Reconfigurable Computing at UQ

Prof. Neil Bergmann
School of ITEE, University of Queensland
Contents

• FPGAs and RSOC
  – Motivating our technology choice

• HW/SW Task Migration
  – Managing Computing & Communications

• OCF and RC
  – Using hardware accelerators transparently

• OS support on RSOC
  – For high speed real time systems
University of Queensland

Research intensive University, in top 4 (out of 40) for research in Australia,

THES list of world top 50,

Shanghai Jiao Tong University World World Rankings: Top 9-17 in Asia-Pacific, 101-152 in World
Contents

• FPGAs and RSOC
  – Motivating our technology choice

• HW/SW Task Migration
  – Managing Computing & Communications

• OCF and RC
  – Using hardware accelerators transparently

• OS support on RSOC
  – For high speed real time systems
FPGA
Programmable Hardware
Largest around 10M programmable gates
Gate functions & interconnection customised by downloading bit stream to RAM cells:
Reconfigurable System-on-Chip

System of chips on a board

System of modules hardwired on a chip

System of modules reprogrammed on an FPGA
System-on-Chip ASIC

- IP Blocks from third parties
- Soft IP (VHDL) or Hard IP (Mask Layout)
- Designer selects and places
- Also designs some glue logic
- Maybe designs some custom peripherals
- Hard to verify and test
Reconfigurable System-on-Chip

- Some built-in vendor functions – processor, memory, multipliers
- Soft (VHDL) IP blocks, can be edited, some “open-source”
- Hard (fixed layout) IP blocks, not designed to be edited
- Design adds custom blocks
- Easier to test & verify, can be debugged.
Contents

• FPGAs and RSOC
  – Motivating our technology choice

• HW/SW Task Migration
  – PhD Project of Mr Ian Clough

• OCF and RC
  – Using hardware accelerators transparently

• OS support on RSOC
  – For high speed real time systems
HW/SW Task Migration

• RSOC allows tasks to run in HW or in SW
• Can we move running tasks between HW and SW, as application and environmental demands change?
Computational Model

- Dataflow model
- Tasks have data streams as input and output
- Streams can be divided into packets, which could vary from words to video frames
- Computation consists of tasks and logical connections between tasks.
Tasks

• Can be hardware or software
• Self-contained – all communications happens over I/O streams, no global data.
• Can be implemented in HW or SW
• For software, Linux-based tasks, using “pipes” for I/O
• For hardware, custom logic blocks, using Xilinx FSL for I/O
Network on Chip

- **SW Task 1**
- **SW Task 2**
- **SW Task 3**
- **HW Task 4**
- **HW Task 5**

**SW Router**

**HW Router**

Input/Output (I/O) to Other Chips
Task Migration

- Stop sending inputs to OLD
- Execute until migration point
- Output completed results
- Transfer any state to NEW
- Redirect inputs to NEW
Progress

- Network on chip under construction
- Migration protocols being developed
Contents

• FPGAs and RSOC
  – Motivating our technology choice

• HW/SW Task Migration
  – Managing Computing & Communications

• OCF and RC
  – PhD Project of Mr Adel Alyousef

• OS support on RSOC
  – For high speed real time systems
Problem

• The same software can run on many different platforms, perhaps with a recompilation and with new libraries.
• How can we use the same software with and without FPGA-based accelerators?
Open Crypto Framework

- Crypto accelerator chips are complex to use, but this can be helped by device drivers.
- OCF was designed to allow UNIX applications to use crypto accelerator chips if available, use software when they are not, also load balance.
- Originally in FreeBSD, now in Linux.
OCF

• OCF is not restricted to crypto applications, could be used for any hardware accelerators.
• This project investigates OCF to interface to FPGA-based hardware accelerators
• The project uses FPGA-based crypto accelerators, but could be extended to other function.
Logical Architecture

Application

OCF

Device Driver
Software
3DES

Device Driver
Hardware
3DES

Device Driver
Hardware
3DES

Device Driver
Software
AES

Device Driver
Hardware
AES

Device Driver
Software
RC6
Physical Architecture
Slot-Based

MICROBLAZE / LINUX

State (eg key, IV)

FSL
Advantages

• Accelerators for slots can be changed as data mix changes (eg more AES, less 3des)
• Load Balancing for multiple accelerators
• Works with no hardware accelerators (eg during reconfiguration)
Progress

- Architecture completed, and being debugged
- About to start trials with different data mixes
Contents

• FPGAs and RSOC
  – Motivating our technology choice

• HW/SW Task Migration
  – Managing Computing & Communications

• OCF and RC
  – Using hardware accelerators transparently

• OS support on RSOC
  – PhD Project of Mr Yi Tang
Motivation

• In multitasking embedded systems, task scheduling and context switching are software overheads.
• Hardware support could assist but:
  – How many tasks?
  – What scheduling algorithm?
• Would only be viable for large volumes
• FPGAs allows application specific support
Context Switching

• The context for a task typically consists of a snapshot of registers (including Program Counter, Stack Pointer, ...)
• Normally need to copy out registers from old task into task control block, and copy in registers from new task.
FPGA Context Switching

- For small number of tasks (8 or so), we can have multiple copies of all registers, context switch is just a change to a pointer.
- For larger numbers of tasks, we can have two register copies. Predictively copy next task ahead of context switch, save old task after context switch from separate memory.
Scheduling

- Static scheduling – assigns static priorities to decide which task should next be run. Not very flexible.
- Dynamic scheduling (EDF - Earliest Deadline First; LSF – Least Slack First) can provide better real-time performance, but requires more complex scheduling to be run more often.
FPGA Task Scheduling

• Use hardware scheduler to continually decide which is the next task to run, and to decide when that task should run.
• Can also link to interrupt handler to change task states (suspended to runnable) without needed ISR in software.
• Can use complex dynamic schedules
Progress

• Hardware Context Switch and Hardware Scheduler designs complete
• Still need to interface to software operating system (eCOS)
• Then run performance tests
• No clear metrics to compare operating system performance – throughput, latency, predictability, utilisation, ...
Conclusions

- FPGAs and RSOC have significant potential advantages over ASIC, especially in terms of design flow.
- Still much harder to design than microprocessor-based systems, but multicore processors are helping bridge the gap.
- Need both tools and designers