Platform8051 Development Kit

User's Guide

Actel®

Table of Contents

Introduction

The Platform8051 Development Kit is intended as a demonstration and evaluation medium for Actel's Core8051 Microcontroller IP and Core10/100 Ethernet MAC IP. The kit is delivered pre-programmed with a simple web server demonstration project (running over the Ethernet connection). However, Platform8051 is not limited to this application. The Flash-based ProASIC$^{\underline{\text{PLUS}}}$ FPGA that is at the core of the design may be reprogrammed and used to evaluate other Actel-based intellectual property (IP) or to implement any number of possible designs that may or may not make use of Actel IP.

Using This Document

This User's Guide describes the contents, architecture, and guidelines for working with the Platform8051 Development Kit. Use this document for the following:

- Installation and setup instructions for Platform8051 and related tools
- Detailed user information and description of Platform8051 example designs
- Detailed reference material to be used when implementing a new design using Platform8051, Platform8051 Development Kit, and/or Core8051

Platform8051 Development Kit Contents

- ProASIC$^{\underline{\text{PLUS}}}$-based Platform8051 board
- All cables and power supplies
- Actel Libero® Integrated Design Environment (IDE) software, evaluation license (a full license may be purchased separately from Actel)
- Programming files for example FPGA designs (source files and IP for designs must be purchased separately)
- Programming and source files for Platform8051 example software designs
- FS2® CD with 8051 debugger software and Windows® drivers
- Keil Software® µVision2™/C51 C compiler/debugger, evaluation license (a full license may be purchased separately from Keil Software)

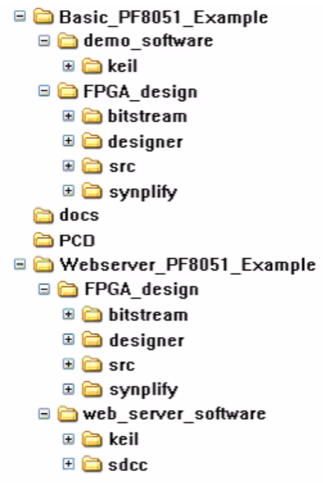
Recommended Additional Products

- Actel FlashPro Lite (Actel ordering code “FlashPro Lite”)
- Core8051 IP license (Actel ordering code “Core8051-SN/AN/AR”)
- Core10/100 IP license (Actel ordering code “Core10/100-SN/AN/AR”)

- Actel Libero software license (Actel ordering code “Libero-Platinum-PC-N-1year”)
- Optional: Keil µVision2 software license (Keil part number “PK51,” “DK51,” or “C51” see www.keil.com)
- Optional: FS2 Debugger Upgrades (www.fs2.com)

Actel Platform8051 Development Kit CD ROM

Along with CDs for Actel and third-party EDA tools, the Platform8051 Development Kit includes a CD containing Actel example files, documentation, and programming files for the example design. (See Figure 1.) This CD contains folders for the Platform8051 Development Kit documentation (including a QuickStart guide, Testing and Programming Procedure, and this User’s Guide). Other folders contain printed circuit board (PCB) information (schematic and bill of material), example design software (C-based projects and source code), and FPGA example designs source files for the hardware design (minus the Core8051 and Core10/100 IP, which must be purchased separately). While all the source code for the FPGA designs is not on the CD, the top-level netlist is a useful reference for understanding and evaluating the components of Platform8051. The FPGA design folder also contains a *bitstream* subfolder. This is where a copy of the programming files for the example FPGA designs can be found.



```
Basic_PF8051_Example
    demo_software
        keil
    FPGA_design
        bitstream
        designer
        src
        synplify
docs
PCD
Webserver_PF8051_Example
    FPGA_design
        bitstream
        designer
        src
        synplify
    web_server_software
        keil
        sdcc
```

Figure 1. Contents of PF8051 Development Kit v2.1 CD

1 Platform8051 PCB

The Platform8051 Development Kit is intended as a demonstration and evaluation medium for a variety of Actel IP, including Core8051, Core10/100, CoreSPI, CoreI2C, CoreSDLC, Core16X50, CoreUART, and the Utopia family cores (slight modification required). With the daughter card it is a versatile board that can demonstrate a variety of other cores, including Core1553BRM. See Figure 1-1.

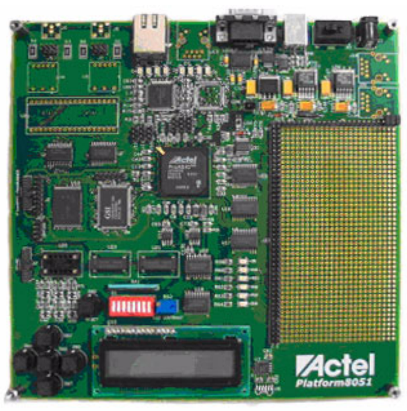


Figure 1-1. PF8051 Development Board

The Platfom8051 board contains the following:

- ProASIC$^{\underline{PLUS}}$ APA600-FG676
- 4 MB SRAM memory
- 32 MB Flash memory
- 10/100 Ethernet PHY (Physical Layer Device)
- RS-232 serial interface
- LCD display
- 10-bit ADC

All subsystems can be addressed by the APA600. In addition, there are several unpopulated component footprints on the Platform8051 board, including locations for a second Ethernet PHY, a

1553B Transceiver, and USB circuitry. A complete bill-of-material and schematic are on the Platform8051 CD under the *PCB* folder. A block diagram of the PCB is shown in Figure 1-2.

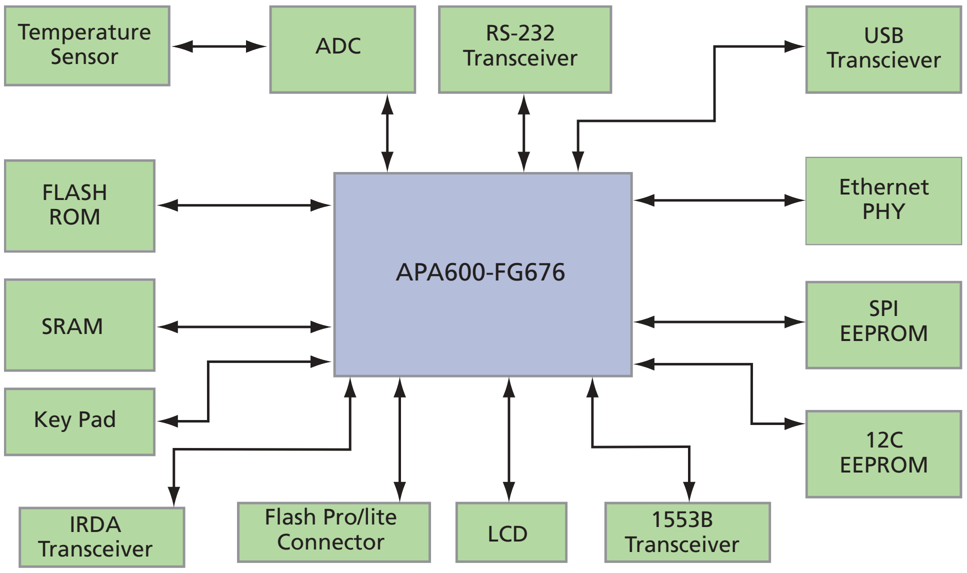


Figure 1-2. Platform8051 Development Board Block Diagram

APA600 Overview

The Platform8051 Development Kit is based on Actel's second-generation Flash product family, ProASIC$^{\underline{\text{PLUS}}}$. ProASIC$^{\underline{\text{PLUS}}}$ expands on all the features and benefits offered by the ProASIC 500K family. Based on 4 LM .22 µ Flash technology, ProASIC$^{\underline{\text{PLUS}}}$ offers the combination of reprogrammability and nonvolatility in a 75,000- to 1 million-system gates programmable logic product. ProASIC$^{\underline{\text{PLUS}}}$ combines the advantages of ASICs with the benefits of FPGAs, enabling engineers to leverage their existing ASIC or FPGA design flows and tools.

ProASIC$^{\underline{\text{PLUS}}}$ key features:

- Reprogrammable (In-System Programming, or ISP) and nonvolatile
- FlashLock™ security
- Live at power-up
- Low power
- ASIC design flow

The ProASIC$^{\underline{PLUS}}$ APA600 device used in Platform8051 has 600,000 available system gates, 21,504 registers, 126 kbits of RAM, and four global clock networks, with up to 56 total low-skew clock segments. More detailed information about ProASIC$^{\underline{PLUS}}$ devices is available in the *ProASIC$^{\underline{PLUS}}$ Flash Family FPGA Data Sheet*.

LCD Display

The on-board, 1 x 16 LCD display is controlled through a simple memory-mapped interface. The LCD controller is contained in the LCD module. See "LCD Command and Data Register" on page 47 for details of how the Core8051 interfaces with the LCD display.

ADC

An analog-to-digital converter (ADC) is also part of the Platform8051 Development Board and is controlled through a register-mapped interface inside the FPGA.

DIP Switches

A bank of eight DIP switches is also on the board. These switches are wired to inputs of the FPGA so that *open* will correspond to logic 1 (4.6 k pull-up to 3.3 V) and *closed* will correspond to logic 0 (GND). In the example designs, these inputs to the FPGA are used to determine which of four possible memory configurations the Core8051 will use. See "Basic Platform8051 Demonstration Project" on page 51 and "Web Server Demonstration Project" on page 59 for more details on available memory configurations.

Jumpers

There are a variety of jumpers and headers on the board used for configuration and debugging purposes. The functions of these jumpers are defined in Table 1-1 on page 10.

Table 1-1. Jumpers and Headers

Jumper	Signal	Description	Function
J1	V2.5	Connects 2.5 V power to APA600 core	Always connect pins 1 and 2.
J2	{8:1} = VJTAG_RCK, TMS, TDO, TDI, TCK, GND, VPN_AD24_IO VPP_Y20	Aux. JTAG	Auxiliary JTAG head. In parallel to J3
J3		FlashPro JTAG header	FlashPro or FlashPro Lite connection for APA600 programming and in-system debugging of Core8051
J4	{6:1} = BUIP_IO_GLMX_P22 BUIP_IO_GLMX_N3 AUIP_PPECL_P24 AUIP_PPECL_P5 AUIN_NPECL_N24 AUIN_NPECL_N5	PECL inputs to APA600	Optional additional clock inputs to FPGA
J10	AN_EN	Configures Ethernet PHY U13 Pin 1 = VCC, Pin 3 = GND, Pin 2 = signal	See Table 1-2 on page 11.
J11	AN_1		
J12	AN_2		

Table 1-1. Jumpers and Headers (Continued)

Jumper	Signal	Description	Function
J14	AN_EN	Configures Ethernet PHY U14 Pin 1 = VCC, Pin 3 = GND, Pin 2 = signal	See Table 1-2.
J15	AN_1		
J16	AN_0		
J17	{12,14,16,18,22} = FPGA_HDR_TRST, FPGA_HDR_TMS, FPGA_HDR_TDO, FPGA_HDR_TDI, FPGA_HDR_TCK	Spare FlashPro JTAG header for FS2CLAM®	{22} = APA600 pin# L5 {18} = APA600 pin# L4 {16} = APA600 pin# L3 {14} = APA600 pin# L2 {12} = APA600 pin# L1
J22	{4:1} = FPGA_OPPINS[4:1]	OPPINS	{4} = APA600 pin# K4 {3} = APA600 pin# K3 {2} = APA600 pin# K2 {1} = APA600 pin# K1

Table 1-2. Jumper J10-J16 Function Description

AN_EN	AN1	AN0	Forced Mode
0	0	0	10Base-T, Half-Duplex
0	0	1	10Base-T, Full-Duplex
0	1	0	100Base-TX, Half-Duplex
0	1	1	100Base-TX, Full-Duplex
AN_EN	AN1	AN0	Advertised Mode
1	0	0	10Base-T, Half/Full-Duplex
1	0	1	100Base-T, Half/Full-Duplex
1	1	0	10Base-T, Half-Duplex 100Base-TX, Half-Duplex
1	1	1	10Base-T, Half/Full-Duplex 100Base-TX, Half/Full-Duplex

2

8051 Implementation

There are many design decisions that must be made in configuring the Core8051 and peripheral components in Platform8051 implementation. General information about configuring Core8051 is defined in this section. Specific information about the configuration used in the Platform8051 example projects can be found in sections "Basic Platform8051 Demonstration Project" on page 51 and "Web Server Demonstration Project" on page 59.

Core8051 Memory Map Considerations

A brief explanation of the 8051 memory configuration options is provided in this document. The design engineer will need to understand both this and the specific requirements of the application to plan the design of the memory map.

There are four separate memory regions used in a Core8051:

- DATA – 256 bytes x 8 bits wide – used for dynamic storage of program data (registers, stack, variables)
- CODE – 64 k bytes x 8 bits wide – used for program storage and interrupt vectors
- XDATA – 64 k bytes x 8 bits wide – used for storage of large data sets, custom-designed peripherals, and extended stack space, if necessary
- SFR – 128 bytes x 8 bits wide – a combination of internal (to the core) and external memory for special function registers

CODE, DATA, and XDATA memory spaces are not a part of Core8051 and therefore must be implemented by the user, either internal or external to the FPGA. The SFR memory space consists of two parts: internal to the core and external. The internal SFR memory space is implemented in the core. The user must implement the optional, external SFR memory space outside the core.

The Core8051 memory spaces can be implemented either as synchronous or asynchronous. The recommended mode when using RAM that is inside of ProASIC$^{\underline{\text{PLUS}}}$ devices is to make them synchronous on write and asynchronous on read. This is the default configuration that should be considered first. Use the same clock source as the Core8051 for the RAM WCLOCK. Some designs have improved timing by changing to synchronous reads as well as writes. When using off-chip RAM, the timing between the Write Enable, Output Enable, and Data should be carefully considered and reviewed after layout.

CODE and XDATA Memory Implementation

The CODE memory space is 64 k bytes x 8 bits wide, and is used for program storage and interrupt vectors. After reset, Core8051 always starts at address 0x0000 in the CODE space. The XDATA memory space is also 64 k bytes x 8 bits wide and is used for storage of large data sets, custom-designed peripherals, and extended stack space, if necessary. XDATA and CODE can be constructed to consume less than the entire 64 k memory map, allowing for more efficient memory usage. However, the user must ensure that sufficient memory space is allocated for the program. The Keil compiler/debugger always fills the CODE space, starting at 0x0000 (or 0x0800 in the eval version) and goes upward. The upper limit of CODE space is dependent on the size of the application. Similarly, the size of XDATA RAM needed is defined by the application code.

The Core8051, DATA, XDATA, and CODE each have their own write-enable (WR) and read-enable (RD) signals (see Table 2-1). It is feasible and common practice to combine the CODE and XDATA memory spaces into the same physical memory. In some applications, XDATA and CODE are even overlaid in the same physical memory region. In these cases, the user must prevent the CODE memory space from being overwritten by XDATA variables.

Table 2-1. Core8051 Memory Bus Signal Summary

Memory Space	Data Bus	Address Bus	RD/WR Control	Other Controls	Description
XDATA	memdatao[7:0] – output memdatai[7:0] – input	memaddr[15:0]	memwr – write enable memrd – read enable	memacki – acknowledge (for multi-cycle operation)	Memory bus for large data sets in RAM, Flash programming, or custom 8051 peripherals
CODE	memdatao[7:0] – output memdatai[7:0] – input	memaddr[15:0]	dbgmempswr – write enable mempsrd – read enable	mempsacki – acknowledge (for multi-cycle operation)	Memory bus for program code memory in RAM or Flash
DATA	ramdatao[7:0] – output ramdatai[7:0] – input	ramaddr[7:0]	ramwe – write enable ramoe – read enable		Memory bus for internal Core8051 registers and stack space in RAM
XSFR	sfrdatao[7:0] – output sfrdatai[7:0] – input	sfraddr[6:0]	sfrwe – write enable sfrre – read enable		(Optional) Bus for mapping custom peripherals into Core8051 SFR space

Core8051 treats the CODE memory space as read only. The CODE memory can only be written by the OCI® (On-Chip Instrumentation) debug circuitry. If the OCI is not used, the application designer will have to develop his own method of loading the CODE memory. Intentionally mapping XDATA over CODE memory space so that the 8051 can update all or part of its own code is an advanced application that is beyond the scope of this document. Code/data banking, the use of additional 64 k blocks for CODE or XDATA memory space to accommodate applications with large code or data requirements, can be easily implemented with Core8051. This is due to the flexibility of Actel FPGAs to create chip select signals differentiating between 64 k banks. However, code/data banking is not covered in this document.

The Platform8051 demo board has a switch in the run-time library (RTL) that enables the memory map to be changed by either the P3 register from Core8051 or from DIP switches on the board. By default, XDATA and CODE share common space overlaying the address ranges.

DATA Memory Implementation

Internal Data (DATA) may be created on-chip or off-chip. However, it requires single-cycle access. This may be a determining factor in your final system clock speed. DATA must always be kept in a separate memory as it may be accessed in the same cycle as CODE or XDATA. It is common to create DATA memory as 128 or 256 bytes. The lower 128 bytes are required in all Core8051 designs and respond to direct or indirect addressing. The upper 128 bytes provide additional space for variables and the Core8051 stack, however, it can only be addressed indirectly. Direct addressing to the upper 128 bytes will result in an SFR access, not a DATA access. (An example of indirect addressing is an instruction where the target address is stored in a register. Direct addressing is when the address is explicitly expressed in the instruction.)

Customizations to SFR Memory

The SFR internal memory region in Core8051 gives the user access to various functions of the core such as data pointers, timers, interrupts, etc. These registers are defined in the RTL code for Core8051 and no additional logic needs to be added by the user for implementation. See the *Core8051 Datasheet* for more information.

External SFR (XSFR) memory is an optional feature that provides a means for connecting custom peripherals or for implementing custom features into Core8051. An alternative method for integrating custom features is to construct them so that they respond to external data (XDATA) memory addresses. The choice between creating custom memory mapped features in XDATA vs. XSFR is up to the designer.

The ways in which SFR memory is used by a standard Core8051 system are limited to the registers listed in Table 2-2 on page 16. All other SFR memory addresses may be decoded and used to implement custom peripherals. If an address is chosen that is already implemented inside Core8051, the external register will be ignored.

Table 2-2. SFR Register List

Register	Address	Description
p0	80h	Port 0
Sp	81h	Stack Pointer
dpl	82h	Data Pointer Low 0
dph	83h	Data Pointer High 0
dpl1	84h	Dual Data Pointer Low
dph1	85h	Dual Data Pointer High
pcon	87h	Power Control
tcon	88h	Timer/Counter Control
tmod	89h	Timer Mode Control
tl0	8Ah	Timer 0, low byte
tl1	8Bh	Timer 1, high byte
th0	8Ch	Timer 0, low byte
th1	8Dh	Timer 1, high byte
ckcon	8Eh	Clock Control
p1	90h	Port 1
dps	92h	Data Pointer Select Register
scon	98h	Serial Port 0, Control Register
sbuf	99h	Serial Port 0, Data Buffer
p2	A0h	Port 2
ien0	A8h	Interrupt Enable Register
ip0	A9h	Interrupt Enable Register
p3	B0h	Port 3

Table 2-2. SFR Register List (Continued)

Register	Address	Description
ien1	B8h	Interrupt Enable Register
ip1	B9h	Interrupt Enable Register
psw	D0h	Program Status Word

Core8051 Clock Configuration for Platform8051

To ensure proper operation of Core8051, the input clocks must be placed on low-skew global buffers, such as RCLK or HCLK in Actel's antifuse devices, or on clock spines in Actel's ProASIC$^{\text{PLUS}}$ devices. One method for ensuring proper placement of the input clocks is to use the set_global command in the ProASIC$^{\text{PLUS}}$ GCF constraints file used by Actel's Designer software.

Core8051 has three clock inputs: CLK, CLKPER, and CLKCPU. The most efficient way to implement the Core8051 clocking is to drive all three of these inputs with the same global signal, as in Figure 2-1 on page 18. Combining all clocks will save FPGA resources and ensure a high-speed design with easy-to-analyze timing. However, unifying the three clocks has the drawback of disabling the Core8051's low power modes IDLE and STOP. The IDLE and STOP modes depend on having three independent clocks inputs.

If IDLE and/or STOP modes are required, then CLKPER and CLKCPU must be implemented as shown in Figure 2-2 on page 18. This will make cross-clock domain skew analysis more difficult and consume more of the FPGA's global clock resources (all three clock inputs to Core8051 must be on low-skew global buffers or clock spines).

Peripherals in the FPGA that are related to Core8051, but not part of it, should use the same clock as the CLKPER input. If unified clocking is used, then that one clock should be used for all user logic. If the gated clocking is used, then the gated clock CLKPER should be used for user logic.

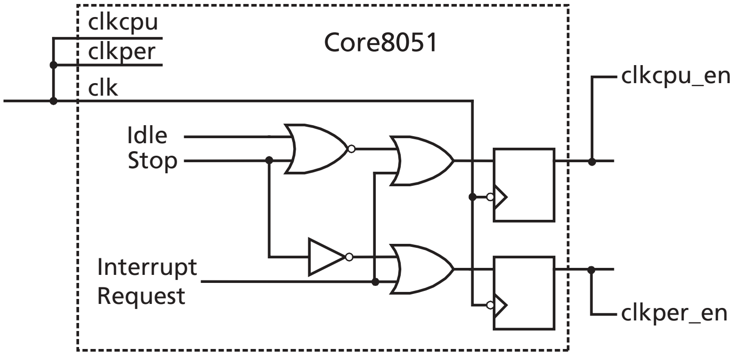


Figure 2-1. Core8051 with Unified Clocks (no IDLE/STOP mode)

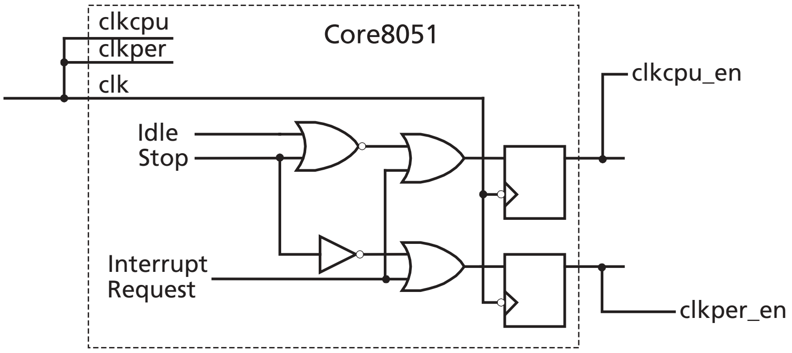


Figure 2-2. Core8051 with Gated Clocks (enables IDLE/STOP modes)

Core8051 FPGA Resource Utilization

The performance numbers in the Core8051 datasheet are based on timing analysis of the Core8051 in a device without any memories or additional logic (Table 2-3 on page 19). In most ProASIC$^{\underline{PLUS}}$ Platform8051 designs, it will be advantageous to use the RAM resources embedded in the FPGA. In these designs (such as the demonstration projects), the memory access time is the limiting factor for core performance. Of course, any additional logic, such as a memory controller, will also affect the size and utilization of the design.

Table 2-3. Device Utilization for Core8051 (only)

Family	Cells or Tiles		Utilization		Performance
	Sequential	Combinational	Device	Total	
ProASIC<u>PLUS</u>	547	3994	APA150-STD	74%	16 MHz
Axcelerator®	690	2560	AX250-3	77%	45 MHz
SX-A	685	2790	A54SX72A-3	58%	30 MHz
RTSX-S	685	2790	RT54SX72S-2	58%	12 MHz

Core8051 OCI Debugger Interface Implementation

Introduction

The On-Chip Instrumentation extension to the Actel Core8051 enhances the CPU core functionality, providing run-time control, memory and register visibility, complex breakpoint capability, and a trace history feature, all without using any resources from the core CPU.

OCI Features

The OCI module inside Core8051 has many powerful capabilities, including the following:

- Control via a 4-pin IEEE-1149.1 (JTAG) port, compatible with daisy-chained multi-core systems
- Start/stop run control through debugreq/debugack signals to core
- Unlimited number of software breakpoints available via 0xA5 opcode
- Single-step by assembly instruction
- Access to all 8051 registers and memory spaces (CODE, XDATA, SFR, and DATA)
- Scalable number of hardware breakpoints (zero to four) consisting of one address/data value and one of the following modes:
  - Code memory execution

    - Code memory read or write
    - External data memory read or write
    - SFR read or write
- Ability to read and write to internal data memory
- Capability to combine two hardware breakpoints to form an address range (lower and upper bound) and masked data value
- Hardware breakpoints may be configured to break emulation, start or stop the trace, or assert a trigger-out signal
- Optional break bus can be used to synchronize operation of multiple Core8051 devices connected in a multi-core configuration
- Optional AuxOut signal available to control on-chip test modes or other system-specific functions
- Optional trace history of the most recent branch points allows software reconstruction of execution flow. Memory is configurable in powers of two from 2 to 256 frames. Branches record both branch-from and branch-to addresses. Trace start/stop actions from the trigger also allocate a trace frame
- Support for code memory bank switching systems. Additional bits denoting the bank number are supplied by user logic and participate in breakpoint decisions and trace
- Presence of the OCI does not impact processor performance significantly (see the *Core8051 Datasheet* for detailed performance estimates)

Configuring the OCI

Actel customers who purchase either the complete Core8051 source code or Core8051 netlist have access to the OCI module (see the *Core8051 User's Guide* on the CD for details). FPGA resources can be saved by omitting the OCI from the design; however, this will prevent in-system debugging. Omitting the OCI will also force the user to implement another means of programming the CODE memory (which is normally done through the debugger and OCI). Netlist customers are provided netlists with and without the OCI module included.

Note: The number of triggers and trace memory size is encoded into the netlist name, as described in Figure 2-3.

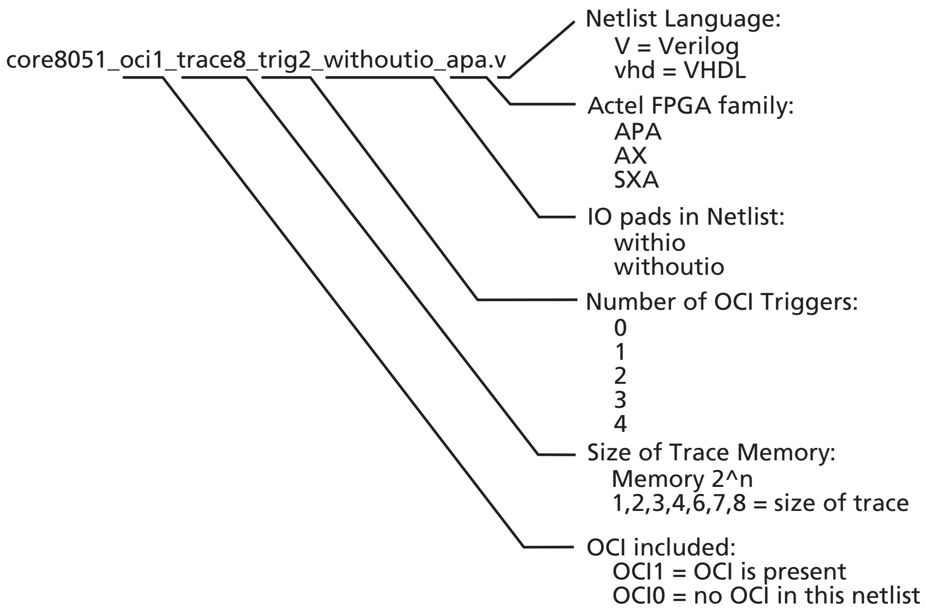


Figure 2-3. Core8051 Netlist Naming Conventions

The demonstration projects for the Platform8051 Development Kit have been created using the file *core8051_oci1_trace8_trig1_withoutio_apa.vhd*. This gives the design an OCI with one hardware breakpoint, 256-line x 20-bit trace buffer.

Customers working from the source RTL (rather than netlists) must set the generic parameters for Core8051 appropriately for their needs. These map directly to the netlist options and are defined in great detail in the *Core8051 Datasheet*, and the *Core8051 User's Guide* (located on the CD). The generic parameters used by RTL customers to configure Core8051 are defined in the following code segment:

```
component CORE8051
    generic (
    -- set this to 1 to instantiate OCI logic
    USE_OCI    : integer := 0;
    -- set this to 1 to use ProASIC+ UJTAG macro for OCI logic
    USE_UJTAG: integer := 0;
    -- TRACE_DEPTH
    -- no trace:  Set value to 0
    -- 256 depth: Set value to 8
    TRACE_DEPTH: integer := 0;
    -- TRIG_NUM
    -- no triggers:  set value to 0
    --  1 trigger:   set value to 1
    --  2 triggers:  set value to 2
    --  4 triggers:  set value to 4
    TRIG_NUM: integer := 0;
    -- set this to 1 to make nrsto an output from here
    NRSTOUT    : integer := 0;
    -- set this to 1 to enable flip-flop optimizations (default is 0)
    EN_FF_OPTS: integer := 0
    );
...
```

JTAG Interface

The JTAG connection for the debug interface to the Core8051 OCI is dependent on the device family. For ProASIC$^{\underline{PLUS}}$ devices, the designer should connect the Core8051 JTAG ports (TCK, TDI, TDO, TMS, TRSTB) to top-level ports in the design. These will pass through synthesis as ports/pins to the design. However, when imported into Actel's Designer software, these ports will be deleted automatically, as the software will detect the internal connection to User JTAG (UJTAG). UJTAG is a special JTAG connection internal to the FPGA that makes use of the same external JTAG pins as the FlashPro Lite interface.

In Antifuse FPGAs, UJTAG does not exist. The JTAG ports of Core8051 must be treated as user I/O pins and placed at pin locations appropriate to the board design.

If the OCI will not be used for downloading or debugging code in this design, then the JTAG pins should be driven with constant values (within the RTL code):

TCLK	<= 1
TMS	<= 0
TDI	<= 0
TRSTB	<= 1
TDO	=> unconnected

Limitations and Getting Access to FS2 Value-Added OCI Features

When using the OCI in-system debugger for Core8051, there are two types of breakpoints: software and hardware.

Software breakpoints are generated by exchanging the opcode at a particular line in memory with a breakpoint symbol 0xA5. When the Core8051 reaches one of these symbols, it halts operation and waits for the debugger to take control. The debugger tool (either Keil µVision2 or FS2) monitors core operation for occurrences of these software breakpoints, and when one is encountered, the normal opcode is exchanged for the breakpoint symbol. The debugger relinquishes control to the Core8051, which will resume operation with the normal opcode that replaced the breakpoint symbol. The number of software breakpoints that can be placed in an application at one time is only limited by the amount of CODE space allocated to the application.

Note: Software breakpoints cannot be used when the CODE memory space is placed in FLASH memory. This is due to the fact that FLASH memory cannot be easily rewritten to swap opcodes.

Hardware breakpoints are generated by special hardware in the OCI that monitors the address bus and halts Core8051 operation when a particular CODE address is fetched. In the Core8051 OCI block, hardware breakpoints are implemented using triggers (a set of registers that can cause a breakpoint or tracepoint by matching either an address or data bus pattern). Hardware breakpoints have the advantage of working well no matter what type of memory is used to store CODE. The disadvantage is that debugger support becomes more complicated, and only a limited number of hardware breakpoints are available (a maximum of four).

The number of breakpoints and trace memory size available is based upon the type of FS2 license. With the standard FS2 license only a single breakpoint with no trace functionality is available. For users to gain access to multiple hardware breakpoints with trace memory, an enhanced license from FS2 is required. Contact FS2 (www.fs2.com) directly to purchase an upgrade. The maximum for all users is four hardware breakpoints and 256 lines of trace memory.

3

Core10/100 Implementation

Core10/100 Overview

Core10/100 is an Ethernet Media Access Controller that connects Local Area Networks at data rates of 10 or 100 Mb/s (see Figure 3-1 and Figure 3-2 on page 26). It has a Media Independent Interface (MII) for physical connection and implements Carrier Sense Multiple Access with Collision Detection (CSMA/CD) algorithms per IEEE 802.3. Ethernet is a common standard, which is used in computer, communications, industrial, and other applications.

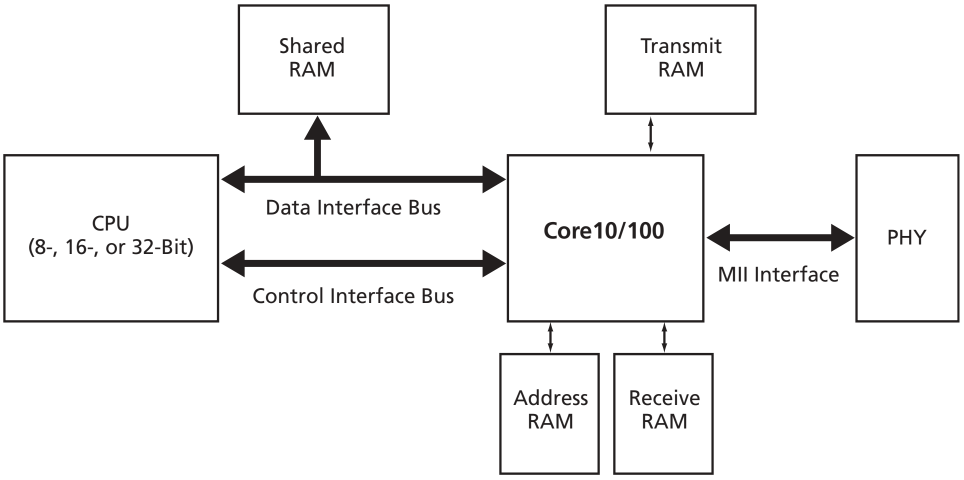


Figure 3-1. Overview of a Typical Core10/100 System

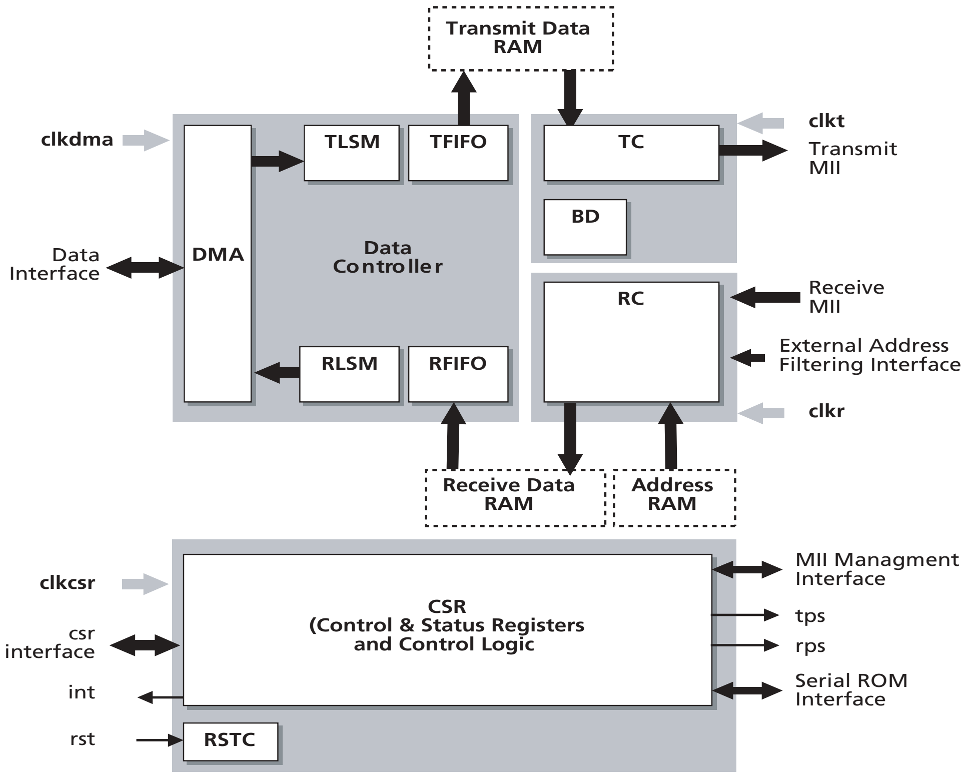


Figure 3-2. Detailed Core10/100 Block Diagram

Detailed Core10/100 information is available in *Ethernet Media Access Controller Core10/100 data sheet*.

The sections *Transmit Process* and *Receive Process* in the Actel *Core10/100 data sheet* provide an overview of the operation of Core10/100.

Ethernet PHYs

The MII interface to Core10/100 works with most Ethernet PHY chips. Due to the analog circuit requirements of an Ethernet PHY, this cannot be implemented in an Actel FPGA.

On the Platform8051 Development Board, a DP83846AVHG PHY from National Semiconductor is used. Additional outputs and jumpers for the PHY are described in Table 3-1.

Table 3-1. Additional Ethernet PHY Circuitry

LED	Color	Signal	Description
D1	Green	LED_FDPLX	Full Duplex LED Status; indicates Full-Duplex status
D2	Green	LED_COL	Collision LED Status; indicates collision activity in Half-Duplex mode
D3	Green	LED_SPEED	Speed LED Status; indicates link speed – high for 100 MB/sec, low for 10 MB/sec
D4	Green	LED_RX	Receive LED Status; indicates receive activity. LED is on for activity, off for no activity.
J9 Left	Green	LED_TX	Transmit LED Status; indicates transmit activity. LED is on for activity, off for no activity.
J9 Right	Green	LED_GCLNK	Good Link LED Status; indicates good link status for 10BaseT and 100BaseT

4

Platform8051 Design Development

Actel recommends that the user run one of the two demonstration projects described in "Platform8051 Demonstration Projects" on page 47 to achieve confidence in the tools and gain familiarity with the Core8051 and debug environment before developing a Platform8051 design. This section covers a suggested design flow (Figure 4-1) for users wishing to develop their own evaluation or product design.

The Suggested Design/Verification Flow

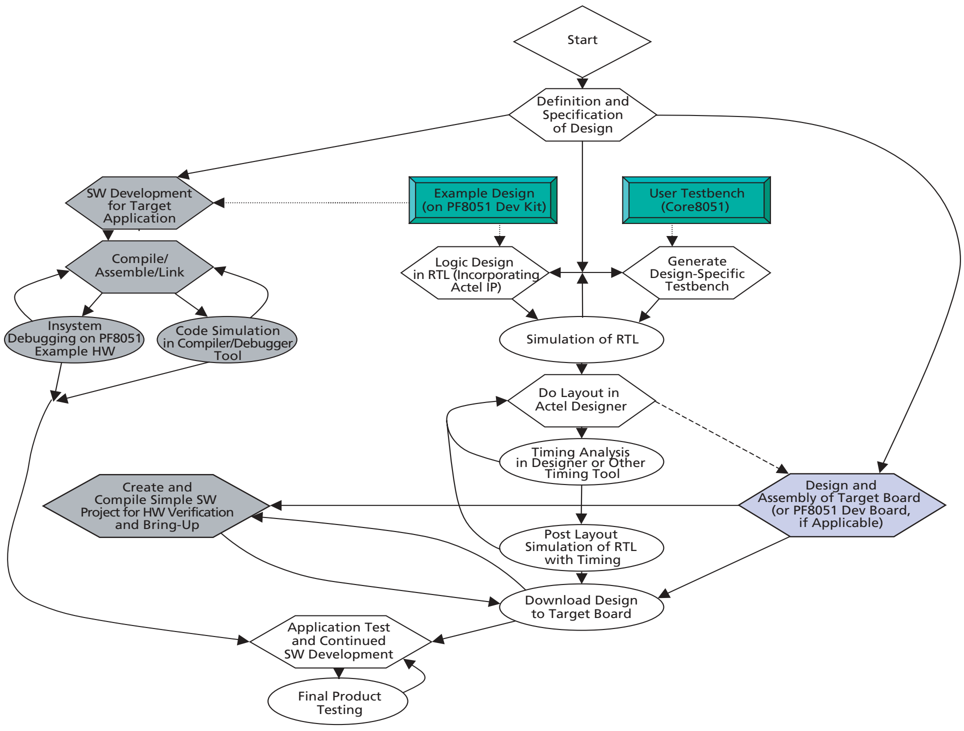

Start
Definition and Specification of Design
SW Development for Target Application
Example Design (on PF8051 Dev Kit)
User Testbench (Core8051)
Compile/ Assemble/Link
Logic Design in RTL (Incorporating Actel IP)
Generate Design-Specific Testbench
Insystem Debugging on PF8051 Example HW
Code Simulation in Compiler/Debugger Tool
Simulation of RTL
Do Layout in Actel Designer
Timing Analysis in Designer or Other Timing Tool
Create and Compile Simple SW Project for HW Verification and Bring-Up
Design and Assembly of Target Board (or PF8051 Dev Board, if Applicable)
Post Layout Simulation of RTL with Timing
Download Design to Target Board
Application Test and Continued SW Development
Final Product Testing

Figure 4-1. Design Flow

Once familiarity has been gained with the core and related tools, RTL coding of an original evaluation design can begin. Simulation can be greatly assisted by referencing the User Testbench that is delivered to Core8051 license holders. Ideally, simulation should be done both before and after layout. As a minimum, post-layout simulation including timing back-annotation should be performed. It is further recommended that the user complete a static timing analysis.

Software development can be started in parallel with hardware design. Initially software can be tested using a software simulator. Using the RTL simulation for software development is not recommended. If this is attempted, it is important that the test code be kept extremely simple. Running large amounts of code in simulation will be extremely slow. RTL simulation of a Core8051 system is best restricted to individual specific features. Full system verification is best accomplished by testing the actual hardware.

When the customized RTL is completed and downloaded to a board for the first time, it is highly recommended that a very simple software program be used to test out the hardware features before a complex application code is loaded into the Core8051 memory. System debug can then be accomplished using the OCI module and either the Keil or FS2 debugger.

A flow chart for this suggested development flow is shown in Figure 4-1 on page 29.

How to Obtain RTL or Netlists for Core8051 and Core10/100

Contact Actel's worldwide staff of sales professionals who are ready to assist you with the purchase of the IP or the evaluation kit, and help you instantiate your design on an Actel FPGA. "Recommended Additional Products" on page 5 lists the full part numbers and ordering codes for single-use netlists and RTL (-SN, -SR) and multiple-use netlists and RTL (-AN, -SN). In the numbering scheme, S represents single-use, A represents multiple-use on Actel devices, N represents netlists, and R represents RTL source code.

RTL or Netlist versions of Core8051 have the same OCI functionality. However, each netlist has fixed parameters and cannot be adjusted. Netlist customers will receive netlists with a variety of OCI parameter settings and targeting all supported Actel device families. If a specific set of parameters is not in the default netlist release, it may be requested from Actel. However, new parameters or changes to the design beyond the defined parameters will not be an option. All of the Core8051 parameters can be controlled by customers in the RTL version.

Working with Synplicity®

The PF8051 CD includes the folder */FPGA_design/src*, which contains the top-level netlist and Synplicity project files. However, these files cannot be compiled without the Core8051 and Core10/100 sub-modules, which are not included on the CD and must be purchased separately. The top-level netlist is a useful reference for how to create memories, both on- and off-chip, as well as integrating other parts of the Platform8051 system.

For more information about working with Synplicity, see the *Core8051 User's Guide* (located on the CD). Also, more Synplicity information specific to the demonstration designs is available in "Basic Platform8051 Demonstration Project" on page 51.

Design Layout

As with Synthesis, layout can only be performed by Actel customers who have purchased either netlist or RTL versions of Core8051 and/or Platform8051. The source code (netlist or RTL) is not included in the Platform8051 Development Kit.

After synthesizing the design, use Actel's Designer software to place-and-route the FPGA. During the layout process, Designer will compile, place, route, and generate device-specific programming files for physical implementation of the design into an Actel FPGA. Refer to the *Designer User's Guide* for details on using the Designer software.

Tcl and GCF Files

Along with the EDIF netlist that is generated by Synplicity, Actel's Designer software can also take Tcl scripts and GCF constraints files as inputs. To facilitate layout and provide a step-by-step example, a Tcl script (**.tcl*) is included on the PF8051 CD for each of the example projects. See "Basic Platform8051 Demonstration Project" on page 51 for example project details.

To run a Tcl script in Designer, go to the **File** menu and select **Execute Script…** This will open the *Execute Script* dialog box shown in Figure 4-2. Select **platform.tcl** and click **Run**. No arguments are typically necessary.

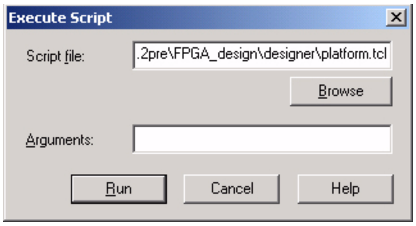


Figure 4-2. Execute Script... Dialog in Designer

The Tcl script will re-import the EDIF netlist, apply the **.gcf* constraints file, and run Layout. The **.gcf* constraints file will set the timing constraints and place the I/Os to the appropriate fixed pin locations to match the board design.

Layout for Core8051 Netlist and RTL Customers

The netlist and RTL releases of Core8051 include the *layout* directory, which contains example ADB (Actel Database) files for fully placed-and-routed versions of the macro in each of the following supported Actel device families:

- ProASIC$^{\text{PLUS}}$

- Axcelerator
- SX-A
- RTSX-S

The example layout files included with the netlist and RTL releases of Core8051 are listed in Table 4-1.

Table 4-1. Core8051 Example Layout Files

Core8051 Layout
layout/core8051_oci1_trace8_trig1_apa.adb
layout/core8051_oci1_trace8_trig1_ax.adb
layout/core8051_oci1_trace8_trig1_sxa.adb
layout/core8051_oci1_trace8_trig1_sxs.adb

The *netlists* directory is also included with the netlist and RTL releases of Core8051, and contains the corresponding netlist file of each ADB file in the *layout* directory. For example, the file *netlists/core8051_oci1_trace8_trig1_withio_ax.v* corresponds to the layout database file *layout/core8051_oci1_trace8_trig1_ax.adb*.

Note: The netlist files that contain "_withio_" in their names in the *netlists* directory are for use as stand-alone versions of the Core8051 macro. They are not for use within a design since they already have I/O pads inserted. To integrate the netlist version of the Core8051 macro into a design, use the netlist files that contain "_withoutio_" in their names.

Programming the Design into the FPGA

The Platform8051 Development Kit is centered around an Actel APA600 device. This Flash-based FPGA can be reprogrammed to contain any design the user wishes to develop using standard Actel tools (an evaluation copy of the Actel Libero FPGA design tool is provided with the Platform8051 Development Kit). In addition, a FlashPro Lite download cable must be purchased separately to reprogram the FPGA or use the 8051 development/debug tools.

Although the FlashPro Lite is used both to program the FPGA and to load application code compiled for the Core8051, the two processes should not be confused.

To program the ProASIC^{PLUS} FPGA on the Platform8051 Development Board:

1. **Connect a FlashPro Lite to the parallel port of your PC.**

2. Connect the FlashPro Lite ribbon cable to header J3 on the Platform8051 Development Board.
3. Start the Actel FlashPro programming software on the PC.
4. Select *Connect…* from the *File* menu.
5. Ensure that the *Configuration* value is set to ProASIC$^{\text{PLUS}}$, and click *Connect* (Figure 4-3).

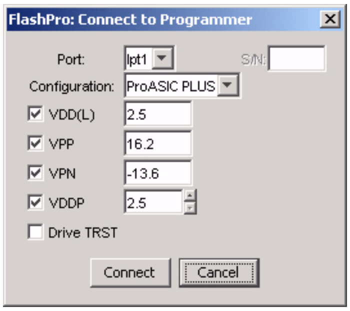
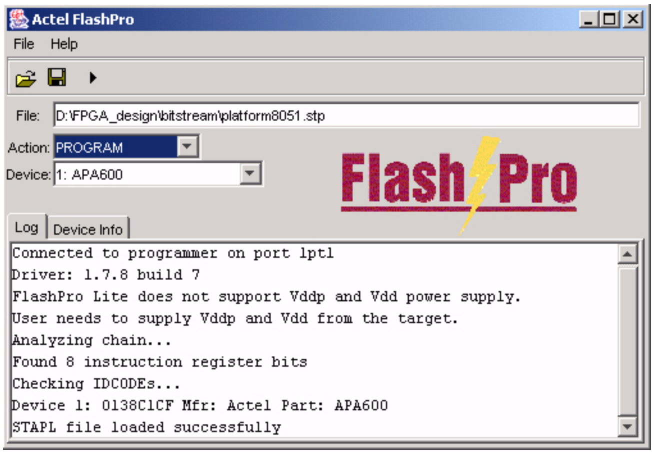


Figure 4-3. FlashPro “Connect...” Dialog

6. Select *Analyze Chain* from the *File* menu. Verify that an APA600 device is identified.
7. Select *Open STAPL File…* from the *File* menu. Find the **.stp* file for the design you intend to program and click *Open*.
8. In the main FlashPro window, set the *Action* field to *Program* (Figure 4-4).



```
Connected to programmer on port lpt1
Driver: 1.7.8 build 7
FlashPro Lite does not support Vddp and Vdd power supply.
User needs to supply Vddp and Vdd from the target.
Analyzing chain...
Found 8 instruction register bits
Checking IDCODEs...
Device 1: 0138C1CF Mfr: Actel Part: APA600
STAPL file loaded successfully
```

Figure 4-4. FlashPro, Connected, Chain Established, Ready to Program

9. **Click on the *Execute* button ▶ to initiate programming.**

Available Simulation Testbenches

Testbenches are an essential part of the design process. The deliverables of Core8051 include two comprehensive testbenches:

- One originally designed for verification of the 8051 core. This is naturally the more comprehensive and provides fault coverage near 100%.
- The second is intended for use by the designer. This testbench is easier to modify into a form that will test a typical system implementation. It exercises more of the system submodules that would be typically created outside the Core8051 and is simple and easy to understand.

The Core8051 testbenches are constructed to execute a series of 8051 instructions and verify the resulting operation of the core. However, while possible, modifying the application code in the simulation is beyond the scope of this document.

Core10/100 has its own separate testbenches. These are not integrated with the Core8051 testbenches.

Note: There is one difference between the simulation packages delivered with the netlist and the RTL versions of Core8051. With the netlist version, the more comprehensive verification testbench is pre-compiled and cannot be modified (no source code is provided). RTL customers are provided the source code to the verification testbench.

Once hardware simulation and verification has been completed, the task of verifying the application code can begin. For application code simulation, a C compiler with simulation capabilities, for example Keil µVision2, is highly recommended.

See the *Core8051 User's Guide* (located on the CD) for detailed information about hardware simulation files.

5

Software Development Tools Setup

Core8051 will run the same code and will use the same compilers and assemblers as most other 8051-compatible microcontrollers. Typically, the user will write code in C and then use a compiler to create a Hex or OMF file. Either of these files can be downloaded to the CODE memory space to be run by Core8051.

Alternatively, the user may choose to write code in assembly language. In this case, the input to the compiler/assembler will be a set of ASM51 assembly language commands. The legal opcodes and their functions are listed in the *Core8051 Datasheet*. Further documentation on 8051 opcodes and the syntax of assembly language can be found in the compiler/assembler manuals.

Actel recommends the Keil (www.keil.com/dd/cl/actel/8051.htm) or SDCC (http://sdcc.sourceforge.net) compilers and assembler for Core8051. While others may work, only the Keil and SDCC compilers and assemblers have been tested and verified by Actel.

For in-system debugging of application code, either Keil or FS2 (www.fs2.com) tools may be used. FS2's ISA-Actel51 Debugger is included with the purchase of Core8051 or the Platform8051 Development Kit. An evaluation version of Keil µVision2 is included in the development kit (or can be downloaded directly from Keil), however, this version has limited functionality. Limitations of the Keil evaluation version are published at www.keil.com/demo/limits.htm. The Keil µVision2 simulator is instruction- and cycle-accurate with Core8051 (not all 8051s are supported with cycle-accurate simulation).

Note: Actel has two solution partners who distribute Keil software at a substantial discount: FS2 and Capital Automation. Contact your Actel sales representative for more information.

Installing and Configuring the C Compiler and Debugging Tools

Keil µVision2 Installation

Follow the Keil installation instructions for the µVision2 C51 compiler. The Keil compiler should be installed prior to installing the FS2 ISA-ACTEL51 software. The µVision2 compiler may be used independently; however, if the µVision2 debugger is desired, it requires the FS2 ISA-ACTEL51 parallel port drivers and software translation layer be installed. For assistance with installation or general use of Keil tools, contact Keil at www.keil.com or call (800) 348-8051.

SDCC Installation

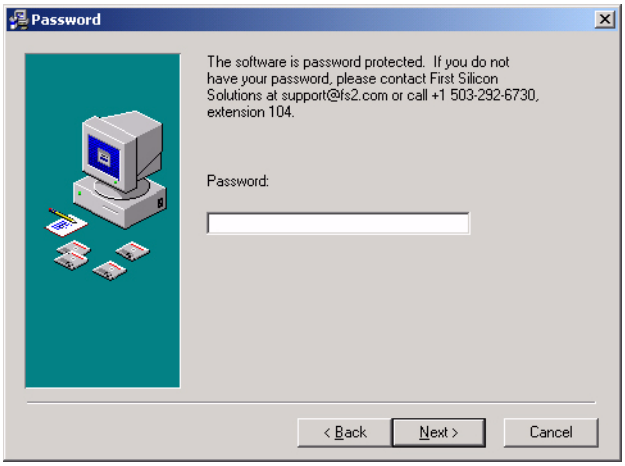
The SDCC C compiler for 8051 is available at http://sdcc.sourceforge.net. Directions and product support are available from SDCC. There are no special requirements or features for using SDCC with Actel Core8051 or Platform8051.

FS2 ISA-Actel51 Installation

Again, if FS2 tools are to be used in conjunction with the Keil µVision2 compiler/debugger, it is important that µVision2 be installed first.

To install ISA-Actel51:

1. **Run *Setup.exe* from the FS2 ISA-ACTEL51 CD**
2. **When prompted, enter one of the two setup passwords printed on the CD case** (Figure 5-1). Use the *With Keil* password if the Keil µVision2 has previously been installed, or the *Without Keil* password if the Keil software is not being used.



Password

The software is password protected. If you do not have your password, please contact First Silicon Solutions at support@fs2.com or call +1 503-292-6730, extension 104.

Password:

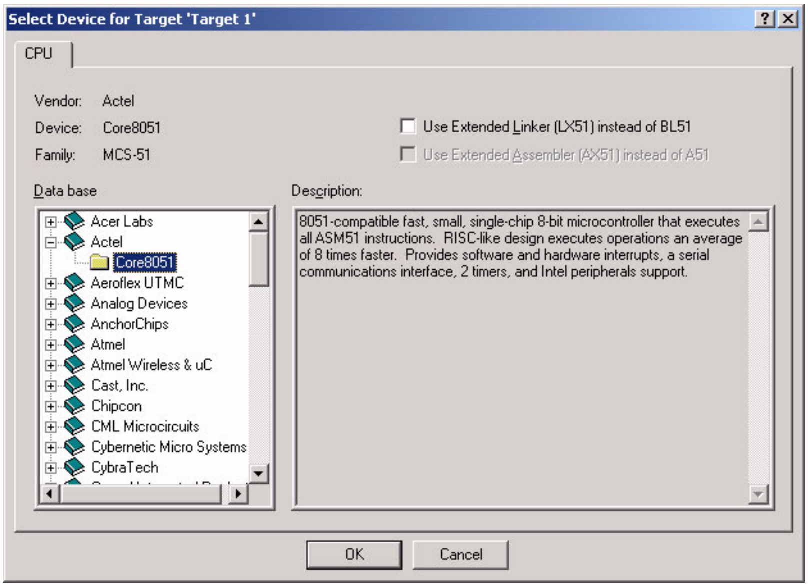
Figure 5-1. FS2 Installation Password Dialog

3. **Click *Next* to complete the installation process.**

Note: For assistance in installing the FS2 tools, contact support@fs2.com, or call (503) 292-6730.

Setting Up a Project in Keil µVision2

When a new project is created in µVision2, the first step is to select a target device (Figure 5-2). Early versions of µVision2 do not have a selection for Actel Core8051. While targeting a similar 8051 may work, it is highly recommended that µVision2 version 7.07 or later be used to target Core8051 specifically.



Vendor: Actel
Device: Core8051
Family: MCS-51

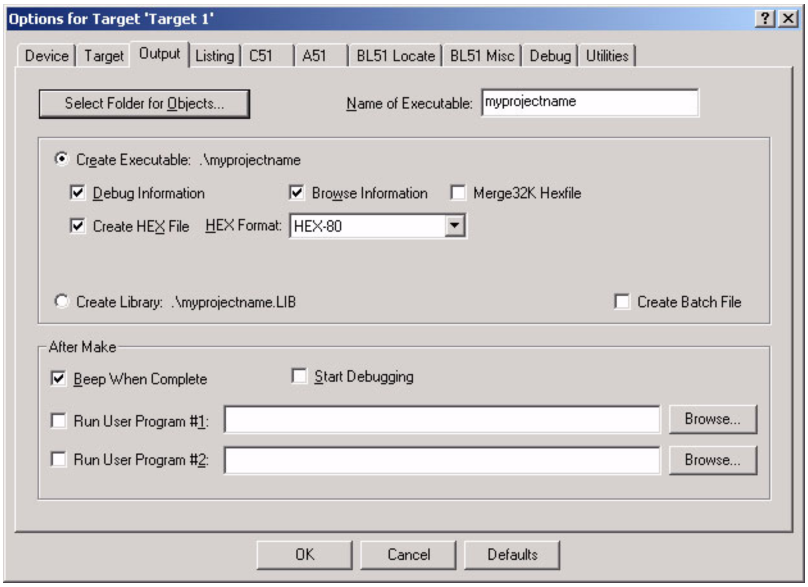
Use Extended Linker (LX51) instead of BL51
Use Extended Assembler (AX51) instead of A51

8051-compatible fast, small, single-chip 8-bit microcontroller that executes all ASM51 instructions. RISC-like design executes operations an average of 8 times faster. Provides software and hardware interrupts, a serial communications interface, 2 timers, and Intel peripherals support.

Figure 5-2. Select Target Device, Creating New Project

After selecting the target device, the *Options for Target…*, settings under the *Project* menu must be modified.

1. **On the *Output* tab, *Create HEX File* must be selected with the *Hex Format* set to *HEX-80*.** If the name for the output files (either Hex or OMF) is different from the project name, it should be typed into the *Name of Executable* field. The *Debug Information*, *Browse Information*, and *Create Executable* check boxes should be selected. (See Figure 5-3 on page 38.)



Options for Target 'Target 1'

Device | Target | Output | Listing | C51 | A51 | BL51 Locate | BL51 Misc | Debug | Utilities

Select Folder for Objects...   Name of Executable: myprojectname

● Create Executable: .\myprojectname

[✓] Debug Information   [✓] Browse Information   [ ] Merge32K Hexfile

[✓] Create HEX File   HEX Format: HEX-80

○ Create Library: .\myprojectname.LIB   [ ] Create Batch File

After Make

[✓] Beep When Complete   [ ] Start Debugging

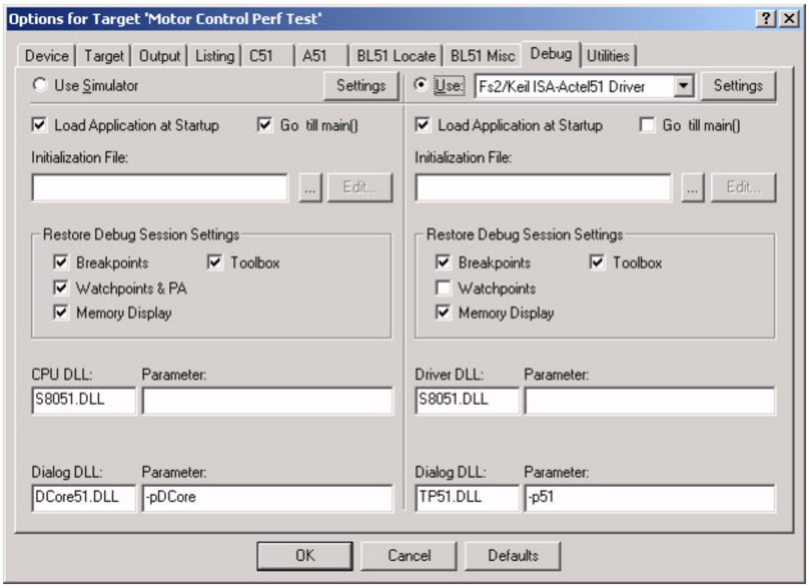
[ ] Run User Program #1:   Browse...

[ ] Run User Program #2:   Browse...

OK | Cancel | Defaults

Figure 5-3. Keil µVision2 Options for Target, Output Tab

2. **On the *Debug* tab, select the proper driver for communicating with the target board.** Select *Fs2/ Keil ISA-Actel51 Driver* from the drop-down list and ensure that the *Use* button next to it is selected (otherwise the debugger will open the simulator instead of connecting to the target board). *Load Application at Startup* should also be selected. (See Figure 5-4 on page 39.) If the FS2 driver is not present in the drop-down list, exit the Keil software and reinstall the FS2 tools.


Options for Target 'Motor Control Perf Test'

Use Simulator | Settings

[✓ Load Application at Startup] [✓ Go till main()]

Initialization File:

Restore Debug Session Settings

[✓ Breakpoints] [✓ Toolbox]
[✓ Watchpoints & PA]
[✓ Memory Display]

CPU DLL: S8051.DLL Parameter:

Dialog DLL: DCore51.DLL Parameter: -pDCore

[✓ Use:] Fs2/Keil ISA-Actel51 Driver | Settings

[✓ Load Application at Startup] Go till main()

Initialization File:

Restore Debug Session Settings

[✓ Breakpoints] [✓ Toolbox]
Watchpoints
[✓ Memory Display]

Driver DLL: S8051.DLL Parameter:

Dialog DLL: TP51.DLL Parameter: -p51

Figure 5-4. Keil μVision2 Options for Target, Debug Tab

Real Time Operating Systems (RTOS) and the Core8051

There are several RTOSs available for 8051 processors. These can be useful in some cases; however, it should be remembered that the 8051 is a small and relatively limited microcontroller. Limitations in its ability to manage memory, large stacks, and overall data throughput can make an RTOS a substantial limiter in system performance. The vast majority of Core8051 applications can be implemented without an RTOS.

Some common RTOSs are listed at www.keil.com/rtos/default.htm.

Special Non-Standard C for 8051

The complexity of the 8051 memory map, and the lack of an operating system place unusual requirements on C compilers written for the 8051. Some ANSI rules must be violated in order to accommodate this processor. Detailed information about these syntax variances should be obtained from the compiler vendor.

Some frequently used C syntax is defined in the following sections.

Placing Variables or Constants in Different Memory Spaces

By default, any globally defined variable or constant will be placed in the DATA memory space. However, it is possible to define it in either DATA, XDATA, or CODE memory spaces as well.

Example:

```
int i1; /* will be placed in default memory space (DATA)*/
data int i2;  /* will be placed in DATA explicitly */
xdata int i3;  /* will be placed in XDATA explicitly */
code const int i4=0;  /* will be placed in CODE explicitly (READ ONLY) */
```

Placing Variables at Specific Addresses

Along with selecting the memory space for variables, it is often required that a variable be placed at a specific address (to correspond to a memory-mapped register). This is typically done with the **_at_** syntax.

Example:

```
data char LCD_DATA_REG _at_ 0xF101;
    /*defines LCD_DATA_REG at xdata address 0xF101 */
```

This can also be accomplished by casting pointers or setting linker commands, but the **_at_** syntax is often the most convenient.

Stack Placement

In a normal C application, all local variables are stored either in registers or in the stack. This is not always the case when working with the 8051. The 8051 normally keeps its stack near the top of DATA space. DATA space is only 128 bytes total (for the stack, registers, and global variables). In some cases, DATA space can be extended to 256 bytes, but is still not large enough to fit the large stacks that most non-8051 C compilers tend to require.

In an 8051 application, the stack is kept as small as possible by keeping many of the local variables in special data structures in DATA space. Complex graph-coloring algorithms are used to determine exactly when the variables must be kept in scope and when they can be overwritten. This can be much more efficient in terms of performance and stack space than typical C compilers.

Another option for designers that require large amounts of stack space is moving the stack into XDATA. This makes a great deal of space available, but limits performance. DATA stacks can be accessed in one instruction/cycle. XDATA may take up to three or four instructions/cycles to access.

Recursive or Re-entrant Code

Because of the way the stack is handled, recursive or re-entrant code is not recommended in Core8051 applications. Many 8051 C compilers (including Keil) will detect re-entrant functions and create an error message. If there is no alternative, the keyword *reentrant* must be added to the function definition to force all of its variables to be stored in the stack. In this case, careful analysis should be done to guarantee the stack space will not overflow.

Example:

```
void my_recursive_function () reentrant {
        my_recursive_function();
        …
        }
```

Pointer types

Due to the complexity of the 8051 memory space, compilers must manage unusual pointer types. A pointer must not only contain an address, but also a memory space.

Example:

```
int *i;  /* GENERIC POINTER: the pointer "i" will be 3 bytes in size, the
first byte will be a representation of the memory space (XDATA, CODE, DATA),
the next to will be the 16-bit address */

xdata int *i2;  /* SPACE SPECIFIC POINTER: the pointer "i2" will be 2 bytes
in size, just a 16-bit address, the type is always XDATA * for this pointer
*/
```

6 Downloading Software

After the application code has been compiled, downloading to the target device is accomplished with either FS2 ISA-Actel51 or Keil µVision2. Either of these tools can also be used to do in-system debugging with single-step, breakpoint, register, and data control. This section gives an overview of the use of these tools as well as some specific information related to the Platform8051 Development Board.

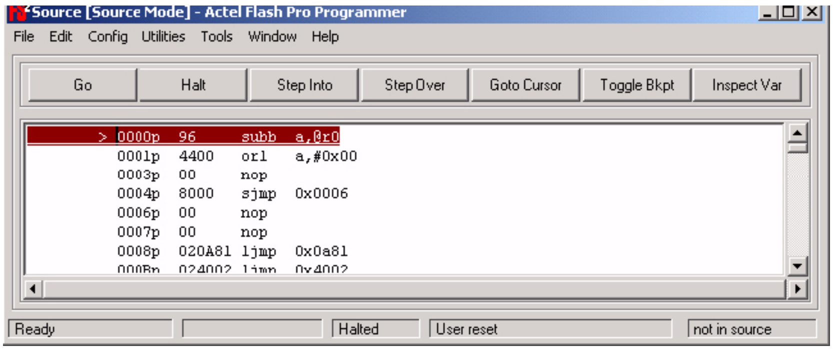
Downloading and Debugging from SRAM with FS2

Before opening the debugger, the Platform8051 Development Board should be powered on and connected to the PC through a FlashPro Lite. In addition, the FPGA must be programmed with a Core8051 with OCI design prior to attempting to download code (see "Programming the Design into the FPGA" on page 32). Before downloading code into the design, the DIP switches must be verified to be in the *debug* memory map mode: Switch[1:8] = 00000011**.

Note: **These switches are wired to inputs of the FPGA so that *open* will correspond to a logic 1 (4.6 k pull-up to 3.3 V) and *closed* will correspond to a logic 0 (GND).

To download the code to the target device:

1. **Open FS2 ISA-Actel51 Debugger.** This application is typically found under the *FS2* folder in the Windows *Start* menu. The first window that should be visible is shown in Figure 6-1. If this window does not open, but the warning "Parallel port device does not support IEEE-1284 negotiation protocol" is displayed, check the connections and try again.


Source [Source Mode] - Actel Flash Pro Programmer
File Edit Config Utilities Tools Window Help
Go
Halt
Step Into
Step Over
Goto Cursor
Toggle Bkpt
Inspect Var
Ready
Halted
User reset
not in source

```
> 0000p  96     subb  a,@r0
  0001p  4400   orl   a,#0x00
  0003p  00     nop
  0004p  8000   sjmp  0x0006
  0006p  00     nop
  0007p  00     nop
  0008p  020A81 ljmp  0x0a81
  000Bp  024002 ljmp  0x4002
```

Figure 6-1. ISA-Actel51 Debugger Startup Window

2. **After establishing a connection to Core8051, the application code should be downloaded.** Code is outputted by the compiler as either a *Hex*, or *OMF* file that contains the raw binary as well as symbols necessary for conducting source-level debugging. The compiler output can be

downloaded into Core8051 CODE memory space by selecting the *Load Hex…* or *Load OMF…* commands under the *Tools* menu. The Hex file will have an extension of *.ihx* or *.hex*, depending on what compiler you have used. The OMF file typically does not have an extension. When using the Keil compiler, the OMF file will be the project name with no extension. Browse to find this file and click *OK* (Figure 6-2).

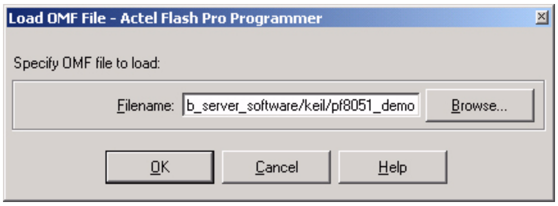

Load OMF File - Actel Flash Pro Programmer
Specify OMF file to load:
Filename:
b_server_software/keil/pf8051_demo
Browse...
OK
Cancel
Help

Figure 6-2. ISA-Actel51 Debugger Load OMF Dialog

This will load the CODE and reset the Core8051. The debugger is now ready for normal operations such as single stepping, inserting breakpoints, clearing breakpoints, and watching data values.

For details on debugging with ISA-Actel51 Debugger, see the ISA-Actel51 help files or the *ISA-Actel51 Getting Started Manual*, both of which are installed along with the ISA-Actel51software. Additional assistance can be obtained by contacting FS2 directly (www.fs2.com).

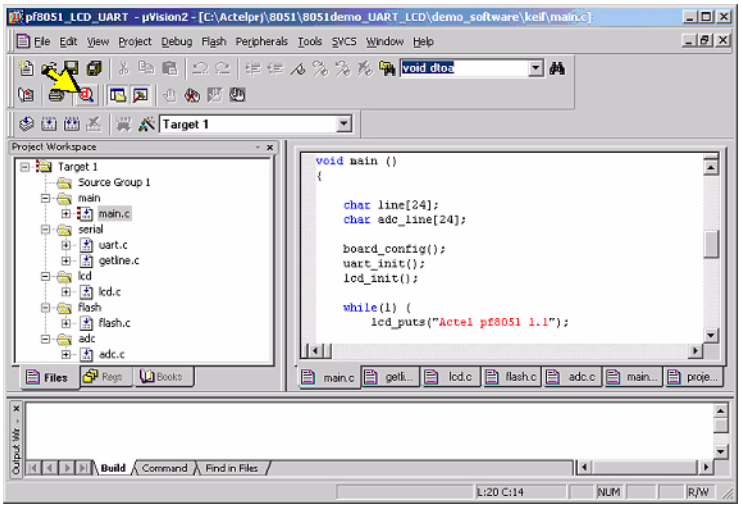
Downloading and Debugging from SRAM with Keil µVision2

Before opening the debugger, the Platform8051 Development Board should be powered on and connected to the PC through a FlashPro Lite. In addition, the FPGA must be programmed with a Core8051 with OCI design prior to attempting to download the code. (see "Programming the Design into the FPGA" on page 32). Before downloading code into the design, the DIP switches must be verified to be in the *debug* memory map mode: Switch[1:8] = 00000011**.

Note: **These switches are wired to inputs of the FPGA so that *open* will correspond to a logic 1 (4.6 k pull-up to 3.3 V) and *closed* will correspond to a logic 0 (GND).

The Keil µVision2 project should be set up as described in "Setting Up a Project in Keil µVision2" on page 37. Also, the debug options must be set to *use FS2/Keil ISA-Actel51 Driver.*

After successfully compiling the code and generating a Hex file, click the *Debug* icon (Figure 6-3) to start the debugger (or select *Start/Stop Debug Session* from the *Debug* menu).


pf8051_LCD_UART - µVision2 - [C:\Actelprj\8051\8051demo_UART_LCD\demo_software\keil\main.c]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
void dtoa
Target 1
Project Workspace
Target 1
Source Group 1
main
main.c
serial
uart.c
getline.c
lcd
lcd.c
flash
flash.c
adc
adc.c
Files
Regs
Books
main.c getli... lcd.c flash.c adc.c main... proje...
Output Win
Build
Command
Find in Files
L:20 C:14
NUM
R/W

```
void main ()
{

    char line[24];
    char adc_line[24];

    board_config();
    uart_init();
    lcd_init();

    while(1) {
        lcd_puts("Actel pf8051 1.1");
```

Figure 6-3. Keil µVision2

Starting the debug session will automatically download the code to the device, reset the Core8051, and start an interactive debug window. The debugger is now ready for normal operations such as single stepping, inserting breakpoints, clearing breakpoints, and watching data values. For details on debugging with µVision2, use the *Help* menu, or contact Keil directly for support (www.keil.com).

Copying Code from SRAM to Flash in Example Designs

The Keil and FS2 debuggers do not support Flash programming for code download. If the application requires that the code be stored in Flash memory, the application design must create Flash programming routines.

In the Platform8051 example designs, a Flash programming algorithm has been included in the code. To copy the code into Flash using this example, first download to SRAM (in *Debug* memory mode). Then set DIP switch [1] to *open*. This input is connected to the Port2 inputs and monitored by the Core8051 during application startup. Next press the Reset button once (SW2). This will restart the code (detecting the change to Port2) and start the Flash programming algorithm. Wait at least 10 seconds for the Flash programming to complete.

Once the Flash is programmed, the DIP switches may be set to 00000000 (all *closed*). When reset is pressed again, or the board power-cycled, it will restart in *Shadow* mode (see section "Memory Map Architecture Implementation in Basic Platform8051 Example Design" on page 52 for details).

To copy the code into Flash using this example:

1. **Set DIP switches [1:8] to 00000011 and download the application code to SRAM.**
2. **Switch DIP switch [1] to 1.** This input is connected to Port2 inputs and monitored by the Core8051 during application startup.
3. **Press Reset (SW2) once.** This will restart the Core8051 (detecting the change to the DIP switch) and initiate the Flash programming sub-routine.
4. **Wait at least 10 seconds for the Flash Programming to complete.**
5. **Set DIP switches [1:8] to 00000000 and power-cycle the board.**

Debugging from Flash

The FS2 and Keil debuggers will debug code from Flash memory in the same manner as with internal memory. The exceptions are that code updates cannot be accomplished as easily (as described in "Copying Code from SRAM to Flash in Example Designs" on page 45), and only one break point is available. The limitation to the number of available breakpoints is because software breakpoints cannot be used in Flash memory and only a limited number of OCI triggers are available (see "Core8051 OCI Debugger Interface Implementation" on page 19 for details).

7
Platform8051 Demonstration Projects

The Platform8051 CD contains two demonstration projects for use with the Platform8051 board and Core8051:

- Basic Terminal Server
- Web Server

The terminal server project connects via RS-232 to a PC COM port (running HyperTerminal or similar text-based terminal window). The server will run the PC terminal, display terminal data on the Platform8051 LCD screen, and read back data from the on-board ADC. This project is used to illustrate the interaction between Actel Libero design tools, Actel Platform/Core8051, and third-party tools for microcontroller application compilation and debugging.

The web server project connects to a PC via Ethernet and provides voltage, power, and temp data from the board via a simple web page. This project is designed to demonstrate the various features and capabilities of the Platform8051.

Common Design Aspects – LCD and ADC Implementation

The Platform8051 Development Board has an LCD display. This display is controlled by a simple register-mapped interface design inside the APA600 FPGA. The commands for this interface are defined in this section. The LCD drivers are not contained in the FPGA, just the XDATA register interface.

An analog-to-digital converter (ADC) is also part of the Platform8051 Development Board. Similarly to the LCD implementation, the ADC is not part of the APA600 FPGA. Instead the ADC is on the board as a separate component that is controlled through a register-mapped interface design inside the FPGA.

LCD Command and Data Register

Core8051 will write LCD instructions to the LCD Command Register (0xF100) and LCD data to the LCD Data Register (0xF101). Table 7-1 on page 48 lists all LCD commands. Table 7-2 on page 49 lists the LCD module instruction bit descriptions. Data written to the LCD data register will be transferred to the LCD's Data RAM at the address pointed to by the Data Address counter in the LCD module.

Reading from the LCD command address returns the LCD Status. Bit 7 shows the Busy-Flag (BF) indicating internal operation. If this bit is 1, the LCD's internal operation is in progress and it is Busy. During this time the LCD cannot accept any instruction or data.

Table 7-1. LCD Module Instruction Set

D7	D6	D5	D4	D3	D2	D1	D0	Instruction	Description
0	0	0	0	0	0	0	1	Clear Display	Clear display and returns cursor to the home position
0	0	0	0	0	0	1	*	Cursor Home	Returns cursor to home position
0	0	0	0	0	1	I/D	S	Entry Mode Set	Set Cursor Move direction (I/D), specifies to shift the display (S)
0	0	0	0	1	D	C	B	Display On/Off	Sets On/Off of Display (D), Cursor (C), and blink of cursor position (B)
0	0	0	1	S/C	R/L	*	*	Cursor/Display shift	Set cursor-move or display-shift (S/C)
0	0	1	DL	N	F	*	*	Function Set	Sets interface data length (DL), number of display line (N), and character font (F)
0	1	CGRAM Address						Set CGRAM address	Sets the CGRAM address
1	DDRAM Address							Set DDRAM address	Sets the DDRAM address

* Value can be either 0 or 1.

Table 7-2. LCD Module Instruction Bit Description

Bit Name	Settings	
I/D	0 = Decrement cursor position	1 = Increment cursor position
S	0 = No display shift	1 = Display shift
D	0 = Display off	1 = Display on
C	0 = Cursor off	1 = Cursor on
B	0 = Cursor blink off	1 = Cursor blink on
S/C	0 = Move cursor	1 = Shift display
R/L	0 = Shift left	1 = Shift right
DL	0 = 4-bit interface	1 = 8-bit interface
N	0 = 1 Line	1 = 2 Lines
F	0 = 5x7 dots	1 = 5x10 dots

ADCIN and ADCOUT registers

Core8051 can read and write the ADCIN register at 0xF200. Table 7-3 describes the meaning of each bit in this register.

Table 7-3. ADCIN Register Field Definition

Mnemonic	Bits	Default	R/W	Description
CHNSEL	7:5	3'h0	R/W	Channel Number. ADC channel number
UNIBIP	4	1'h1	R/W	UNI BIP bit. Unipolar, Bipolar bit
SGLDIF	3	1'h1	R/W	SGL BIP. Single-Ended Differential
PD	2:1	2'h2	R/W	Power Down Mode. Power-down mode
START	0	1'h0	R/W	Start Bit. Start ADC read. This bit will reset after ADC read is completed.

The ADCOUT register returns 10-bit values of ADC output. ADCOUT1 (0xF201 in the example designs) returns the eight MSB bits of ADC out, and ADCOUT0 (0xF202 in the example designs) returns the two LSB bits (see Table 7-4).

Table 7-4. ADCOUT Register Contents

ADCOUT1								ADCOUT0							
9	8	7	6	5	4	3	2	1	0	-	-	-	-	-	-

Basic Platform8051 Demonstration Project

This project illustrates the interaction between Actel Libero design tools, Actel Platform/Core8051, and third-party tools for microcontroller application compilation and debugging. The design is downloaded onto a Platform8051 development board where it can be run from on-board SRAM or Flash.

Once the example application has been downloaded, it will act as a simple terminal server via RS-232 to a PC COM port running HyperTerminal or a similar text-based terminal window (Figure 7-1). The server will run the PC terminal, display terminal data on the Platform8051 LCD screen, and read back data from the Platform8051 ADC.

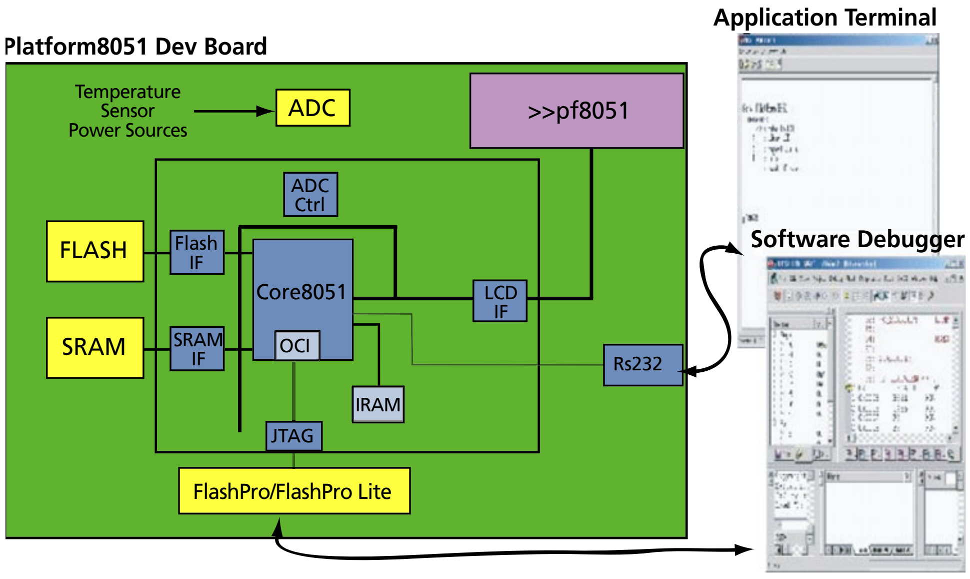


Figure 7-1. Overview for Basic Core8051 Demonstration Project

Memory Map Details

Memory Map Architecture Implementation in Basic Platform8051 Example Design

Core8051's memory map is implemented in four different operating modes to support all possible configurations. These modes will be selected by memmode[1:0] signal in the FPGA design. The following sections describe the detailed operation of the memory map modes.

Memory Map Mode Selection

Memory mode will be generated by using Core8051's Port3 [7:5] outputs and external User Switch [8:7] settings. When the Core8051 drives Port3 [7] output to HIGH (default/reset state), memmode will be controlled by the external switch settings. When Port3 [7] is LOW, memmode will be generated by Port3 [6:5], which allows the user to switch between all possible memory map modes by software control. Table 7-6 on page 53 shows the memory map modes.

There are four possible memory modes in the demonstration designs:

- Shadow - Used when running the application code from Flash
- Normal - Application code is in SRAM, peripherals are all mapped in XDATA space
- Flash Program - Used to copy application code from SRAM to Flash (while CODE space is mapped and running from SRAM)
- Debug - Both XDATA and CODE are overlaid in SRAM

Memory Resource Mapping

Table 7-5 on page 53 describes the mapping of the memory resources modes of operation.

Resources available on-board for Platform8051 design are:

- 64 kbytes of Flash
- 64 kbytes of LRAM (lower part of SRAM)
- 64 kbytes of URAM (upper part of SRAM)

Table 7-5. Memory Map Mode Definitions

Mode	8051 Program Memory (64 Kbytes)	8051 Data Memory (64 Kbytes)
Shadow	0000:FFFF – Flash	0000:FFFF – LRAM
Normal	0000:FFFF – LRAM	0000:CFFF – URAM (52 kbytes) D000:EFFF – TX, RX memory for Core10/100 (8 kbytes) F000:FFFF – ADC/LCD registers (4 kbytes)
Flash Program	0000:FFFF – LRAM	0000:FFFF – Flash
Debug	0000:FFFF – LRAM	0000:FFFF – LRAM

Table 7-6. Memory Map Modes

Port 3[7]	Port 3[6:5]	**User SW[8:7]	Memmode[1:0]	Memory Map Mode
1	XX	00	00	Shadow
1	XX	01	01	Normal
1	XX	10	10	Flash Program
1	XX	11	11	Debug
0	00	XX	00	Shadow
0	01	XX	01	Normal
0	10	XX	10	Flash Program
0	11	XX	11	Debug

**These switches are wired to inputs of the FPGA so that *open* will correspond to a logic 1 (4.6 k pull-up to 3.3 V) and *closed* will correspond to a logic 0 (GND).

Memory Map for Normal Operation Mode

Core8051's Program Memory will be mapped to an external SRAM's lower 64 kbytes of memory space during normal operation mode. Data memory is mapped to various memories and registers as shown in Table 7-7.

Table 7-7. Memory Map for Normal Operation

Data Address	Memory	R/W	Description
0000:CFFF (52 kbytes)	URAM	R/W	Upper portion on external SRAM
F100	LCD Command	R/W	LCD Command register
F101	LCD Data	R/W	LCD Data register
F102:F1FF	Reserved	–	Reserved for future use
F200	ADCIN (1 byte)	R/W	ADC In register
F201:F202	ADCOUT(2 bytes)	R	ADC Out register
F203:FFFF	Reserved	–	Reserved for future use

Address range 0x0000 to 0xCFFF accesses the external SRAM's upper portion of 128 kbytes.

Clock Configuration in the Platform8051 Basic Example Design

The Platform8051 Web Server Example Design is implemented using a unified clock (as shown in Figure 2-1 on page 18). This makes IDLE and STOP modes not available in this design. Modules in the example designs, which are peripheral to the Core8051, are driven by the main system clock. Minimizing the number or clocks enables easy timing analysis and increased performance.

The clock input to the APA600 in the Platform8051 example projects is divided by two inside the chip before being used. The design is running at 8 MHz, not the 16 MHz being supplied by the crystal oscillator on the board.

Design Files

The design files for this project can be found on the PF8051 CD. This includes a software C code project and source files for hardware design (minus the Core8051 and Core10/100, which must be purchased separately). Constraints files and scripts for layout and synthesis are also provided.

Below is the Synplicity project file (*platform_act.prj*). This defines all the files and operations necessary to synthesize this project.

```
#-- Synplicity, Inc.
#-- Version 7.3
#-- Project file
C:\Actelprj\8051\8051demo_UART_LCD\FPGA_design\synplify\platform_act.prj
#-- Written on Wed Mar 24 15:08:23 2004


#add_file options
add_file -vhdl -lib work "$LIB/proasic/proasicplus.vhd"
add_file -vhdl -lib work "../src/top/user_ram.vhd"
add_file -vhdl -lib work "../src/top/adc_def_pkg.vhd"
add_file -vhdl -lib work "../src/top/adc_regbank.vhd"
add_file -vhdl -lib work "../src/top/adcctrl.vhd"
add_file -vhdl -lib work "../src/top/addrdatactrl.vhd"
add_file -vhdl -lib work "../src/top/clkgen.vhd"
add_file -vhdl -lib work "../src/top/phyclk.vhd"
add_file -vhdl -lib work "../src/top/csrctrl.vhd"
add_file -vhdl -lib work "../src/top/datamux.vhd"
add_file -vhdl -lib work "../src/top/dpram2kx8.vhd"
add_file -vhdl -lib work "../src/top/dpram4kx8.vhd"
add_file -vhdl -lib work "../src/top/dpram64x16.vhd"
add_file -vhdl -lib work "../src/top/flashctrl.vhd"
add_file -vhdl -lib work "../src/top/lcdctrl.vhd"
add_file -vhdl -lib work "../src/top/memopmode.vhd"
add_file -vhdl -lib work "../src/top/ram256x20.vhd"
add_file -vhdl -lib work "../src/top/rs232ctrl.vhd"
add_file -vhdl -lib work "../src/top/sramctrl.vhd"
add_file -vhdl -lib work "../src/top/txdatamem.vhd"
add_file -vhdl -lib work "../src/top/rxdatamem.vhd"
add_file -vhdl -lib work "../src/top/waitctrl.vhd"
add_file -vhdl -lib work "../src/top/traceram.vhd"
add_file -vhdl -lib work "../src/oci/ujtag_syn.vhd"
add_file -vhdl -lib work "../src/core8051/
core8051_oci1_trace8_trig1_withoutio_apa.vhd"
add_file -vhdl -lib work "../src/top/sfr_misc.vhd"
add_file -vhdl -lib work "../src/top/platform8051_act.vhd"
add_file -constraint "platform_act.sdc"
add_file -constraint "core8051.sdc"


#implementation: "synplify"
impl -add synplify
```

```
#device options
set_option -technology PA
set_option -part APA600
set_option -speed_grade Std

#compilation/mapping options
set_option -default_enum_encoding default
set_option -symbolic_fsm_compiler 1
set_option -resource_sharing 1

#map options
set_option -frequency 100.000
set_option -fanout_limit 12
set_option -maxfan_hard 0
set_option -disable_io_insertion 0
set_option -retiming 0
set_option -report_path 4000

#simulation options
set_option -write_verilog 0
set_option -write_vhdl 0

#automatic place and route (vendor) options
set_option -write_apr_constraint 1

#set result format/file last
project -result_file "./platform8051_act.edn"

#implementation attributes
set_option -vlog_std v2001
impl -active "synplify"
```

After synthesis, the Actel Designer software tool must be used for layout. The following Tcl script (*platform.tcl*) can be run inside Designer to complete layout of the design. See "Design Layout" on page 31 for more information about this process.

```
##########################################################################
# platform.tcl - Designer Tcl script to compile & place & route
# Platform8051 design
# TFB 9/22/03
##########################################################################
```

```
set design_base "platform"
set design_gcf_file $design_base.gcf
set design_adb_file $design_base.adb
set design_log_file $design_base.log
set design_netlist_in "../synplify/platform8051_act.edn"

### setup various clocks

# import, compile, setup timing constraints
new_design -name $design_base -family "PA" -path {.}
set_device -die "APA600" -package "676 FBGA" -speed "STD" \
-voltage "2.5" -jtag "yes" -trst "yes" -temprange "COM" -voltrange "COM"
import_source -format "edif" -edif_flavor "GENERIC" $design_netlist_in \
-format "gcf" $design_gcf_file
compile
save_design $design_adb_file
layout -placer "On" -place_incremental "Off" -router "Off" -timing_driven
save_design $design_adb_file
layout -placer "Off" -router "On" -route_incremental "Off" -timing_driven
save_design $design_adb_file
export -format "log" $design_log_file
```

Downloading and Running the Basic Platform8051 Demonstration Project

Required Software Tools, and Hardware

To fully run this example design the following is required:

- HyperTerminal or similar terminal server PC software (settings: 1200 bps, 8-bit, no parity, one stop bit, no flow control)
- ISA-Actel51 Debugger software package
- Actel Libero IDE (design software for customizing the logic (if desired))
- FlashPro software for programming the Platform8051 board
- Platform8051 Development Board
- FlashPro Lite Programmer
- Windows PC (with ISA-Actel51 Debugger and Libero installed)
- Serial port
- Null modem cable

Downloading the Design

To download the design, program the FPGA image first:

1. Open the FlashPro PC software.
2. From the *File* menu, select *Connect…*
3. From the *File* menu, select *Analyze Chain*.
4. From the *File* menu, select *Open STAPL File*.
5. Select *8051demo_UART_LCD\FPGA_design\designer\platform.stp* and click *Open*.
6. Set *Action* to *Program* and click the *Play* button.

Downloading the Application

To download the application:

1. Set DIP switch S1 [1:8] to 00000011, where 0 is closed and 1 is open.
2. Open the FS2 ISA-Actel51 Debugger software.
3. Set *Port* to the appropriate port – usually LPT1.
4. Accept the default TckRate, Tvcc threshold.
5. From the *Tools* menu, select *Load Hex…*
6. Set the address to load at as 0x0000x.
7. Set the Filename as */8051demo_UART_LCD/demo_software/keil/pf8051_LCD_UART.hex*.
8. Click *Okay*.

Running the Design

To run the design:

1. Open HyperTerminal and connect a null modem cable to the PC serial port with the following settings: 1200 bps, 8-bit, no parity, one stopbit, no flow control.
2. In the FS2 ISA-Actel51 Debugger click *GO*.

   In the HyperTerminal window, the following information and command prompt will appear:

```
Actel Platform 8051 v1.01
   commands:
E <> : echo to LCD
C    : clear LCD
R    : report status
H    : help
```

```
T    : read ADC temp
       pf8051>
```

To echo text from the PC onto the LCD screen, use the "E <arg>" command:

1. **At the pf8051 prompt, type *E hello world!* and press return.**

   "hello world!" will then be displayed on the LCD screen.

```
pf8051> E hello world!
E hello world!
pf8051>
```

Web Server Demonstration Project

This project illustrates the many features of the Platform8051 Development Kit (Figure 7-2). The design is downloaded onto a Platform8051 development board where it can be run from on-board SRAM or Flash. Once downloaded and run, the supplied C code will act as a web server over the Ethernet connection, supplying temperature, power, and voltage data from the on-board ADC.

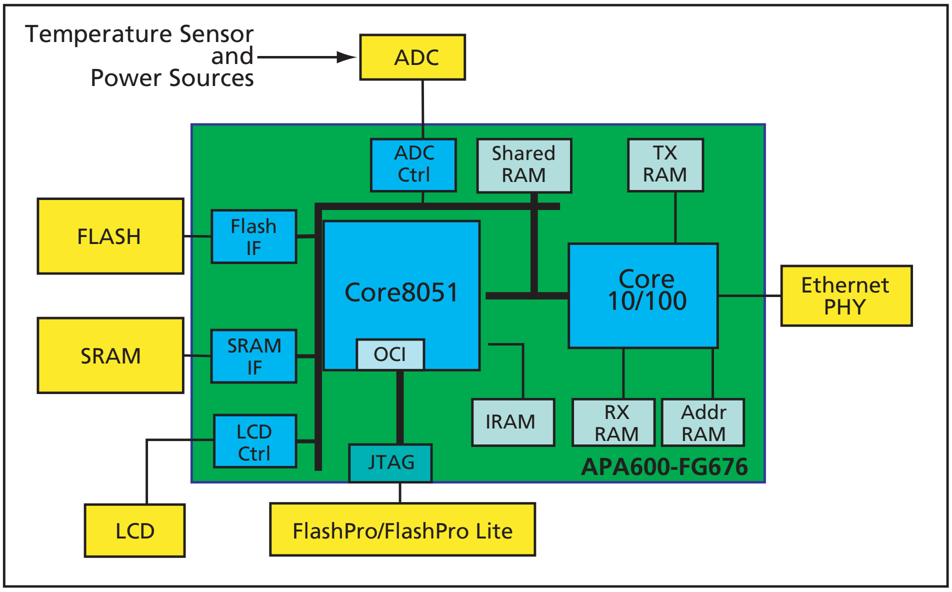


Figure 7-2. The FPGA Block Diagram of the Web Server Design

Memory Map Details

Memory Map Architecture Implementation in Platform8051 Web Server Example Design

Core8051's memory map is implemented in four different operating modes to support all possible configurations. These modes will be selected by memmode[1:0] signal in the FPGA design. The following sections describe the detailed operation of the memory map modes.

Memory Map Mode Selection

Memory mode will be generated by using Core8051's Port3 [7:5] outputs and external User Switch [8:7] settings. When the Core8051 drives Port3 [7] output to HIGH (default/reset state), memmode will be controlled by the external switch settings. When Port3 [7] is LOW, memmode will be generated by Port3 [6:5], which allows the designer to switch between all possible memory map modes by software control.

There are four possible memory modes in the demonstration designs:

- Shadow – Used when running the application code from Flash
- Normal – Application code is in SRAM, peripherals are all mapped in XDATA space
- Flash Program – Used to copy application code from SRAM to Flash (while CODE space is mapped and running from SRAM)
- Debug – Both XDATA and CODE are overlaid in SRAM

Memory Resource Mapping

Table 7-8 on page 61 and Table 7-9 on page 61 describe the mapping of the memory resources modes of operation.

Resources available on board for Platform8051 design are:

- 64 kbytes of Flash
- 64 kbytes of LRAM (lower part of SRAM)
- 64 kbytes of URAM (upper part of SRAM)

Table 7-8. Memory Map Mode Definitions

Mode	8051 Program Memory (64 Kbytes)	8051 Data Memory (64 Kbytes)
Shadow	0000:FFFF – Flash	0000:FFFF – LRAM
Normal	0000:FFFF – LRAM	0000:CFFF – URAM (52 kbytes) D000:EFFF – TX, RX memory for Core10/100 (8 kbytes) F000:FFFF – ADC/LCD registers (4 kbytes)
Flash Program	0000:FFFF – LRAM	0000:FFFF – Flash
Debug	0000:FFFF – LRAM	0000:FFFF – LRAM

Table 7-9. Memory Map Modes

Port 3[7]	Port 3[6:5]	**User SW[8:7]	Memmode[1:0]	Memory Map Mode
1	XX	00	00	Shadow
1	XX	01	01	Normal
1	XX	10	10	Flash Program
1	XX	11	11	Debug
0	00	XX	00	Shadow
0	01	XX	01	Normal
0	10	XX	10	Flash Program
0	11	XX	11	Debug

**These switches are wired to inputs of the FPGA so that *open* will correspond to a logic 1 (4.6 k pull-up to 3.3 V) and *closed* will correspond to a logic 0 (GND).

Memory Map for Normal Operation Mode

Core8051's Program Memory will be mapped to the External SRAM's lower 64 kbytes of memory space during normal operation mode. Data memory is mapped to various memories and registers, as shown in Table 7-10.

Table 7-10. Memory Map for Normal Operation

Data Address	Memory	R/W	Description
0000:CFFF (52 kbytes)	URAM	R/W	Upper portion on External SRAM
D000:DFFF (4 kbytes)	RX (MAC) Memory	R/W	MAC Receive Shared Memory
E000:EFFF (4 kbytes)	TX (MAC) Memory	R/W	MAC Transmit Shared Memory
F000:F0FF (256 Bytes)	CSR (MAC) Registers	R/W	MAC CSR registers.
F100	LCD Command	R/W	LCD command register
F101	LCD Data	R/W	LCD data register.
F102:F1FF	Reserved	–	Reserved for future use
F200	ADCIN(1 Byte)	R/W	ADC IN register
F201:F202	ADCOUT(2 Bytes)	R	ADC Out register
F203:FFFF	Reserved	–	Reserved for future use

Address range 0x0000 to 0xCFFF accesses the external SRAM's upper portion of 128 kbytes. Eight kbytes of internal dual-port memory is shared between Core8051 and Core10/100 to store receive and transmit Ethernet data. This internal dual-port memory can be accessed by Core8051 at address 0xD000 to 0xEFFF. Core8051 will be able access the Core10/100 MAC's CSR registers at address range 0xF000 and 0xF0FF. See the MAC user document [Is this doc on the CD?] for more details about the CSR registers.

Clock Configuration in the Platform8051 Web Server Example Design

The Platform8051 Web Server Example Design is implemented using a unified clock (as shown in Figure 2-1 on page 18). This makes IDLE and STOP modes unavailable in this design. Modules in the example designs, which are peripheral to Core8051, are driven by the main system clock. Minimizing the number of clocks enables easy timing analysis and increased performance.

The clock input to the APA600 in the Platform8051 example projects is divided by two inside the chip before being used. The design is running at 8 MHz, not the 16 MHz being supplied by the crystal oscillator on the board.

A second 25-MHz clock is used in this design for the Core10/100 only, and is also placed on a global clock buffer.

Core10/100 Configuration in Platform8051 Web Server Example Design

The Core10/100 supports various 8-, 16-, and 32-bit host interfaces and TX/RX FIFO sizes as required by the system. In the web server demonstration project, the 8-bit interface of Core10/100 was chosen because Core8051 has an 8-bit bus. In addition, the 8 kbytes XDATA memory space is shared with the Core10/100. More specifically, 4 kbytes of the 8 kbytes shared RAM space is allocated for transmit data, and the other 4 kbytes of the 8 kbytes shared RAM space is allocated for receive data. The TX FIFO is configured to use 2 kbytes of embedded FPGA RAM. Similarly, the RX FIFO is also configured to use 2 kbytes of embedded FPGA RAM. The Address RAM uses a 192 lines of 16-bit word-embedded FPGA RAM.

Table 7-11 shows the Core10/100 RTL parameters used for the web server demonstration project design.

Table 7-11. Core10/100 Parameters in Web Server Example

Core Parameter	Value	Description
CSRWIDTH	8	CSR interface data width
DATAWIDTH	8	DATA interface data width
DATADEPTH	13	DATA interface address width, support 8 kbytes
TFIFODEPTH	11	TX DATA RAM address width, support 2 kbytes
RFIFODEPTH	11	RX DATA RAM address width, support 2 kbytes

Detailed Core10/100 information is available in *Ethernet Media Access Controller Core10/100 data sheet*.

Core10/100 Memory Map in Platform8051 Web Server Example Design

The Core10/100 related resources can be reached when the Platform8051 Example Design is running in Normal mode (Normal mode is defined in "Memory Map Details" on page 52). The Core8051 memory map for Ethernet drivers is defined in Table 7-12.

Table 7-12. Ethernet Related Memory Addresses

Data Address	Memory	R/W	Description
D000:DFFF (4 kbytes)	Shared Memory	R/W	MAC Receive Shared Memory
E000:EFFF (4 kbytes)	Shared Memory	R/W	MAC Transmit Shared Memory
F000:F003	CSR0	R/W	MAC Bus mode
F008:F00B	CSR1	R/W	MAC Transmit poll demand
F010:F013	CSR2	R/W	MAC Receive poll demand
F018:F01B	CSR3	R/W	MAC Receive list base address
F020:F023	CSR4	R/W	MAC Transmit list base address
F028:F02B	CSR5	R/W	MAC Status
F030:F033	CSR6	R/W	MAC Operation mode
F038:F03B	CSR7	R/W	MAC Interrupt enable
F040:F043	CSR8	R/W	MAC Missed frames and overflow counters
F048:F048	CSR9	R/W	MAC MII management
F058:F05B	CSR11	R/W	MAC Timer and interrupt mitigation control

Detailed information about Core10/100 Control and Status Registers (CSR registers) is available in the *Ethernet Media Access Controller Core10/100 data sheet*.

Design Files

The design files for this project can be found on the PF8051 CD. These files include a C code project and source files for the hardware design (minus the Core8051 and Core10/100, which must be licensed separately). Constraints files and scripts for layout and synthesis are also provided.

Below is the Synplicity project file (*platform_act.prj*). This defines all the files and operations necessary to synthesize this project. Table 7-13 on page 68 contains a summary of the resources used when the below script is followed to perform the layout of the example project.

```
#-- Synplicity, Inc.
#-- Version 7.3.4
#-- Project file
Y:\deleons\master\pf8051devkit2.1\FPGA_design\synplify\platform_act.prj
#-- Written on Thu May 27 14:03:07 2004


#add_file options
add_file -vhdl -lib work "$LIB/proasic/proasicplus.vhd"
add_file -vhdl -lib work "../src/top/user_ram.vhd"
add_file -vhdl -lib work "../src/top/adc_def_pkg.vhd"
add_file -vhdl -lib work "../src/top/adc_regbank.vhd"
add_file -vhdl -lib work "../src/top/adcctrl.vhd"
add_file -vhdl -lib work "../src/top/addrdatactrl.vhd"
add_file -vhdl -lib work "../src/top/addrfiforam.vhd"
add_file -vhdl -lib work "../src/top/clkdiv.vhd"
add_file -vhdl -lib work "../src/top/csrctrl.vhd"
add_file -vhdl -lib work "../src/top/datamux.vhd"
add_file -vhdl -lib work "../src/top/macshmemctrl.vhd"
add_file -vhdl -lib work "../src/top/dpram2kx8.vhd"
add_file -vhdl -lib work "../src/top/dpram4kx8.vhd"
add_file -vhdl -lib work "../src/top/dpram64x16.vhd"
add_file -vhdl -lib work "../src/top/flashctrl.vhd"
add_file -vhdl -lib work "../src/top/idataram.vhd"
add_file -vhdl -lib work "../src/top/lcdctrl.vhd"
add_file -vhdl -lib work "../src/top/lcdfsm.vhd"
add_file -vhdl -lib work "../src/top/memopmode.vhd"
add_file -vhdl -lib work "../src/top/ram256x8.vhd"
add_file -vhdl -lib work "../src/top/ram256x20.vhd"
add_file -vhdl -lib work "../src/top/rs232ctrl.vhd"
add_file -vhdl -lib work "../src/top/rxfiforam.vhd"
add_file -vhdl -lib work "../src/top/sramctrl.vhd"
add_file -vhdl -lib work "../src/top/txdatamem.vhd"
add_file -vhdl -lib work "../src/top/rxdatamem.vhd"
add_file -vhdl -lib work "../src/top/txfiforam.vhd"
add_file -vhdl -lib work "../src/top/waitctrl.vhd"
add_file -vhdl -lib work "../src/top/traceram.vhd"
add_file -vhdl -lib MAC_LIB "../src/core10100/
Core10100_synplify_PROASICPLUS_noio_scb.vhd"
add_file -vhdl -lib work "../src/core8051/
core8051_oci1_trace8_trig4_withoutio_apa.vhd"
add_file -vhdl -lib work "../src/top/sfr_misc.vhd"
add_file -vhdl -lib work "../src/top/platform8051_act.vhd"
add_file -constraint "platform_act.sdc"


#implementation: "synplify"
impl -add synplify

#device options
set_option -technology PA
set_option -part APA600
set_option -speed_grade Std

#compilation/mapping options
set_option -default_enum_encoding default
set_option -symbolic_fsm_compiler 1
set_option -resource_sharing 1

#map options
set_option -frequency 25.000
set_option -fanout_limit 12
set_option -maxfan_hard 0
set_option -disable_io_insertion 0
set_option -report_path 4000

#simulation options
set_option -write_verilog 0
set_option -write_vhdl 0

#automatic place and route (vendor) options
set_option -write_apr_constraint 1

#set result format/file last
project -result_file "./platform_act.edn"
```

```
#implementation attributes
set_option -vlog_std v2001
impl -active "synplify"
```

After synthesis, the Actel Designer software tool must be used for layout. The following Tcl script (*platform.tcl*) can be run inside Designer to complete layout of the design. See "Design Layout" on page 31 for more information about this process.

```
##########################################################################
# platform.tcl - Designer Tcl script to compile & place & route
# Platform8051 design
# TFB 9/22/03
##########################################################################
set design_base "platform"
set design_gcf_file $design_base.gcf
set design_adb_file $design_base.adb
set design_log_file $design_base.log
set design_netlist_in "../synplify/platform8051_act.edn"

### setup various clocks

# import, compile, setup timing constraints
new_design -name $design_base -family "PA" -path {.}
set_device -die "APA600" -package "676 FBGA" -speed "STD" \
-voltage "2.5" -jtag "yes" -trst "yes" -temprange "COM" -voltrange "COM"
import_source -format "edif" -edif_flavor "GENERIC" $design_netlist_in \
-format "gcf" $design_gcf_file
compile
save_design $design_adb_file
layout -placer "On" -place_incremental "Off" -router "Off" -timing_driven
save_design $design_adb_file
layout -placer "Off" -router "On" -route_incremental "Off" -timing_driven
save_design $design_adb_file
export -format "log" $design_log_file
```

Table 7-13. Device Utilization for Platform8051 Demo Design (Core8051 + Core10/100)

Family	Cells or Tiles			Utilization		Performance
	Sequential	RAM	Combinatorial	Device	Total	
ProASIC^{PLUS}	2789	432	12528	APA600-STD	62%	16 MHz

Changing the Ethernet MAC Address

If more then one Platform8051 web server demonstration project needs to be run on the same Ethernet LAN, it may be necessary to change the Ethernet MAC address in the project. To generate a design with a different MAC address, search for *my_mac* in the file *pf8051_demo.c*, change the values to reflect a unique MAC address, and then re-compile the code.

Downloading and Running the Web Server Demonstration Project

Directions for Running the Web Server over a LAN with a DHCP Server

Follow these instructions when you connect the Platform8051 development board to your company's Ethernet network. Your Ethernet network must have an active DHCP server.

1. **Connect the power plug into the power connector of the board.**
2. **Connect the Ethernet Cat5 cable to an Ethernet RJ-45 jack.**
3. **The position of the DIP switch should be set to *all open*.**
4. **Turn on the power switch of the board;** the default address (10.0.0.198) is displayed on the LCD.
5. **Wait for up to 2-3 minutes, until the IP address <*A.B.C.D*> obtained from the DHCP server appears on the LCD display.**
6. **Enter the following http request in your web browser according to the IP address shown on the LCD of the board:**

   http://A.B.C.D, e.g. *http://190.10.15.118*
7. **The web page shown in** Figure 7-3 on page 70 **will appear.**
8. **Enter a text string in the *Text to display in LCD* text box in the web page and click the *Submit* button.** The entered text string should be displayed on the LCD of the board. The maximum number of the characters you can enter is 16.

Directions for Directly Connecting the Web Server to a PC

Follow these instructions when connecting the Platform8051 development board to your PC directly without using a DHCP server:

1. **Connect the power plug into the power connector of the board.**
2. **Connect the Ethernet Cat5 cable to the crossover adaptor and then to your PC.** Or connect the Ethernet crossover cable to your PC's Ethernet jack directly.
3. **The position of the DIP switch should be set to *all open*.**
4. **Turn on the power switch of the board;** the default address (10.0.0.198) is displayed on the LCD.
5. **Wait for 2-3 minutes.**
6. **Enter the following http request in your web browser: *http://10.0.0.198*.**
7. **The web page shown in** Figure 7-3 on page 70 **will appear.**
8. **Enter a text string in the *Text to display in LCD* text box in the web page and click the *Submit* button.** The entered text string should be displayed on the LCD of the board. The maximum number of the characters you can enter is 16.

Web Server Application Description

A sample web server application is included in the Platform8051 Development Kit to demonstrate the functionality of a Core8051 system.

Actel IP Core8051 and Core10/100 are used in the demonstration design shown in Figure 7-2 on page 59.

Top-level source code for the FPGA design can be found on the Platform8051 Development Kit CD.

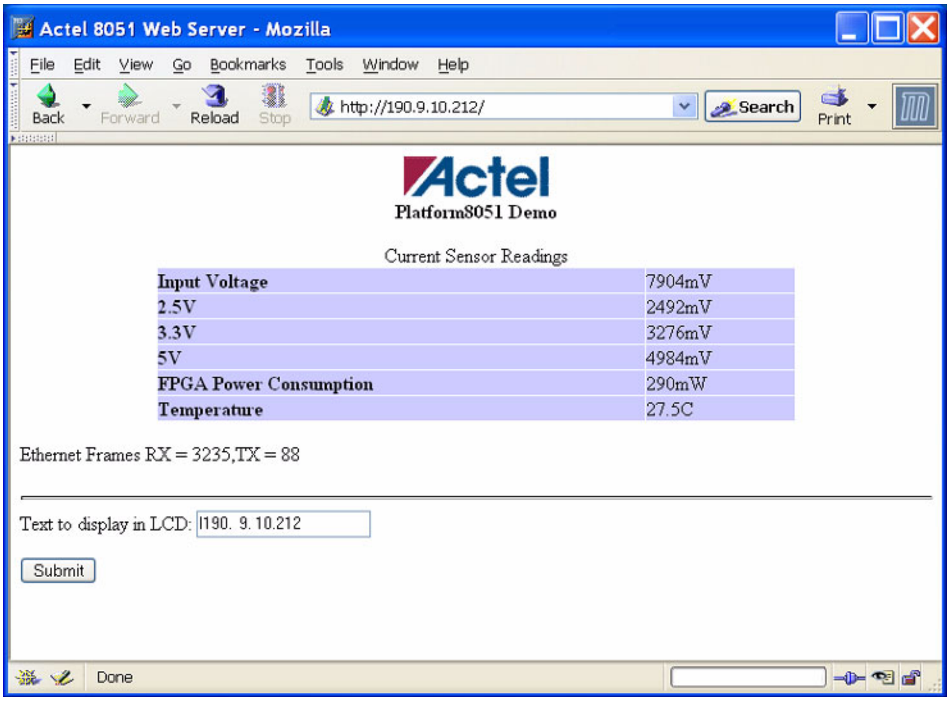
Core8051 obtains analog readings from eight analog channels, as listed below:

- Channel 0 for FPGA power consumption estimation
- Channel 1 for 2.5 V source
- Channel 2 for 3.3 V source
- Channel 3 for input power voltage
- Channel 4 for 5.0 V source
- Channel 5 and 6 are connected to daughter board area
- Channel 7 for temperature sensor

The web server displays the above information on a user web page.

Web Server Software

The web server software is written in C and provided on the Platform8051 Development Kit CD. It is example code and is not supported or warranted by Actel. The Platform8051 Web Server user interface is a web page shown in Figure 7-3.



Platform8051 Demo

Current Sensor Readings

Input Voltage	7904mV
2.5V	2492mV
3.3V	3276mV
5V	4984mV
FPGA Power Consumption	290mW
Temperature	27.5C

Ethernet Frames RX = 3235,TX = 88

Text to display in LCD: l190. 9. 10.212

Submit

Figure 7-3. The Web Page Served by the Example Design

A

Platform8051 FPGA Design Pinout (u1)

```
*****************************************************************
Pin Report - Date: Mon May 10 15:41:57 2004  Pinchecksum: NOT-AVAILABLE
  Design Name: platform  Family: PA  Die: APA600  Package: 676 FBGA
*****************************************************************

Name          Number      FixedTypeDescription
------------  ----------  ---------------------


ADC_CSn       J24         FIXEDOutADC Chip Select
ADC_DIN       J22         FIXEDInADC Data Input
ADC_DOUT      J25         FIXEDInADC Data Output
ADC_SCLK      J23         FIXEDOutADC Clock
ADC_SSTRB     J26         FIXEDInADC Strobe
CLK_16MHZ     N22         FIXEDIn16Mhz Input Clock
COL           A6          FIXEDInMAC Collision Detect
CRS           A5          FIXEDInMAC Carrier Sense
FLASH_BYTE    M5          FIXEDOutFlash Word/Byte Detect
FLASH_CE      M1          FIXEDOutFlash Chip Enable, Active Low
FLASH_OE      M2          FIXEDOutFlash Output Enable, Active Low
FLASH_RP      M4          FIXEDOutFlash Reset, Active Low
FLASH_WE      M3          FIXEDOutFlash Write Enable, Active Low
KEYPAD(0)     AF23        FIXEDInPushbutton Kewpad Switch Input
KEYPAD(1)     AF24        FIXEDIn
KEYPAD(2)     AF22        FIXEDIn
KEYPAD(3)     AF21        FIXEDIn
KEYPAD(4)     AF20        FIXEDIn
LCD_DB(0)     AB18        FIXEDIn/OutLCD Data Bus
LCD_DB(1)     AB17        FIXEDIn/Out
LCD_DB(2)     AB16        FIXEDIn/Out
LCD_DB(3)     AB15        FIXEDIn/Out
LCD_DB(4)     AB14        FIXEDIn/Out
LCD_DB(5)     AA19        FIXEDIn/Out
LCD_DB(6)     AA18        FIXEDIn/Out
LCD_DB(7)     AA17        FIXEDIn/Out
LCD_E         AA14        FIXEDOutLCD Enable
LCD_RS        Y16         FIXEDOutLCD Data/Instruction Control Signal
LCD_RWn       Y18         FIXEDOutLCD Read/Write
LCD_TR        Y14         FIXEDOutLCD Data Bus Direction Control
LED_OUT(0)    AF14        FIXEDOutLED Output
LED_OUT(1)    AF15        FIXEDOut
LED_OUT(2)    AF16        FIXEDOut
LED_OUT(3)    AF17        FIXEDOut
LED_OUT(4)    AF18        FIXEDOut
LED_OUT(5)    AF19        FIXEDOut
MAC_RESETn    B12         FIXEDOut
MDC           A3          FIXEDOutMAC Management Clock
MDIO          A4          FIXEDIn/OutMAC Management Data I/O
MEM_ADDR(0)   P3          FIXEDOutExternal Memory Address Bus
MEM_ADDR(1)   P4          FIXED Out
MEM_ADDR(2)   P6          FIXED Out
MEM_ADDR(3)   P7          FIXED Out
MEM_ADDR(4)   R1          FIXED Out
MEM_ADDR(5)   R2          FIXED Out
MEM_ADDR(6)   R3          FIXED Out
MEM_ADDR(7)   R4          FIXED Out
MEM_ADDR(8)   R5          FIXED Out
MEM_ADDR(9)   R6          FIXED Out
MEM_ADDR(10)  T1          FIXED Out
MEM_ADDR(11)  T2          FIXED Out
MEM_ADDR(12)  T3          FIXED Out
MEM_ADDR(13)  T4          FIXED Out
MEM_ADDR(14)  T5          FIXED Out
MEM_ADDR(15)  T6          FIXED Out
MEM_ADDR(16)  T7          FIXED Out
MEM_ADDR(17)  U1          FIXED Out
MEM_ADDR(18)  U2          FIXED Out
MEM_ADDR(19)  U3          FIXED Out
MEM_ADDR(20)  U4          FIXED Out
MEM_ADDR(21)  U5          FIXED Out
MEM_DATA(0)   U6          FIXED In/OutExternal Memory Data Bus
MEM_DATA(1)   V1          FIXED In/Out
MEM_DATA(2)   V2          FIXED In/Out
MEM_DATA(3)   V3          FIXED In/Out
MEM_DATA(4)   V4          FIXED In/Out
MEM_DATA(5)   V5          FIXED In/Out
MEM_DATA(6)   V6          FIXED In/Out
MEM_DATA(7)   V7          FIXED In/Out
RESETn        P1          FIXED InGlobal Hardware Reset Input, Active Low
RS232_RTS     D25         FIXEDOutRTS for RS232/UART port
RS232_RXDATA  C25         FIXEDInIncoming Data for RS232/UART
RS232_TXDATA  D24         FIXEDOutOutgoing Data for RS232/UART
RXCLK         B5          FIXEDInMAC Receive Clock
RXDATA(0)     B9          FIXEDInMAC Receive Data
```

RXDATA(1)	B8	FIXEDIn
RXDATA(2)	B7	FIXEDIn
RXDATA(3)	B6	FIXEDIn
RXDV	B11	FIXEDInMAC Receive Data Valid
RXER	B10	FIXEDInMAC Receive Error
SRAM_ADSC	AF8	FIXEDOutSRAM Address Strobe, Active Low
SRAM_ADSP	AF9	FIXEDOutSRAM Address Strobe, Active Low
SRAM_ADV	AF10	FIXEDOutSRAM Advance Enable, Active Low
SRAM_BA	AE11	FIXEDOutSRAM Byte Write Enable A, Active Low
SRAM_BB	AE12	FIXED OutSRAM Byte Write Enable B, Active Low
SRAM_BC	AE13	FIXED OutSRAM Byte Write Enable C, Active Low
SRAM_BD	AF5	FIXED OutSRAM Byte Write Enable D, Active Low
SRAM_BW	AF6	FIXEDOutSRAM Byte Write Enable, Active Low
SRAM_CLK	AF13	FIXEDOutSRAM Clock Input, Active High
SRAM_E	AE9	FIXEDOutSRAM Chip Enable, Active High
SRAM_FT	AE8	FIXEDOutSRAM PipeLine/FlowThrough Mode
SRAM_G	AE10	FIXEDOutSRAM Output Enable, Active Low
SRAM_GW	AF7	FIXEDOutSRAM Global Write Enable, Active Low
SRAM_LBO	AF12	FIXEDOutSRAM Linear Burst Order Mode, Active Low
SRAM_ZZ	AF11	FIXEDOutSRAM Sleep Mode Control, Active High
TXCLK	A7	FIXEDInMAC Transmit Clock
TXDATA(0)	A11	FIXEDOutMAC Transmit Data
TXDATA(1)	A10	FIXED Out
TXDATA(2)	A9	FIXEDOut
TXDATA(3)	A8	FIXEDOut
TXEN	A12	FIXEDOutMAC Transmit Enable
TXER	A13	FIXEDOutMAC Transmit Error
USER_SW(0)	AC10	FIXEDInDIP Switch Input
USER_SW(1)	AC11	FIXEDIn
USER_SW(2)	AC12	FIXEDIn
USER_SW(3)	AC13	FIXEDIn
USER_SW(4)	AD4	FIXEDIn
USER_SW(5)	AD5	FIXEDIn
USER_SW(6)	AD6	FIXEDIn
USER_SW(7)	AD7	FIXEDIn

B

PCD Daughter Card Connections (j23)

```
PIN NAME       CONNECTED TO
--- ----------------------------------------------------
1   GND
2   GND
3   CLK_25MHZ_2u22 74LVT245 pin# 4 (u21 XTALOSC)
4   GND
5   GND
6   GND
7   V2.5
8   V2.5
9   FPGA_DBC_IO1u1a APA600 pin# AD26
10  FPGA_DBC_IO2u1a APA600 pin# AD25
11  FPGA_DBC_IO3u1a APA600 pin# AC26
12  FPGA_DBC_IO4u1a APA600 pin# AC25
13  FPGA_DBC_IO5u1a APA600 pin# AB26
14  FPGA_DBC_IO6u1a APA600 pin# AB25
15  FPGA_DBC_IO7u1a APA600 pin# AB24
16  FPGA_DBC_IO8u1a APA600 pin# AA26
17  FPGA_DBC_IO9u1a APA600 pin# AA25
18  FPGA_DBC_IO10u1a APA600 pin# AA24
19  FPGA_DBC_IO11u1a APA600 pin# Y26
20  FPGA_DBC_IO12u1a APA600 pin# Y25
21  FPGA_DBC_IO13u1a APA600 pin# Y24
22  FPGA_DBC_IO14u1a APA600 pin# Y23
23  FPGA_DBC_IO15u1a APA600 pin# W26
24  FPGA_DBC_IO16u1a APA600 pin# W25
25  FPGA_DBC_IO17u1a APA600 pin# W24
26  FPGA_DBC_IO18u1a APA600 pin# W23
27  FPGA_DBC_IO19u1a APA600 pin# W22
28  FPGA_DBC_IO20u1a APA600 pin# W21
29  FPGA_DBC_IO21u1a APA600 pin# V26
30  FPGA_DBC_IO22u1a APA600 pin# V25
31  FPGA_DBC_IO23u1a APA600 pin# V24
32  FPGA_DBC_IO24u1a APA600 pin# V23
33  FPGA_DBC_IO25u1a APA600 pin# V22
34  FPGA_DBC_IO26u1a APA600 pin# V21
35  FPGA_DBC_IO27u1a APA600 pin# V20
36  FPGA_DBC_IO28u1a APA600 pin# U26
37  FPGA_DBC_IO29u1a APA600 pin# U25
38  FPGA_DBC_IO30u1a APA600 pin# U24
39  FPGA_DBC_IO31u1a APA600 pin# U23
```

```
40  FPGA_DBC_IO32u1a APA600 pin# U22
41  FPGA_DBC_IO33u1a APA600 pin# U21
42  FPGA_DBC_IO34u1a APA600 pin# T26
43  FPGA_DBC_IO35u1a APA600 pin# T25
44  FPGA_DBC_IO36u1a APA600 pin# T24
45  FPGA_DBC_IO37u1a APA600 pin# T23
46  FPGA_DBC_IO38u1a APA600 pin# T22
47  FPGA_DBC_IO39u1a APA600 pin# T21
48  FPGA_DBC_IO40u1a APA600 pin# T20
49  FPGA_DBC_IO41u1a APA600 pin# R26
50  FPGA_DBC_IO42u1a APA600 pin# R25
51  FPGA_DBC_IO43u1a APA600 pin# R24
52  FPGA_DBC_IO44u1a APA600 pin# R23
53  FPGA_DBC_IO45u1a APA600 pin# R22
54  FPGA_DBC_IO46u1a APA600 pin# R21
55  FPGA_DBC_IO47u1a APA600 pin# P23
56  FPGA_DBC_IO48u1a APA600 pin# P21
57  FPGA_DBC_IO49u1a APA600 pin# P20
58  FPGA_DBC_IO50u1a APA600 pin# N26
59  FPGA_DBC_IO51u1a APA600 pin# N23
60  FPGA_DBC_IO52u1a APA600 pin# N21
61  FPGA_DBC_IO53u1a APA600 pin# M26
62  FPGA_DBC_IO54u1a APA600 pin# M25
63  FPGA_DBC_IO55u1a APA600 pin# M24
64  FPGA_DBC_IO56u1a APA600 pin# M23
65  FPGA_DBC_IO57u1a APA600 pin# M22
66  FPGA_DBC_IO58u1a APA600 pin# M21
67  FPGA_DBC_IO59u1a APA600 pin# M20
68  FPGA_DBC_IO60u1a APA600 pin# L26
69  DBC_IO5VOUT0u19 74LVT245 pin#18 (u1a APA600 pin# L25)
70  DBC_IO5VOUT1u19 74LVT245 pin#17 (u1a APA600 pin# L24)
71  DBC_IO5VOUT2u19 74LVT245 pin#16 (u1a APA600 pin# L23)
72  DBC_IO5VOUT3u19 74LVT245 pin#15 (u1a APA600 pin# L22)
73  DBC_IO5VIN0u18 74LVT245 pin#18 (u1a APA600 pin# L21)
74  DBC_IO5VIN1u18 74LVT245 pin#17 (u1a APA600 pin# K26)
75  DBC_IO5VIN2u18 74LVT245 pin#16 (u1a APA600 pin# K25)
76  DBC_IO5VIN3u18 74LVT245 pin#15 (u1a APA600 pin# K24)
77  ADC_CH5_DBC_P77u31 MAX1204 pin# 6
78  ADC_CH6_DBC_P78 u31 MAX1204 pin# 7
79  CLK_16MHZ_2u22 74LVT245 pin# 8 (u23 XTALOSC)
80  CLK_USERu22 74LVT245 pin# 9 (u25)
81  FPGA_EE_I2C_SCLu12 M24256 pin# 6
```

```
82  FPGA_EE_I2C_SDA u12 M24256 pin# 5
83             NC
84             NC
85             NC
86             NC
87  VCC
88  FPGA_DBC_GCLKu1c APA600 pin# N25
89  V3.3
90  V3.3
```

C
References

Core8051 Data Sheet

www.actel.com/ipdocs/Core8051DS.pdf

Core8051 Instruction Set User's Guide

PF8051 CD: /docs/Core8051UG.pdf

www.actel.com/ipdocs/Core8051UG.pdf

Core8051 User's Guide

PF8051_CD: /docs/Core8051_UG.pdf

Core10/100 Data Sheet

www.actel.com/ipdocs/Core10100DS.pdf

Core10/100 User's Guide

PF8051_CD: /docs/Core10/100UG.pdf

Platform8051 Development Kit Quick Start Guide

PF8051_CD: /docs/Platform8051_devkit_QuickStart.pdf

Platform8051 Programming and Testing Procedure

PF8051_CD: /docs/Platform8051_programming_testing_procedure10.pdf

ProASIC^{PLUS} Flash Family FPGA Data Sheet

www.actel.com/documents/ProASIC^{PLUS}DS.pdf

Designer User's Guide

www.actel.com/documents/designerUG.pdf

Actel Application Notes

www.actel.com/techdocs/appnotes/index.html

D

Product Support

Actel backs its products with various support services including Customer Service, a Customer Technical Support Center, a web site, an FTP site, electronic mail, and worldwide sales offices. This appendix contains information about contacting Actel and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From Northeast and North Central U.S.A., call **650.318.4480**
From Southeast and Southwest U.S.A., call **650. 318.4480**
From South Central U.S.A., call **650.318.4434**
From Northwest U.S.A., call **650.318.4434**
From Canada, call **650.318.4480**
From Europe, call **650.318.4252** or **+44 (0)1276 401 500**
From Japan, call **650.318.4743**
From the rest of the world, call **650.318.4743**
Fax, from anywhere in the world **650. 318.8044**

Actel Customer Technical Support Center

Actel staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions. The Customer Technical Support Center spends a great deal of time creating application notes and answers to FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Actel Technical Support

Visit the Actel Customer Support website (www.actel.com/custsup/search.html) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the Actel web site.

Website

You can browse a variety of technical and non-technical information on Actel's home page, at www.actel.com.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. Several ways of contacting the Center follow:

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is tech@actel.com.

Phone

Our Technical Support Center answers all calls. The center retrieves information, such as your name, company name, phone number and your question, and then issues a case number. The Center then forwards the information to a queue where the first available application engineer receives the data and returns your call. The phone hours are from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. The Technical Support numbers are:

650.318.4460
800.262.1060

Customers needing assistance outside the US time zones can either contact technical support via email (tech@actel.com) or contact a local sales office. Sales office listings can be found at www.actel.com/contact/offices/index.html.

Index

50200045-0/12.04