TEMAC: Tri-mode Ethernet MAC

Chris Fletcher
CS294-48 Show and Tell
Protocol Implementation Patterns

• Datapath
  • General Patterns: Feed Forward / Pipe-and-filter, Duplex (pattern?)
  • Reliability: CRC (pattern?)
  • Data Width conversion
    – Pattern: Serial ↔ Parallel Conversion
    – Options: Registers, Muxes
  • Data Clock conversion
    – Pattern: Clock crossings
    – Options: Registers, Asynchronous Buffers

• Control
  • Data Width (Control – auto negotiation)
    – Patterns: FSM, Serial ↔ Parallel (MDIO), Control/Status register interface
  • MDIO
    – Pattern: Bus & Control/status register interface

• Multi-port/lane MACs
  • Patterns: “Single [Fixed] Controller” (that controls MDI on a per-port basis)
  • Multiple-data (Single Controller Multiple Data - SCMD)
Extra Slides
Design Approach

- **Objective**
  - Pass data (or packets?) from system \(\rightarrow\) sink[s]
  - Minimize number of “User constraints”

- **Patterns broken into “Protocol {Decision, Implementation}”**
Protocol Implementation

• Dictated by physical constraints
  – Optimize to system platform
  – Faithfully implement the protocol

• Building the datapath
  – How can patterns tell us how to build this?
  – Separate constraints into functional blocks
    • What order should the blocks be arranged in?
    • What other patterns must be employed based on constraints that arise from each top-level constraint? (Auto negotiation can dynamically change the protocol width)
Functional Unit Order

• Four units:
  – data width conversion, clock crossing, packet buffering, frame construction

• Arguments:
  – Clock Crossing first:
    • Clock crossing between arbitrarily separated (in frequency) clocks
    • Buffer data after the crossing so that when a packet has been clock-crossed and is fully ready, system can send 1 word / cycle guaranteed
  – Packet Buffering next
    • After frame is constructed, we would have to buffer Preamble, CRC, SFD
    • Data width can change: doing parallel $\rightarrow$ serial conversion after the buffer means that the buffer’s data width need not change dynamically
  – Frame construction & Parallel $\rightarrow$ Serial conversion last:
    • Frame fields (preamble, SFD, CRC, IFG (if you count that)) are independent of system-side data width
    • Injecting fields into a parallel$\rightarrow$serial converter along with data is natural for parallel $\rightarrow$ serial converters
    • If fields were injected into the data stream before the buffer, the final data stream wouldn’t necessarily be word-aligned to the buffer