CS 6501 - HW2 - CACTI

Huy Nguyen - mpc5ya

February 16, 2025

1 Question 1

Associativity	Access Time (ns)	Data Area Efficiency (%)	Area (mm^2)
2	1.47098	78.3192	2.24949
4	1.59863	69.0385	2.29881
8	2.65835	54.9431	2.72061
16	4.57620	19.1033	7.56952

Table 1: Impact of Associativity on Cache Performance

As cache associativity increases, access time, data area efficiency, and total area are significantly affected due to increased complexity in data lookup and storage.

1. Access Time (ns) Increases: Higher associativity increases access latency because more tag comparisons must be performed during a lookup. A 2-way cache requires only two tag checks, whereas a 16-way cache must check 16, increasing the critical path delay. Additionally, the extra multiplexing and selection logic further slows down data retrieval.

2. Data Area Efficiency (%) Decreases: With more associativity, a larger portion of the cache is used for tag storage and control logic rather than actual data. The increased tag array size reduces the space available for data storage, leading to lower data area efficiency. This trend is especially evident in highly associative caches, where tag management overhead grows significantly.

3. Cache Area (mm^2) Increases: Higher associativity demands more hardware resources, such as larger tag arrays and multiplexers, increasing the total cache area. This makes high-associativity caches more expensive in terms of silicon area and power consumption. The data shows a clear growth in cache area as associativity increases, with a 16-way cache being significantly larger than a 2-way cache.

Higher associativity reduces cache misses but comes at the cost of higher access latency, lower efficiency, and increased area. Lower associativity caches, on the other hand, are faster and smaller but may suffer from more conflict misses. Thus, L1 caches typically favor lower associativity for speed, while LLCs use higher associativity to improve hit rates.

2 Question 2:

Technology Node (nm)	Access Time (ns)	Area (mm^2)	Energy (nJ)
22	0.431572	0.20131	0.0341372
32	0.590766	0.28458	0.0569571
40	0.691583	0.444199	0.0812997
90	1.47098	2.24949	0.303592

Table 2: Impact of Technology Node on Cache Performance

The technology node, which refers to the manufacturing process size of transistors, has a significant impact on access time, area, and energy consumption in cache memory. As the node size increases (from 22 nm to 90 nm), access time, area, and energy consumption also increase.

As the technology node increases, access time becomes significantly longer. The data shows that at 22 nm, the access time is 0.431572 ns, whereas at 90 nm, it increases to 1.47098 ns. This is because smaller nodes have shorter transistor gate lengths, allowing for faster switching speeds and lower propagation delay. In contrast, larger nodes suffer from higher parasitic capacitance and resistance, slowing down the overall operation. This trend explains why modern processors rely on smaller technology nodes (e.g., 5 nm, 7 nm) to achieve faster cache performance.

Another key trend is the increase in cache area as the technology node gets larger. The results indicate that at 22 nm, the area is only 0.20131 mm^2, whereas at 90 nm, it expands to 2.24949 mm^2, more than 10 times larger. This happens because smaller transistors enable higher density integration, allowing more cache storage to fit within a smaller physical space.

Energy efficiency is another critical factor affected by technology scaling. The dynamic read energy at 22 nm is 0.0341372 nJ, but at 90 nm, it rises to 0.303592 nJ, nearly 9 times higher. This occurs because larger nodes operate at higher voltages and suffer from increased leakage current, leading to higher power dissipation.

3 Question 3:

Cache Size (Bytes)	Access Time (ns)	Area (mm^2)	Energy (nJ)
32,768	1.0104	0.735189	0.206864
131,072	1.47098	2.24949	0.303592
1,048,576	3.59361	21.291	1.23064
2,097,152	5.05455	37.9972	1.79205

Table 3: Impact of Cache Size on Performance Metrics

As cache capacity increases from 32 KB to 2 MB, we observe significant changes in access time, area, and energy consumption. These variations are due to the increased complexity of managing larger caches, which affects lookup time, hardware footprint, and power consumption.

A key trend is the increase in access time as cache size grows. At 32 KB, the access time is 1.0104 ns, whereas at 2 MB, it increases to 5.05455 ns. This happens because larger caches require more time to locate data within a bigger address space. Additionally, with more cache lines, tag comparisons take longer, and data may need to traverse a larger cache structure, introducing more delay. This trade-off explains why L1 caches are kept small (e.g., 32 KB) for low latency, while LLCs (Last-Level Caches) can be much larger but tolerate higher latency.

The physical area of the cache increases dramatically as capacity grows. The data shows that at 32 KB, the cache area is only 0.735 mm^2, but at 2 MB, it expands to 37.9972 mm^2, nearly 50 times larger. This is because a larger cache requires more SRAM cells to store additional data, increasing the silicon footprint. Additionally, supporting circuitry, such as decoders and multiplexers, also scales up, further contributing to the increase in area.

As cache capacity increases, energy consumption rises significantly due to the additional power required for data retrieval and leakage current from more transistors. The dynamic read energy at 32 KB is 0.206864 nJ, but it grows to 1.79205 nJ at 2 MB, showing a nearly nine-fold increase. This occurs because larger caches consume more power for tag checks, data movement, and maintaining multiple cache banks. Moreover, standby leakage also increases, as seen from the significant jump in leakage power at higher capacities.

4 Question 4:

Read-Write Ports	Access Time (ns)
1	1.47098
2	1.85786
4	2.75917

Table 4: Impact of Read-Write Ports on Access Time

b) As the number of read-write ports increases from 1 to 4, we observe a significant increase in access time. Initially, with one read-write port, the access time is 1.47098 ns. When the number of ports increases to 2, the access time rises to 1.85786 ns, and at 4 ports, it further increases to 2.75917 ns. This trend indicates that adding more ports increases access latency, making the cache slower.

The primary reason for this increase is the higher complexity of multi-port cache architectures. With more read-write ports, the cache must handle multiple simultaneous accesses, requiring additional control logic, larger multiplexers, and more complex wiring to ensure correct data access. This increases the critical path delay, leading to longer access times. Additionally, multi-port caches often require banking or duplication of data arrays to allow parallel reads and writes, which adds further latency.

c) A cache with more read-write ports is chosen despite increased access latency because it allows multiple simultaneous accesses, reducing contention and improving multi-core performance. In high-performance computing, AI, and graphics processing, workloads require high data throughput, making multi-port caches essential for minimizing memory stalls and maintaining efficient execution. While single-port caches may have lower latency, they cause queuing delays in multi-threaded

systems. Multi-port caches help balance parallel access and system-wide efficiency, making them crucial for shared caches and high-bandwidth applications.

5 Question 5:

Smallest area configuration:

1. **Associativity 2** - Area: 2.24949
2. **Technology Node 22nm** - Area: 0.20131
3. **Cache Size 32768** - Area: 0.735189
4. **Read-Write Ports 1** - Area: 2.24949

Parameter	Value
Associativity	2
Technology Node (nm)	22
Cache Size (Bytes)	32,768
Read-Write Ports	1
Access Latency (ns)	0.256185
Area (mm^2)	0.0443551
Energy (nJ)	0.0188471
Data Array Efficiency (%)	53.9647
Tag Array Efficiency (%)	68.8739

Table 5: Configuration with the Least Area

6 Question 6:

Base on some of above configurations, we can propose some configuration for L1 and LLC cache due to some of the reasons.

Parameter	L1 Cache Choice
Associativity	2-way
Cache Size	32 KB
Cache Block Size	64 Bytes
Technology Node	22 nm
Read-Write Ports	1
Access Time	0.256 ns
Area	0.044 mm^2
Energy	0.0188 nJ

Table 6: Optimized L1 Cache Configuration

L1 cache is designed for ultra-fast access since it is used in every CPU cycle. A 2-way set-associative cache provides a balance between hit rate and latency, ensuring fast lookups without excessive complexity. A small 32 KB size minimizes access latency, while a 64-byte block size optimizes data fetching. The 22 nm technology node is chosen for its low power consumption and high efficiency. Using only one read-write port reduces **hardware complexity, keeping latency at just 0.256 ns. This configuration is ideal for high-speed processing in modern CPUs.

LLC (Last-Level Cache) is designed to reduce memory accesses, minimizing expensive DRAM fetches. A 16-way set-associative cache significantly reduces cache misses, making it highly efficient for multi-threaded workloads. The 2 MB cache size ensures sufficient data storage for large applications, while a 128-byte block size improves prefetching efficiency. The 90 nm node keeps power consumption manageable, and 4 read-write ports allow multi-core access without excessive bottlenecks. While the access time (13.06 ns) is higher than L1, it is acceptable for LLC's role in caching large working sets.

Parameter	LLC Configuration
Associativity	16-way
Cache Size	2 MB (2,097,152 Bytes)
Cache Block Size	512 Bytes
Technology Node	90 nm
Read-Write Ports	1
Access Time	13.0654 ns
Area	206.887 mm^2
Energy	18.7537 nJ

Table 7: Revised LLC Configuration

However, these are just the proposed configuration. In reality when designing a cache, we will need more information such as additional details about the processor type, workload characteristics, memory bandwidth, multi-core architecture, and power constraints are essential. Mobile CPUs prioritize low power and smaller caches to reduce energy consumption, while HPC and server processors benefit from larger LLCs and higher associativity (16-way or more) to handle large datasets efficiently. Workload type also matters—real-time applications need fast L1 access, whereas data-heavy tasks require larger LLCs to minimize memory stalls. Multi-core systems need shared LLCs with multiple read-write ports, while memory bandwidth and DRAM latency impact cache sizing—if memory is slow, a larger LLC is necessary. Thermal constraints also play a role, as larger caches consume more power, making smaller LLCs preferable for energy-efficient designs. The final choice depends on whether the system prioritizes speed, efficiency, or scalability.

One information I think that would be important is the target clock frequency (or cycle time) of the processor. This dictates how many cycles we can afford for L1 access (and sometimes L2 or LLC) before it becomes a performance bottleneck. If the clock is very fast (short cycle time), the L1 must be small and/or have fewer ways of associativity to meet the single-cycle or two-cycle access requirement. Conversely, if the clock is slower, we might tolerate a slightly larger or more associative L1.