Virtualizing FPGAs in the Cloud

ASPLOS'20

Yue Zha

Jing Li

Presenter: 黃明瀧

Outline

Introduction

Objectives

Simultaneously solve 2 problems:

1. Bad utilization
   - Cloud providers currently allocate **physical FPGA** for user app
   - Causes **internal fragmentation**
2. Scaling up is difficult (for software devs)
   - Vendor tools focuses on single-device programming
   - Devs have to:
     - Manually partition app into multiple FPGAs
     - Manually handle inter-FPGA communication

Existing Solutions

Problems to solve: (1) Bad utilization and (2) Difficult to use multi-FPGA

- The overlay architecture
  - Create “abstract FPGA”on top of different FPGAs
  - Analogous to JVM for software
  - Does not solve (1)
- Slot-based method
  - Analogous to MIG for GPU
  - Does not solve (2)
- AmorphOS
  - 2 modes: slot-based and combined-compile
  - Does not solve (2)

Methods

Overview of ViTAL

- Carefully divide FPGA into identical physical blocks
- Compile apps into virtual blocks
- Map virtual blocks to physical blocks at runtime
- **Runtime allocation** $\Longrightarrow$ Good utilization
- Illusion of **single, infinitely large FPGA** $\Longrightarrow$ Good scale-out

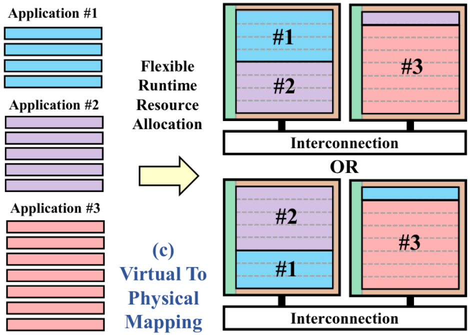

Figure 1: Virt-to-phys mapping

Programming Model

- Reuse HLS tool from Xilinx
- Programming model: **single, large FPGA**
- “Portable across FPGAs”[1].->I don’t see that true

Architecture Layer

- **Homogeneous abstraction**: “User region”blocks are the same
- “Service”and “communication”region for interconnect

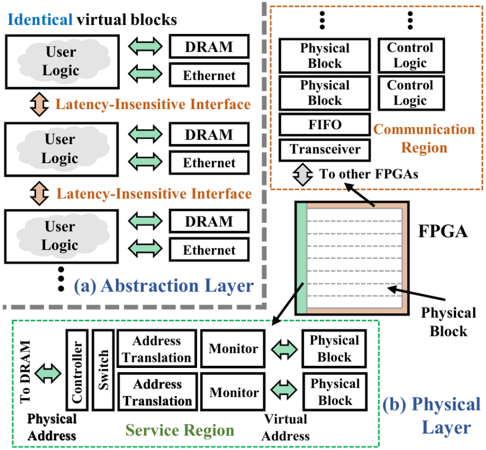

Figure 2: Architecture details

Compilation Layer

1. Synthesis (HLS $\rightarrow$ RTL $\rightarrow$ Netlist)
2. **Partition**
   - Into virtual blocks
   - Done at the **netlist level** for **max resource usage**
3. Communication interface generation
4. P&R and Relocation (at runtime)

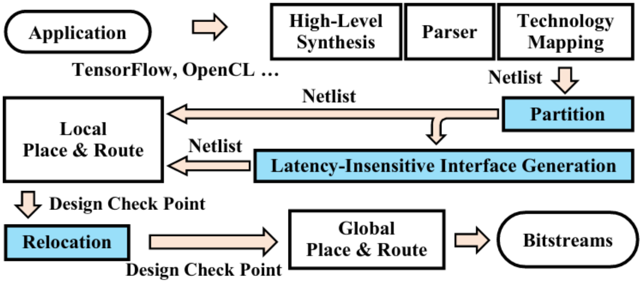

Figure 3: Compilation flow

System Layer

- Uses *partial reconfiguration* to deploy virtual blocks
- Tries to allocate **single FPGA** for one app
  - ···then 2 FPGAs, then 3 FPGAs, ···
  - Lowers communication overhead within app
- DRAM access is virtualized & managed (for isolation)

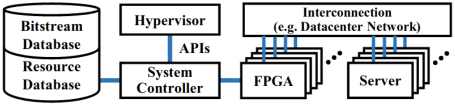

Figure 4: System controller

Evaluation

Conclusion