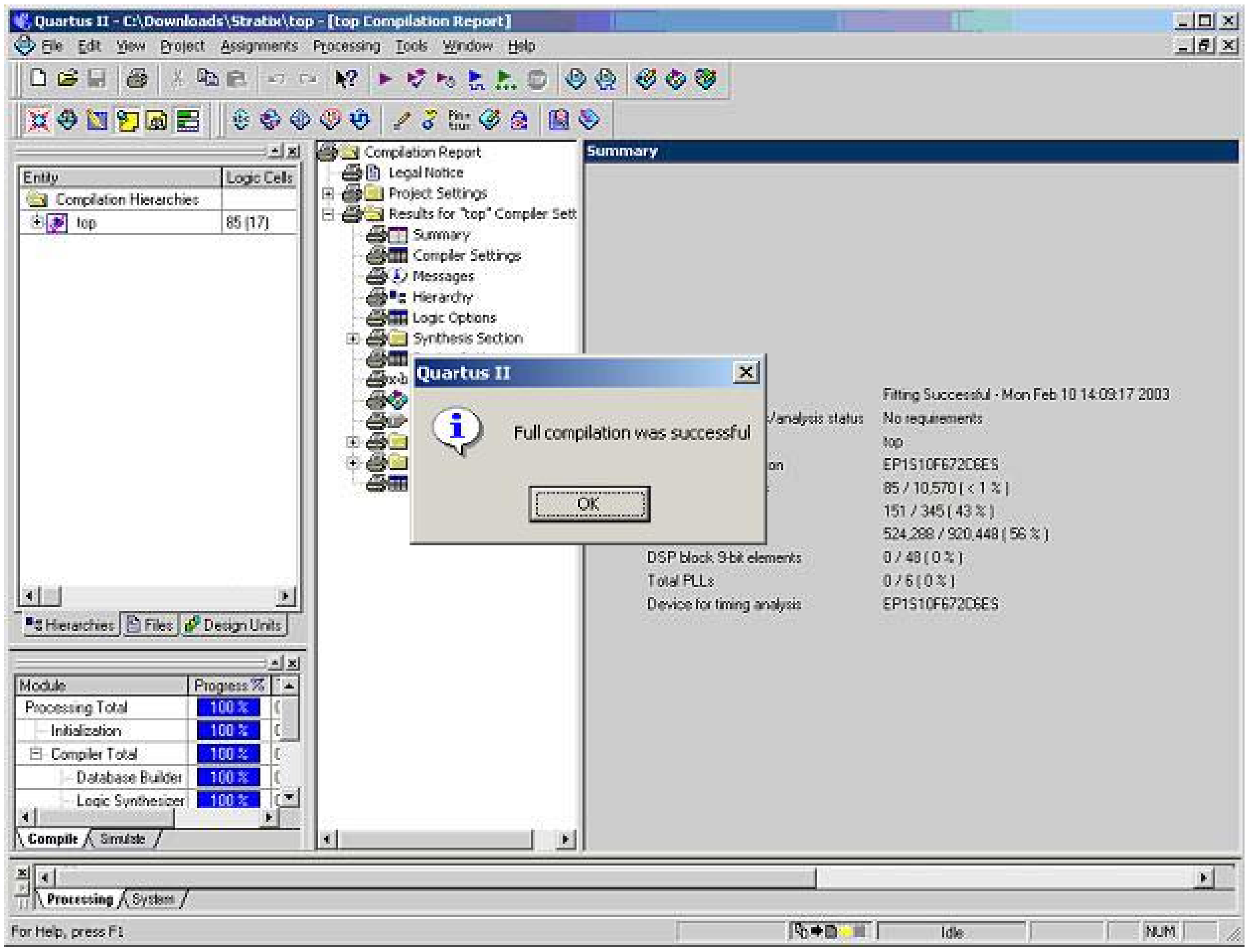
5. Use File/Open Project and find the TOP Quartus II project file in either the EP1C3 (FastPack), EP1C20, EP1S10, or EP1S25 subdirectory. Once opened, you will see the 'Compilation Hierarchies' box in the upper-left area. Double-click the 'top' section to open TOP.TDF, the top-level design file.



```
---------
-- TOP --
---------

include "pxl_.inc";
include "altsyncram.inc";

subdesign top
(
    clkb        : input;        -- Clock socket inputs
    clka        : input;        -- (connect to Enhanced PLLs)

    nclka       : input;        -- IO pin clock inputs
    nclkb       : input;        -- (connect to Fast PLLs)
    eclka       : input;
    eclkb       : input;
    sclka       : input;
    sclkb       : input;
    wclka       : input;
    wclkb       : input;

    n[31..0]    : bidir;        -- IO pins
    e[31..0]    : bidir;
    s[31..0]    : bidir;
    w[31..0]    : bidir;

    rx          : input;        -- DB9 serial port signals
    tx          : output;
```

Quartus II file "top.tdf"

6. Compile the design using the "play" button. Upon completion, a TOP.RBF file will be generated which contains the compiled design.



Quartus II - C:\Downloads\Stratix\top - [top Compilation Report]

Full compilation was successful

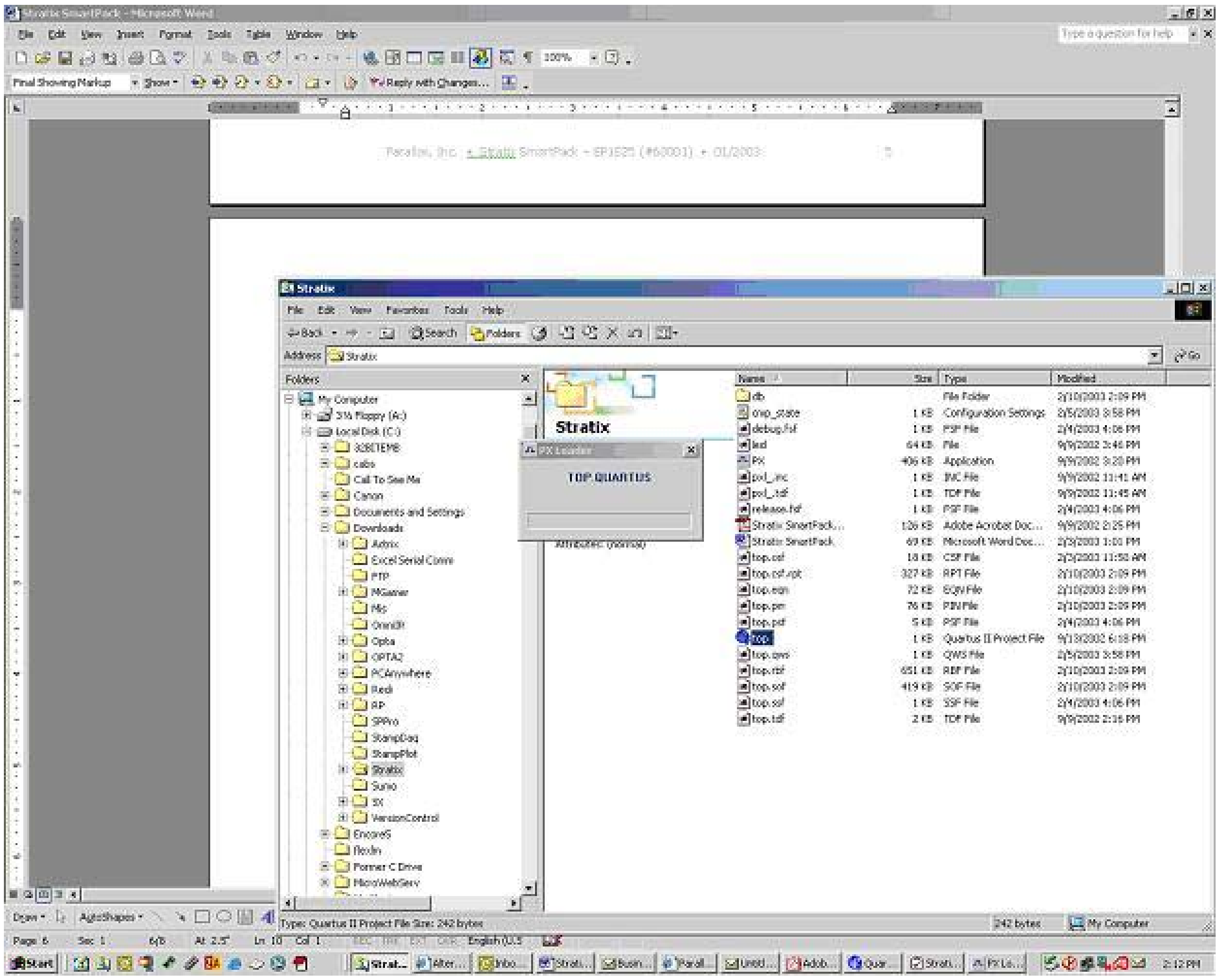
	Fitting Successful - Mon Feb 10 14:09:17 2003
/analysis status	No requirements
	top
on	EP1S10F672C6ES
	85 / 10,570 (< 1 %)
	151 / 345 (43 %)
	524,288 / 920,448 (56 %)
DSP block 9-bit elements	0 / 48 (0 %)
Total PLLs	0 / 6 (0 %)
Device for timing analysis	EP1S10F672C6ES

Module	Progress %
Processing Total	100 %
Initialization	100 %
Compiler Total	100 %
Database Builder	100 %
Logic Synthesizer	100 %

Generating the "top.rbf" File

7. Parallax has built a serial loader used to load the RBF file into the FPGA. From a Command Prompt, with PX.EXE in the path, type: PX yourfile.RBF. Or, within a Window, drag the TOP.RBF file onto PX.EXE for serial downloading. If you are not using COM1, add /# after the command, where # is the COM port number (1-4).

8. The Parallax loader is not a complete GUI application – it's only a command-line interface. During configuration loading, a window temporarily appears indicating status.



Name	Size	Type	Modified
db		File Folder	2/10/2003 2:09 PM
cmp_state	1 KB	Configuration Settings	2/5/2003 3:58 PM
debug.fsf	1 KB	PSF File	2/4/2003 4:06 PM
led	64 KB	File	9/9/2002 3:46 PM
PX	406 KB	Application	9/9/2002 3:20 PM
pxl_.inc	1 KB	INC File	9/9/2002 11:41 AM
pxl_.tdf	1 KB	TDF File	9/9/2002 11:45 AM
release.fsf	1 KB	PSF File	2/4/2003 4:06 PM
Stratix SmartPack...	126 KB	Adobe Acrobat Doc...	9/9/2002 2:25 PM
Stratix SmartPack	69 KB	Microsoft Word Doc...	2/3/2003 1:01 PM
top.csf	18 KB	CSF File	2/3/2003 11:50 AM
top.csf.rpt	327 KB	RPT File	2/10/2003 2:09 PM
top.eqn	72 KB	EQN File	2/10/2003 2:09 PM
top.pin	76 KB	PIN File	2/10/2003 2:09 PM
top.psf	5 KB	PSF File	2/4/2003 4:06 PM
top	1 KB	Quartus II Project File	9/13/2002 6:18 PM
top.qws	1 KB	QWS File	2/5/2003 3:58 PM
top.rbf	651 KB	RBF File	2/10/2003 2:09 PM
top.sof	419 KB	SOF File	2/10/2003 2:09 PM
top.ssf	1 KB	SSF File	2/4/2003 4:06 PM
top.tdf	2 KB	TDF File	9/9/2002 2:16 PM

Additional Notes:

To program the boot flash, add /P after the command.

To use the memory loader, integrate the PXL_.TDF code into your design and specify the memory file in the command line (non-.RBF). You may specify both an .RBF and a memory file in the command line. This works with /P, too. This way, at power-on, the PX Loader can load the FPGA's memory immediately after configuring the device.

The included Quartus II project TOP.QUARTUS contains all appropriate device and pin settings for the SmartPack. It is recommended that you copy and use this as a base for your own application.

Memory Loader Details

When PX.EXE is given a non-.RBF file, it treats it as a memory file. The signals PX0, PX1, and PX2 are used as "data", "clock", and "load/run", respectively.

The memory loading process is as follows:

1) RES is pulled low, >40us elapses.
2) PX2 is pulled low, >40us elapses.
3) RES is released high, >40us elapses.
    - At this point, the configured FPGA "wakes up" and sees that PX2 is low and knows that "memory-mode" is active.
4) Bit 0 of the first byte of the memory file is placed on PX0 while PX1 is driven low.
5) PX1 is driven high to clock in the data on PX0.
6) Bits 1-7 are sequentially clocked out in the same way.
7) The next byte in the memory file is shifted out, and so on until the file is completely sent.
8) RES is driven low, >40us elapses
9) PX2 is released high, >40us elapses
10) RES is released high
    - The FPGA "wakes up" again, but this time sees that PX2 is high and goes about "run-mode" operations

The PXL_.TDF module facilitates memory loading by accepting PX0-PX2 inputs and a clock, and providing a load signal and clock-synchronized write-enable, address, and data signals.

This parameterized module can be set for varying data-bus and address-bus widths, enabling any width and length of memory to be loaded. All data is ordered least- to most-significant from the memory file to the target memory. For example, if you had a 32-bit wide memory, data bytes #0-#3 would make up word #0, data byte #0 being the least-significant. On the other hand, if you had a two-bit-wide memory, bits #0-#1 of byte #0 would make up the first word, bit #0 being the least-significant.

It's called PXL_ so that you can instantiate it simply as PXL.

Files

PX.EXE	program for loading Stratix and memory
<Schematic>.PDF	PDF schematic files
TOP.QUARTUS	example project
TOP.CSF	example project's configuration file
TOP.TDF	example project's top-level AHDL file
LED	example memory file – Type "PX LED" to load it
PXL_.TDF	PXL module for integration into your application
PXL_.INC	PXL module's include file

*** Special Note

The early versions of the EP1S25 have a latch-up problem, hence the 5A requirement. Typically, the entire SmartPack consumes less than 300ma, but to power it up takes a current surge peaking at over 3A. One remedy for the latch-up problem is to connect the wall pack to an AC outlet, then plug it into the SmartPack power jack. This way, the wall pack can provide, via its internal capacitors, the current surge needed to push the Stratix through its latch-up voltage of 450mV.

Some Advice About Altera FPGA Development

We have been using Altera FPGA's internally at Parallax for over 3 years now. We have written extensive application code and have come to some surprising determinations about the best course of development. Here's what we recommend:

For programming languages, Quartus II supports Verilog, VHDL, and AHDL. The first two are common standards, while AHDL is Altera's own language. Use AHDL as your programming language. It's simple, direct, and can always be turned back into a schematic. Think of it as an ideal assembly language for FPGA's. We have written in Verilog, since it's portable, but didn't find it nearly as easy to use as AHDL. We took one look at VHDL and saw that for whatever you were trying to accomplish, you had way too much typing to do. You don't need your ideas getting lost amid a sea of text.

When using AHDL, avoid IF-THEN-ELSE-ELSEIF-YOURGOINGTOGETABIGHEADACHE. It's too easy to innocently specify what will become unnecessarily complex logic. Altera's manuals warn of this. Plus, this is a bad allegory to the software IF-THEN, as it suggests procedure and singularity – both invalid concepts here. In retrospect, we have used every AHDL language construct in the course of our own development, but as things evolved, there were only TWO constructs used in thousands of lines of code: assignments and CASE statements. Those two constructs will let you accomplish anything, keep you efficient and sane, and help the compiler do its job. Yes, it's that simple – assignments and CASE statements.

If you're coming from a software background, you are entering a new and exciting world. All of your lines of code will now be executing simultaneously. It takes a little while to get a handle on this. No matter what you design, this is what it will do: It will use combinatorial logic to set up the 'data' (and maybe 'enable') inputs to flipflops. These flipflops are clocked by some signal, probably the 50MHz oscillator that comes with the SmartPack. The outputs of these flipflops will probably get used, along with external inputs, to again set up the 'data' and 'enable' lines on the flipflops. Some of these flipflop outputs may go to pins. Where does speed come in? How fast can your logic settle out before the clock goes high again, registering the next state? Too much logic and you can't run as fast. Simple logic can be clocked at 400MHz+. To get those high speeds, you will use on-chip PLL's to rev up your clock input to something faster. Oh, yes, be careful about introducing external pin signals into your clean machine. You may need to register them (run them through a flipflop) to keep them steady for your internal logic, within the brief clock cycle. Actually, you need to read this and ponder it:

http://www.fpga.com.cn/advance/skill/commandments.pdf

The Quartus II Compiler is pretty smart. It boils down all your logic equations, throws out what's not needed, optimizes what's left, and then reconchunkulates it to fit into LE's (Logic Elements – the atomic 4-to-1 function with flipflop and adder chain). The Altera FPGA's are massive assemblages of LE's. There is a hierarchy to their arrangement to make routing efficient (or even possible), and there's other stuff, too, like RAMs and IO Pins, PLL's, and DSP Blocks. You will not need to worry about those, though, as all these things get put to use by the compiler. After a compilation, you can view the physical result in a graphical window. You can see how and where LE's got employed, what RAMs got used, etc. This helps you gain an appreciation for what is going on, which may make you a better designer.

I suppose there's a lot more to be said, but until we at Parallax, or perhaps you, make some good texts on this subject, you will need to study the on-line Help to learn about AHDL. Remember: assignments and CASE statements ('group' notation makes these very powerful). Learn the boolean operators and their order of precedence. Also, look at our example code. Its main point is to exercise the hardware and teach by example. It should give you a good idea about how to begin an application.

It would have been great if someone who knew what they were talking about would have told us these things 3 years ago and we would have listened to them, the latter being less likely. We'd have been way ahead now. So, this is your opportunity…

"A wise person does at once, what a fool does at last. Both do the same thing; only at different times."
– John Dalberg Acton.

Lucky you! Have fun…

P.S. What will be built from Cyclone and Stratix FPGA's are things not yet conceived. The gap between FPGA capability and applications is huge! Think of the FPGA as the quantum step up from the microcontroller. You can build several simple microcontrollers within even the minimal Cyclone EP1C3 – and you can make all of them run at 100 MIPS. Things are possible that are not yet dreamed of. It will be at least 10 years before mainstream applications catch up to even today's technology. In the near future, Altera will introduce versions of the Stratix and Cyclone in 90nm process technology. That could easily mean 32-bit adder chains made from LE's running at 600MHz! Things are getting so fast that time-of-flight measurements are becoming practical for light! Down the spectrum, RF signals can be numerically synthesized and processed with just a handful of LE's. Wild things are possible. It's a great time to be alive and getting into FPGA's!