ReconOS: Extending OS Services Over FPGAs

Marco Platzner
Computer Engineering Group
University of Paderborn

marco.platzner@upb.de
Reconfigurable Hardware Operating Systems

- Introduce a new layer of abstraction
  - turn hardware accelerators into hardware tasks (threads)
  - rely on an operating system to schedule, place, and execute these tasks

- Motivation
  - increase productivity and portability
  - exploit partial reconfigurability
  - use reconfigurable hardware for dynamic task sets

- Operating system services
  - task management
    - load/remove/preempt/resume
    - communication, synchronization
    - scheduling
  - resource management
  - time management
Main goal: extend the **multithreaded programming model** to reconfigurable hardware

- threads communicate and synchronize using programming model primitives, e.g., semaphores, mutexes, mailboxes, shared memory
- established model in software-based systems (e.g., POSIX pthreads)
Hardware Threads

- A hardware thread consists of two parts
  - OS synchronization finite state machine
  - user logic

- A hardware thread is connected to the
  - OS on the main CPU via the OSIF
  - main memory via the MEMIF
Hardware Threads

- Function library (VHDL) for implementing the OS synchronization FSM

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;

entity fsm is
  port(
    clk, reset : in std_logic;
    oif : inout std_logic;
  );
end entity fsm;

architecture behav of fsm is
begin
  process (clk, reset)
  variable ack : boolean := false;
  begin
    if reset = '1' then
      state <= GET_DATA;
      run <= '0';
      oif_reset (oif);
      if state = GET_DATA then
        if rising_edge (clk) then
          mbox_get (oif,MB_IN,data_in,days); -- receive new packet
          next_state <= COMPUTE;
          run <= '1';
          when COMPUTE =>
            run <= '0';
            if ready = '1' then
              run <= '0';
              next_state <= PUT_DATA;
            end if;
            when PUT_DATA =>
              mbox_put (oif,MB_OUT,data_out,done);
              next_state <= LOCK;
              when LOCK =>
                mutex_lock (oif,CNT_MUTEX,days); -- acquire lock
                next_state <= READ;
                when READ =>
                  read (o_memif,i_memif,addr,count,done);
                  next_state <= WRITE
                  when WRITE =>
                    write (o_memif,i_memif,addr,count + 1,done); -- update counter
                    next_state <= UNLOCK;
                    when UNLOCK =>
                      mutex_unlock (oif,CNT_MUTEX,days); -- release lock
                      next_state <= GET_DATA;
                    end case;
                    if done then state <= next_state; end if;
                  end if;
              end case;
          end if;
        end if;
      end if;
    end if;
  end process;
end architecture behav;
```
Delegate Threads

- A SW delegate thread is associated with every hardware thread
  - calls the OS kernel on behalf of the hardware thread
Example ReconOS Architecture

- Hardware threads can be loaded / removed by partial reconfiguration
- Hardware threads use cooperative multitasking
ReconOS Toolflow

Sources
- C Source Code
  - Software Thread
  - Software Thread
- Base System Design
  - SoC Specification
  - IP-Cores
  - Target Platform Specification
- VHDL Source Code
  - Hardware Thread
  - Hardware Thread

Build Process
- Compile & Link
- System Libraries
- Root File System
- Linux Kernel
- ReconOS Device Drivers
- Vendor IP
- ReconOS System Builder
- ReconOS Hardware Components
- ReconOS VHDL Package
- Hardware Synthesis

Binaries
- Root File System
- Executable
- Kernel Image
- Kernel Modules
- Bitstream
ReconOS Versions

- **Version 1.0**
  - eCos/PowerPC, Virtex-2Pro (XUPV2P), Virtex-2 (Erlangen Slot Machine) and Virtex-4 (Avnet Virtex-4 PCIe Kit, ML403)

- **Version 2.0**
  - Linux, eCos / PowerPC, Virtex-2Pro (XUPV2P) and Virtex-4 (ML403)
  - Virtual memory support, FIFO interconnect

- **Version 3.0**
  - Linux, xilkernel / MicroBlaze, Virtex-6 (ML605)

- **Version 3.1**
  - Linux / ARM, Xilinx Zynq (Zedboard)

- **Version ?**
  - Linux / ARM, Xilinx Zynq (Zedboard)
  - Vivado HLS for hardware thread design,
    direct communication between hardware threads
The ReconOS operating system for reconfigurable computing offers a unified multithreaded programming model and OS services for threads executing in software and threads mapped to reconfigurable hardware. By semantically integrating hardware accelerators into a standard OS environment, ReconOS allows for rapid design-space exploration, supports a structured application development process, and improves the portability of applications between different reconfigurable computing systems.

ReconOS is an operating system approach for reconfigurable computing.

Multithreaded Programming Model
Easy to understand programming model based on hardware and software threads.

Active development and support
Many developers are working with ReconOS and form a community you want to join.

Extended and easy to use Toolchain
A complete and easy to use toolchain supports you while developing your ReconOS applications.
Experience with ReconOS (1)

1. ReconOS supports a step-by-step application design process

1. desktop/Linux

1a. desktop/Linux

2. Zynq/Linux-ARM

3. Zynq/Linux-ARM+accelerators
Experience with ReconOS (2)

2. ReconOS facilitates design space exploration

- Example: video object tracker
  - Virtex-4 FPGA (2 x PPC 405)
  - sw: all threads run in software
  - hw*: a number of threads run in hardware
  - sw*: a number of threads run on second (worker) CPU
Experience with ReconOS (3)

3. ReconOS enables (self-)adaptive systems

- Example: video object tracker
  - performance in [7,10] fps
  - minimize number of cores
Why isn’t ReconOS used more?

- ReconOS is an academic project
  - good as playground for research ideas, but we have limited resources for making it easily usable for others
  - out of the box only a few platforms are supported
  - more tutorials and examples needed on the website

- ReconOS is (still) a complex environment
  - requires understanding of platform FPGA architectures and tool flows
  - requires some hardware design skills (for creating hardware threads)

- Performance more important than productivity / flexibility
  - designers tend to optimize to the max, at the end they often have one big hardware thread and thrown away the OS abstractions

- The multi-threading model of ReconOS is obviously not suitable for all types of applications
OS Abstractions for Heterogeneous Nodes

- Delegate threads, cooperative multitasking (like in ReconOS) for tasks on FPGA and GPU
- Allows for preemption and heterogeneous migration of tasks
  - based on a programming pattern with check-pointing and strip-mining
Scheduling for Heterogeneous Nodes: HETSCHED

- Experiment
  - sets of 32 tasks: Heat Distribution, Correlation Matrix, Gauss Blur, Markov Chain
  - all tasks implemented on CPU, FPGA, GPU
  - all schedulers are work-conserving

- Based on task runtimes
- Based on task affinities
- Preemption and heterogeneous migration
Summary: OS Services for FPGAs

- ReconOS: multithreaded programming for software and hardware
- Heterogeneous node: preemption and heterogeneous migration
- Does ReconOS get software programmers on FPGAs?
- Which OS services are useful for FPGAs in …
  - embedded systems
  - high-performance computing
  - warehouse scale computers

Thank You!
Questions?