18-643 Lecture 4: 
SoC FPGA

James C. Hoe  
Department of ECE  
Carnegie Mellon University

Housekeeping

• Your goal today: be one with the SoC way
• Notices
  – Handout #2: lab 0, due noon, 9/9
  – Make friends, make teams, due noon, 9/9
  – Handout #3: lab 1, due noon, 9/23
• Readings
  – Ch 2 and 6, The Zynq Book (skim Ch 3&10 if interested)
What is a System-on-Chip (SoC)?

“In short, for larger systems, the term system on a chip is hyperbole, indicating technical direction more than reality . . .” --- Wikipedia
SoC is the Natural Course of Scaling

- There was a time when even a “component” is a “chip set”
- Eventually, entire component fit in one chip
  ⇒ system ≡ board-level connection of chips
  e.g. CPU, cache, DRAM, controller, I/O, . . .
- Later when more could fit on one chip
  – integrate for performance (CPU+cache)
  – integrate for cost/reliability (reducing parts count)
- Still more and more (including analog)
  Your “system” is a component in another system

SoC is IP-Based Chip Design

- Complexity wall
  – designer productivity grows slower than Moore’s Law on logic capacity
  – cannot scale design team size to compensate
  ⇒ must stop designing individual transistors
- A chip design as a collection of IPs
  – each IP fits in a manageable design complexity
  – IP integration fits in a manageable design complexity

Analogous to board-level chip integration in the 80s/90s
SoC is Systematic Interconnect

- More IPs, more elaborate IPs ⇒ intractable to design wires at bit- and cycle-granularity
- On-chip interconnect standards (e.g. AMBA)
  - familiar “bus” abstraction (addr/data, master/slave)
  - some even with cache-coherence
  - point-to-point signaling underneath, even full-fledge network-on-chip
- Plug-and-play integration of interface-compatible IPs

  Analogous to expansion bus and I/O cards on a motherboard

SoC is Heterogeneous Computing

- SoC integrates diverse specialized functionalities
- When power is more precious than transistor count, heterogeneous computing trades off
  - relatively small cost of unused transistors (gated off) at any given time
  versus
  - large power and performance gains from having the right tool at the right time
SoC is HW/SW Co-Design

- SoC mixes HW and SW computing substrates
- An application is partitioned for mapping to
  - HW: what needs to be better than in SW
  - SW: everything else
- SW is the heart and soul
  - in control of HW
  - enables product differentiation
- SW can be harder than HW (Is this surprising?)
  - embodying most of the complexity
  - dominating development time/effort

Meet the Zynq SoC
Yeap, it is a system (where is the FPGA?)

Actually, it is still mostly fabric even for a small fabric (XC7z020)
Range of Zynq Offerings

PS/PL Interface Options

Zynq terminology:
PL=programmable logic
PS=processing system

something for every occasion; boon or bane?
Concept: Bus and Transactions

- Bus connects bus devices
  - masters: devices who initiate transactions
  - slaves: devices who respond to transactions
- Transaction based on a memory-like paradigm
  - “address”, “data”, “reading vs. writing”
  - master issues read/write transaction to an address
  - each slave is assigned an address range to respond in a “memory-like” way, i.e., returning or accepting data

AXI is the standard interface in Zynq

Concept: Memory Mapped I/O

- Think of normal ld/st as how processor “communicates” with memory
  - ld/st address identifies a specific memory location
  - ld/st data conveys information
- Can communicate with devices the same way
  - assign an address to register of external device
  - ld/st from the “mmap” address means reading/writing the register
  - BUT remember, it is not memory, e.g.,
    - additional side-effects
    - not idempotent
Fabric Module as AXI Slave

- ARM core issues ld/st instructions to addresses corresponding to “mmapped” AXI device registers aka programmed I/O or PIO
- Nothing is simpler
- Very slow (latency and bandwidth)
- Very high overhead
  - ARM core blocks until ld response returns
  - many 10s of cycles
  
  Best for infrequent, simple manipulation of control/status registers

Fabric Module as AXI Master

- Fabric can also issue mmap read/write as master
- AXI HP
  - dedicated 64-bit DRAM read/write interfaces
    - fastest paths to DRAM (latency and bandwidth)
  - no cache coherence
    - if data shared, ARM core must flush cache before handing off
    - major performance hiccup from (1) flush operation and (2) cold-cache restart
    - best for fabric-only data, DRAM-only data, or very coarse-grained sharing of large data blocks
Fabric Module as ACP Master

- “Accelerator Coherence Port”
- Fabric issues memory read/write requests through ARM cores’ cache coherence domain
- Shortest latency on cache hits
  - ARM core could even help by prefetching
  - if not careful, ARM cores and fabric could also interfere through cache pollution
- Not necessarily best bandwidth (only one port)

Best for fine-grained data sharing between ARM cores and fabric

DMA Controller

- AXI-slave programming interface
  - programmable from ARM core and fabric
  - source and dest regions given as <base, size>
  - source and dest could be memory (cache coherent) or mmapped regions (e.g., ARM core scratch-pad or mmapped accelerator interface)
- Need to move large blocks to “amortize” DMA setup costs (PIO writes)
- Corollary: need to start moving well ahead of use

Best for predictable, large block exchanges
Remember this poor guy?

- everything else is soft
- two hierarchies of soft-logic busses (slow and slower)
- special on-chip memory (OCM) port allows ld/st directly into fabric
- CoreGen Library of IPs to hang off the busses

Vivado Demo
Vivado/Zynq Way of Thinking

- Processor-first development
- All peripherals AXI-interfaced
- Extensive library of infrastructural and functionality IPs
- You could add custom-built soft IPs
  - compile from C and Simulink, or hack RTL
  - adhere to pre-defined interface schemes
- You could program pins-and-gates too (ISE) if you enjoy “grunt work”

Parting Thoughts

- SoC is a way to manage complexity if we are to consume Moore’s Law transistors
- SoC’ness complements FPGA traditional value-added
  - hardware performance that is flexible
  - fast design turnaround (time-to-market)
  - low NRE investments
  - in-the-field update/upgrades
- Tools is an integral part of the picture . . .
  . . . will see more on this
The Zedboard Kit

- ECE is loaning to you
  - 1x Avnet ZedBoard 7020 baseboard
  - 1x 12 V power supply
  - 1x 4-GB SD Card (don’t need to use it)
  - 2x Micro-USB cable (1 in box; 1 additional)
  - 1x USB Adapter (don’t need to use it)
  - 1x set of documentations
  - 1x voucher to Xilinx Vivado® Design Edition license for 7Z020-only (keep/use it or give it back)

- You will treat it like your grade depended on it
- You will return all of it in perfect condition, or else

Looking Ahead

- Lab 1 (wk3/4): get cozy with Vivado
  - most important: learn logic analyzer and eclipse debugger
- Lab 2 (wk5/6): meet Vivado HLS
  - most important: decide if you would use it
- Lab 3 (wk7/8): hands-on with AFU
  - most important: have confidence it can work
- Project . . .