Embedded Design with FPGAs and ARM Cortex-M1

Dominic Pajak, ARM
Jean Labrosse, Micrium
Mike Thompson, Actel

April 2008 – Embedded Systems Conference
Agenda

- ARM Cortex-M1
  - Dominic Pajak

- Micrium uC/OS-II
  - Jean Labrosse

- Using Cortex-M1 in FPGA
  - Mike Thompson
Embedded systems in FPGA

- FPGAs are available at increasingly low-cost
- Rapid development of time-to-market critical designs
- Initial product runs, design entry, prototyping
- Long-life applications – no risk of obsolescence
- Networking, Consumer, Industrial/Auto, Aero, Portable apps

Reduce risk, costs and time-to-market combined with ARM

- ARM Cortex-M1 optimised for FPGA implementation
- ARM proven in billions of ARM Powered devices
- Familiar, high-quality development tools and OS support
- Migration path to ASIC through software compatibility

Available in Actel Fusion, Igloo, and ProASIC3 devices
ARM Cortex processor family

- Common architecture across the performance spectrum
  - Thumb®-2 blended 16/32-bit ISA
  - Performance and efficiency
- ARM Cortex A Series - Applications CPUs focused on the execution of complex OS and user applications
- ARM Cortex R Series - Deeply embedded processors focused on real-time environments
- ARM Cortex M Series - Microcontroller cores focused on very area sensitive, deterministic, interrupt driven environments
- Cortex-M1 is upward compatible with Cortex family
ARM Cortex-M1 is easy to use

- A standard processor architecture for all FPGAs
  - Enabling code and tool reuse
  - Dramatically lowers the cost of migrating across FPGAs
  - Users can seamlessly migrate to leading edge FPGA device

- Designed to simplify software development
  - Familiar, high-quality development tools and OS
    - e.g. RealView, Keil, GNU....
  - Everything can be written in C
    - No need for assembler for vector table or interrupt handlers

- Compatibility roadmap to ASIC/ASSP and MCU
  - Object code upwards compatibility with Cortex processors in ASIC
  - Compatible with legacy Thumb code from ARM7TDMI onwards
ARM Cortex-M1 processor features

- **A 3-stage, 32-bit RISC processor**
  - Highly configurable to enable design trade-offs
  - Retains the same programmers model for software simplicity

- **Tightly Coupled Memories**
  - Internal FPGA block RAM used as single-cycle access memory
  - ITCM, DTCM configurable from 0k to 1024kBytes

- **Configurable debug**
  - JTAG or reduced pin-count SWD interface
  - Full – 2 watchpoints, 4 breakpoints
  - Small – 1 watchpoint, 2 breakpoints
  - None – removable for cost reduction and security
ARM Cortex-M1 processor features (2)

- **Integrated Interrupt Controller**
  - Fast interrupt response
  - Configurable 1, 8, 16, 32
  - Software programmed priority levels (1-4)
  - Non-Maskable Interrupt

- **AMBA AHB-lite 32-bit bus interface**
  - Connection to external memory and peripherals

- **Big or little endian**
  - Synthesis time configurable
ARM Cortex-M1 compatibility

- ARM Cortex-M1 implements a lightweight Thumb-2 profile
  - Processor executes blended 16-bit, 32-bit in one execution mode

- Can execute existing Thumb code
  - For example Thumb code from ARM7TDMI, ARM926EJ-S onwards
  - Most microcontroller applications use predominately Thumb code
  - Upwards compatible with Cortex processors

- Migration to ASIC/ASSP & MCU at all performance points

Instruction set compatibility:

- Thumb®
- ARM7TDMI
- ARM926EJ-S
- ARM1176EJ-S
- Cortex-M1
- Cortex-M3
- Cortex-R4
- Cortex-A8

Thumb-2
# Cortex-M1 instruction set

- **Bringing 32-bit performance, with advantages of 16-bit code density**
  - More software can be squeezed into on-chip FPGA RAM
  - Can allow a drop in FPGA size or remove the need for additional off-chip RAM
  - Reduces cost, improves performance

- **Simple Instruction Set Architecture**
  - Based on 16-bit Thumb, plus 32-bit Thumb-2 system instructions

<table>
<thead>
<tr>
<th>16-bit</th>
<th>32-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>ADD</td>
</tr>
<tr>
<td>B</td>
<td>BIC</td>
</tr>
<tr>
<td>CMN</td>
<td>CMP</td>
</tr>
<tr>
<td>LDM</td>
<td>LDR</td>
</tr>
<tr>
<td>LDRSH</td>
<td>LSL</td>
</tr>
<tr>
<td>MVN</td>
<td>NEG</td>
</tr>
<tr>
<td>PUSH</td>
<td>REV</td>
</tr>
<tr>
<td>RSB</td>
<td>SBC</td>
</tr>
<tr>
<td>STRB</td>
<td>STRH</td>
</tr>
<tr>
<td>SXTH</td>
<td>TST</td>
</tr>
<tr>
<td>WFI</td>
<td>YIELD</td>
</tr>
<tr>
<td></td>
<td>BL</td>
</tr>
<tr>
<td></td>
<td>DMB</td>
</tr>
<tr>
<td></td>
<td>DSB</td>
</tr>
<tr>
<td></td>
<td>ISB</td>
</tr>
<tr>
<td></td>
<td>MRS</td>
</tr>
<tr>
<td></td>
<td>MSR</td>
</tr>
<tr>
<td></td>
<td>32-bit</td>
</tr>
</tbody>
</table>

- More software can be squeezed into on-chip FPGA RAM
- Can allow a drop in FPGA size or remove the need for additional off-chip RAM
- Reduces cost, improves performance

- **Simple Instruction Set Architecture**
  - Based on 16-bit Thumb, plus 32-bit Thumb-2 system instructions

- **16-bit**
- **32-bit**
Cortex-M1 is very easy to program

- Vector Table
- Interrupt entry/exit stubs
- Exception Handler
- Initialization code
- High quality tools
- Optimized compiler
Interrupt handling is fast and simple

Other processors

ARM Cortex-M1

- Core automatically handles
  - Saving corruptible registers
  - Exception prioritization
  - Exception nesting

- No need for assembly, just C
  - Pointer to C routine at vector
  - ISR is a C function

- Faster interrupt response
  - With less software effort

Interrupt handling is fast and simple

Other processors

ARM Cortex-M1

- Core automatically handles
  - Saving corruptible registers
  - Exception prioritization
  - Exception nesting

- No need for assembly, just C
  - Pointer to C routine at vector
  - ISR is a C function

- Faster interrupt response
  - With less software effort

Interrupt handling is fast and simple

Other processors

ARM Cortex-M1

- Core automatically handles
  - Saving corruptible registers
  - Exception prioritization
  - Exception nesting

- No need for assembly, just C
  - Pointer to C routine at vector
  - ISR is a C function

- Faster interrupt response
  - With less software effort
RealView Tools support

- **Cortex-M1 supported first by ARM RealView tools**
  - High-quality, intuitive software development tools
  - Fully exploits performance and code density advantage of Cortex-M1

- **RealView Compiler Tools v3.1 feature MicroLib**
  - C libraries for embedded and memory constrained applications

- **Optimized for embedded apps**
  - To get the most out of Cortex-M1

### RealView Tools support

<table>
<thead>
<tr>
<th>Processor</th>
<th>Object</th>
<th>Library Total</th>
<th>Standard</th>
<th>MicroLib</th>
<th>% saving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cortex-M1</td>
<td>Thumb-2</td>
<td>Library Total</td>
<td>16452</td>
<td>5996</td>
<td>64%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RO Total (bytes)</td>
<td>19472</td>
<td>9016</td>
<td>54%</td>
</tr>
</tbody>
</table>

*Dhrystone 2.1 Benchmark*
ARM Cortex-M1 Summary

- Cortex-M1 and tools available as a free download from Actel
  - Encrypted version of ARM Cortex-M1 Processor
  - Supporting Actel Fusion, ProASIC3 and Igloo devices
  - www.actel.com

- Complete solution
  - ARM Cortex-M1
  - Peripherals
  - Hardware Design
  - Software Design
  - Synthesis
  - Development Boards
Agenda

- ARM Cortex-M1
  - Dominic Pajak

- Micrium uC/OS-II
  - Jean Labrosse

- Using Cortex-M1 in FPGA
  - Mike Thompson
Who is Micrium?

- **Florida Corporation**
  - Started in 1999
  - Satellite office in Montreal, Canada
  - Worldwide distributors

- **Provider of:** High Quality Embedded Software Components
  - Some products are FAA/FDA certified
  - Outstanding support and documentation

- **Code provided in source form**
  - Cleanest Source Code in the Industry!
µC/OS-II
- Embedded RTOS
- µC/OS-MMU
- µC/OS-MPU

µC/FS
- Embedded File System

µC/GUI
- Embedded Graphical User Interface

µC/BuildingBlocks
- Software time-of-day clock (µC/CLK)
- Character-based LCD (µC/LCD)
- Shell (µC/Shell)
- CRC Calculation (µC/CRC)

µC/Probe
- Run-Time Data Monitor

µC/TCP-IP
- Embedded TCP/IP v4 stack
- DHCPc, DNSc, FTP, HTTPs, POP3c, SMTPc, SNTPc, etc.

µC/USB-Device
- Bulk-device stack
- Mass-Storage Class

µC/USB-Host
- HID
- CDC
- Mass-Storage Class

µC/Modbus
- Master and/or Slave
- RS-232C or RS-485
- ASCII and/or RTU

µC/CAN
- CAN Framework
Type of Operating System Used for the Current Project
Summary of Chart in Volume 1 Report
(Percent of Respondents, N = 523)

Components are Royalty-Free

Licensed on a ‘per-end-product’ basis

- Perpetual use on that end-product
- Unlimited number of units

Each ‘different’ product that embeds our software requires a license

Other licensing schemes are available:

- Product Line (i.e. Family)
- Per-CPU type (ARM, Cortex, etc.)
- Site
Preemptive Multitasking

Supports up to 255 tasks and 255 priorities

Written in ANSI C

Highly portable
  - Ported to over 45 different CPU architectures

Scalable and ROMable
  - 5K to 20K bytes code, 1K to 3K bytes data (Cortex-M1)

Provides standard ‘services’ to the application
  - Semaphores, Mutexes, Queues, Task, Time, etc.

High Performance
µC/OS-II
The Real-Time Kernel

- Used in 1000s of products all over the world

- Book describing internals

- Adopted by 100s of Colleges & Universities

- Supports most of the MISRA C rules

Third Party Certified
- FAA - DO178B Level A
- FDA - 510(k)
- IEC - 61508

DO-178B SOFTWARE CERTIFICATION LEVELS

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Software that could cause or contribute to the failure of the system resulting in a catastrophic failure condition.</td>
</tr>
<tr>
<td>B</td>
<td>Software that could cause or contribute to the failure of the system resulting in a hazardous or severe failure condition.</td>
</tr>
<tr>
<td>C</td>
<td>Software that could cause or contribute to the failure of the system resulting in a major failure condition.</td>
</tr>
<tr>
<td>D</td>
<td>Software that could cause or contribute to the failure of the system resulting in a minor failure condition.</td>
</tr>
<tr>
<td>E</td>
<td>Software that could cause or contribute to the failure of the system resulting in no effect on the system.</td>
</tr>
</tbody>
</table>
Cleanroom design
- Based on RFCs
- ANSI C

Easy to Port
- Works with any RTOS (but needs an RTOS)
- Works with most 32 bit CPUs
- Works with any NIC
- Small Footprint (Scalable: 75 to 120 Kbytes)

- Critical sections kept to a minimum

- Zero copy

- Small and Large packet buffers
  - Efficient use of memory
  - Small and Large buffer size adjustable at compile time
A universal embedded system monitoring tool

Windows app. that collects and displays target data at run-time:
  - Any variable
  - Any memory location
  - Any I/O port

Works with ANY processor
  - 8-, 16-, 32-, 64-bit or DSP

Works with ANY compiler
  - Compiler/linker needs to generate a .ELF file

Works with any interface
  - RS-232C, TCP/IP, USB, J-Tag, etc.
- ‘Visualize’ a live embedded system using:
  - Gauges and Meters
  - Numeric indicators
  - Graphs and Plots
  - Barcharts
  - Tables
  - Virtual LEDs
  - Etc.

- Change variables in the target using
  - Sliders
  - Dials
  - Switches
  - Buttons
  - Etc.
Agenda

■ ARM Cortex-M1
  ● Dominic Pajak

■ Micrium uC/OS-II
  ● Jean Labrosse

■ Using Cortex-M1 in FPGA
  ● Mike Thompson
Actel Cortex-M1 Design Flow

Hardware and Software Requirements

HARDWARE DEVELOPMENT FLOW

- **CoreConsole**
  - Select Processor
  - Choose Peripherals
  - Add Optional User Custom IP
  - Auto Stitch to Build System

- **Libero IDE**
  - Combine System with Non-Processor Code
  - Synthesis, Simulation, and Layout
  - Optional Timing/Power Analysis
  - Easy to use GUI

SOFTWARE DEVELOPMENT FLOW

- **SoftConsole**
  - Eclipse-Base IDE
  - GNU C-Compiler
  - GNU Debugger
  - Memory Map and Peripheral Core Driver File Import

OR

Industry Standard Support

- Compilers – RealView, Keil, IAR, GNU
- RTOS – uC/OS II, uClinux, Nucleus
- APIs, Drivers – Jungo, GAO Research
- Debuggers – RealView, GDB, IAR

Libero IDE

- Combine System with Non-Processor Code
- Synthesis, Simulation, and Layout
- Optional Timing/Power Analysis
- Easy to use GUI

Software Program and Debug

FlashPro3 Programmer

Cortex-M1, CoreMP7, B051 Development Kits
Actel Cortex-M1 Design Flow

Hardware and Software Requirements

HARDWARE DEVELOPMENT FLOW

CoreConsole
- Select Processor
- Choose Peripherals
- Add Optional User Custom IP
- Auto Stitch to Build System

Libero IDE
- Combine System with Non-Processor Code
- Synthesis, Simulation, and Layout
- Optional Timing/Power Analysis
- Easy to use GUI

SOFTWARE DEVELOPMENT FLOW

SoftConsole
- Eclipse-Based IDE
- GNU C-Compiler
- GNU Debugger
- Memory Map and Peripheral Core Driver File Import

Industry Standard Support
- Compilers – RealView, Keil, IAR, GNU
- RTOS – uC/OS II, uClinux, Nucleus
- APIs, Drivers – Jungo, GAO Research
- Debuggers – RealView, GDB, IAR

Select Processor
Choose Peripherals
Add Optional User Custom IP
Auto Stitch to Build System

Combine System with Non-Processor Code
Synthesis, Simulation, and Layout
Optional Timing/Power Analysis
Easy to use GUI

FlashPro3 Programmer
Cortex-M1, CoreMP7, B031 Development Kits

Eclipse-Base IDE
GNU C-Compiler
GNU Debugger
Memory Map and Peripheral Core Driver File Import

OR

Software Program and Debug
STAPL Hardware Programming File

Embedded Design in FPGA with Cortex-M1
© 2008 Actel
April 2008
27
**CoreConsole - HW Development Tools**

- **SOC Builder and IP Deployment**
  - Fast assembly and configuration of user designs
  - Easy-to-use graphical user interface
  - Wide range of AMBA peripheral IP

- **CoreConsole v1.4**
  - Can be downloaded from [www.actel.com](http://www.actel.com)

- **System output as configured RTL**
  - Allows easy system setup and configuration
  - Cortex-M1 output as a blackbox

- **Automatic IP vault web update**

- **Full support for Cortex-M1**

- **Seamless integration with Libero IDE**
System-on-Chip - *Cortex-M1*

**Processor System**
- Processor
- Bus Fabric
- Components

**Components**
- Cortex-M1
- AMBA
- IP Cores

**CoreConsole**

Automatically Creates Basic System
- ... OR ... User Can Create System Manually
IP Cores Available in CoreConsole

- **Processors**
  - Cortex-M1, CoreMP7
  - Core8051s, CoreABC

- **AMBA Interfaces**
  - CoreAHB, CoreAHBLite
  - CoreAPB, CoreAPB3
  - CoreAHB2APB

- **Other Interfaces**
  - Core10/100, Core429, CorePCIF
  - Core1553BRT, Core1553BRM

- **Subsystem Cores**
  - CoreAHBNvm
  - CoreAHBSram
  - CoreAI
  - CoreCFI
  - CoreDDR
  - CoreFMEE
  - CoreFROM
  - CoreGPIO
  - CoreI2C
  - CoreInterrupt
  - CoreMemCtrl
  - CorePWM
  - CoreRemap
  - CoreSDR
  - CoreSMBus
  - CoreTimer
  - CoreUART, CoreUARTapb
  - CoreWatchdog
Building an SoC with CoreConsole

- Decide on components needed to meet system requirements.
- Add busses and bridge as necessary
- Add components
- Connect to busses
- Configure components and memory map placement.
- Generate system
- Test and Verify the system
**Cortex-M1 - CoreConsole Configuration**

- **Select Debug Interface**
  - None (Default)
  - RealView JTAG
  - FlashPro3

- **Select Die**
  - M1AFS600 (Default)
  - M1A3P1000
  - Future Project-Wide Setting

- **Other Options Inactive**
Stitching an SOC Together

- CoreConsole enables components to be stitched to the AHB and APB busses
- Components ‘advertise’ the interfaces they have available
- Auto-Stitching supported to accelerate this task
- User can add and remove individual connections
- Ad-Hoc connections are selected from drop down configuration boxes
Adhoc Signal Connections

- To connect
  - Right Click on component and click configure
  - Or click

- Label the connection

- Select ‘From’ component and pin

- Select ‘To’ component and pin

- Click connect

- Observe the connection label added to schematic
Rapid SOC Generation

- Items to be Generated are Selected in This Tab

- Output Folder Tree is $C:\text{CoreConsole}\LiberoExport\MyDesign$

- Details of the Files Output are Communicated to Libero in an XML File
  - Libero uses this to import a design

- All the Files Generated by CoreConsole Can Be Located on the Disk And Manually Edited
Actel Cortex-M1 Design Flow

Hardware and Software Requirements

**HARDWARE DEVELOPMENT FLOW**

- **CoreConsole**
  - Select Processor
  - Choose Peripherals
  - Add Optional User Custom IP
  - Auto Stitch to Build System

- **Libero IDE**
  - Combine System with Non-Processor Code
  - Synthesis, Simulation, and Layout
  - Optional Timing/Power Analysis
  - Easy to use GUI

**SOFTWARE DEVELOPMENT FLOW**

- **SoftConsole**
  - Eclipse-Base IDE
  - GNU C-Compiler
  - GNU Debugger
  - Memory Map and Peripheral Core Driver File Import

- **Industry Standard Support**
  - Compilers – RealView, Keil, IAR, GNU
  - RTOS – uC/OS II, uClinux, Nucleus
  - APIs, Drivers – Jungo, GAO Research
  - Debuggers – RealView, GDB, IAR

**FlashPro3 Programmer**

**Cortex-M1, CoreMP7, B051 Development Kits**

- Combine System with Non-Processor Code
- Synthesis, Simulation, and Layout
- Optional Timing/Power Analysis
- Easy to use GUI
Libero IDE

- Libero Project Manager
  - Manages design flow and files

- Design Creation/Verification
  - HDL, SmartGen Cores, Schematic
  - Optimization
  - Test Bench
  - Verification

- Design Implementation
  - Floor planning & physical constraint
  - Place & Route
  - Timing constraints & analysis
  - Power analysis
  - Program file generation

- Programming and Debug
  - FlashPro3 supports Programming for all Fusion devices
Libero IDE

Design Entry Tools

Design Hierarchy View

Catalog:
- Configurable Cores
- HDL Templates
- Macros
- Bus Interfaces

Interactive Design Flow Management Tools

Log File
Embedded Design in FPGA with Cortex-M1

Libero IDE - Project Manager

Catalog:
- HDL Templates
- Simple click to insert into HDL code
- Proven/tested

File Manager View
Synthesize with Synplicity’s Synplify AE

- Synplify AE
  - Leading edge synthesis from the market leader
  - Close OEM partnership provides optimal benefit to Actel users
  - Integration with Libero IDE ensures seamless operation
  - Optimized performance and area utilization for all Actel FPGAs
  - Available in Free Libero Gold
Mentor Graphics ModelSim HDL Simulator

- HDL simulation in VHDL or Verilog
  - Pre-synthesis simulation
  - Post-synthesis simulation
  - Post-layout simulation

Simulate
Libero IDE - Designer User Interface

Physical Implementation Tools

Constraint & Analysis Tools

Log & Device Information

Design Flow:
- Compile
- Layout
- Deck Annotate
- Programming File

Multiview Navigator:
- Netlist Viewer
- PinEditor
- ChipPlanner
- I/O Attribute Editor

SmartTime:
- Constraint Editor
- Timing Analyzer
- Smart Power

Log Information:
- The Export-map command succeeded (00:01:32)
- Wrote to the file: c:\counter16.stp
- The Generate programming file command succeeded (00:01:36)
Designer Functions

- Import Netlist, Compile, and Design Rule Check
- Floor planning and physical constraints
- Timing driven Place and Route
- Back annotated timing for full timing simulation
- SmartTime setup of Timing Constraints and Timing Analysis
- SmartPower analysis of power consumption
- Generate bitstream or STAPL programming files
- Comprehensive log file and reports
Programming Software

- FlashPro
  - In System Programming (ISP) for Actel Flash devices
    - Supports all FlashPro hardware programmers
  - Includes ChainBuilder
    - Generates a merged STAPL file for programming Actel FLASH devices in a mixed IC environment

- Silicon Sculptor
  - Supports all Actel devices
    - Use with Silicon Sculptor hardware programmers
  - Launch from Libero Project Manager or stand alone

Device Debugging

- Synplicity Identify Debugger
Actel Cortex-M1 Design Flow

Hardware and Software Requirements

HARDWARE DEVELOPMENT FLOW

CoreConsole
- Select Processor
- Choose Peripherals
- Add Optional User Custom IP
- Auto Stitch to Build System

Libero IDE
- Combine System with Non-Processor Code
- Synthesis, Simulation, and Layout
- Optional Timing/Power Analysis
- Easy to use GUI

SOFTWARE DEVELOPMENT FLOW

SoftConsole
- Eclipse-Base IDE
- GNU C-Compiler
- GNU Debugger
- Memory Map and Peripheral Core Driver File Import

Industry Standard Support
- Compilers – RealView, Keil, IAR, GNU
- RTOS – uC/OS II, uClinux, Nucleus
- APIs, Drivers – Jungo, GAO Research
- Debuggers – RealView, GDB, IAR

OR

STAPL Hardware Programming File
FlashPro3 Programmer
Cortex-M1, CoreMP7, B051 Development Kits
**Software development environment**
- Eclipse-based IDE - easy user interface
- Supports Cortex-M1, CoreMP7, Core8051/s
- Can be downloaded from [www.actel.com](http://www.actel.com)

**C/C++ programming and debug**
- CodeSourcery G++ ARM tools
- SDCC 8051 compiler
- Programming and debug with Actel’s FlashPro3

**Can import existing code**
- Open platform for application development

**Support for RTOS and stacks**
- uC/OC, uClunix
- TCP/IP, USB, IPMI
SoftConsole GNU C/C++ Compiler

- Extensive intelligent ARM optimization
  - Built from CodeSourcery G++ GNU/GDB

- Includes many features useful for embedded systems
  - Powerful inline assembly syntax
  - Comprehensive linker script language permitting exact placement of code and data

- Large developer base results in tool stability

- ISO C and C++ language support
  - Complete runtime libraries
  - Aggressive code usage analysis and syntax warnings
  - Supports ARM EABI for better portability
Support for source- and assembly-level debugging

Live debugging of new code
- In an FPGA or in the GDB ARM simulator

Breakpoints can occur when certain conditions are met

Intelligent access to hardware
- Register banks and memory ranges
- Hover over a variable to read its current value
- Changes in value are obvious for any variable, memory or register
- Current stack frame displayed while debugging

Evaluation of expressions at runtime
Actel Cortex-M1 Design Flow

Hardware and Software Requirements

**HARDWARE DEVELOPMENT FLOW**

**CoreConsole**
- Select Processor
- Choose Peripherals
- Add Optional User Custom IP
- Auto Stitch to Build System

**Libero IDE**
- Combine System with Non-Processor Code
- Synthesis, Simulation, and Layout
- Optional Timing/Power Analysis
- Easy to use GUI

**SOFTWARE DEVELOPMENT FLOW**

**SoftConsole**
- Eclipse-Base IDE
- GNU C- Compiler
- GNU Debugger
- Memory Map and Peripheral Core Driver File Import

**Industry Standard Support**
- Compilers – RealView, Keil, IAR, GNU
- RTOS – uC/OS II, uClinux, Nucleus
- APIs, Drivers – Jungo, GAO Research
- Debuggers – RealView, GDB, IAR

**STAPL Hardware Programming File**

**FlashPro3 Programmer**

**Cortex-M1, CoreMP7, 8051 Development Kits**
On-Chip Debugging via FlashPro3

- **Download and debug executable programs to development boards using FlashPro3**
  - Can program and debug processor memory and FPGA fabric with FlashPro3
  - Reduces pin-count – Utilizes dedicated FPGA JTAG pins via UJTAG versus GPIO RVI-ME configuration (10-pins)

- **Full debugging of code on remote target**
  - View internal registers, memory locations, variables, etc.

- **Uses the same interface as Instruction Set Simulator**
  - Only one tool to learn
**Actel M1 Development Boards**

- **M1-SYSGMT-DEV-KIT**
  - M1AFS600 device
  - System Mgmt GUI
  - Demonstration designs
  - Full PCI interface
  - Also available with FP3

- **M1 SOC Boards**
  - Powered by USB or wall supply
  - 1M SRAM, 4M Flash memory
  - FP3 programmer built into board
  - Expansion connectors

- **M1A3P-DEV-KIT-SCS**
  - M1A3P1000 device

- **M1AFS-DEV-KIT-SCS**
  - M1AFS600 device

- **M1AGL-DEV-KIT-SCS**
  - M1AGL600 device
Cortex-M1 in Fusion FPGA
ATCA IPMC/carrierIPMC

- ATCA Blade and ATCA AMC Carrier reference designs using Cortex-M1
  - IPMC Blade and AMC Carrier (carrier-IPMC) designs running on benchtop development board using ARM Cortex-M1 processor
  - Serial-over-LAN support for Intel 82575/1, Broadcom BCM5714C network controllers in all four modes (NC-SI, DE, SPT, PT)
  - Firmware proven in high reliability applications
    - Firmware in C and built with GCC and GNU tools in SoftConsole
    - Firmware upload over IPMB-0 or the serial port is implemented
  - Supports all mandatory and a wide range of optional IPMI commands

- In development now
  - Fusion AMC carrier needs 1/3 less board area than H8S design (1,167mm² vs 1,748mm²)
  - Available Q3 2008
Where to Go for More Information

- **Cortex-M1 on the Web**

- **Key Documents**
  - Cortex-M1 Handbook (*Actel*)
  - ARMv6-M Architecture Reference Manual (*ARM*)
  - Cortex-M1 Technical Reference Manual (*ARM*)
  - uC/OS-II Datasheet (*Micrium*)
Summary

- **Cortex-M1** allows designers to benefit from hassle-free, industry-standard ARM architecture
  - Optimized for use in M1 devices (ProASIC3, IGLOO, Fusion)

- A huge number of tools are available like Micrium’s uC/OS-II RTOS to simplify design

- **Actel FPGA tools** offer seamless development flow
  - CoreConsole, Libero, dev kit hardware development tools
  - SoftConsole with GNU software development tools
  - M1 development boards

- Brings flexibility and fast time to market to system-level designs