Designing a Multi-Processor based system with FPGAs

Frank de Bont
Trainer / Consultant
Why Use Processors?

- 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
µP versus µC

A microprocessor (µP) is just one component of many in a complex system of digital & analog I/O

Most simple system components are contained completely within a microcontroller (µC)
Rarely the Ideal Mix

- Difficult to find the required mix of peripherals in Off the Shelf (OTS) microcontroller solutions

Today System Requirements
- UART
- USB
- TIMER
- Ethernet
- SPI
- GPIO
- FLASH
- DDR

Microcontroller #1
- Lacks Ethernet & Includes RAM vs DDR

Microcontroller #2
- Lacks USB & Includes Unnecessary IP
Difficult to find the required mix of peripherals in Off the Shelf (OTS) microcontroller solutions

Selecting a single processor core with long term solution viability is difficult at best

Today's System Requirements
- 1 GHz
- UART
- USB
- TIMER
- 10/100 Ethernet
- SPI
- GPIO
- FLASH
- DDR3

Meets current system requirements

Future System Requirements
- 2.4 GHz
- UART
- USB
- TIMER
- 10/100/1000 Ethernet
- SPI
- GPIO
- FLASH
- DDR4

?
Here Today, Gone Tomorrow

- Difficult to find the required mix of peripherals in Off the Shelf (OTS) microcontroller solutions
- Selecting a single processor core with long term solution viability is difficult at best
- Without direct ownership of the processing solution, obsolescence is always a concern
Difficult to find the required mix of peripherals in Off the Shelf (OTS) microcontroller solutions

Selecting a single processor core with long term solution viability is difficult at best

Without direct ownership of the processing solution, obsolescence is always a concern

Many microprocessor based solutions provide limited On-Chip peripheral support
Embedded Design with FPGAs

- FPGA allow for the implementation of an ideal mix of peripherals and system infrastructure
- New system requirements can be supported without changing the processor core
- Longevity of FPGAs approaches the longest available microcontrollers in the market
- FPGAs are used to augment µP functionality absorbing the core is the next natural step
Softcore Processor System

Processor Core

System Infrastructure

System Peripherals

BRAM

ILMB Bus

clock

reset

DLMB Bus

AXI Interconnect

interrupts

FLASH

DDR

Timer

UART

GPIO

MicroBlaze

BRINGING YOU THE NEXT LEVEL IN EMBEDDED DEVELOPMENT
Hybrid Processor System

Processor Cores

- iMX6
- DDR3

System Infrastructure

- Clock
- Reset
- Interrupts

- eMMC
- QSPI
- DDR3

System Peripherals

- UART
- USB
- RS-422
- HDMI
- Ethernet
- UART
- USB
- Camera Link
- Motor Control

Ethernet Switch

CVBS / HD-SDI
Hybrid Processor System

Processing System

Static Memory Controller
Quad-SPI, NAND, NOR

Dynamic Memory Controller
DDR3, DDR2, LPDDR2

AMBA® Switches

ARM® CoreSight™ Multi-core & Trace Debug

NEON™/FPU Engine

Cortex™-A9 MPCore™
32/32 KB I/D Caches

512 KB L2 Cache

Snoop Control Unit (SCU)

Timer Counters

General Interrupt Controller

DMA

Configuration

AMBA® Switches

2x SPI

2x I2C

2x CAN

2x UART

GPIO

2x SDIO with DMA

2x USB with DMA

2x GigE with DMA

I/O MUX

Multi-Standards I/Os (3.3V & High Speed 1.8V)

System Peripherals

EMIO

XADC

S_AXI_GP0/1

M_AXI_GP0/1

Multi-Gigabit Transceivers

Programmable Logic:
System Gates, DSP, RAM

S_AXI_HP0

S_AXI_HP1

S_AXI_HP2

S_AXI_HP3

S_AXI_ACP

Multi-Standards I/Os (3.3V & High Speed 1.8V)
FPGA based Processor System

Processor Core

System Peripherals

eMMC
QSPI
DDR4

UART
USB
RS-422
HDMI
Ethernet
Camera Link
Motor Control
CVBS / HD-SDI

clock
reset
interrupts
FPGA based Processor System

Processing System

Application Processing Unit
- ARM® Cortex™-A53
  - 32 KB I-Cache w/Parity
  - 32 KB D-Cache w/ECC
  - Memory Management Unit
  - Embedded Trace Macrocell
- NEON™ Floating Point Unit
  - 32 KB I-Cache w/Parity
  - 32 KB D-Cache w/ECC
  - Memory Management Unit
  - Embedded Trace Macrocell

Memory
- DDR4/3/3L, LPDDR4/3 ECC Support
- 256 KB OCM with ECC
- 1 MB L2 w/ECC

Graphics Processing Unit
- ARM Mali™-400 MP2
  - Geometry Processor
  - Pixel Processor
  - Memory Management Unit
  - 64 KB L2 Cache

Real-Time Processing Unit
- ARM Cortex™-R5
  - 128 KB TCM w/ECC
  - 32 KB I-Cache w/ECC
  - 32 KB D-Cache w/ECC
- GIC

Platform Management Unit
- System Management
- Power Management
- Functional Safety

Configuration and Security Unit
- Config AES, Decryption, Authentication, Secure Boot
- Voltage/Temperature Monitor
- TrustZone

System Functions
- Multi-channel DMA
- Timers, WDT, Resets, Clocking, & Debug

Programmable Logic

Storage & Signal
- Block RAM
- UltraRAM
- DSP

General-purpose I/O
- High-Performance I/O
- High Density (Low Power) I/O

High-Speed Connectivity
- 100G EMAC
- PCIe® Gen4

High-Speed Connectivity (Up to 6Gb/s)
- DisplayPort
- USB 3.0
- SATA 3.1
- PCIe 1.0 / 2.0

High-Speed Connectivity
- GigE
- USB 2.0
- CAN
- UART
- SPI
- Quad SPI NOR
- NAND
- SD/eMMC

Video Codec
- H.265/H.264

AMS

BRINGING YOU THE NEXT LEVEL IN EMBEDDED DEVELOPMENT
A full complement of tools are required to design an embedded processor system:

- Processor system generation
- Hardware implementation tools
- Software compilers
- Hardware debugger tools
- Software debugging tools
Necessary Tools

- A full complement of tools are required to design an embedded processor system
  - Processor system generation
  - Hardware implementation tools
  - Software compilers
  - Hardware debugger tools
  - Software debugging tools

- HW implementation & SW compilation are the two main flows that must be addressed
  - The embedded flows should mirror traditional flows
Traditional Embedded Design Flow

Standard Embedded SW Development Flow

- Code Entry
- C/C++ Cross Compiler
- Linker
- Load Software Into FLASH
- Debugger

Compiled ELF

Download Combined Image to FPGA

Compiled BIT

Board Support Package

System Netlist

Data2MEM

Embedded Developers Kit

Vivado

SDK

Traditional Embedded Design Flow

VHDL or Verilog

Standard FPGA HW Development Flow

- HDL Entry
- Simulation/Synthesis
- Implementation
- Download Bitstream Into FPGA
- ILA

Download Combiined Image to FPGA

RTOS, Board Support Package

C Code

اماكن النص والنص الكامِل. النص الرئيسي: "Traditional Embedded Design Flow

Standard Embedded SW Development Flow

- Code Entry
- C/C++ Cross Compiler
- Linker
- Load Software Into FLASH
- Debugger

Compiled ELF

Download Combined Image to FPGA

Compiled BIT

Board Support Package

System Netlist

Data2MEM

Embedded Developers Kit

Vivado

SDK

Traditional Embedded Design Flow

VHDL or Verilog

Standard FPGA HW Development Flow

- HDL Entry
- Simulation/Synthesis
- Implementation
- Download Bitstream Into FPGA
- ILA

Download Combiined Image to FPGA

RTOS, Board Support Package

C Code
Design Flow and Tools

- **FPGA hardware**
  - Spartan - Artix
  - Kintex - Virtex
  - Zynq
    - 7000 - 7000S
    - Zynq UltraSCALE+
      - Dual A53 / Dual R5
      - Quad A53 / Dual R5 / Codec
      - Quad A53 / Dual R5 / GPU

- **Software**
  - Real-time OS
  - Drivers
  - APIs
  - ...

- **Tools**
  - Vivado SDK
  - Matlab
  - SDSoC
  - ...

- **Tools**
  - Vivado HLx
  - Matlab
  - ...

BRINGING YOU THE NEXT LEVEL IN EMBEDDED DEVELOPMENT
Embedded Design Flow with SDSoC

- Migrate C/C++ functions to hardware
- System-level debug and profile
- Simple hardware-software partitioning
- Full system generation including driver and hardware connectivity
Design Flow without SDSoC

- **System Spec (C/C++)**
- **HW / SW Partition**
- **HW Design (Verilog / VHDL / HLS)**
- **HW Connectivity (IPI Block Design)**
- **SW Driver (Low-level C)**
- **SW Connectivity (C/C++)**
- **Req. Met?**

**Next Steps:**
- **Vivado / HLS**
- **Vivado IPI**
- **SDK / OS Tools**
- **SDK**

**Final Step:**
- **Application**
- **Drivers / Middleware**
- **Data path**

**Tools:**
- Vivado / HLS
- Vivado IPI
- SDK / OS Tools
- SDK

**Deployment:**
- **IP**
- **PS**
- **PL**
Design Flow with SDSoC

- System Spec (C/C++)
  ▶ Function Selection
  ▶ Refine Code
  ▶ Code typically needs to be refined to achieve optimal results
  ▶ Req. Met?

- SDSoC Environment
- IP
- Glue Logic
- Driver / Middleware
- Application
- ZYNQ
- PL
- PS

BRINGING YOU THE NEXT LEVEL IN EMBEDDED DEVELOPMENT
SDSoC System Level Profiling

- **Rapid system performance estimation**
  - Full system estimation (programmable logic, data communication, processing system)
  - Reports SW/HW cycle level performance and hardware utilization
- **Automated performance measurement**
  - Runtime measurement by instrumentation of cache, memory, and bus utilization
SDSoC System Level Profiling

Performance, speedup and resource estimation report for the 'Topic' project

Note: Performance estimation assumes worst-case latency of hardware accelerators, it also assumes worst-case data transfer size for arrays (if transfer size cannot be determined at compile time). If the accelerator latency and data transfer size at run-time is smaller than such assumptions, the performance estimation will be more pessimistic than the actual performance.

Summary

<table>
<thead>
<tr>
<th>Performance estimates for 'main' function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW-only (Measured cycles)</td>
</tr>
<tr>
<td>HW accelerated (Estimated cycles)</td>
</tr>
<tr>
<td>Estimated speedup</td>
</tr>
</tbody>
</table>

Details

<table>
<thead>
<tr>
<th>Performance estimates for functions 'sobel_filter, sharpen_filter and rgb_2_gray'</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW-only (Measured cycles)</td>
</tr>
<tr>
<td>HW accelerated (Estimated cycles)</td>
</tr>
<tr>
<td>Estimated speedup</td>
</tr>
</tbody>
</table>

Resource utilization estimates for hardware accelerators

<table>
<thead>
<tr>
<th>Resource</th>
<th>Used</th>
<th>Total</th>
<th>% Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP</td>
<td>3</td>
<td>220</td>
<td>1.36</td>
</tr>
<tr>
<td>BRAM</td>
<td>6</td>
<td>140</td>
<td>4.29</td>
</tr>
<tr>
<td>LUT</td>
<td>715</td>
<td>53200</td>
<td>1.34</td>
</tr>
<tr>
<td>FF</td>
<td>600</td>
<td>106400</td>
<td>0.56</td>
</tr>
</tbody>
</table>
MPSoC HW/SW Considerations

- Quad-core ARM Cortex-A53
- Dual-core ARM R5
- ARM Mail-400MP GPU
- DDRx and SMC controllers
- Security firmware
- Platform Management Unit
- FSBL, uBoot
- ARM trusted firmware
- XEN hypervisor
- Software test libraries
- Inter-processor Framework
- Multi-OS boot image
Example Default Configuration: APU-Linux / RPU

- **System software**
  - FSBL: First Stage Boot Loader
  - uBoot: Open source

- **APU: Non-secure mode**
  - ARM trusted firmware: From Xilinx, verified on APU
  - SMP Linux
  - No hypervisor: Non-secure mode

- **RPU**
  - In split mode (default)
  - R5-0: FreeRTOS
  - R5-1: Bare-metal
Example Configuration: APU-Hypervisor Linux / RPU

- **System software**
  - FSBL: First Stage Boot Loader
  - uBoot: Open source

- **APU: Non-secure mode**
  - ARM trusted firmware/XEN hypervisor, verified on APU
  - Guest OS
  - Domain (1): Linux
  - Domain (2): Bare-metal

- **RPU**
  - In split mode (default)
  - R5-0: FreeRTOS
  - R5-1: Bare-metal
Run-Time Software

Growing Ecosystem

<table>
<thead>
<tr>
<th></th>
<th>APU</th>
<th>RPU</th>
<th>MicroBlaze</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xilinx, Mentor, Ubuntu</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xen Hypervisor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Android</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Baremetal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FreeRTOS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GHS – Integrity, uVelocity</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LynxOS7, LynxSecure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mentor Nucleus, AutoSAR, Hypervisor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Micrium - uC/OS-II &amp; III</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QNX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sysgo – PikeOS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wind River – VxWorks7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows EC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Qemu Emulation Platform

- QEMU emulation platform
  - Provides Linux hosted emulation platform
  - Accelerates and scales embedded software development
  - Enables architecture and porting of software
  - Emulates multiple blocks of the processing system

- QEMU enables you to start working on designs before hardware is available
How Qemu Works

QEMU maps ARM v8 CPU instructions to one or more equivalent x86 functions.

QEMU runs on your laptop, desktop or server.

- Quad Core Cortex A-53
- Dual Core Cortex R5
- Peripherals
Our competences

Core|Vision has more than 125 man years of design experience in hard- and software development. Our competence areas are:

- System Design
- FPGA Design
- Consultancy / Training
- Digital Signal Processing
- Embedded Real-time Software
- App development, IOS Android
- Data Acquisition, digital and analog
- Modeling & Simulation
- PCB design & Layout
- Doulos & Xilinx Training Partner
Q&A

Cereslaan 10b
5384 VT Heesch
+31 (0)412 660088

www.core-vision.nl
Email: info@core-vision.nl
Training Program

- Essentials of FPGA Design 1 day
- Designing for Performance 2 days
- Advanced FPGA Implementation 2 days
- Design Techniques for Lower Cost 1 day
- Designing with Spartan-6 and Virtex-6 Family 3 days
- Essential Design with the PlanAhead Analysis Tool 1 day
- Advanced Design with the PlanAhead Analysis Tool 2 days
- Xilinx Partial Reconfiguration Tools and Techniques 2 days
- Designing with the 7 Series Families 2 days
Training Program

- Designing FPGAs Using the Vivado Design Suite 1  2 days
- Designing FPGAs Using the Vivado Design Suite 2  2 days
- Designing FPGAs Using the Vivado Design Suite 3  2 days
- Designing FPGAs Using the Vivado Design Suite 4  2 days
- Designing with the UltraScale and UltraScale+ Architecture  2 days
- Vivado Design Suite for ISE Software Project Navigator User  1 day
- Vivado Design Suite Advanced XDC and Static Timing Analysis for ISE Software User  2 days
Training Program

- Designing with Multi Gigabit Serial IO  3 days
- High Level Synthesis with Vivado  2 days
- C-Based HLS Coding for Hardware Designers  1 day
- C-Based HLS Coding for Software Designers  1 day
- DSP Design Using System Generator  2 days
- Essential DSP Implementation Techniques for Xilinx FPGAs  2 days
Training Program

- Embedded Systems Design 2 days
- Embedded Systems Software Design 2 days
- Advanced Features and Techniques of SDK 2 days
- Advanced Features and Techniques of EDK 2 days
- Zynq All Programmable SoC Systems Architecture 2 days
- Zynq All Programmable SoC Accelerators 1 day
- C Language Programming with SDK 2 days
- Embedded Design with PetaLinux Tools 2 days
- Zynq UltraScale+ MPSoC for the System Architect 2 days
- Embedded C/C++ SDSoC Development Environment and Methodology 1 day
Training Program

- VHDL Design for FPGA 3 days
- Advanced VDHL 2 days
- Comprehensive VHDL 5 days
- Expert VHDL Verification 3 days
- Expert VDHL Design 2 days
- Expert VHDL 5 days
- Essential Digital Design Techniques 2 days