

# Advanced Computer Systems Architecture

## Chip-Multiprocessors: Applications and Architectures

CSE 526M

Prof. Patrick Crowley

## Plan for Today

- Questions
- Today's discussion

## Outline

- IXP Tools tutorial
- MSF Operation
- MSF Receive and Transmit state machines

## Tutorial Objective

- How to create a project
- How to set parameters
- How to build a project
- How to simulate a project
- Pointers to SDK Documentation

## IXP Tools Tutorial

- [Link](#)

## MSF Operation

- Recall, the MSF breaks incoming packets into *mpackets*, which are fixed-length segments (configurable as 64, 128 or 256 bytes)
- A given mpacket can be marked as the start of packet (SOP), end of packet (EOP), or both (if the packet fits within one mpacket).
- mpackets are stored in a buffer (RBUF and TBUF) at the MSF
- The microengines are responsible for re-assembling mpackets into packets

## Receive State Machine

1. Microengine places a thread on the `RX_THREAD_FREELIST` at the MSF
2. The MSF signals the thread when it has an mpacket ready in an RBUF element, and writes that packet's *receive status words* – RBUF entry, EOP, SOP and status bits (i.e., error bits) -- into the thread's transfer registers
3. Upon receiving the signal, the microengine reads the receive status from the transfer registers
4. The microengine copies the data into memory, and notifies the MSF to release the RBUF entry

## Transmit State Machine

- Analogous to receive, except the microengine is responsible for segmenting data into mpackets and populated the TBUF populated
- The transmit code must also be concerned with flow control on the outgoing link (e.g., the microengine must wait for a free TBUF entry).

## Next steps

- Before developing the code, we need to
  - Get more assembler experience
  - See how to implement buffers with queues and rings
- These are Tuesday's topics

## Assignment

- Tuesday
  - Go through the tutorial on your own (in the lab)
  - Submit a commentary on the following:
    - Find (or write) a small piece of C code (it can be absolutely anything, completely trivial if you like) to drop into the tutorial project. Simulate that code, check that it works, and determine it's run-time and microengine utilization. In your commentary, describe your program, include the source, and describe the performance results you gathered. If you had any strange problems that kept you from simulating, describe those. (Spend no more than 45 minutes in the lab doing this.)