Computer Structure

System and DRAM

Lecturer:
Aharon Kupershtok

Prepared by Lihu Rappoport
Personal Computer System

- USB
- PCIe x1, x2, x4 Slots
- SATA
- Display Port
- PCI Express x1, x2, x4
- DMI x4
- HDMI
- LAN 10/100/1000
- Audio Codec
- BIOS
- Line out
- Microphone
- Display Port
- 2ch DDR3
- Graphics
- System Agent
- Core
- LLC
- Core
- LLC
- Core
- LLC
- Core
- LLC
- Display
- IMC
6th Generation Intel Core™ – Skylake

- 14nm process
- Quad core die, with Intel HD Graphics 530
Intel Core™ Uncore Subsystem

- The uncore subsystem includes
  - The System Agent (SA) handles I/O
    - Contains PCI Express, Memory Controller, Display Engine, and more
  - The Graphics Unit (GT)
  - The Last Level Cache (LLC) Slices

- A high bandwidth ring bus
  - Connects between the Cores, LLC slices, Graphics, and System Agent

- Server uncore is different
  - No Graphics Unit
  - Large capacity LLC with snooping capabilities that support multiple processors
  - Intel® Quick Path Interconnect (QPI) interfaces support multi-socket platforms
  - SA supports high BW traffic from memory and I/O
Last Level Cache – LLC

- The LLC consists of multiple cache slices
  - The number of slices is equal to the number of IA cores
  - Each slice can supply 32 bytes/cycle
  - Each slice may have 4/8/12/16 ways
    - Corresponding to 0.5M/1M/1.5M/2M block size

- Physical addresses are uniformly distributed among cache slices using a hash function
  - The LLC acts as one shared cache
    - With multiple ports and BW that scales with the number of cores
    - Ring/LLC not likely to be a BW limiter
  - Prevents hot spots

- LLC is shared among all Cores, Graphics and Media
  - GFX/media may in compete with cores on LLC
Last Level Cache – LLC (cont.)

- LLC hit latency is 26-31 cycles
  - Depends on Core location relative to the LLC Slice (how far the request needs to travel on the ring)

- Fetching data from LLC when another core has the data (in E/M states)
  - Clean hit – data is not modified in the other core – 43 cycles
  - Dirty hit – data is modified in the other core – 60 cycles

- LLC is fully inclusive of all core internal caches
  - Eliminates unnecessary snoops to cores
  - Per core “Core Valid bit” indicates if the internal core caches need to be snooped for a given cache line

- Traffic that cannot be satisfied by the LLC
  - LLC misses, dirty line write-back, non-cacheable operations, MMIO/IO operations
  - Travels through the ring to the IMC
Scalable Ring Interconnect

- High bandwidth scalable ring bus
  - Interconnects Cores, Graphics, LLC and SA
  - Composed of 4 bidirectional rings
    - 32 Byte Data ring, Request ring, Acknowledge ring, and Snoop ring
  - Fully pipelined at core frequency/voltage
    - bandwidth, latency and power scale with cores
  - Ring wires run over LLC without area impact
  - Distributed arbitration, coherency and ordering

- Each Core/LLC can get data from two stations
  - One connected to the “down“ direction,
  - and one connected to the “up”

- Ring access always picks the shortest path
  - E.g., Core3 to LLC2 data uses the “up” stream in 1 hop
    - Rather than from the "down" stream in 7 hops
  - Ave. trip from Core to LLC is \((0+1+2+3)/4 = 1.5\) hops
Ring Traffic Control

- Each stop can pull one request off the ring per clock
  - Either from the "up" or from the "down"

- To avoid a case of data arriving to a given stop from both "up" and "down" directions at the same time
  - Each ring stop has a given polarity ("even" on "odd")
    - A ring stop can only pull data from the direction that matches that polarity
  - The ring changes polarity once per clock
  - The sender knows the receiver and its polarity is, and what the hop count is between them
  - By delaying sending the data for at most one cycle, a sender can assure the receiver can read it
    - and the case of two packets arriving at the same time never occurs
Both Core's requests get a hit in the LLC, and the CVBs (core valid bits) indicate that no snoop is needed.

The Core's requests each get to the right LLC slice. Both Core's requests get a hit in the LLC, and the CVBs indicate that no snoop is needed.
System Agent Components

- **PCIe controllers that connect to external PCIe devices**
  - Support different configurations: \( \times 16 + \times 4 \), \( \times 8 + \times 8 + \times 4 \), \( \times 8 + \times 4 + \times 4 + \times 4 \)

- **DMI (Direct Media Interface) controller**
  - Connects to the PCH (Platform Controller Hub)

- **Integrated display engine**
  - Handles delivering the pixels to the screen

- **Flexible Display Interface (FDI)**
  - Connects to the PCH, where the display connectors (HDMI, DVI) are attached

- **DisplayPort (used for integrated display)**
  - e.g., a laptop’s LCD

- **Integrated Memory Controller (IMC)**
  - Connects to and controls the DRAM

- **An arbiter that handles accesses from Ring and from I/O (PCIe & DMI)**
  - Routes the accesses to the right place
  - Routes main memory traffic to the IMC
DRAM
DRAM Operation

- DRAM cell consists of transistor + capacitor
  - Capacitor keeps the state; Transistor guards access to the state
  - Reading cell state: raise access line AL and sense DL
    - Capacitor charged ⇒ current to flow on the data line DL
  - Writing cell state: set DL and raise AL to charge/drain capacitor
  - Charging and draining a capacitor is not instantaneous
- Leakage current drains the capacitor even when transistor is closed
  - The DRAM cell must be periodically refreshed (every 64ms)
Basic DRAM chip

- **DRAM access sequence**
  - Put Row on addr. bus
  - Assert RAS# (Row Addr. Strobe) to latch Row
  - Put Column on addr. bus
  - Wait RAS# to CAS# delay and assert CAS# (Column Addr. Strobe) to latch Col
  - Get data on address bus after CL (CAS latency)
Page Mode DRAM

- Allows Multiple accesses to different columns within the same row
  - Saves RAS, RAS to CAS delay, and Row pre-charge

- CAS-to-CAS delay: delay between two CASs within the same row

- tRP: Row pre-charge time:
  - the time to close current row, and open a new row

- DRAM is not true random access
  - Accessing multiple columns within the same row is much faster than accessing addresses from different rows
DIMMs

• DIMM: Dual In-line Memory Module
  – A small circuit board that holds memory chips

• 64-bit wide data path (72 bit with parity)
  – Single sided: 9 chips, each with 8 bit data bus
  – Dual sided: 18 chips, each with 4 bit data bus
  – Data BW: 64 bits on each rising and falling edge of the clock

• Other pins
  – Address – 14, RAS, CAS, chip select – 4, VDC – 17, Gnd – 18,
    clock – 4, serial address – 3, …
Synchronous DRAM – SDRAM

- All signals are referenced to an external clock (100MHz-200MHz)
  - Makes timing more precise with other system devices

- 4 banks – multiple pages (rows) open simultaneously (one per bank)
  - Accessing columns within each one of the 4 open rows is fast

- Command driven functionality instead of signal driven
  - ACTIVE: selects both the bank and the row to be activated
    - ACTIVE to a new bank can be issued while accessing current bank
  - READ/WRITE: select column

- Burst oriented read and write accesses
  - Successive column locations accessed in the given row
  - Burst length is programmable: 1, 2, 4, 8, full-page (may end by burst terminate)

- A user programmable Mode Register
  - CAS latency, burst length, burst type

- Auto pre-charge: may close row at last read/write in burst

- Auto refresh: internal counters generate refresh address
SDRAM Timing

- $t_{RCD}$: ACTIVE to READ/WRITE gap = $\left\lceil t_{RCD}(\text{MIN}) / \text{clock period} \right\rceil$
- $t_{RC}$: successive ACTIVE to a different row in the same bank
- $t_{RRD}$: successive ACTIVE commands to different banks
DDR-SDRAM

- **2n-prefetch architecture**
  - DRAM cells are clocked at the same speed as SDR SDRAM cells
  - Internal data bus is twice the width of the external data bus
  - Data capture occurs twice per clock cycle
    - Lower half of the bus sampled at clock rise
    - Upper half of the bus sampled at clock fall

- **Uses 2.5V (vs. 3.3V in SDRAM)**
  - Reduced power consumption
DDR SDRAM Timing

200MHz clock

cmd

ACT NOP NOP RD NOP ACT NOP NOP RD NOP ACT NOP NOP

Bank

Bank 0 X X Bank 0 X Bank 1 X X Bank 1 X X Bank 0 X X X

Addr

Row j X X Col j X X Row m X X X Col n X X Row l X X X

Data

j +1 +2 +3 n +1 +2 +3

t_{RCD} > 20ns

t_{RRD} > 20ns

t_{RC} > 70ns

CL = 2
DDR2

- DDR2 doubles the bandwidth
  - 4n pre-fetch: internally read/write 4× the amount of data as the external bus
  - DDR2-533 cell works at the same freq. as a DDR266 cell or a PC133 cell
  - Prefetching increases latency

- Smaller page size: 1KB vs. 2KB
  - Reduces activation power – ACTIVATE command reads all bits in the page

- 8 banks in 1Gb densities and above
  - Increases random accesses

- 1.8V (vs 2.5V) operation voltage
  - Significantly lower power
DDR3

• 30% power consumption reduction compared to DDR2
  – 1.5V supply voltage (vs. 1.8V in DDR2)
  – 90 nanometer fabrication technology

• Higher bandwidth
  – 8 bit deep prefetch buffer (vs. 4 bit in DDR2)

• 2× transfer data rate vs DDR2
  – Effective clock rate of 800–1600 MHz
    ➢ using both rising and falling edges of a 400–800 MHz I/O clock
    ➢ DDR2: 400–800 MHz using a 200–400 MHz I/O clock

• DDR3 DIMMs
  – 240 pins, the same number as DDR2, and are the same size
  – Electrically incompatible, and have a different key notch location
The Memory Controller
The Memory Controller

• The memory controller supports two channels of DDR3
  – Data rates of 1066MHz, 1333MHz and 1600MHz

• Each channel has its own resources
  – Handles memory requests independently
  – Contains a 32 cache-line write-data-buffer
  – Supports 8 bytes per cycle

• A hash function distributes addresses between channels
  – Attempts to balance the load between the channels in order to achieve maximum bandwidth and minimum hotspot collisions

• An out-of-order scheduler maximizes BW and minimize latency
  – Writes to the memory controller are considered completed when they are written to the write-data-buffer
  – The write-data-buffer is flushed out to main memory at a later time, not impacting write latency
Partial Memory Writes

- Memory reads and writes are done at full Cache Line granularity

- *Partial Write* transactions are writes to a subset of a Cache Line
  - Non-cacheable writes, e.g., write requests driven by IO devices
  - Would require adding more pins to indicate to memory which bytes to write
  - Not supported with ECC

- Partial writes are not common enough to justify the extra cost
  - The MC supports a partial write, by a RMW operation
    - Reads the current value of the full CL
    - Replaces the bytes that have to be modified
    - Writes back the full CL to memory

- Software should avoid creating partial write transactions whenever possible
  - E.g., buffer the partial writes into full cache line writes
How to get the most of Memory?

- **For best performance**
  - Populate both channels with equal amounts of memory
    - Preferably the exact same types of DIMMs
  - Use highest supported speed DRAM, with the best DRAM timings

- **Each DIMM supports 4 open pages simultaneously**
  - The more open pages, the more random access
  - It is better to have more DIMMs: \( n \text{ DIMMs} \Rightarrow 4n \text{ open pages} \)
  - Dual sided DIMMs may have separate CS of each side
    - Support 8 open pages
    - Dual sided DIMMs may also have a common CS
Motherboard Layout (Sandy Bridge)

- IEEE-1394a header
- PCI add-in card connector
- PCI express x1 connector
- PCI express x16 connector
- Back panel connectors

- Audio header
- High Def. Audio header
  - S/PDIF

- PCH

- Front panel USB headers

- Bios setup config jumper

- Reset, power, Disk LED

- Speaker

- Battery

- Main Power connector

- SATA connectors

- Rear chassis fan header
- LGA775 processor socket
- Processor fan header
- DIMM Channel A sockets
- DIMM Channel B sockets
- Front chassis fan header
- Chassis intrusion header
- Serial port header
- Rear chassis fan header
Backup
DRAM Access Sequence Timing

- Put row address on address bus and assert RAS#
- Wait for RAS# to CAS# delay (tRCD) between asserting RAS and CAS
- Put column address on address bus and assert CAS#
- Wait for CAS latency (CL) between time CAS# asserted and data ready
- Row pre-charge time: time to close current row, and open a new row
**DRAM controller**

- **DRAM controller gets address and command**
  - Splits address to Row and Column
  - Generates DRAM control signals at the proper timing

- **DRAM data must be periodically refreshed**
  - DRAM controller performs DRAM refresh, using refresh counter
The Memory Controller

- The memory controller supports high-priority isochronous requests
  - E.g., USB isochronous, and Display isochronous requests
  - High bandwidth of memory requests from the integrated display engine takes up some of the memory bandwidth
  - Impacts core access latency to some degree
SRAM vs. DRAM

- Random Access: access time is the same for all locations

<table>
<thead>
<tr>
<th></th>
<th>DRAM – Dynamic RAM</th>
<th>SRAM – Static RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refresh</td>
<td>Refresh needed</td>
<td>No refresh needed</td>
</tr>
<tr>
<td>Address</td>
<td>Address muxed: row+ column</td>
<td>Address not multiplexed</td>
</tr>
<tr>
<td>Access</td>
<td>Not true “Random Access”</td>
<td>True “Random Access”</td>
</tr>
<tr>
<td>density</td>
<td>High (1 Transistor/bit)</td>
<td>Low (6 Transistor/bit)</td>
</tr>
<tr>
<td>Power</td>
<td>low</td>
<td>high</td>
</tr>
<tr>
<td>Speed</td>
<td>slow</td>
<td>fast</td>
</tr>
<tr>
<td>Price/bit</td>
<td>low</td>
<td>high</td>
</tr>
<tr>
<td>Typical usage</td>
<td>Main memory</td>
<td>cache</td>
</tr>
</tbody>
</table>