Embedded Processing Using FPGAs
Agenda

• Why FPGA Platform Based Embedded Processing
• Embedded Use Models And Their FPGA Based Solutions
• Architecture/Topology Choices
• A Reoccurring Question: Hardware Or Software
• Reconfigurable Hardware
• Tool Flows For FPGA Based Embedded Systems
Agenda

• Why FPGA Platform Based Embedded Processing
  • Embedded Use Models And Their FPGA Based Solutions
• Architecture/Topology Choices
• A Reoccurring Question: Hardware Or Software
• Reconfigurable Hardware
• Tool Flows For FPGA Based Embedded Systems
Why Use Processors In the First Place

• Microcontrollers (µC) and Microprocessors (µP) Provide a Higher Level of Design Abstraction
  – Most µC functions can be implemented using VHDL or Verilog - downsides are parallelism & complexity
  – Using C/C++ abstraction & serial execution make certain functions much easier to implement in a µC

• Discrete µCs are Inexpensive and Widely Used
  – µCs have years of momentum and software designers have vast experience using them
Why Embedded Design using FPGAs

In Addition To The Universal Drive Towards Smaller Cheaper Faster With Less Power….

1. Difficult to Find the Required Mix of Peripherals in Off the Shelf (OTS) Microcontroller Solutions
2. Selecting a Single Processor Core with Long Term Solution Viability is Difficult at Best
3. Without Direct Ownership of the Processing Solution, Obsolescence is Always a Concern
4. Many Microprocessor Based Solutions Provide Limited On-Chip Peripheral Support
Rarely the Ideal Mix

1. Difficult to Find the Required Mix of Peripherals in Off the Shelf (OTS) Microcontroller Solutions
   - Even with variants, compromises must still be made

   System Requirements
   - 2@ UART
   - TIMER
   - I2C
   - SPI
   - GPIO
   - FLASH
   - DDR SDRAM

   Microcontroller #1
   - UART
   - FLASH
   - GPIO
   - UART
   - Timer
   - Lacks I2C & Includes RAM vs DDR SDRAM

   Microcontroller #2
   - UART
   - FLASH
   - SPI
   - Lacks a Second UART & Includes Unnecessary IP

   - Must decide between inventory impact of multiple configurations or device cost of comprehensive IP
Changing Requirements

Selecting a Single Processor Core with Long Term Solution Viability is Difficult at Best
– Future proofing system requirements is difficult at best

Changing processor cores to accommodate new requirements consumes valuable design resources
Here Today, Gone Tomorrow

Without Direct Ownership of the Processing Solution, Obsolescence is Always a Concern

- A single core is used to create a family of µCs

- The sheer number of µC configurations can lead to obsolescence of lower volume configurations/variants
Many Microprocessor Based Solutions Provide Limited On-Chip Peripheral Support

- Manufacturers create I/O devices to extend the functionality of the base processor core

- An external, pre-defined interface between a µP and its support device limits overall system performance
FPGA Embedded Design

1. FPGAs Allow For The Implementation Of An Ideal Mix Of Peripherals And System Infrastructure
2. New System Requirements Can Be Supported Without Changing Core Design
3. Longevity Of FPGAs Approaches The Longest Available Microcontrollers In The Market
4. FPGAs Are Used To Augment µp Functionality - Absorbing The Core Is The Next Natural Step
Agenda

• Why FPGA Platform Based Embedded Processing

• Embedded Use Models And Their FPGA Based Solutions

• Architecture/Topology Choices

• A Reoccurring Question: Hardware Or Software

• Reconfigurable Hardware

• Tool Flows For FPGA Based Embedded Systems
Processor Use Models

State Machine
- Lowest Cost, No Peripherals, No RTOS & No Bus Structures
- VGA & LCD Controllers
- Low/High Performance

Microcontroller
- Medium Cost, Some Peripherals, Possible RTOS & Bus Structures
- Control & Instrumentation
- Moderate Performance

Custom Embedded
- Highest Integration, Extensive Peripherals, RTOS & Bus Structures
- Networking & Wireless
- High Performance

Range of Use Models
Xilinx Processor Solutions

1. State Machine
2. Microcontroller
3. Custom Embedded

Range of Solutions
PicoBlaze 8-Bit Microcontroller Reference Design

- Free PicoBlaze macro for implementing complex state machines and micro sequencers
- Supports Spartan-3, Virtex-4, Virtex-II/Pro FPGAs and CoolRunner-II CPLDs
- Minimal logic size
  - Only 96 Spartan-3 slices, 5% of XC3S200 device
- High performance
  - Up to 44 MIPS in Spartan-3 and 100 MIPS in Virtex-4
- 100% embedded capability
  - Everything in FPGA – no external components required

http://www.xilinx.com/picoblaze
- 3-Stage Parallel Pipeline
- Optional Instruction and Data Caches – 2KB to 64KB Each
- Optional Multiply Unit
- 32 x 32-bit GPRs
- Dedicated Fast Simplex Links - FSL
  - 32-bit, Point-to-Point Links between GPRs in the Register File and User Logic
  - 8 Input & 8 Output
  - Blocking & Non-Blocking Transfers – Predefined Insts
- Local Memory Bus Interfaces - LMB
  - Provide Access to Inst/Data Memory Without Bus Access
- 100 MHz+ On-Chip Peripheral Bus
  - 32-bit Semi-Duplex Bus
- Fast access “Cache Link” interface
PowerPC Processor Block

- 10/100/1000 EMAC Core
- Instruction Side On Chip Memory Interface
- ISOCM Control
- PowerPC 405 Core
- APU Control
- Processor Local Bus Interface
- Auxiliary Processor Unit
- Host
- Statistics
- EMAC
- DCR I/F
- 10/100/1000 EMAC Core
- ISOCM
- Processor Block
- DCR Control
- Reset, Debug
- Data Side On Chip Memory Interface
- Device Configuration Register Interface
- 10/100/1000 EMAC Core
- Host
- Statistics
- EMAC
- DCR I/F
Integrating a PowerPC 405

- **IP-Immersion™**
  - Advanced Technology Metal "Headroom" Enables True IP Immersion
  - IP-Immersion Tiles Provide Direct Connectivity to Virtex-II Pro Fabric

- **ActiveInterconnect™**
  - Segmented Routing Enables High Bandwidth Interface
  - General Purpose Routing Passes Over Hard IP (Hex & Long)

Replaces only 1,024 LUTs and 8 Block RAMs
UltraController

Simple processor/fabric interface uses minimal FPGA resources

Code stored in block RAM

JTAG
gpio_in
gpio_out

Easy to use 5-port HDL module

sys_clk
sys_rst
UltraController II

- Interrupt
- sys_clk
- sys_rst_out
- sys_rst
- JTAG

Easy to use HDL module

Code Loaded and stored in Cache

Operates at maximum CPU frequency

Simple processor/fabric interface uses minimal FPGA resources

PowerPC 405

Operates at maximum CPU frequency
Virtex 5 Processor Block

- 5 x 2 Crossbar Connection (128-bit) including Arbiters
  - Configure through bit-stream or DCR
  - Up to 1:1 freq w/ processor
- 1 PLB master interface to FPGA fabric
- 2 PLB slave interfaces from FPGA fabric
- PLB Master/Slave ports
  - operate at diff freq
- 4 Thirty-two bit DMA Channels
- Memory Ctrl Interface to connect soft logic based Memory controller
- Auxiliary Processing Unit Controller
  - 128 bit interface, pipelining supported
- DCR Controller Module
- CPM and Control Module
The System Infrastructure

Bus infrastructure ranges from logic efficient to maximum bandwidth

**Bi-directional Bus**
Single channel of communication, least amount of logic required and moderate routing requirement

**Centrally Arbitrated**
Full duplex communication, moderate amount of logic required and minimal routing required

**Slave-side Arbitration**
N-channels of communication, most expensive logic requirement and extensive routing required
System Peripherals - IP

Slower peripherals such as I/O functions are typically placed on secondary bus structures.

High speed memory and system interfaces are typically placed on primary bus structures.

Infrastructure
I/O Based Peripherals
Memory Interfaces
High Level Functions
High Levels of Integration

Instruction

Data

Arbiter

Processor Local Bus - PLB

Memory Controller

PCI 32/33

PCI 32/33

10/100/1G Ethernet

Memory

10/100 Ethernet

GPIO

UART

User Peripheral

IPIF (master or slave)

Data Control Register Bus - DCR

BRAM

User Logic

System Reset

JTAG Block

10/100/1G Ethernet

Full System Customization & High Performance
Complete Customization

Instruction-Side Local Memory Bus
- System Reset
- JTAG Block

Instruction LMB

Dual Port Block RAM

Data-Side Local Memory Bus
- DDR SDRAM
- ZBT SSRAM
- SDRAM

Data LMB

Processor Local Bus - PLB
- Memory Controller
- UART1
- UART 7
- Ethernet or CAN
- SPI
- IPIF

Arbiter

User Peripheral

FPGA Fabric
Gigabit System Reference Design (GSRD)

- Designed for maximum TCP/IP performance with the PowerPC 405
- GSRD building blocks
  - LocalLink Gigabit Ethernet MAC
  - Communications DMA Controller
  - Multi Port Memory Controller
- High Performance TCP transmit
  - Over 900Mbps (with Treck TCP/IP stack)
- Applications
  - High performance bridging
  - TCP/IP and user data interfaces
  - IP over GigE Ethernet Switching
  - High Speed Remote Monitoring
  - Remote Image/Video Capture
Multiported Memory Controller
Today You Have Multiple Topology Choices

MicroBlaze or PowerPC

- Local Memory
- Custom Coprocessors
- Multi Port Memory Controller
- GPIO
- CAN/MOST
- PCI
- Custom I/O Peripherals
- Ethernet MAC
- Interrupt Controller
- Timer/PWM
- I²C/SPI
- UART
- GPIO
- USB 2.0
- Generic Peripheral Controller

Virtex or Spartan FPGA

- Debug
- Trace
- JTAG

Point-to-Point Connections for higher performance

Shared Bus for smaller area
Multicore Topologies

- Memory Map. SW responsibility for non-conflicting resource partition
- Pass small sized messages between a sender and a receiver
- Use synchronous-polling or async interrupts
- You can still use shared memory techniques with BRAM

Diagram:
- Processor Subsystem 1
  - Other XPS Peripherals UART, GPIO, TIMER
- Processor Subsystem 2
  - Other XPS Peripherals UART, GPIO, TIMER

Shared external memory
- MPMC (External Memory)
  - Data_IRQ_A
  - Data_IRQ_B

Shared BRAM memory
- XPS Mailbox Core
  - Mutex array

Private boot memory
- XPS BRAM

Partitioning clocking domains
- Treatment of CPU and system reset

 CPUs with shared resources need to sync.
- Configure number of mutex registers to test and set
Virtex 5 Processor Block

- 5 x 2 Crossbar Connection (128-bit) including Arbiters
  - Configure through bit-stream or DCR
  - Up to 1:1 freq w/ processor
- 1 PLB master interface to FPGA fabric
- 2 PLB slave interfaces from FPGA fabric
- PLB Master/Slave ports
  - operate at diff freq
- 4 Thirty-two bit DMA Channels
- Memory Ctrl Interface to connect soft logic based Memory controller
- Auxiliary Processing Unit Controller
  - 128 bit interface, pipelining supported
- DCR Controller Module
- CPM and Control Module
Building a System

Significant reduction in amount of FPGA logic required to build a system using integrated functionality of Virtex-5 Processor Block
Agenda

• Why FPGA Platform Based Embedded Processing
• Embedded Use Models And Their FPGA Based Solutions
• Architecture/Topology Choices
• The Reoccurring Question: Hardware Or Software
• Reconfigurable Hardware
• Tool Flows For FPGA Based Embedded Systems
Two Ways to meet Performance Requirements

Brut Force

Intelligent

5 GHz

μC

5 MHz

CPU

FPGA

APU or FSL

HW Accelerator
Co-Processor Interfacing

Conventional approach

μC

Bus Interface

FPGA, ASIC or ASSP

Xilinx Solutions

μC

FSL

Co-Processor / Peripheral

• Fast
• Deterministic
• Easy implementation

APU

Co-Processor / Peripheral

Xilinx Solutions

Fast

Deterministic

Easy implementation
Conventional vs. APU Implementations

Benefits
- Less overhead logic
- Faster transfer
- Efficient software implementation

Overhead required to connect soft logic
Bus Cycle Improvement via APU

Reduce number of bus cycles by factor of 10 by using APU
Up to 13x Performance Boost

**PPC 405 Single Precision FPU**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Performance</th>
<th>Speed Up</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Software Emulation of Floating Pt *</td>
<td>Single Precision FPU</td>
</tr>
<tr>
<td>FIR Filter</td>
<td>846.09 ms</td>
<td>63.94 ms</td>
</tr>
<tr>
<td>Video Editing Algorithm</td>
<td>248.38 ms</td>
<td>27.93 ms</td>
</tr>
<tr>
<td>PID Loop</td>
<td>1.66 us</td>
<td>0.37 us</td>
</tr>
<tr>
<td>1024pt FFT</td>
<td>19.48 ms</td>
<td>5.42 ms</td>
</tr>
</tbody>
</table>

(*C-Code is compiled using IBM Performance Libs delivered with EDK 8.2i)*
Floating Point Accelerated by APU

- Algorithm description - FPU w/ FIR filter
- PowerPC operating frequency – 400 MHz
- APU with direct interface to fabric - XtremeDSP blocks

APU improves algorithm execution 20X over software emulation
Accelerated MPEG De-Compression

Example: Video application - IDCT MPEG de-compression

- Leverages Virtex-4 Integrated Features
  - PowerPC, APU, XtremeDSP Slice
  - Software Emulation – 13000 IDCTs/sec
- HW acceleration over software
  - Lower latency and high bandwidth
- Efficient HW/SW design partitioning
  - Optimized implementation
- Significant performance increase
  - 20x improvement over software
  - Utilizing APU and ExtremeDSP
Tailor the System to Achieve Performance
Take MP3 Decoding with Custom Hardware Logic

- 100MHz MicroBlaze™, pure software = 146 seconds
- 100MHz MicroBlaze™ + FSL + LL MAC = 9 seconds
- 100MHz MicroBlaze™ + FSL + DCT + IMDCT + LL MAC = 7 seconds

Performance Improvement

Note: MicroBlaze™ v4.00 core, ML40x board, 100MHz system clock, EDK8.1
An Example: MicroBlaze™ FPU in Motor Control

- Design implications:
  - You will get more done for a given clock frequency
  - Reducing the clock frequency (if possible) reduces power dissipation

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Function Calls/Second</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>MicroBlaze™ without FPU</td>
<td>Over 5,300</td>
<td>1X</td>
</tr>
<tr>
<td>MicroBlaze™ with FPU</td>
<td>Over 82,000</td>
<td>Over 15X</td>
</tr>
</tbody>
</table>

Benefit: The customer now have the extra processor bandwidth for use on other things.

Extra cycles needed for specialized peripherals

Note: Used MicroBlaze™ v4.00 core on Spartan 3S1500.
Auxiliary Processing Solutions

Hard Processor Block

Fabric Control Module Interface

Floating Point Unit

UltraController®V4

FSL

C to HDL

Impulse Technologies

Poseidon Design Systems

Celoxica Solution

PLB

OCM

405 Core

APU Control

Auxiliary Processor

FPGA Fabric

Application Specific

General Solution
C to HDL Tools

- Create Code
- Profile Code
- Identify Critical Code Segments
- Translate Critical Code to HDL
- Build HW Accelerators from HDL
- Replace Critical Code with Calls to HW Accelerators
Impulse C Programming Model

- System of Independent sequential execution units
  - Called Processes
  - Can be executed in HW or SW and run concurrently
- Processes communicate with each other
  - Via self-synchronizing FIFO buffers called streams, or
  - Via shared memories
Impulse C-HDL Tool Flow

Develop and debug the application using standard C tools such as Visual Studio or gcc/gdb.

Use the Impulse C tools to generate synthesizable hardware and APU interfaces for specific C-language subroutines.

Combine the generated hardware and software IP with other components in Platform Studio. Generate a bitmap and download to the target board.

<table>
<thead>
<tr>
<th>Application</th>
<th>PowerPC only (300MHz)</th>
<th>PowerPC/APU (300/50MHz)</th>
<th>Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Filter (512X512 image)</td>
<td>.1414 sec</td>
<td>0.0124 sec</td>
<td>11.4X</td>
</tr>
<tr>
<td>Encryption (8M characters)</td>
<td>2.257 sec</td>
<td>0.0667 sec</td>
<td>33.84 X</td>
</tr>
<tr>
<td>Fractal image (1K max iterations)</td>
<td>119 sec</td>
<td>6 sec</td>
<td>19.83 X</td>
</tr>
</tbody>
</table>
Agenda

- Why FPGA Platform Based Embedded Processing
- Embedded Use Models And Their FPGA Based Solutions
- Architecture/Topology Choices
- A Reoccurring Question: Hardware Or Software
- Reconfigurable Hardware
- Tool Flows For FPGA Based Embedded Systems
What is Partial Reconfiguration?

• Dynamically modify blocks of logic in an FPGA by downloading partial bit files while unmodified logic continues to operate

• Logic is LUTs, FFs, BRAMs, DSP blocks, MGTs...
PR Applications Today

- JTRS Software Defined Radio
  - SWaP and multi-channel concurrency

- Microwave Communications
  - Flexibility and extended functionality

- Cryptography
  - Physical isolation in one FPGA

\[ \text{Ciphertext (CT)} \quad \rightarrow \quad \text{Plaintext (PT)} \]
SDR Development Platform

Core Connect

- Interpolation filter
- TX NCO
- D/A Interface (14 bits -- 300 MHz)

Ethernet interface

- Decimation filter
- RX NCO
- A/D Interface (12 bits -- 200 MHz)

Waveform combiner

- WAVEFORM A
- WAVEFORM B

Ethernet Switch

DAC

ADC

- RJ45 – 100BT (Data)
- RJ45 – 100BT (Voice)
- RJ45 – 100BT (Ctrl)

www.xilinx.com 51 - Embedded Processing using FPGAs
PR Application: Dynamic Network Interface Protocol Switching
Partial Reconfiguration Applications

- Reduce FPGA size by multiplexing FPGA hardware resources
Agenda

• Why FPGA Platform Based Embedded Processing
• Embedded Use Models And Their FPGA Based Solutions
• Architecture/Topology Choices
• A Reoccurring Question: Hardware Or Software
• Reconfigurable Hardware

• Tool Flows For FPGA Based Embedded Systems
Necessary Tools

• A Full Complement of Tools are Required to Design an Embedded Processor System
  – Processor System Generation
  – Hardware Implementation & Software Compilation
  – Hardware & Software Debug

• HW Implementation & SW Compilation are the two Main Flows that Must be Addressed
  – The Embedded Flows Should Mirror Traditional Flows
Tools are needed to create the system infrastructure, connect peripherals and configure the peripherals as desired.

Once the system is defined, the hardware can be implemented and software compiled.

- Databits = 8
- Baudrate = 9600
- Parity = Off
Co-processor Creation

Auto-generated HW & SW Templates

```plaintext
BEGIN feb_v10
  PARAMETER INSTANCE = feb_v10
  PARAMETER HW_VER = 1.00 a
  PARAMETER C_RST_RESET HIGH = 1
  PORT SYS_RST = sys_bus_reset
  PORT PCU_CLK = sys_clk
END

BEGIN febfsal_bridge
  PARAMETER INSTANCE = febfosal_bridge
  PARAMETER HW_VER = 1.00 a
  PARAMETER C_FSR_LINKS = 3
  BUS_INTERFACE NSIS = fcb_v10
  BUS_INTERFACE NSIF = ppc405_virtex4_to_pcpu_corel
  BUS_INTERFACE NSIL = pcpu_corel_to_ppc405_virtex4
  PORT PCU_CLK = sys_clk
END

BEGIN fsl_v20
  PARAMETER INSTANCE = ppc405_virtex4_to_pcpu_corel
  PARAMETER HW_VER = 1.00 a
  PARAMETER C_RST_RESET HIGH = 1
  PORT SYS_RST = sys_bus_reset
END

BEGIN apu_corel
  PARAMETER INSTANCE = apu_corel
  PARAMETER HW_VER = 1.00 a
  BUS_INTERFACE NSIS = pcpu_corel_to_ppc405_virtex4
  BUS_INTERFACE NSIF = ppc405_virtex4_to_pcpu_corel
  PORT PCU_CLK = sys_clk
END

BEGIN fsl_v20
  PARAMETER INSTANCE = pcpu_corel_to_ppc405_virtex4
  PARAMETER HW_VER = 1.00 a
  PARAMETER C_RST_RESET HIGH = 1
  PORT SYS_RST = sys_bus_reset
END

BEGIN PROC
  - PARAMETER DRIVER_NAME = pcpu
  - PARAMETER DRIVER_VER = 1.00 a
  - PARAMETER HW_INSTANCE = ppc405_virtex4
  - PARAMETER ARCHIVER = powerpc-eabi
END

BEGIN DRIVER
  - PARAMETER DRIVER_NAME = generic
  - PARAMETER DRIVER_VER = 1.00 a
  - PARAMETER HW_INSTANCE = fcb_v10
END

BEGIN DRIVER
  - PARAMETER DRIVER_NAME = apu_corel
  - PARAMETER DRIVER_VER = 1.00 a
  - PARAMETER HW_INSTANCE = apu_corel
END
```

Include "spu_corel.h"

```c
#include "parameters.h"

// Following is an example driver function that is called in the main function.
// This example driver writes all the data in the input arguments into the input FSL bus through blocking writes. FSL peripheral will
// automatically read from the FSL bus. Once all the inputs have been written, the output from the FSL peripheral is read
// into output arguments through blocking reads.

void apu_corel_app()
{
  // Instance name specific Macros. Defined for each instance of the peripheral.
  // 
  // #define WRITE_APU_COREL0_INPUT0(x) write_int_APU_COREL0_INPUT0(x)
  // 
  // #define READ_APU_COREL0_INPUT0(x) read_int_APU_COREL0_INPUT0(x)

  unsigned int* input0, /* Array size = 8 */
  unsigned int* output0 /* Array size = 8 */
  }
```
Hardware Implementation

Source Code → Synthesis → System Netlist → Build & Map → Merged & Mapped Design → Place & Route → Memory Map for Internal Code Storage → Place & Route → Configuration File

VHDL or Verilog

Standard FPGA

HW Development Flow

Simulation/Synthesis

Build & Map

Place & Route

Configuration File
Eclipse Based SDK for the Software Savvy Engineer

Powerful Software Development
- Eclipse open source Software IDE allows an industry standard method for seamless tool integration
- Consists of:
  - Perspectives
  - Views
  - Editors
- Environment window displays one or more perspectives
  - A perspective contains views (like the Navigator) and editors
Software IP

- Device drivers are provided for each hardware device
- Device drivers are written in ANSI C and are designed to be portable across processors
- Device drivers allow the user to select the desired functionality to minimize the required memory
- Device drivers in all layers have common characteristics and API
Board Support Package

Board Support Packages (BSPs) are collections of parameterized drivers for a specific processor system.

libxil.a

XEmac_mWriteReg ( ... );
XEmac_SendFrame ( ... );
...
XSpi_mSendByte ( ... );
XSpi_GetStatusReg ( ... );
...
XTmrCtr_mWriteReg ( ... );
XTmrCtr_IsExpired ( ... );
...
XUartLite_SendByte ( ... );
XUartLite_GetStats ( ... );
...

www.xilinx.com
Software Compilation

- C Code
- C/C++ Cross Compiler
- Assembler
- Linker
- Standard Embedded SW Development Flow

Flow Diagram:
- Compiler receives Source Code
- Assembler receives Source Code
- Linker receives Relocatable Object Code and Mapfile and Linker Script
- Machine Code is the final output
System Design Flow

**Standard Embedded SW Development Flow**
- Code Entry
- C/C++ Cross Compiler
- Linker
- Load Software Into FLASH
- Debugger

**Embedded Developers Kit**
- Board Support Package
- System Netlist
- Data2MEM
- Compiled ELF
- Compiled BIT
- Download Combined Image to FPGA
- RTOS, Board Support Package
- HDL Entry
- Simulation/Synthesis
- Implementation
- Download Bitstream Into FPGA
- Chipscope

**Standard FPGA HW Development Flow**
- VHDL or Verilog

67 - Embedded Processing using FPGAs
HW & SW Debug

Internal & external logic analysis is used to determine where problems reside.

Integrated Bus Analyzer

SW debug typically consists of JTAG based runtime control of the target processor core.

External Connection
“Platform Debug” Integrated HW/SW Debuggers

- Cross Trigger HW and SW Debuggers to Find and Fix Bugs Faster!
- Enable better insight into the HW / SW code dynamics
Key Messages

• FPGA Based Embedded Systems Solve Classic Integration Issues (Size, Cost, Power)
• They Also Creating Novel Solutions Not Easily Realized Using Other Technologies
• FPGA Platforms Enable Flexible and Dynamic Systems
• Intelligent Tools Accelerate Development And Enable You To Optimize The Balance Of Feature Set, Performance, Area & Cost Of Your Design
Q&A