Mobile Systems

– real-time computation

COMP28512

Steve Furber & Barry Cheetham

Real-time computation

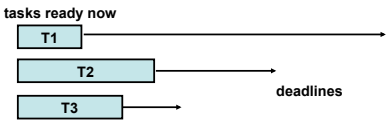
- Often computing "as soon as possible" is OK
  - but not for:
    - control, e.g. anti-lock braking
    - streaming media, e.g. mp3 players
    - telephony
  - these all require "real-time" computing
- Hard real-time
  - a missed deadline is a system failure
- Soft real-time
  - a missed deadline is a loss of service quality

Real-time computation

- Real-time requirements
  - an event should not occur ***until*** a given time has elapsed
  - an event must take place ***before*** a given time has elapsed
  - a certain computational activity must take place ***within a specified period*** of time
  - responding to one event must take ***priority*** over another event

Scheduling

- Given
  - the readiness of certain tasks
  - knowledge of how long they take to execute
  - knowledge of their respective deadlines
- Determine which task to execute next

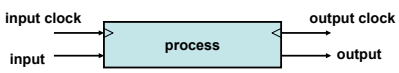


Responding to external events

- Polling
  - cycle through possible input events repeatedly until an active input is found
  - wastes a lot of power!
- Interrupts
  - an event flag causes the processor to switch attention to handling the event
  - incurs context-switch overheads
- DMA (direct memory access hardware)
  - move the data for processing later

Buffering

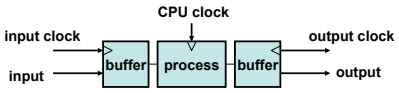
- Handling individual inputs and outputs in real time is near-impossible



  - what do you do if inputs and outputs must be handled at the same time?

Buffering

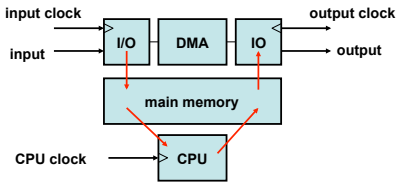
- Hardware input and output processes can meet timing precisely



  - buffers can allow the CPU to process blocks of data
    - more efficient than taking one input at a time

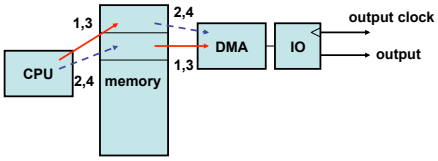
Buffering

- In practice we may use DMA to transfer data to buffers in main memory



Double-buffers

- Assemble data into blocks
  - work on one block while another is output
  - then switch over to work on the other while the first is output



Timing

- Many processor systems incorporate one or more timer/counters
  - run off a system clock
  - can be programmed to interrupt
    - after a certain time
    - at regular intervals
  - can generate output events at precise times
  - provides the processor with a real-time reference

Watchdog timers

- What happens if there is a glitch and the system crashes?
- A 'watchdog' timer can be used to interrupt or reset the system
  - normally the software resets the watchdog regularly before it triggers
  - if the software crashes the watchdog will (eventually) trigger
  - functionality is restored, perhaps after a brief loss of real-time performance

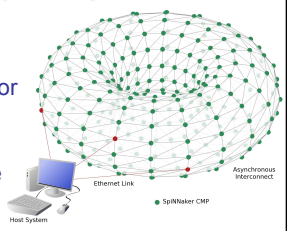
Real-time communications

- Sending real-time data streams over unreliable, variable delay networks
  - e.g. RTP (Real-time Transport Protocol)
  - what can be done about dropped packets?
    - resend will be too late
  - send packets with
    - sequence number (to see what is missing)
    - timestamp (to get accurate real-time output)
    - no flow control, no error control, no acknowledgement, no retransmission request
  - interpolate to approximate lost data

Example: *SpiNNaker*

Objective:

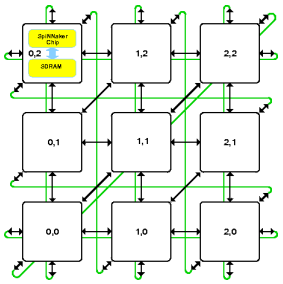
- to design
  - a chip multiprocessor
  - to model up to a billion spiking neurons in real time
- to investigate
  - biological mechanisms
  - fault-tolerant electronics



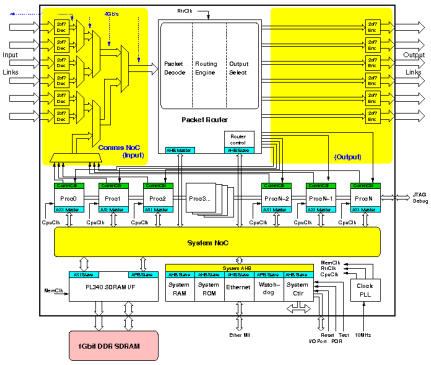
EPSRC ARM Silistix

2D Multi-Chip System

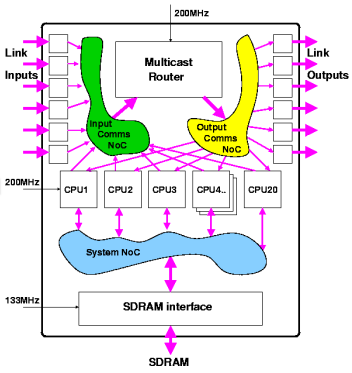
- 2-chip nodes
  - CMP + SDRAM
- Point-to-point links
  - self-timed
  - NoC protocol
- Async packets
  - source key routing
  - key is payload



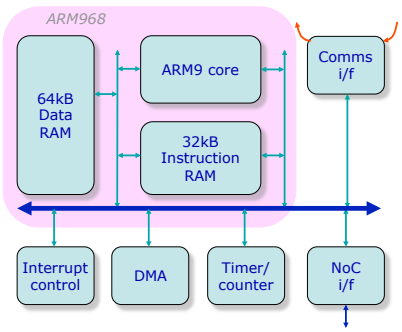
CMP node



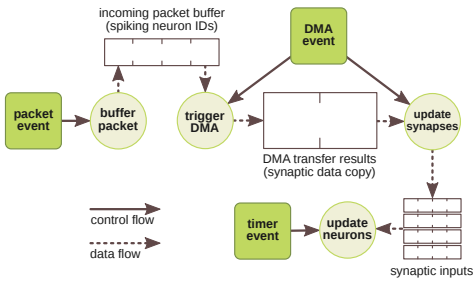
GALS organization



Processor node



Event-driven software

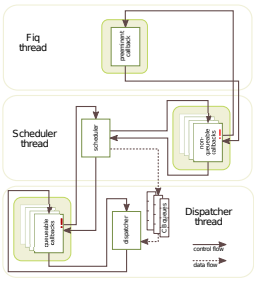


Real-time kernel

priority level = -1
only one callback
cannot be pre-empted

priority level = 0
can only be pre-empted
by priority -1 callback

priority level > 0
can be pre-empted
by priority <= 0 callbacks
scheduled in priority order



Power-efficiency

- In a real-time system there are no bonus points to finishing early!
  - when there are no events to process the CPU should go into a low-power mode
    - clocks disabled
  - interrupt wakes CPU up
    - process event
    - go back to sleep
  - event-driven model of computation

RTOS

- Real-Time Operating Systems
  - class of operating system for real-time systems
  - event-driven priority scheduling
    - switches task only when an event of higher priority needs service
  - task state:
    - ***running:*** currently executing
    - ***ready:*** will run when all higher-priority tasks have completed
    - ***blocked:*** can't run until some enabling event occurs
      - most tasks are blocked most of the time

Context switch

- A major issue in real-time computing is *context-switching* overhead
  - saving the state of the old task
  - restoring the state of the new task
  - this can take 10s to 100s of CPU cycles
- Interrupts
  - cause an automatic context switch for interrupt handling
    - often optimised by dedicated hardware resources
  - may cause a pre-emptive task switch

Co-routines

- An RTOS may reduce context switch overheads through use of *co-routines*
  - share stack and other process state
  - employ cooperative multitasking
    - instead of pre-emptive multitasking
    - a task hands over control to a higher-priority task at a mutually convenient time
  - must observe various programming restrictions

Summary

- Mobile systems often have real-time computation requirements
  - e.g. for streaming media data
- Radio communication is a particular problem
  - because of unavoidable packet loss
- Careful system design minimizes problems
  - buffering, DMA, scheduling, co-routines, ...