Prototyping RISC-V Based Heterogeneous Systems-on-Chip with the ESP Open-Source Platform

Paolo Mantovani, Giuseppe Di Guglielmo, Davide Giri and Luca P. Carloni
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
- RISC-V

SoC Integration
- I/O

Rapid Prototyping

* By Nvidia Corporation
** 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
  - RISC-V Ariane (64 bit)
  - SPARC V8 Leon3 (32 bit)
- L1 private cache
- L2 private cache
  - Configurable size
  - MESI protocol
- IO/IRQ channel
  - Un-cached
  - 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
  o Direct-memory-access
  o Run-time selection of coherence model:
    ▪ Fully coherent
    ▪ LLC coherent
    ▪ Non coherent
  o User-defined registers
  o Distributed interrupt
ESP Accelerator Socket

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

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

read/write
config
done

AXI4 bus
APB bus
IRQ

NoC

Columbia University
in the City of New York
## 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

```c
/* 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 Methodology In Practice

Accelerator Flow

- `init_accelerator.sh`
- `compute()`*
- `load()/store()`*
- `make mac-exe`
- `make mac-hls`
- `make mac-sim`
- `project.tcl`

# Generate accelerator
# Specialize accelerator
* this step is automated
* for ML applications
# Test behavior
# Generate RTL
# Test RTL
# Optimize RTL

SoC Flow

- `make sldgen`
- `make esp-xconfig`
- `make mac-barec`
- `make sim`
- `make vivado-syn`
- `esp_cfg.h`
- `make linux`
- `make fpga-run`

# Generate sockets
# Configure RISC-V SoC
# Compile bare-metal
# Simulate system
# Implement for FGPA
# Configure runtime
# Compile Linux
# Deploy prototype
ESP Accelerator Flow

Developers focus on the high-level specification decoupled from memory access, system communication, hardware/software interface.
ESP Interactive SoC Flow

SoC Integration

accelerator
accelerator
accelerator
NVDLA.org
accelerator

ESP SoC Generator

General SoC configuration:
- vStorage
  - ETH Firmware
    - No FPGA
  - ETH (192.168.1.2)
- Use SOMA
- With synchronizers

Data transfers:
- Local
  - Scatter/Gather
Cache Configuration:
- Cache Enum:
  - L2 SETS: 512
  - L2WAYS: 4
- LLC SETS: 1024
- LLCWAYS: 16
- ACC L2 SETS: 512
- ACC L2 WAY: 4

CPU Architecture:
- Core: arm72e

NoC configuration

NoC Tile Configuration

- Interconnect types:
  - mem
  - cpu

- Num CPUs: 1
- Num memory controllers: 1
- Num IDA pools: 1
- Num accelerators: 0
- Num CLBs: 1
- WR/WE lines: 0

Generate SoC config
ESP: A Flexible Platform for Open-Source Hardware

We hope that ESP will serve the OSH community as a Platform to develop software for RISC-V and accelerators for any application domain.
Thank you from the ESP team!

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

System Level Design Group

Columbia University