EECS150 - Digital Design

Lecture 13 - Accelerators

March 5, 2013

John Wawrzynek

Motivation

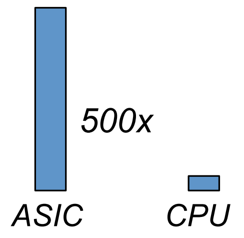
- 90/10 rule:
  - Often 90 percent of the program runtime and energy is consumed by 10 percent of the code (inner-loops).
  - Only small portions of an application become the performance bottlenecks.
  - Usually, these portions of code are data processing intensive with relatively fixed dataflow patterns (little control): cryptography, graphics, video, communications signal processing, networking, ...
  - The other 90 percent of the code not performance critical: UI, control, glue, exceptional cases, ...

  Hybrid processor-core hardware accelerator

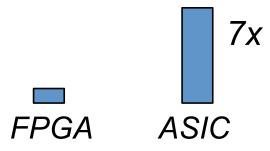
  - Hardware accelerator/economizer implements specialized circuits for inner-loops.
  - Processor packs the noncritical portions (90%), 10% of the computation into minimal space.

Energy Efficiency of CPU versus ASIC versus FPGA

Rehan Hameed, Wajahat Qadeer, Megan Wachs, Omid Azizi, Alex Solomatnikov, Benjamin C. Lee, Stephen Richardson, Christos Kozyrakis, and Mark Horowitz. Understanding sources of inefficiency in general-purpose chips. SIGARCH Comput. Archit. News, 38:37–47, June 2010.



Ian Kuon and Jonathan Rose. Measuring the gap between fpgas and asics. In Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate arrays, FPGA '06, pages 21–30, New York, NY, USA, 2006. ACM

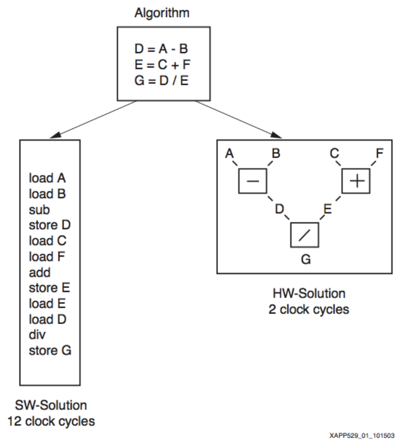


∴ FPGA : CPU = 70x

Similar story for performance efficiency

Why is HW more efficient than processors?

- Performance/cost or Energy/op
  1. exploit problem specific parallelism, at thread and instructions level
  2. custom "instructions" match the set of operations needed for the algorithm (replace multiple instructions with one), custom word width arithmetic, etc.
  3. remove overhead of instruction storage and fetch, ALU multiplexing



What about FPGAs?

“System on Chip” Example

- Three ARM cores, plus lots of accelerators
- Targets smart phones

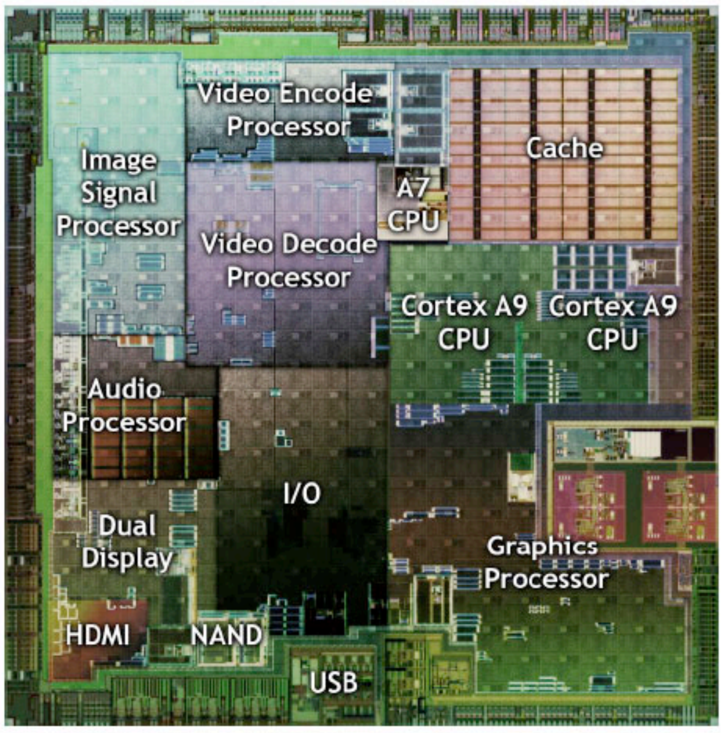
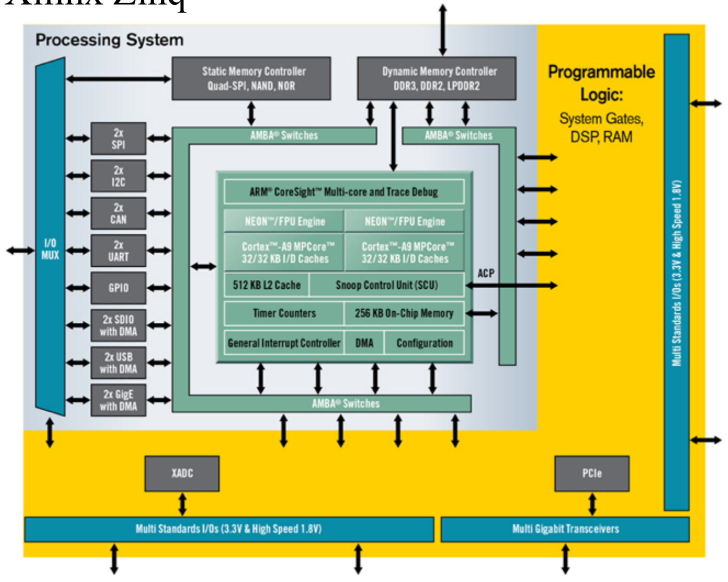


Figure 1 - NVIDIA Tegra 2 System on a Chip

Processors in FPGAs

Xilinx Zinq

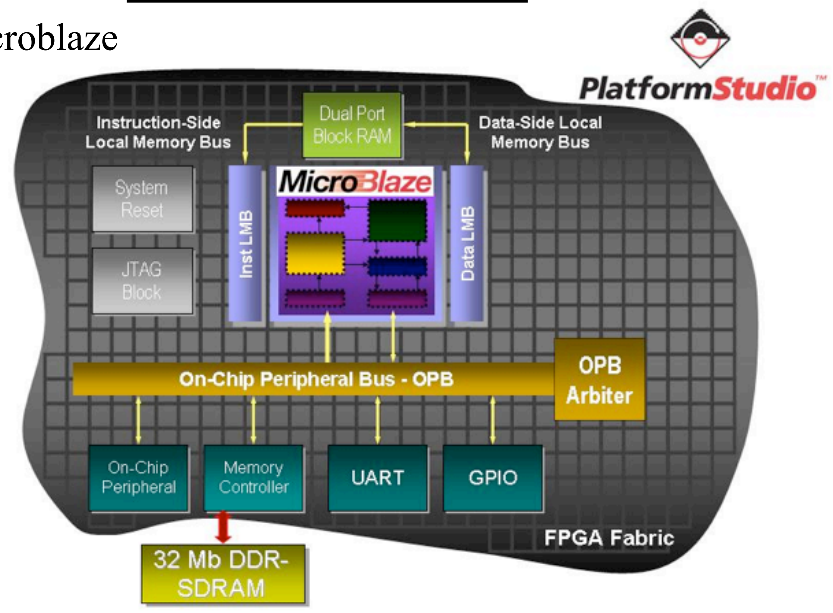

Processing System
Static Memory Controller Quad-SPI, NAND, NOR
Dynamic Memory Controller DDR3, DDR2, LPDDR2
Programmable Logic: System Gates, DSP, RAM
AMBA® Switches
AMBA® Switches
I/O MUX
2x SPI
2x I2C
2x CAN
2x UART
GPIO
2x SDIO with DMA
2x USB with DMA
2x GigE with DMA
ARM® CoreSight™ Multi-core and Trace Debug
NEON™/FPU Engine
NEON™/FPU Engine
Cortex™-A9 MPCore™ 32/32 KB I/D Caches
Cortex™-A9 MPCore™ 32/32 KB I/D Caches
512 KB L2 Cache
Snoop Control Unit (SCU)
ACP
Timer Counters
256 KB On-Chip Memory
General Interrupt Controller
DMA
Configuration
AMBA® Switches
XADC
PCIe
Multi Standards I/Os (3.3V & High Speed 1.8V)
Multi Standards I/Os (3.3V & High Speed 1.8V)
Multi Gigabit Transceivers

- Dual ARM Cortex™-A9 MPCore
  - Up to 800MHz
  - Enhanced with NEON Extension and Single & Double Precision Floating point unit
  - 32kB Instruction & 32kB Data L1 Cache
- Unified 512kB L2 Cache
- 256kB on-chip Memory
- DDR3, DDR2 and LPDDR2 Dynamic Memory Controller
- 2x QSPI, NAND Flash and NOR Flash Memory Controller
- 2x USB2.0 (OTG), 2x GbE, 2x CAN2,0B 2x SD/SDIO, 2x UART, 2x SPI, 2x I2C, 4x 32b GPIO
- AES & SHA 256b encryption engine for secure boot and secure configuration
- Dual 12bit 1Msps Analog-to-Digital converter
  - Up to 17 Differential Inputs
- Advanced Low Power 28nm Programmable Logic:
  - 28k to 235k Logic Cells (approximately 430k to 3.5M of equivalent ASIC Gates)
  - 240kB to 1.86MB of Extensible Block RAM
  - 80 to 760 18x25 DSP Slices (58 to 912 GMACS peak DSP performance)
- PCI Express® Gen2x8 (in largest devices)
- 154 to 404 User IOs (Multiplexed + SelectIO™)
- 4 to 12 12.5Gbps Transceivers (in largest devices)

Altera: **Dual-Core ARM Cortex-A9 MPCore Processor**

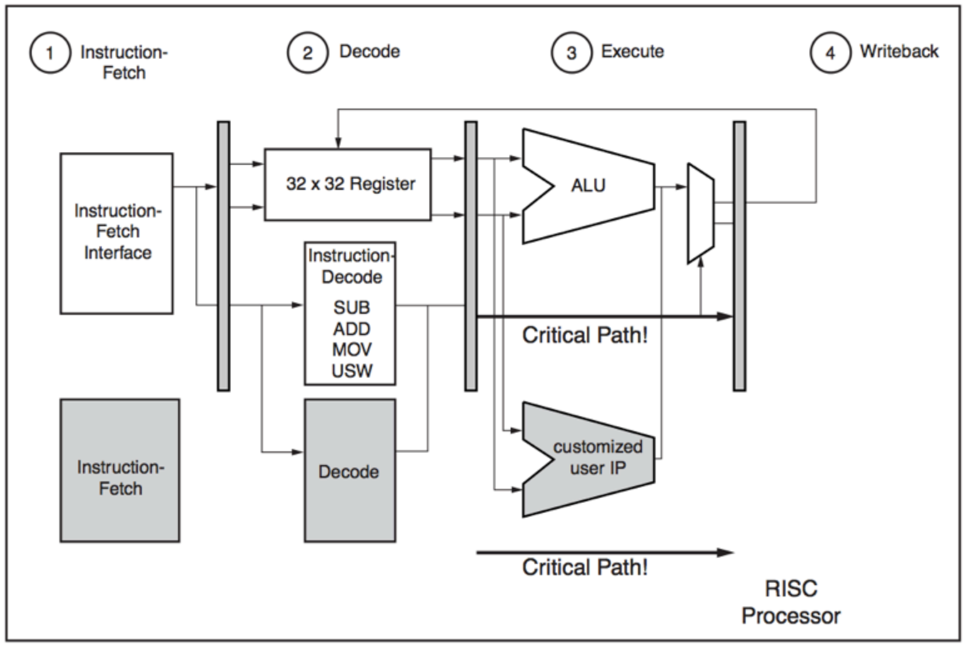
Soft Processor

Xilinx: Microblaze



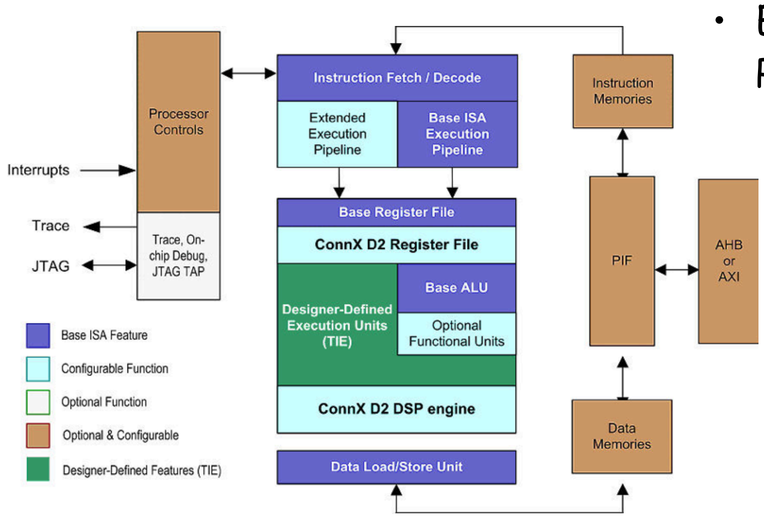
Altera: Nios, MIPS

Custom Hardware in the Pipeline



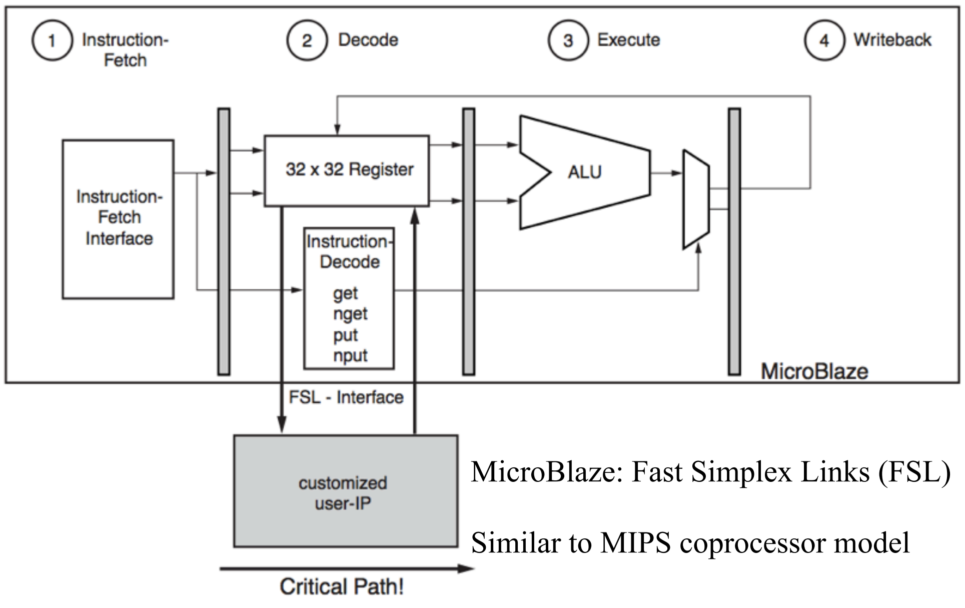
XAPP529_03_101503

Custom Instructions



- Example: Tensilca Product
  - Special language TIE is used for defining special function units
  - Custom architecture automatically compiled
  - Compiler support challenging

Tightly Coupled Co-processor

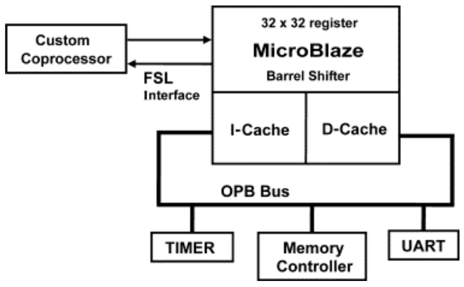


MicroBlaze: Fast Simplex Links (FSL)

Similar to MIPS coprocessor model

XAPP529_04_101503

MicroBlaze Fast Simplex Links



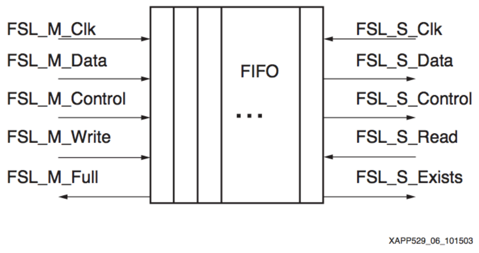
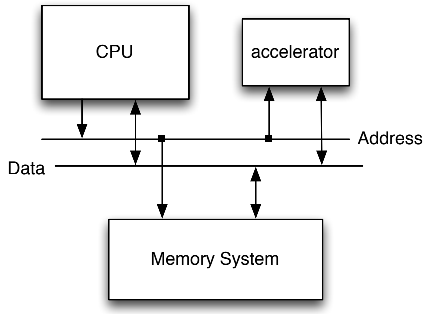


Figure 6: **FSL interface**

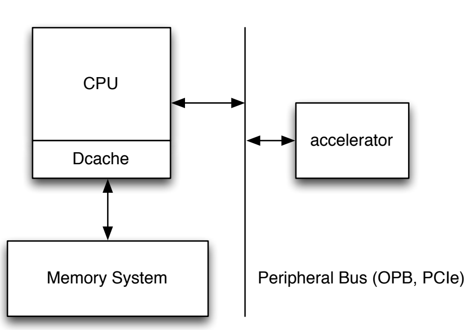
```
// Blocking Data Read and Write to Local Link no. id
microblaze_bread_datafsl(val, id)
microblaze_bwrite_datafsl(val, id)

// Non-blocking Data Read and Write to Local Link no. id
microblaze_nbread_datafsl(val, id)
microblaze_nbwrite_datafsl(val, id)

// Blocking Control Read and Write to Local Link no. id
microblaze_bread_cntlfsl(val, id)
microblaze_bwrite_cntlfsl(val, id)

// Non-blocking Control Read and Write to Local Link no. id
microblaze_nbread_cntlfsl(val, id)
microblaze_nbwrite_cntlfsl(val, id)
```

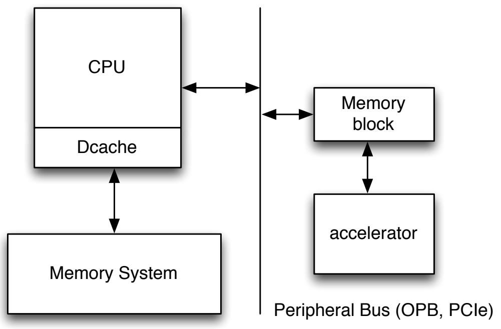
Memory Mapped Accelerator



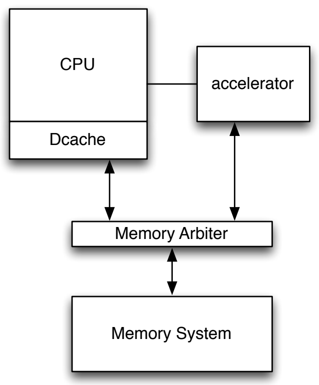
- Memory mapped control/data registers

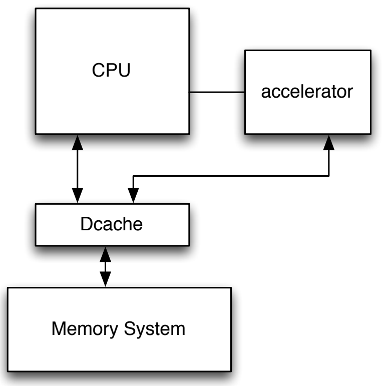
Memory Mapped Accelerator Common Variations





CPU/Accelerator Shared Memory





- Processor instructs accelerator to independently access memory and perform work
- How does processor synchronize with accelerator (how does it know when it is done)
- Data Cache on CPU creates "coherency" issue
- What about a cache in the accelerator?