All programmable SoC platform to enable the smarter embedded design

2014 Nov
Wide Adoption by the Industry

➤ 700+ customers worldwide
  – 250+ customers (APAC)
  – 500+ projects (APAC)

➤ 100+ partners, 30+ dev boards

➤ Diverse applications including industrial control, video surveillance, machine vision, ADAS, wireless, wired, test&measurement, broadcasting, display etc.
Advantages of Zynq Platform

**Lowest NRE, Best Risk Mitigation**
- Already manufactured silicon
- Negligible development & design tool costs
- Xilinx IP library + third-party IP
- Extensive development boards

**Greatest Flexibility & Differentiation**
- All Programmable HW, SW & I/O
- Anytime field reconfiguration
- System Secure (encryption)

**Streamlined Productivity & Fast TTM**
- Instant HW/SW co-development
- All Programmable Abstractions (C, C++, OpenCV, OpenCL, HDL, model-based entry)
- Vivado Design Suite, Vivado HLS, IP Integrator & UltraFast Methodology
- Broad and open OS & IDE support (Open-source Linux & Android, FreeRTOS, Windows Embedded, Wind River, Green Hills, & many others)

**Lowest Cost of Derivatives & Highest Profitability**
- IP standardized on ARM AMBA AXI4
- Reuse precertified code (ISO, FCC, etc.)
- Reuse & refine code & testbenches
- Volume silicon, power circuitry, PCBs & IP licensing
An Example of Use Zynq as a Platform
Zynq-7000 Block Diagram
**Zynq-7000 Development Environment**

**Hardware** + **Software** + **Tools**

**Processors**
- Cortex A9

**Boot loader**
- Sample code

**IP Cores**

**Platforms**
- Boards and Kits

**Xilinx supported BSPs**
- Standard boards

**Commercial & Custom BSPs**
- Product boards

**Hardware Development**
- Vivado, Vivado HLS

**Software Development**
- Linux & bare-metal software development environment
- Xilinx ChipScope Pro
- Linux Ecosystem Tools

**Xilinx Support**
Zynq-7000 All Programmable SoC Design Flow

» Design starts from the Xilinx Vivado design suite
  - Zynq PS centric design with the processing functions

» Design is expanded to include peripherals and PL standard & custom blocks

» Design is then exported to Vivado SDK for BSP and SW development
  - The end result is simply a customized “application board” for further development
Zynq-7000 All Programmable SoC Design Flow

- Design starts from the Xilinx Vivado design suite
  - Zynq PS centric design with the processing functions

- Design is expanded to include peripherals and PL standard & custom blocks

- Design is then exported to Vivado SDK for BSP and SW development
  - The end result is simply a customized “application board” for further development
All Programmable SoC Software Platform

Applications

- Industrial Applications
- Communication Applications
- Automotive Applications
- Medical Applications

Libraries & APIs

- Standard “C” Libraries
- File System Libraries
- Networking Libraries
- Graphics Libraries
- ARM Libraries
- ZYNQ Libraries
- Custom Libraries

OS BSP’s

- TCP/IP
- USB Stack
- Seriel IO
- Interrupt Manager
- File System
- PCIe
- SPI
- PCAP
- SD Card
- SPI
- I2C
- UART
- SMP
- Customizable Accelerators
- Customizable Peripherals

Processing System

Programmable Logic

OS Kernel

High Level and Low Level Drivers

- Custom Libraries:
  - Configuration of the ZYNQ PL
  - Control for clocking resources in ZYNQ PS
  - Performance monitoring
  - HW and SW debug support (e.g., cross-triggering with ChipScope Pro SW)
Programmer’s View of Programmable Logic

Simple memory mapped Interface

Programmer’s View of Custom Accelerators & Peripherals

<table>
<thead>
<tr>
<th>Start Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000_0000</td>
<td>External DDR RAM</td>
</tr>
<tr>
<td>0x4000_0000</td>
<td>Custom Peripherals (Programmable Logic including PCIe)</td>
</tr>
<tr>
<td>0xE000_0000</td>
<td>Fixed I/O Peripherals</td>
</tr>
<tr>
<td>0xF800_0000</td>
<td>Fixed Internal Peripherals (Timers, Watchdog, DMA, Interconnect)</td>
</tr>
<tr>
<td>0xFC00_0000</td>
<td>Flash Memory</td>
</tr>
<tr>
<td>0xFC00_0000</td>
<td>On-Chip Memory</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Start Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x4000_0000</td>
<td>Accelerator #1 (Video Scaler)</td>
</tr>
<tr>
<td>0x6000_0000</td>
<td>Accelerator #2 (Video Object Identification)</td>
</tr>
<tr>
<td>0x8000_0000</td>
<td>Peripheral #1 (Display Controller)</td>
</tr>
</tbody>
</table>

Code Snippet

```c
int main() {
    int *data = 0x1000_0000;
    int *accel1 = 0x4000_0000;

    // Pure SW processing
    Process_data_sw(data);

    // HW Accelerator-based processing
    Send_data_to_accel(data, accel1);
    process_data_hw(accel1);
    Recv_data_from_accel(data, accel1);
}
```
# Best-In-Class Operating Systems

<table>
<thead>
<tr>
<th>OS</th>
<th>Provider</th>
<th>Version</th>
<th>Availability (ZC702 board support)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>ENEA</td>
<td>3.0</td>
<td>Now</td>
</tr>
<tr>
<td>ENEA Linux</td>
<td>ENEA</td>
<td>3.0</td>
<td>Now</td>
</tr>
<tr>
<td>WR Linux 5</td>
<td>Wind River</td>
<td>3.4</td>
<td>Now</td>
</tr>
<tr>
<td>MVL CGE6</td>
<td>Montavista</td>
<td>2.6.32</td>
<td>Now</td>
</tr>
<tr>
<td>LinuxLink</td>
<td>Timesys</td>
<td>NA</td>
<td>Now</td>
</tr>
<tr>
<td>Android</td>
<td>iVeia</td>
<td>4.2.2</td>
<td>Now</td>
</tr>
<tr>
<td>Windows Embedded Compact 7/2013</td>
<td>Adeneo Embedded</td>
<td>7</td>
<td>Now</td>
</tr>
<tr>
<td>VxWorks</td>
<td>Wind River</td>
<td>6.9.3, 7</td>
<td>Now</td>
</tr>
<tr>
<td>INTEGRITY</td>
<td>GreenHills Software</td>
<td>NA</td>
<td>Now</td>
</tr>
<tr>
<td>QNX</td>
<td>Adeneo Embedded/QNX</td>
<td>6.5</td>
<td>Now</td>
</tr>
<tr>
<td>OSE</td>
<td>ENEA</td>
<td>5.5</td>
<td>Now</td>
</tr>
<tr>
<td>ThreadX/NetX</td>
<td>Express Logic</td>
<td>NA</td>
<td>Now</td>
</tr>
<tr>
<td>FreeRTOS</td>
<td>Xilinx</td>
<td>7.x</td>
<td>Now</td>
</tr>
<tr>
<td>RTA-OS SC1-4</td>
<td>ETAS</td>
<td>3.0</td>
<td>Now (single core)</td>
</tr>
<tr>
<td>eCOS</td>
<td>ITR</td>
<td>3.0</td>
<td>Now</td>
</tr>
<tr>
<td>eT-Kernel</td>
<td>eSOL</td>
<td>TBD</td>
<td>Now</td>
</tr>
<tr>
<td>µc/OS</td>
<td>Micrium</td>
<td>II</td>
<td>Now</td>
</tr>
<tr>
<td>Nucleus</td>
<td>Mentor</td>
<td>NA</td>
<td>Now</td>
</tr>
<tr>
<td>Quadros</td>
<td>Quadros</td>
<td>NA</td>
<td>Now</td>
</tr>
</tbody>
</table>
# AMP and SMP Architectures

<table>
<thead>
<tr>
<th></th>
<th>AMP</th>
<th>SMP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overhead</strong></td>
<td>1. None to Small (direct IO access, etc.)</td>
<td>1. Small to Moderate with proper design</td>
</tr>
<tr>
<td><strong>Control</strong></td>
<td>1. Only A9 Core-0 is managed by OS</td>
<td>1. Both cores are managed by OS</td>
</tr>
<tr>
<td></td>
<td>2. Core-1 (i.e., 2(^{nd}) Core) runs independently (either separate OS, or “Bare Metal” SW)</td>
<td>2. SW independence on Core-1 by standard OS functions</td>
</tr>
<tr>
<td><strong>Task Scheduling</strong></td>
<td>1. Task scheduling by OS only on Core-0</td>
<td>1. Task scheduling across both cores under OS control and discretion</td>
</tr>
<tr>
<td></td>
<td>2. Core-1 tasks managed by SW (no OS induced jitter, etc.)</td>
<td>2. Task independence by Core Affinity</td>
</tr>
<tr>
<td><strong>Resource Sharing</strong></td>
<td>1. L2 cache (may) not available to Core-1</td>
<td>1. L2 cache serves both cores</td>
</tr>
<tr>
<td></td>
<td>2. GIC, IO resources available to Core-0</td>
<td>2. Shared IO resources, shared GIC (IRQ, FIQ) all under OS</td>
</tr>
<tr>
<td></td>
<td>3. IO resources for Core-1</td>
<td>3. Memory map shared across-cores</td>
</tr>
<tr>
<td></td>
<td>4. Separate memory map for separate cores</td>
<td></td>
</tr>
<tr>
<td><strong>Resource Independence</strong></td>
<td>1. Separate memory and IO (carefully! manage shared resources)</td>
<td>1. Interrupt and IO shielding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Potential cache miss/loss in latency</td>
</tr>
<tr>
<td><strong>Inter-core Communications</strong></td>
<td>1. Shared mem, msg queues by OCM</td>
<td>1. OS control</td>
</tr>
<tr>
<td></td>
<td>2. Open source community RPMMsg/ Remoteproc/ VirtIO</td>
<td>2. Available in commercial OS products</td>
</tr>
<tr>
<td><strong>Tools, Debug, Diagnostics</strong></td>
<td>1. Commonly on Core-0 with specific OS</td>
<td>1. Commonly on both cores with OS</td>
</tr>
<tr>
<td></td>
<td>2. Custom support on Core-1</td>
<td></td>
</tr>
</tbody>
</table>
An AMP Example in Wireless Application

Processor System

Core 0

Core 1
- NEON™/FPU Engine
- Cortex™-A9 MP Core™ 32/32 KB I/D Caches
- Snoop Control Unit (SCU)
- 256 KB On-Chip Memory
- DMA Configuration

External Memory

AXI Interconnect

S_AXI_HP_0/1 64-bits @ 400MHz

Capture CTRL Logic

DPD Correlation Matrix Accelerator

Correlation Matrix Result Data 64-bits @ 400MHz

Software plus H/W Offloading Implementation

S_AXI_HP_2/3 64-bits @ 400MHz

AXI_FIFO

Cholesky CMAC Accelerator

64-bits @ 400MHz

DPD Coefficients

From DUC

DPD Forward Path

Other Tx Processing

SerDes

Tx Interface

SerDes

DPD Feedback

64-bits @ 400MHz

Cholesky Data

64-bits @ 400MHz

Other Tx Processing

Core 0

Core 1
## Comparison of Cholesky Decomposition

<table>
<thead>
<tr>
<th>Cholesky Decomposition Matrix Size</th>
<th>ARM A9 with NEON Computation Time*</th>
<th>Leading Vendor DSP Computation Time**</th>
</tr>
</thead>
<tbody>
<tr>
<td>96x96</td>
<td>3.7 ms</td>
<td>5.2 ms</td>
</tr>
<tr>
<td>132x132</td>
<td>10.2 ms</td>
<td>14.4 ms</td>
</tr>
</tbody>
</table>

* Neon compiler optimized and running at 1GHz  
** Data derived from vendor public domain documents  
*** Comparison made for Cholesky decomposition only, other functions for DPD may be implemented in ARM, NEON, or PL
Project Ne10

- An open source software library provides a set of the most commonly used functions that have been heavily optimized for ARM CPU with Neon
- The library focuses on matrix/vector algebra and signal processing
- The Ne10 is provided free of charge by ARM Limited
- No assembly coding required to use it
- Ne10 will evolve over time

<table>
<thead>
<tr>
<th>Math Functions</th>
<th>Signal Processing Functions</th>
<th>Image Processing Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector Add / Sub</td>
<td>Complex FFT</td>
<td>Image Resize</td>
</tr>
<tr>
<td>Matrix Add / Sub</td>
<td>Float/Fixed point Complex FFT</td>
<td>Image Rotate</td>
</tr>
<tr>
<td>Vector Multiply / Multiply-Accumulator</td>
<td>Float/Fixed point Real2Complex FFT</td>
<td></td>
</tr>
<tr>
<td>Matrix Multiply / Vector Multiply</td>
<td>Finite Impulse Response (FIR) Filters</td>
<td></td>
</tr>
<tr>
<td>Vector Div / Setc / Len / Normalize / Abs / Dot / Cross / Rsbc</td>
<td>Finite Impulse Response (FIR) Decimator</td>
<td></td>
</tr>
<tr>
<td>Matrix Div / Determinant / Invertible / Transpose / Identify</td>
<td>Finite Impulse Response (FIR) Interpolator</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Finite Impulse Response (FIR) Lattice Filters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Finite Impulse Response (FIR) Sparse Filters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Infinite Impulse Response (IIR) Lattice Filters</td>
<td></td>
</tr>
</tbody>
</table>

How to use it?


An Example with Zynq – Spectrum Analyzer

4096 point FFT – Complex 32 bit floating point

- ARM processor alone – 830 usec
- NEON SIMD engine – 571 usec
- Hardware in PL fabric – 129 usec

45% FFT Acceleration Using NEON Instructions and ARM NE10 DSP Library

6.4x FFT Acceleration Using ACP Attached Coprocessor Accelerator
Zynq EtherCAT Motor Drive Ref Design

Field Oriented Control

Torque loop 1.6 μs,
Speed loop 3.2 μs,
Position loop 6.4 μs

Available from QDESYS; send e-mail to info@qdesys.com

- It includes the EtherCAT® binary evaluation IP in the Zynq bitstream
- It includes the QNX® operating system and the master application; this version of QNX® has been created by QDESYS and is for evaluation only.
- It includes the QNX® operating system and the slave application; this version of QNX® has been created by QDESYS and is for evaluation only.
- It includes the Bare-metal
Xilinx / Xylon HMI Ref Design

Xylon offers several free pre-verified reference designs for Xilinx Zynq AP SoC

Available on hardware from Xilinx (ZC702/ZC706), Avnet and TED

Works “out-of-box”

Includes

- Evaluation logicBRICKS IP cores (run-time limited)
- Complete OS image, software drivers and demo applications
- Documentation

Industrial HMI demo included with:

- logiREF-ZGPU ref. design for the ZC702
- logiREF-ZGPU-ZED ref. design for the ZedBoard

# Zynq RTVE Reference Design

- **Real Time Video Engine**
  - Scaling, Deinterlacing, OSD
- **RTVE 2.1 up to 8 HD channels**
- **RTVE 3.0 up to 4K**
- **Linux O/S**

## Hardware Platforms
- OmniTek OZ745
- Device-locked Vivado

## OmniTek Scalable Video Processor
- From [OmniTek.tv](#):
  - SD Image
  - OZ745 BSP
- OSVP docs & collateral
  - Datasheet
  - User's guide
- OSVP Licensing

## RTVE 2.1 Reference Design
- From Xilinx.com:
  - [XAPP1095](#)
  - Evaluation Lounge
  - OmniTek Ref Design
  - Evaluation Bitstream

---

Demos available – contact local Xilinx sales office

---

[Image of Zynq RTVE Reference Design with accompanying hardware and software specifications.]
Zynq LTE UE Ref Design

- ZC706 Board with Z7045
- RF Board: ADI AD9361 Transceiver
  - Very low power
  - 2x2 MIMO
- SAI LTE UE Ref Design
  - All C PHY code ported with HLS
  - LTE UE Release 9
  - Meets 3GPP Standards
  - Optimized for low power
Zynq HD ADAS Ref Design

- Xilinx - ZC702 board
  - 3D Surround

- Xylon - daughter card + cameras
  - Rear view camera with PiP
  - Blind-spot with optical flow

- Licensible Partner IP
  - Rear LDW
  - Pedestrian detection

- 6x HD cameras for large vehicle
Next-Gen Ultrascale MPSoC Architecture

✓ Scalable to 64 bit with HW Virtualization
✓ 16nm FinFET process
✓ Next-Generation Interconnect and Memory
✓ The Right Engines for the Right Tasks
✓ FPGA ASIC-class Scalability and Performance
✓ Advanced Power Management
✓ Multi-level Security, Safety, and Reliability
✓ High-Level Design Abstractions – C/C++, OpenCL
✓ Compatible with Zynq-7000 Software & Ecosystem
Enabling All Programmable Heterogeneous Multi-Processing
Target Applications

- **Wireless Communications**: Support for multiple spectral bands, smart antennas
- **Wired Communications**: Multiple wired communications standards, context-aware network services
- **Data Centers**: SDN (Software Defined Networks), data pre-processing, and analytics
- **Smarter Vision**: Evolving video-processing algorithms, object detection, and analytics
- **Connected Control/M2M**: Flexible/adaptable manufacturing, factory throughput, quality and safety
SDIntegrator: Complete End-to-End Flow

C/C++

Simple familiar SW development environment

Proven design tools and technologies

A complete C to system flow!

SW Code
Generated driver
OS, BSP

Target Board

Platform

PS
PL
IP
IP
IP
IP

Connectivity

Vivado HLS IP Integrator HSM ARM Compiler

Estimator Compiler Debugger Profiler

Debugger

Estimator

Profiler

Compiler

HLS

IP Integrator

HSM

ARM Compiler
Zynq-7000 AP SoC Developer Community
Building a Robust Developer Ecosystem

  - Technical documentation and guide to design software

  - Exchange ideas and support

- Xilinx support will handle WebCases on Linux BSP
  - Use forums & community for general Linux related issues
The Zynq Book

- The first book about Zynq written in English by a team of authors from the University of Strathclyde, Glasgow, UK, with the support of Xilinx.

- Free download pdf

Summary

Zynq is the ideal SoC platform for embedded design
- All programmable: flexibility, scalability and differentiation
- Xilinx offers best-in-class tool, IP-centric development flow and wide range of embedded OS support
- Easy to use as other ARM SoC from developer’s view

Xilinx & partners offer complete ref designs to make development easier

All Programmable SoC Roadmap
- Xilinx is committed to the future of All programmable SoC
- 16nm FinFET Heterogeneous Ultrascale MP SoC
Thank you.