ECE 697J - Advanced Topics in Computer Networks

Microengine Programming II

11/20/03

Previous Class

- Basic introduction to microengine assembly
  - Syntax
  - Register allocation
  - Scope
  - Macros

Specialized Memory Operations

- Memory hierarchy
  - SDRAM
    - store bulk data (e.g. packet payloads)
  - SRAM, scratchpad
    - store header values
    - control information
    - co-ordination between microengines
- Special memory ops
  - Buffer pool manipulation
  - Processor co-ordination
    - Bit testing
    - Memory locking
  - Atomic memory operations

Specialized Memory Operations

- Buffer pool manipulation – similar to mbufs
  - Automate allocation/deallocation of buffers in SRAM
  - Buffers assigned to linked list (limit 8)
  - Push buffer → sram [push,--,addr1,addr2,listnum]
    - SRAM address = addr1+addr2
  - Pop buffer → sram [pop,$xfer,--,--,listnum]
    - $xfer holds first buffer
- Atomic memory increment
  - Scratchpad ideal for storing counters (why ?)
  - Atomic operations essential (why ?)
  - scratch [incr,--,addr1,addr2]
    - Scratchpad address = addr1+addr2
  - Atomic increment guaranteed by hardware

Processor Co-ordination

- Bit testing (test and set)
  - Atomic operations on individual bits in SRAM and scratchpad
  - scratch [bit_wr,$xfer,addr1,addr2,op]
    - Scratchpad address = addr1+addr2
    - $xfer contains mask (why ?)
    - “op” specifies operation to be performed

Operation	Meaning
set_bits	Set the specified bits to one
clear_bits	Set the specified bits to zero
test_and_set_bits	Place the original word in the read transfer register, and set the specified bits to one
test_and_clear_bits	Place the original word in the read transfer register, and set the specified bits to zero

Processor Co-ordination

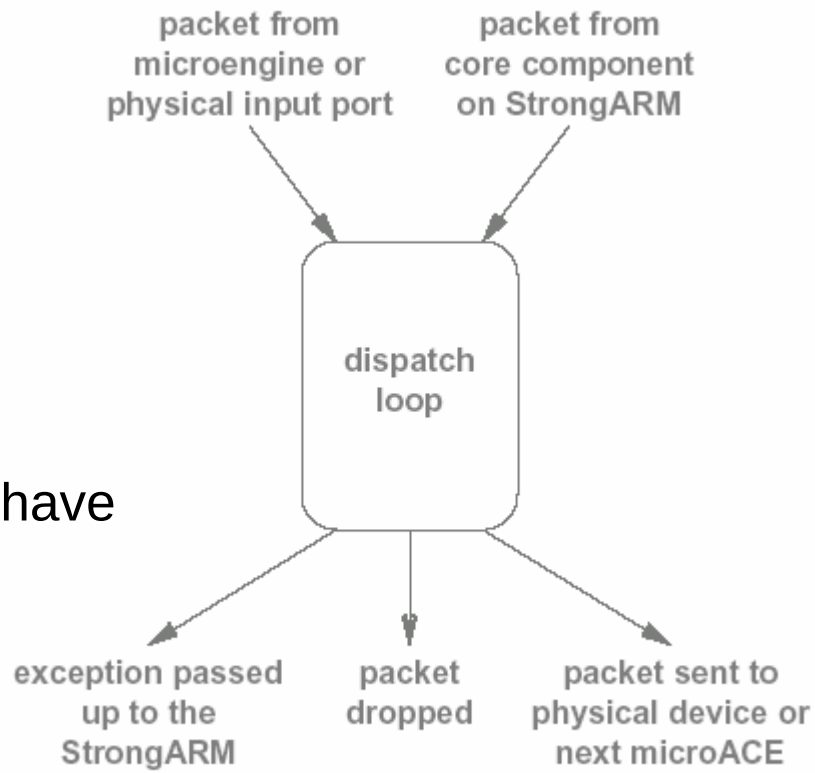
- Memory locking
  - Yet another mechanism
  - SRAM word can act as mutual exclusion lock
  - sram [read_lock,$xfer,addr1,addr2,count], ctx_swap
    - "count" words
    - "ctx_swap" → microengine must block until lock acquired
    - SRAM address = addr1+addr2
  - sram [write_unlock,$xfer,addr1,addr2,count], ctx_swap
    - Similar to read lock
  - Memory location must be unlocked!
    - sram [unlock,--,addr1,addr2]
  - Eight addresses can be locked at one time

Control and Status Registers

- > 150 CSRs
- To access microengine CSRs
  - csr [cmd,$xfer,CSR,count]
    - "cmd" can be read/write from/to "$xfer"
    - "count" words to transfer (= 1 for 32/64 bit transfer!)
  - yet another way: fast_wr [immediate_data,CSR]
    - Use fast path through the FBI
    - "immediate_data" is a 10 bit constant
  - to access local CSRs
    - local_csr_wr[CSR,src]
    - local_csr_rd[CSR]
    - "src" → register or an immediate value
    - one cycle access

Packet Queues

- Source and sink functions implemented as macros
- Macros return "IX_BUFFER_NULL" if no packet available from source
- Sources processed in round robin fashion
- Priority implemented by macros
  - e.g. packets from input port have higher priority



Accessing Packet Headers

- Allocate transfer registers to hold packet header

```
/* Allocate eight SDRAM transfer registers to hold the packet header */
xbuf_alloc[$$hdr, 8]
```

- Compute location of packet within buffer

```
/* Compute the SDRAM address of the data buffer */
Buf_GetData[base, dl_buffer_handle]

/* Compute the byte offset of the start of the packet in the buffer */
DL_GetBufferOffset[offset]

/* Convert the byte offset to SDRAM words by dividing by eight */
/* (shift right by three bits) */
alu_shf[offset, --, B, offset, >>3]
```

Accessing Packet Headers

- Load header into transfer registers

```
/* Load thirty-two bytes of data from SDRAM into eight SDRAM */
/* transfer registers.  Start at SDRAM address  base + offset */
sdram[read, $$hdr0, base, offset, 4]
```

- Free buffer when done

```
/* Free the SDRAM transfer registers when finished */
xbuf_free[$$hdr]
```

- Many conversions necessary
  - Smallest addressable data unit differs for memories

Packet I/O

- Network interface - microengine transfers limited to 64 bytes
- Large frames divided into 64 octet blocks (*mpackets* → MAC packets) by network interface hardware
- Individual *mpackets* transferred to microengine thru Receive FIFO
- Frame re-assembled inside microengine
- Outgoing frame divided again into *mpackets* by microengine
- SOP and EOP bits to identify packet start/end

Packet Ingress

- "Ready Bus Sequencer" polls MAC device to check if *mpacket* available
- "Receive Ready" CSR is set
- How to structure ingress threads ?
  - Static → one thread per MAC device
  - Dynamic → One thread for polling and notification (receive scheduler thread)
    - Pass *mpacket* to next available thread
    - Better resource utilization

Packet Ingress

- Move *mpacket* from Receive FIFO to SDRAM
  - sdram [r_fifo_rd,$$xfer,addr1,addr2,count],indirect_ref
    - SDRAM address = addr1+addr2
    - “count” = number of 8 byte words to transfer
    - “indirect_ref” → address of Receive FIFO = output of previous ALU instruction
- Move *mpacket* from Receive FIFO to SRAM
  - r_fifo_rd [$xfer,addr1,addr2,count]
    - $xfer → starting SRAM transfer register
    - “count” = number of 4 byte words to transfer
    - addr1+addr2 = starting quadword address in the RFIFO

Packet Egress

- Microengine divides frame into *mpackets*
- Steps to handle an *mpacket*
  - Reserve space in Transmit FIFO (TFIFO)
  - Copy *mpacket* from memory into TFIFO
  - Set SOP/EOP bits for *mpacket*
  - Set the valid flag in the XMIT_VALIDATE register
- How to structure egress threads ?
  - Static → one thread per MAC interface
  - Dynamic → scheduler thread picks up next available thread

Packet Egress

- Move *mpacket* from SDRAM to TFIFO
  - sdram [t_fifo_wr,$$xfer,addr1,addr2,count], indirect_ref
    - SDRAM address = addr1+addr2
    - “count” = number of 8 byte words to transfer
    - “indirect_ref” → address of TFIFO = output of previous ALU instruction
- Move *mpacket* from SRAM to TFIFO
  - t_fifo_wr [$xfer,addr1,addr2,count]
    - $xfer → starting SRAM transfer register
    - “count” = number of 4 byte words to transfer
    - addr1+addr2 = starting quadword address in the TFIFO

Packet Egress

- TFIFO data needs to be marked valid
  - Signals that data can be moved to network interface
- Ready Bus Sequencer polls output ports
  - CSR registers set (Transmit Ready)
  - Thread will poll these registers
  - Find out when physical interface is ready to transmit

Summary

- Advanced features of microengines
  - Memory allocation
  - Processor co-ordination
  - Atomic operations
- Packet queues
- Accessing packet header values
- Packet I/O
  - *mpackets*
  - Ingress processing
  - Egress processing