Open ESP
The Heterogeneous Open-Source Platform for Developing RISC-V Systems

Luca P. Carloni with Davide Giri

FOSDEM’ 20,
Brussels Feb 1, 2020
The ESP Vision

ESP is an open-source research platform for heterogeneous system-on-chip design that combines a flexible tile-based architecture and a modular system-level design methodology.

ESP provides three accelerator flows: RTL, high-level synthesis (HLS), machine learning frameworks. All three design flows converge to the ESP automated SoC integration flow that generates the necessary hardware and software interfaces to rapidly enable full-system prototyping on FPGA.
Why ESP?

Heterogeneous systems are pervasive

Integrating accelerators into a SoC is hard

Doing so in a scalable way is very hard

Keeping the system simple to program while doing so is even harder

ESP makes it easy

ESP combines a scalable architecture with a flexible methodology

ESP enables several accelerator design flows

and takes care of the hardware and software integration
ESP Vision: Domain Experts Can Design SoCs

Application Developers:
- K
- PyTorch

Hardware Designers:
- CHISEL
- SystemVerilog

HLS Design Flows:
- `hls 4 ml` + ESP

RTL Design Flows:
- `RTVLA.org`
- `RISC-v`

SoC Integration:
- I/O
- Rapid Prototyping

By Nvidia Corporation

* By lewing@isc.tamu.edu Larry Ewing and The GIMP

** By lewing@isc.tamu.edu Larry Ewing and The GIMP
ESP Architecture

- RISC-V Processors
- Many-Accelerator
- Distributed Memory
- Multi-Plane NoC

The ESP architecture implements a distributed system, which is scalable, modular and heterogeneous, giving processors and accelerators similar weight in the SoC.
ESP Architecture: Processor Tile

• Processor off-the-shelf
  o RISC-V Ariane (64 bit)
    SPARC V8 Leon3 (32 bit)
    o L1 private cache
• L2 private cache
  o Configurable size
  o MESI protocol
• IO/IRQ channel
  o Un-cached
  o Accelerator config. registers, interrupts, flush, UART, ...
ESP Architecture: Memory Tile

- External Memory Channel
- LLC and directory partition
  - Configurable size
  - Extended MESI protocol
  - Supports coherent-DMA for accelerators
- DMA channels
- IO/IRQ channel
ESP Architecture: Accelerator Tile

- Accelerator Socket w/ Platform Services
  - Direct-memory-access
  - Run-time selection of coherence model:
    - Fully coherent
    - LLC coherent
    - Non coherent
  - User-defined registers
  - Distributed interrupt
ESP Accelerator Socket

ESP accelerator
HLS [C/C++, SystemC, Tensorflow*, Pytorch*]
RTL [Chisel, Verilog, ...]

PLM

read/write
config
done

private cache
TLB
DMA ctrl
cfg regs
IRQ

1
coherence planes
2
3
coherent-DMA planes
4
5
DMA planes
6
IO/IRQ plane

NoC

Third-Party Accelerator Socket*
third-party accelerator
(NVDA*, ...)

AXI4 bus
APB bus
IRQ

5
4
DMA planes
5
4
DMA planes
6
IO/IRQ plane

NoC
## ESP Platform Services

### Accelerator tile
- DMA
- Reconfigurable coherence
- Point-to-point
- ESP or AXI interface
- DVFS controller

### Processor Tile
- Coherence
- I/O and un-cached memory
- Distributed interrupts
- DVFS controller

### Miscellaneous Tile
- Debug interface
- Performance counters access
- Coherent DMA
- Shared peripherals (UART, ETH, ...)

### Memory Tile
- Independent DDR Channel
- LLC Slice
- DMA Handler
ESP Software Socket

- **ESP accelerator API**
  - Generation of device driver and unit-test application
  - Seamless shared memory

```
/*
  * Example of existing C application
  * with ESP accelerators that replace
  * software kernels 2, 3 and 5
  */

{  int *buffer = esp_alloc(size);
    for (...) {
        kernel_1(buffer,...); /* existing software */
        esp_run(cfg_k2);      /* run accelerator(s) */
        esp_run(cfg_k3);
        kernel_4(buffer,...); /* existing software */
        esp_run(cfg_k5);
    }
    validate(buffer);      /* existing checks */
    esp_cleanup();         /* memory free */
}
```
ESP
The open-source heterogeneous system-on-chip platform
SystemC and C/C++ Accelerator Design Flow
In Summary: ESP for Open-Source Hardware

• We contribute ESP to the OSH community in order to support the realization of
  – more scalable architectures for SoCs that integrate
  – more heterogeneous components, thanks to a
  – more flexible design methodology, which accommodates different specification languages and design flows

• ESP was conceived as a heterogeneous integration platform from the start and tested through years of teaching at Columbia University

• We invite you to use ESP for your projects and to contribute to ESP!

https://www.esp.cs.columbia.edu
Thank you from the ESP team!

https://esp.cs.columbia.edu
https://github.com/sld-columbia/esp

System Level Design Group