THE UNIVERSITY of WISCONSIN MADISON

ERCBench

An Open-Source Benchmark Suite for Embedded and Reconfigurable Computing

Daniel Chang

Chris Jenkins, Philip Garcia, Syed Gilani, Paula Aguilera, Aishwarya Nagarajan, Michael Anderson, Matthew Kenny, Sean Bauer, Michael Schulte, Katherine Compton

University of Wisconsin - Madison

Dept. of Electrical & Computer Engineering

dwchang@wisc.edu

http://ercbench.ece.wisc.edu

What is ERCBench?

- A benchmark suite for Embedded and Reconfigurable Computing (E&RC)
- Includes a wide variety of application types
  - Currently includes multimedia, cryptography and wireless communications applications
  - Modern applications that are run on E&RC devices
- Open source and freely available on our website

http://ercbench.ece.wisc.edu

Currently Implemented Applications

Audio Processing

- FFT
- Ogg Vorbis

Image Processing

- Face Detection
- JPEG2000

Video Processing

- Xvid

Cryptography

- AES
- Blowfish
- DES
- Elliptic Curve Cryptography
- SHA

Wireless Communication

- Low-density parity-check
- Turbo Decoder
- Viterbi Decoder

Why ERCBench?

- High-quality research in E&RC requires good benchmarks
  - Need up-to-date applications for evaluation
  - Poor benchmarks could lead to inaccurate results
  - Some existing suites have unrealistically small applications
- There are no open-source mixed hardware/software benchmarking suites
  - All existing suites are purely hardware or purely software
  - Several suites charge fees

Other Benchmarking Suites

Hardware-only

- RAW
- Honeywell
- OpenCores
- Groundhog
- RCG
- MCNC
- Toronto 20

Software-only

- SPEC2006
- EEMBC
- MiBench
- MediaBench I/II
- GraalBench

Mixed

- OpenFPGA*

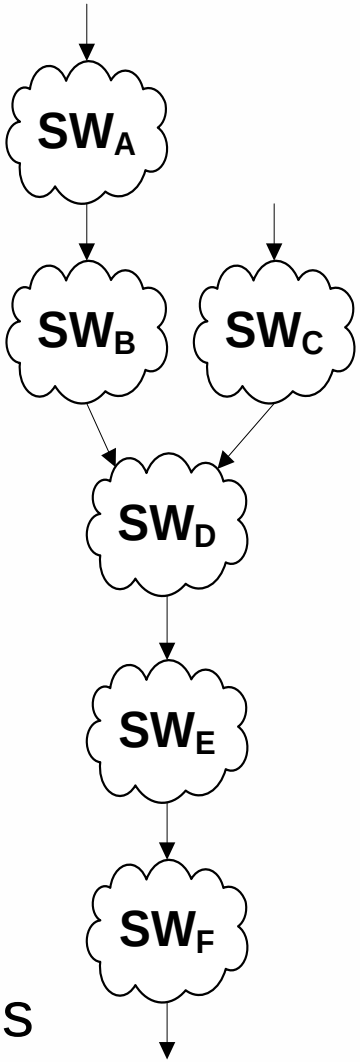
* Not yet available

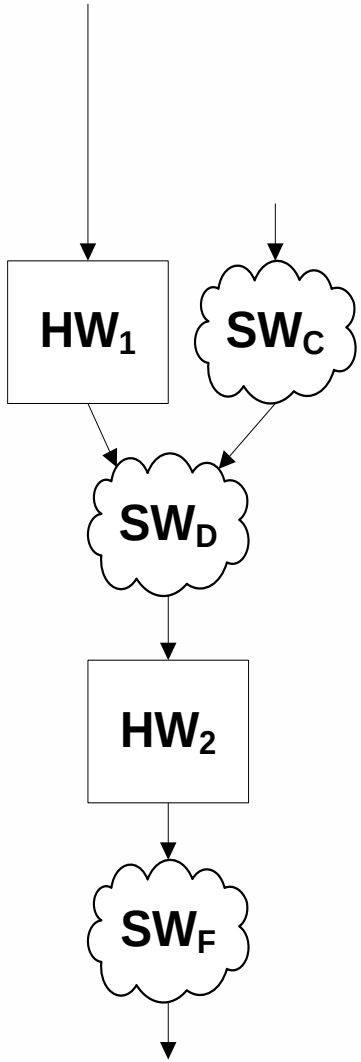
Outline

- Design methodology
- Hardware/software interface
- Using ERCBench
- Example application analysis
  - Xvid, Turbo Decoder
- Future ERCbench plans

Application Profiling

- Profile app. (gprof)
- Find functions with most execution time
- Check function code for HW suitability
- Check function data communication for HW suitability
- Chosen functions implemented as HW
  - For full app, HW modules used by surrounding SW





Implementation & Verification

HDL Implementation

- Candidate functions manually implemented in hardware using synthesizable Verilog
  - Most modules are parameterized
  - Add signals to allow for synchronization with hardware/software interface

Pre-synthesis verification

- Modules verified for functionality in ModelSim
- Use test vectors supplied as part of an application's standard when possible

Synthesis & Post-synthesis Verification

Synthesis

- Modules synthesized for both standard cell and an FPGA to obtain delay and area estimates
  - Standard cell synthesized with TSMC 65 nm
  - FPGA synthesized for Xilinx Virtex-5 LX110

Post-synthesis Verification

- Standard cell and FPGA synthesized designs are re-verified for functionality in ModelSim

Reintegration with Original Software

Reintegration

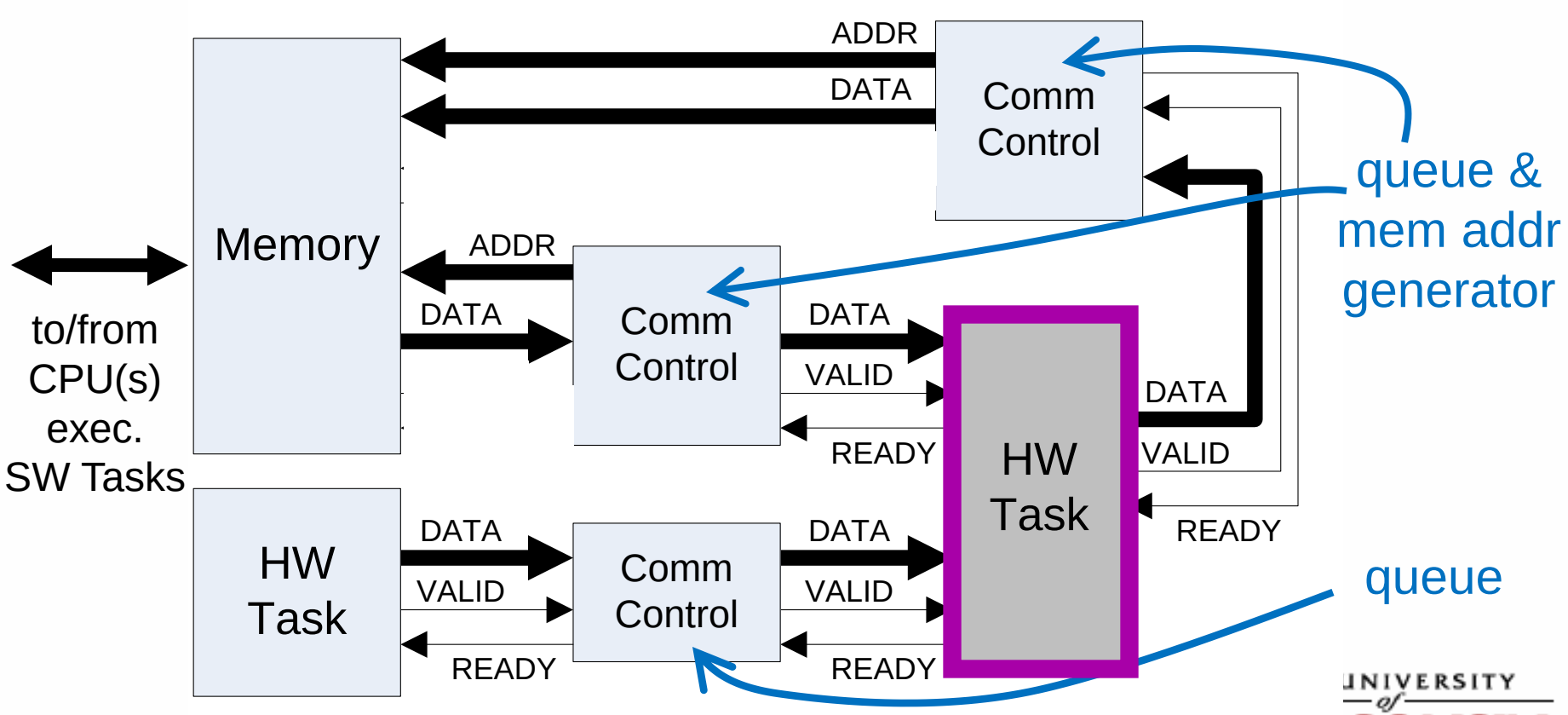
- Steps are system-dependent
  - We use memory-mapped I/O
- Insert instructions or function calls into SW to:
  - Configure hardware (if using reconfigurable hardware)
  - Initialize parameters of the hardware circuit
  - Set up data communication or send input data to HW
  - Test for HW computation completion
  - Receive result data from HW
- We provide original SW, HW kernels and a hybrid version that integrates HW using our setup

Module Interface

- Use a standardized communication interface designed for streaming data
  - Queue-based, producer/consumer paradigm
  - Hand-shaking for inter-task communication
  - Each data port has own interface
  - Flexible and easy to modify
- HW module uses same interface to connect to either HW or SW through a queue
  - Queue module a bit different for SW connection…
- Modules can directly interface without queues
  - HW-queue interface matches at both queue ends

Example: Module Interface

- Highlighted task receives inputs from HW task and SW task
- Highlighted task produces data for SW task
  - In example, HW/SW communication occurs via shared memory



Communication Controllers

- Provide data buffering between tasks
  - Contains FIFO queue and handshaking logic
  - Handles a single stream of data
    - Multiple streams? Multiple communication controllers.
- Currently assume input data width matches output data width (parameterizable), single clock
  - Working on implementation that allows different data widths at each end and that supports two clocks
- Can augment controllers
  - Simple FSM and a few registers to provide a generic memory address generator
  - Interface to memory to issue read/write requests

Using ERCBench

- Benchmarks available on project website
  - Zip files with Verilog source code, test benches, and documentation with usage directions
  - Hybrid benchmarks also point to original software code and modified SW code for Alexandrite system
- Hybrid benchmark SW may require modification
  - HW/SW interface very specific to individual systems
  - In many cases, can convert our code to new system
- Wide variety of studies possible
  - Testing place and route algorithms
  - Reconfigurable computing system research
  - Comparing application implementations
  - Etc…

Xvid Analysis

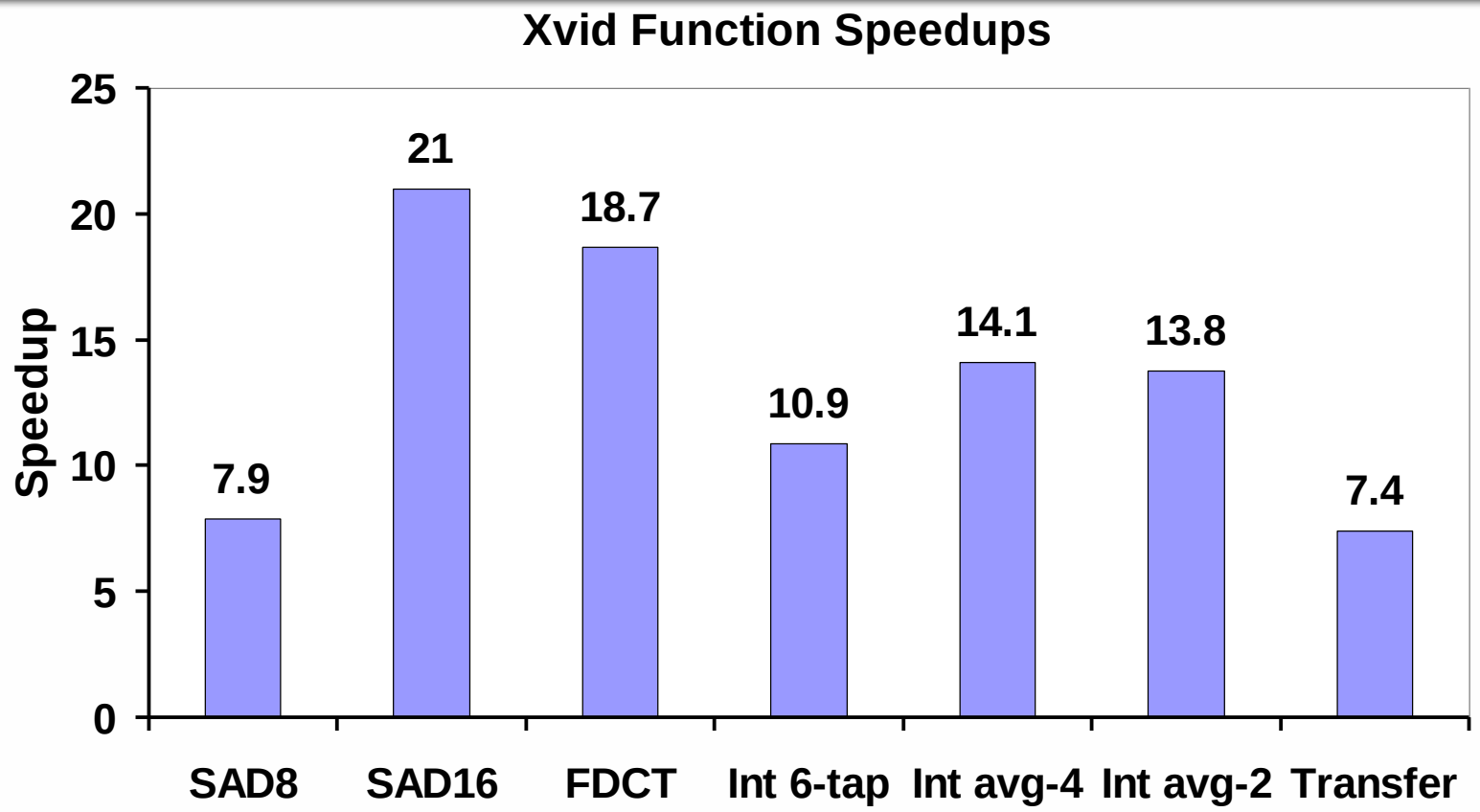
- Xvid is a hardware/software hybrid benchmark
- Seven of the top functions in original software implementation converted to Verilog

Function Name	% of SW-only Exec. Time
SAD8	31.3
SAD16	10.0
FDCT	7.2
Interpolate 8x8 6 tap	7.3
Interpolate 8x8 avg-4	6.2
Interpolate 8x8 avg-2	4.3
Transfer 8 to 16 Sub	2.3
Total	**68.6%**

System Evaluation Using Xvid

- Xvid benchmark used in reconfigurable computing research examining HW scheduling
- System contained multiple CPUs and a shared reconfigurable hardware coprocessor
  - CPUs similar to 900 MHz ARM processor
  - Reconfigurable hardware similar in performance to Xilinx Virtex-5 FPGA
- System was modeled in a full-system simulator
- Executed hybrid Xvid benchmark (full application, with both software and hardware components)

System Evaluation Using Xvid



- Full application has 2.7x speedup over software
  - Application control and some functions still in SW
  - Uses ~20% of the resources on an LX110

Turbo Decoder Analysis

- Hardware-only benchmark that implements the entire communication algorithm
  - Parameterizable # of Processing Elements (PEs)
- Studied differences between FPGA and standard cell implementations
  - Synthesized 2, 4 and 8 PEs on both technologies
  - Required design throughput: **100 Mbps**

Turbo Decoder FPGA Analysis

# of PEs	Frequency (MHz)	Target Frequency	# of Slice Registers	# of Slice LUTs
2	99.41	200	9615	16832
4	99.41	100	19173	33234
8	103.16	50	38289	65852

- Resource usage increases linearly with more PEs
- Two PE design does not meet the 100 Mbps requirement
- Four PEs nearly achieve 100 Mbps while using fewer resources than eight PEs

Turbo Decoder Standard Cell Analysis

# of PEs	Frequency (MHz)	Target Frequency	Cell Area (μm^2)	Normalized Energy
2	200	200	70308	1.00
4	100	100	140563	0.49
8	50	50	274122	0.29

- Unlike FPGA designs, all three designs hit the 100 Mbps requirement
- Four and eight PEs have sufficient throughput to allow voltage scaling and still hit 100 Mbps
  - Normalized energy shows four and eight PE designs can achieve goal for less energy
  - Comes at area cost

Future Work

- More applications to be added…
  - Reed Solomon, Search-and-Sort, Ray Tracing, etc.
- Some HW-only benchmarks to be integrated into applications to create hybrid HW/SW benchmarks
  - Candidates include JPEG2000 and Turbo Decoder
- Extending module interface to support multiple clocks and different data widths at each end
- Convert hybrid benchmarks so that kernels also implemented in OpenCL/CUDA

Conclusion

- Introduced benchmark suite for embedded and reconfigurable computing (E&RC) research
  - Include both hardware circuits and hybrid hardware/software applications
  - Based on real world, high-performance embedded computing applications
  - Designed to facilitate variety of E&RC research topics
- Benchmarks are open-source and publicly-available on ERCbench website
  - http://ercbench.ece.wisc.edu
- Interested in contributing benchmarks to ERCbench? Contact us!

Questions?

The ERCBench suite can be found at:

http://ercbench.ece.wisc.edu