Tools, Architectures and Trends on Industrial all Programmable Heterogeneous MPSoC

29th Euromicro Conference on Real-Time Systems (ECRTS17)
June 27 - 30, 2017
Dubrovnik, Croatia

Dr. Ing. Giulio Corradi Senior System Architect ISM
WHAT IS HAPPENING TO THE SEMICONDUCTOR INDUSTRY?
1 Water drop = 10 Billion Water molecules

$H_2O = 0.275 \text{ nm}$
25 Water molecules

7 nanometers

3D FinFET
Advancement in Memory technology

- **10nm technology**
- **Data transfer rate of 3,200 megabits per second (Mbps),**
- **30 % faster than the 2,400Mbps rate of 20nm DDR4 DRAM**
Semiconductor Industry Consolidation

2007

Semiconductor industry enter a mature stage

Few chipmakers can afford the multibillion dollar investments required of 16nm and below technology

Reasons

- Diminishing returns from:
  - Moore’s Law and,
  - Dennard scaling

Consequences

- Huge computing parallelism
  - Multicores, Manycores
  - Heterogeneous computing

- Memory subsystem changes
  - Faster memories
  - Larger wordlength

All product names, logos, and brands are property of their respective owners

© Copyright 2017 Xilinx
Where are going the industrial applications?
Industrial applications domains challenges

What engineers want…
- Higher predictability (time and delivery)
- High Performance Real time Control
- Real time Networking and Synchronization
- Real time Sensor Fusion
- Mixed traffic: real time, stream and best effort
- Functional Safety and fail operational
- Cybersecurity
- Machine Learning
- Hardware as service

Which challenges they have
- Huge amount of legacy software working as WCET (worst case execution time)
- Few tools for refactoring WCET under new SoC architecture
- Legacy Software working as SCE (single core execution)
- Few tools for repartitioning under HCE (heterogeneous cores architectures)
- Functional Safety Standards using old (proven) paradigms
  - Absolute demonstrable determinism (temporal and spatial)
  - > 90% diagnostic coverage (maximum diagnostic)
  - Fail safe and fail operational
Industrial IoT Devices require multiple domain time

**Edge Devices**
- 100ns...10us
- Sensors
- Motors
- Controllers

**Edge Apps**
- 10us...1ms
- Sensors
- Motors
- Cameras
- Smart sensors
- RF sensors
- Sensor fusion

**Hybrid Apps**
- 10us...100ms
- Enterprise

**Cloud Apps**
- 10ms...days/months
- Cloud

**Acquisition**

**Control**

**Monitoring**

**Optimization**

**Clouds Apps**

© Copyright 2017 Xilinx
<table>
<thead>
<tr>
<th>Consumer/Entertainment/Retail</th>
<th>Transportation/Infrastructure</th>
<th>Enterprise Operations</th>
<th>Oil &amp; Gas/Agriculture</th>
<th>Industrial/Military</th>
<th>Medical/Healthcare</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personal VR/Gaming Smart Displays</td>
<td>Personal Assistants</td>
<td>Ad Targeting and E-Commerce</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Autonomous Cars &amp; Trucks</td>
<td>Transportation &amp; Grid Control</td>
<td>Traffic &amp; Network Analytics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delivery Drones, Warehouse Robots</td>
<td>Cyber Security</td>
<td>Sales, Marketing &amp; Customer Service</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field Drones &amp; Robots</td>
<td>Climate, Water, Energy &amp; Flow Control</td>
<td>Field Sensor Data Analytics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robots/Cobots, UAV, Inspection</td>
<td>Factory Control &amp; Surveillance</td>
<td>Factory &amp; Operations Analytics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medical Imaging &amp; Surgical Robots</td>
<td>Medical Diagnostics</td>
<td>Clinical Analytics &amp; Recommendations</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: Machine Learning Landscape from Moor Insights & Strategy

Edge and Cloud applications integrated
Realization of modern Realtime Systems – all connected

Legacy Industrial Network

IT Network

OT Network (Industrial Ethernet)

IT/OT Alignment

Local Cloud

Remote Service

Edge Node

Display

Contact

Database

Sensor

Temp sensor

Volt meter

IT/OT Alignment

Drive

Sensor

I/O

Display

Button

Contact

Temp sensor

Light sensor

Volt meter

ERP

MES

SCADA

PLC

IO/Drive

© Copyright 2017 Xilinx
Networking time awareness for different traffic classes

TSN Concept
- **Time-aware Gates** make Ethernet cyclic
  - Scheduled Traffic, real-time, cyclic
  - cycle times (intervals): ~100 μs – ms (typical)
  - Reserved Traffic (audio/video)
  - Best effort (IT traffic)

- A Central Network Controller Software calculates open+close for Gates

- Each network element in a TSN must support this concept

Gate Control List
- T0: 01111111
- T1: 00000000
- T2: 10000000
- T3: 01111111
- T4: 00000000
- T5: 10000000
- T6: 01111111
- T125: 10000000
- T126: repeat

Time Aware Shaper (TAS) (IEEE802.1Qbv)
Main take away...

- **Realtime domain extends and becomes pervasive**
  - Multi and Many cores realtime
  - Heterogeneous cores realtime
  - Networked cores realtime
  - Networked systems realtime

- **Interactions between domains**
  - Not clearly defined domains boundaries, something running here today may run elsewhere tomorrow
  - Safety and Security interacts with delivery performances
  - Mixed criticality at system level not just at software level

- **Performances of all systems rising exponentially**
  - Autonomy
  - Machine Learning
  - Artificial Intelligence
Heterogeneous Systems on Chip
All major players have Heterogeneous Systems on Chip

**TI – Jacinto**
- 4 A15
- 2 M4
- 2 C66 DSP

**NXP – iMX8**
- 4 A35 (A53 as well)
- 1 M4

**NVIDIA TX2**
- 2 Denver (ArmV8)
- 4 A57
- 1 A9
- 2 R5
- 1 R5 lockstep
- 1 GPGPU Pascal

**Xilinx – ZU9EG**
- 4 A53
- 2 R5 lockstep
- 2 Tricore (PMU & CSU)
- 2250 DSP (48bitMAC)
- 600K Logic Cells
- many 32bit MB CPUs

All product names, logos, and brands are property of their respective owners.
Some Industrial Use case for Heterogeneous SoC
Industrial USE CASE #1: High Performance PLC

**TRENDS:**
- PLCs continue to evolve
  - Adopting hardware improvements
  - Increasing communications,
  - Being safety enabled
  - Using more memory
  - Using better Human Interfaces
  - Manipulating Video and Camera information
  - Adding
    - Machine Learning
    - Cloud enablement

- Small PLCs will include features of higher-level PLCs,
- Mid- and High-range PLCs will offer a smaller, more compact solution to meet users' needs

If you do not know what is a PLC please look on Wikipedia: [https://en.wikipedia.org/wiki/Programmable_logic_controller](https://en.wikipedia.org/wiki/Programmable_logic_controller)
USE CASE #1 – internals of a PLC and its timing
The basic high level ingredients of heterogeneous SoC

- **APP**: Application Processor
- **RTP**: Real Time Processor
- **TCM**: Tightly Coupled Memory
- **OCM**: On Chip Memory
- **FPGA**: Field Programmable Array
- **GPU**: Graphic Processing Unit
- **GPGPU**: General Purpose GPU
- **FSM**: Finite State Machine
Main take away....

- **Diversified Application Processors**
  - Interference on caches, memory, peripherals
  - Highest performance realtime (some applications cannot use R-class) seek solutions

- **Real time processors**
  - Automatic partition of real-time classes (soft, firm, and hard real-time)
  - Competition with application processors for shared resources
  - Safety integrity among application processors and real-time processors

- **Programmable Logic exploitation**
  - Scheduling offloading
  - Data movers
  - Zero latency synchronizers
  - Direct access (ACE, ACP) to caches helps predictability

- **Different memory traffic**
  - Reservation, realtime, stream and best-effort
  - Purpose specific memory to reduce bottlenecks
All Programmable Platforms
Example of Heterogeneous SoC - ZYNQ7000®

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
NEON™/FPU Engine
Cortex™-A9 MPCore™
32/32 KB I/D Caches
Cortex™-A9 MPCore™
32/32 KB I/D Caches
512 KB L2 Cache
Snoop Control Unit (SCU)
Timer Counters
General Interrupt Controller
DAMA Configuration

S_AXI_GP0/1
M_AXI_GP0/1

M_AXI_ACP

Programmable Logic:
System Gates, DSP, RAM

BRAM
BRAM
> 100
DSP
DSP
> 100
MB
MB

Lockstep

2x SPI
2x I2C
2x CAN
2x UART
GPIO
2x SDIO
with DMA
2x USB
with DMA
2x GigE
with DMA

I/O MUX

XADC

MB = MicroBlaze™ is Xilinx’s full-featured, FPGA optimized 32-bit Reduced Instruction Set Computer (RISC) soft processor, single, dual core lock step and TMR
A naïve use case example of heterogeneous computing
Use Case expanded on software (this is an example, solutions are many)

- **RPU**
  - Safety Loop
  - BlueThoot
  - Wireless WiFi

- **PL**
  - Industrial networking
  - 100M / 1G

- **APU**
  - Hypervisor (XEN Type)
  - SoftPLC
  - RTOS
  - Virtual Machine 1
  - Virtual Machine 2
  - Virtual Machine 3
  - Virtual Machine 4
  - Middleware (OPC/UA Server)
  - Machine Learning and Diagnostic
  - GPOS
  - Machine Learning and Diagnostic
  - GPOS
  - Security
  - HMI

- **SD Memory**
  - Data & Program Store

- **DDR**

- **Linux Kernel**

- **Physical Driver**

- **Network Processor**
  - Link Layer

- **API**

- **API (II)**
  - TSN (Time Sensitive Network)

- **ML**

- **Enterprise Networking**
  - 1Gb

- **ERP/MRP**

- **Field**

- **Proprietary IP**

- **Dedicated DDR**

- **Wired Ethernet**

- **Network Processor**
  - Link Layer

- **Physical Driver**

- **I/O split device driver**

- **TSN (Time Sensitive Network)**

- **3-ports Switch**

- **Wireless WiFi**

- **TDN (Time Sensitive Network)**

- **Native IP**

- **Wireless BlueThoot**

- **Wireless WiFi**

- **Field**

- **Performance bottleneck**

- **Industrial networking**
  - 100M / 1G
Performance challenges for the Heterogeneous systems

Application Processors classic challenges (the old problem)
- AMP (Asymmetric Multiprocessing) resource sharing, still an hard problem to solve
- Resource contention at L2 cache
  - Caches lock, coloring, and other schemes...cache lock no more available in many new AP clusters
- Resource contention at Memory Controller
  - Access policy, different quality of service
- Bus competition
  - SCU (Snoop Control Unit)
  - Word length (64bits,128bits,256bits)

Real Time Processors challenges
- Maximum operating frequency limited (because application processor tricks cannot be used)
- Limited internal memory (technology dependent)
- Resource competition at Memory Controller
- Bus competition with Application Processor
Safety challenges for heterogeneous systems

➤ Are the mixed criticality models addressing the right thing?
   – Many researches focused on:
     • overly simplified models – old architectures or inapplicable results
     • assume that you have full control on the code – often it is impossible
     • use synthetic benchmarks, often far from reality – better collaboration with industry
     • models mathematically dense but no proof of work is delivered.

➤ Are the resources considered and modelled holistically?
   – CCF (Common Cause Failures) one failure here damages the whole system
     • L2 cache shared with all cores, Memory controller shared with all cores, External memory shared with all cores
     • GIC (General Interrupt Controller) failure leads to loss of interrupts
   – Diagnostic difficult to accomplish
     • Latent fault (rarely used functional failure can accumulate unnoticed)
     • BIST (built in test) running concurrently with the applications without disruptions
   – Spatial separation models
     • Certification agencies asks for proof (how do you prove it if you do not know the silicon or tested it?)
   – Temporal separation models
     • Certification agencies asks for proof (to the researchers how do you prove it if you do not know the failure modes)
Partitioning challenges / solutions

➤ How to allocate an application to the best resource
  – Application processor
  – Realtime processor
  – Programmable Logic
  – Containerized (dockers)
    • Real time response in sand box – how do you guarantee it?
  – Containerized (hypervisors)
    • Real time response in peripherals – how do you manage it with bounded latency?
  – Offloaded to Hardware Workers (good for researchers)
    • Migration fully in hardware with high level compilers HLS
    • Migration with soft processors – on demand processing

➤ What programming paradigm is most effective
  – Classical C/C++ and assisted EDA (for automatic partitioning)
  – Use of OpenMP – Embedded engineers have limited exposure to it
  – Use of OpenCL – Lacks some friendliness…
  – Use of SYCL – new and untested…
Enhancing your research with SW to HW transformation
Offloading some of your algorithms in Hardware
- Using compiler from C/C++ to hardware – today the quality of such tools is very good!
- Improve your theory with hardware assistance – it is not that difficult
- Reduce impact of your modifications
  - Use your modules like peripherals
  - Map them in memory space
  - Create new type of specialized cores

Experiment with cores using C/C++ and a few hardware templates
- Ad hoc cores like the RISC-V5 in FPGA
- Soft cores like Microblaze for creation of workers

Use PL memories from C/C++ as
- Mailboxes
- FIFO
- Rings
Vivado HLS: Framework for C/C++ hardware compiler

New hardware specified by software

- Bus master (initiators)
- Memory mover
- Hashing
- Lists
- Pattern matching
- Math
- Arrays
- Graphs
How to take advantage of HLS in SoC ZYNQ7000®
Zynq® UltraScale+™ Use of PL and additional workers
Examples at work...
Example of system with Zynq-7000 - Top (1)

- Dual Core A9
- Dual Core Microblaze (MB)
- Shared segment in DDR between the A9s and MB
Example of system with Zynq-7000 – A9 Hierarchy (2)
Example of system with Zynq-7000 – MB Hierarchy (3)

MicroBlaze subsystem
**Example of system with Zynq-7000 – HLS module and its code (5)**

- **Functional blocks fully in software**
  - You program your model in C/C++
  - You validate it in C/C++
  - You declare the memory and command interface
  - You connect the module to your system
  - You generate your platform

```c
bool memfill( volatile float *write_pnt, volatile unsigned short num_inputs, volatile unsigned short num_outputs, volatile unsigned short cmd, unsigned int size) {
  #pragma HLS INTERFACE s_axilite port=cmd bundle=FILL_CTRL
  #pragma HLS INTERFACE s_axilite port=num_inputs bundle=FILL_CTRL
  #pragma HLS INTERFACE s_axilite port=num_outputs bundle=FILL_CTRL
  #pragma HLS INTERFACE m_axi port=write_pnt  offset=slave bundle=MDM_PORT

  switch (cmd) {
    case 1: forward(write_pnt, num_inputs, num_outputs, size); break;
    case 2: initialize_activation(write_pnt); break;
    case 3: set_layer(num_inputs, num_outputs); break;
    case 4: all_forward(write_pnt, size); break;
    case 5: backpropagation(write_pnt, num_inputs, num_outputs, size, &nn_target_cache[0], &nn_desired_cache[0], 0, &r); break; //sigmoid
  }
  return false;
}
```

**Memory interface and commands with few lines of C code**

- Pointer in DDR you access directly without processor intervention
  - Commands as set of registers (framework produces the mapping)
Example of system with Zynq-7000 – Board in example (6)

The board amongst many…

- UltraZed
- MicroZed
- PicoZed
- ZC702
- Zybo
- Pynq
- ArtyZ
- MiniZed
The Memory optimization for scheduled traffic
Six Port DDRC for better Effectiveness

- 1 port dedicated to RPU (64-bit)
- 2 ports (128-bit) dedicated to CCI traffic:
  - APU (quad A53), RPU (dual R5),
  - HP Coherent and ACE ports from PL
  - GPU, SATA, PCIe, USB3
  - I/O Peripherals
- 1 port (128-bit)
  - Display Port, HP0
- 1 port (128-bit)
  - HP1, HP2
- 1 port (128-bit)
  - HP3, FP-DMA

Legend
- 128 Bit
- 64 bit
- 32 bit APB
- Other
- Master ➔ Slave
- Data in both directions
Traffic Classes in the QoS System

» Isochronous Channel (V)
  – Fixed bandwidth
  – Guaranteed Worst Case latency
    • Required to set FIFO sizes and stream delay timing
  – Regular Traffic Pattern
  – Multiple Outstanding Transactions

» High Priority Read or Low Latency (HPR/LL)
  – High Priority
  – Read only – has to be read, use the HPR

» Best Effort (BE)
  – Lowest priority
  – Shares queue with video
  – Aging counter prevents starvation
Interconnect with Traffic Classes

Legend
- 128 Bit
- 64 bit
- 32 bit
- 32/64/128 bit
- 32/64 bit
- 32 bit APB
- GT bus
- Other
- Master ➔ Slave
- Data in both directions

V* - V or BE if DP is not used
BE* - BE or V if GDMA not used
Safety and beyond
Zynq-7000

Device Domains

- Processing System (PS)
- Programmable Logic (PL)

- Dual Channel A9
- Diverse Channel PS/PL
- Microblaze Lockstep
- SEM IP
- Temperature Monitor
- Voltage Monitor
Zynq-7000 Functional Safety Design

- 2 Channel Architecture (HFT=1)
- Cross Channel Monitoring
- Isolated Sensors (PS & PL)
- Isolated Actuators (PS & PL)
- Isolated Load Power and Load Device
- Independent Fault injection
Zynq Ultrascale +

**Device Domains**
- Full Power Domain (FPD)
- Low Power Domain (LPD)
- Programmable Logic (PL)

**PS Processing Units**
- Applications Processor Unit (APU) = A53 Complex
- Real-Time Processing Unit (RPU) = R5 Complex
- Graphic Processing Unit (GPU) = Mali-400MP Complex

- Configuration Security Unit (CSU): Configuration & Security
- Platform Management Unit (PMU): Power & Safety
Zynq Ultrascale + Low Power Domain Functional Safety Coverage

1. Lockstep for R5s
2. Triple Modular Redundancy (TMR) for Platform Management Unit (PMU) and Configuration & Security Unit (CSU)
3. ECC for TCM, OCM, CSU and PMU RAMs
4. Memory & Peripheral Protection Units provide functional isolation
5. CCF coverage by clock, voltage, and temperature monitors
6. Logic Built In Self Test (LBIST) for checkers & monitors at power-on
   - Peripherals coverage by end-to-end software protocols
7. Software Test Library (STL) for GIC, interconnect, SLCRs & error injection
Functional Safety Design Support and Artifacts

- Zynq Ultrascale + was designed with safety in mind
- Developed as an Safety Element out of Context (SEooC)
- ISO-26262 ASIL-C certifiable design example
- IEC-61508 SIL3 certifiable design example
- Vivado tool chain support in 2017.1
  - ISO-26262 & IEC-61508
  - Isolation Design Flow and verification tools
- Extensive Freedom from Interference hardware in the PL (XPPU, XMPPU) for peripheral and memory isolation
- Third Party Safety Certified compiler (ARM DS5) for A53 and R5
The Partial Reconfiguration (for task switching) for who would like full exploitation of FPGA for real-time, scheduling, and hardware time sharing
Partial Reconfiguration what is it?

Partial Reconfiguration is the ability to dynamically modify blocks of hardware modules in FPGA.
- downloading partial bit files while the remaining logic continues to operate without interruption.

Partial Reconfiguration technology allows designers to:
- change functionality on the fly,
- eliminating the need to fully reconfigure the FPGA
- re-establish links, dramatically enhancing the flexibility that FPGAs offer.

Partial Reconfiguration can:
- allows designers to move to fewer or smaller devices,
- reduce power, and
- improve system upgradability.
- make more efficient use of the silicon by only loading in functionality that is needed at any point in time.
- Swap decoders on the fly
  - One channel remains up while the other changes
- Released “flat” version first
  - Two decoders per channel
- Expanded functionality with existing hardware
  - Deployed new bitstreams for more decoders without changing hardware
Partial reconfiguration of hardware

- Partition methodology enables Partial Reconfiguration
  - Allows clear separation of static logic and Reconfigurable Modules
  - Floorplan to identify silicon resources to be reconfigured

- Design preservation accelerates design closure
  - Lock static design database while implementing new modules
Models in the Cloud for who would like to connect the Edges and the Cloud processing systems
Amazon Web Services EC2 F1 Instances

- Powered by the Xilinx Reconfigurable Acceleration Stack
- Deploy acceleration kernels in the cloud across many F1 instances

F1 partners have solutions across a wide range of applications

- Edico Genome, Maxeler, National Instruments, NGCodec, Ryft, TeraDeep and more

- Learn more here: https://aws.amazon.com/ec2/instance-types/f1
Accelerate your research with PL in the cloud

**Xilinx University Program**

- Run Custom FPGAs in the AWS Cloud

**Accelerate Your Research on the Xilinx AWS Cloud**

- A compute instance with Virtex VU9P UltraScale+ FPGAs
- AWS Cloud pre-configured with Vivado Design Suite
- Get started with AWS Educate and Xilinx University Program

© Copyright 2017 Xilinx
Looking forward

➤ Take advantage of such new technology to expand your research
  – Tools for partitioning of system element into the proper core
  – Tools for refactoring of old code
  – Look under the hood, such machines have many things you can exploit

➤ Connect with industry
  – It is difficult I know… but being proactive is better than reactive

➤ Extend the horizon
  – Sometime a clever function repartition do in hardware what you cannot do in software and vice-versa may save years of frustration in finding the holy grail

➤ In the meantime… Enjoy life, Dubrovnik and its beautiful sea
Thank you!
Special thanks to the program committee
Contact

Giulio.Corradi@Xilinx.com

Put in the email header ECRTS17 (otherwise you will be Bayesian filtered)
Expect delayed response, if none within reasonable time insist, if still none probably you have been filtered
Some Examples how to learn more (beginner and advanced)


- [http://www.pynq.io/](http://www.pynq.io/) (How to get a full many core Zynq based Python enabled framework)

- [https://github.com/Xilinx/HLx_Examples](https://github.com/Xilinx/HLx_Examples) (Highly interesting HLS examples)
  - Memcached implementation in HLS fully in hardware
  - TCP/IP implementation in HLS fully in hardware
  - Video streams
  - Matrix multiplications offloaded in hardware

- [https://www.xilinx.com/products/design-tools/software-zone/sdsoc.html](https://www.xilinx.com/products/design-tools/software-zone/sdsoc.html)
  - Full in software environment for **what if analysis and performance measurement**

- [https://www.xilinx.com/support/university.html](https://www.xilinx.com/support/university.html) (University Program)
Follow Xilinx

facebook.com/XilinxInc

twitter.com/XilinxInc

youtube.com/XilinxInc

linkedin.com/company/Xilinx

plus.google.com/+Xilinx