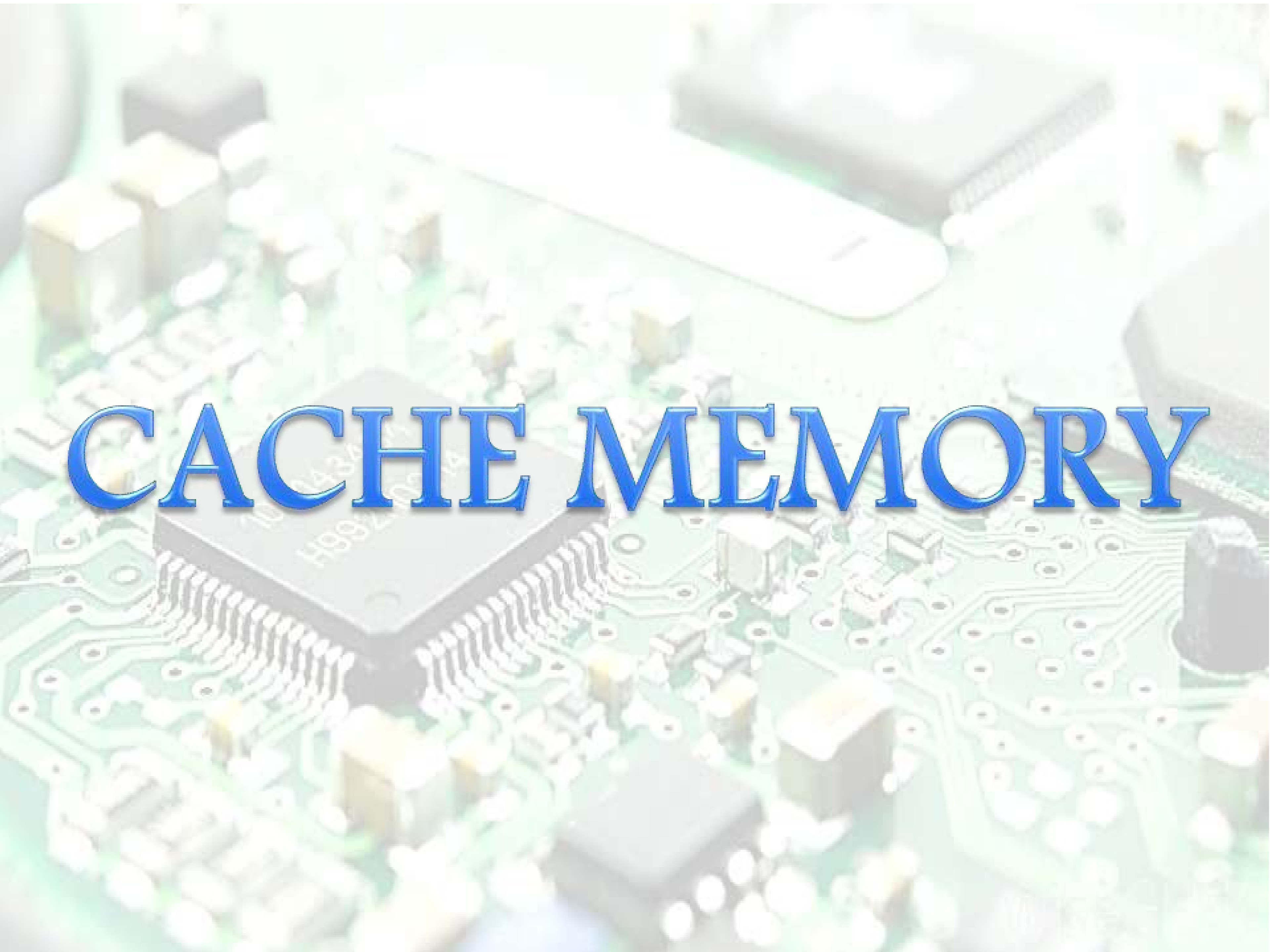
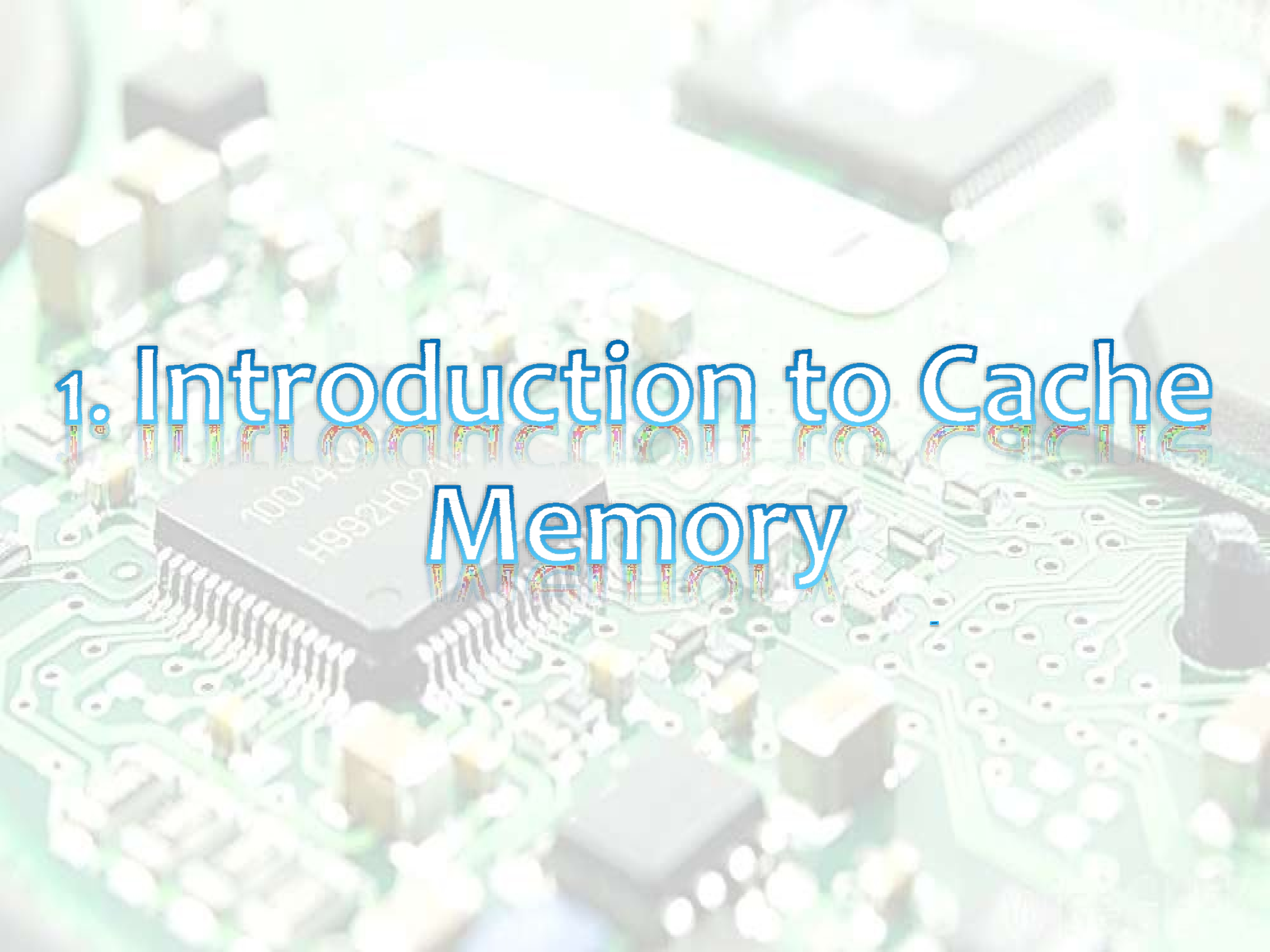
CACHE MEMORY



What is Cache Memory?

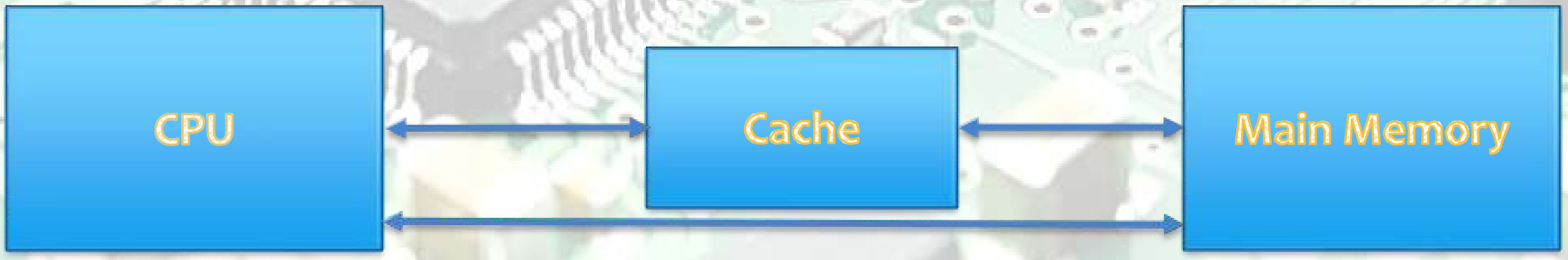
Cache is a small-sized type of volatile memory computer memory that provides high-speed data access to a processor and stores frequently used computer programs, applications and data. It stores and retains data only until a computer is powered up.



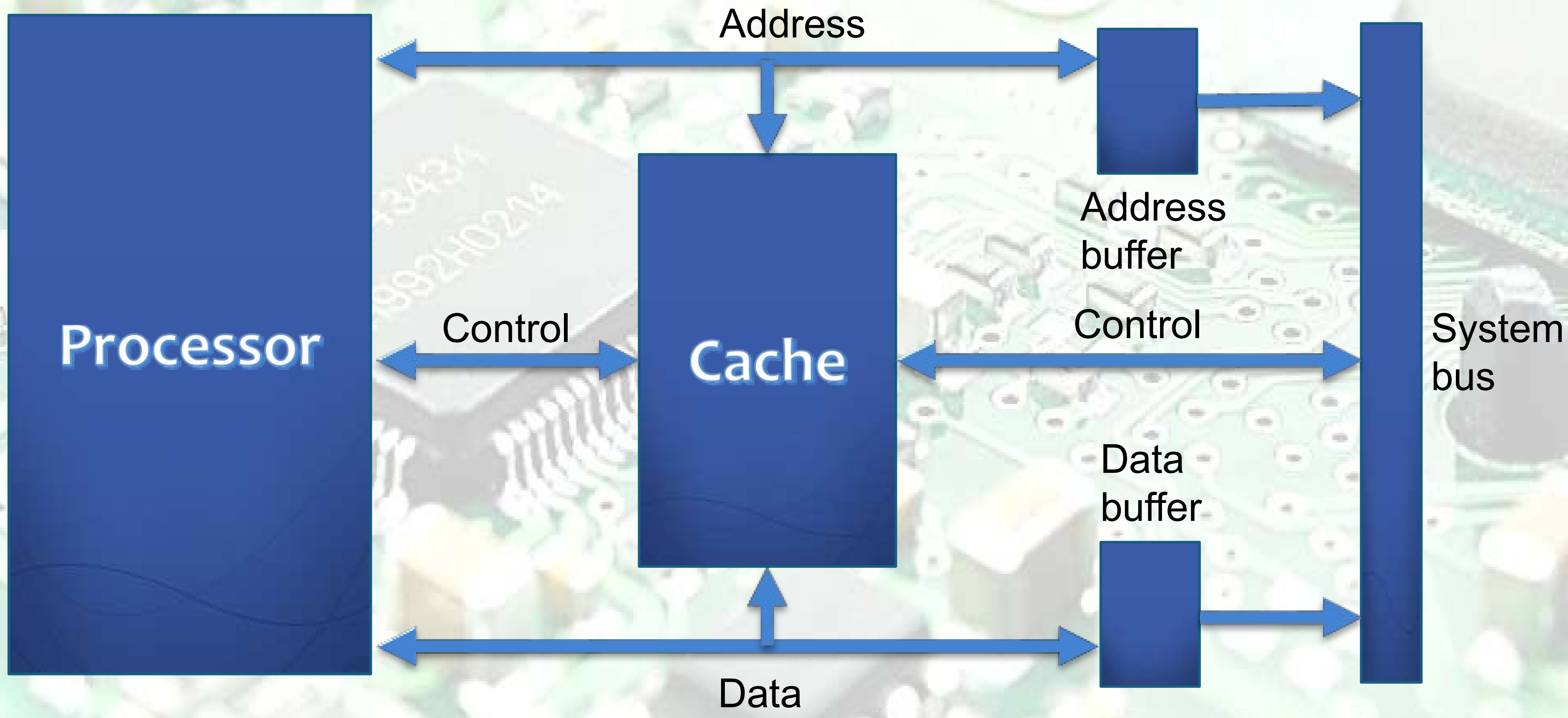
1. Introduction to Cache Memory

CACHE MEMORY

- Small amount of fast memory
- Sits between normal main memory and CPU
- May be located on CPU chip or module

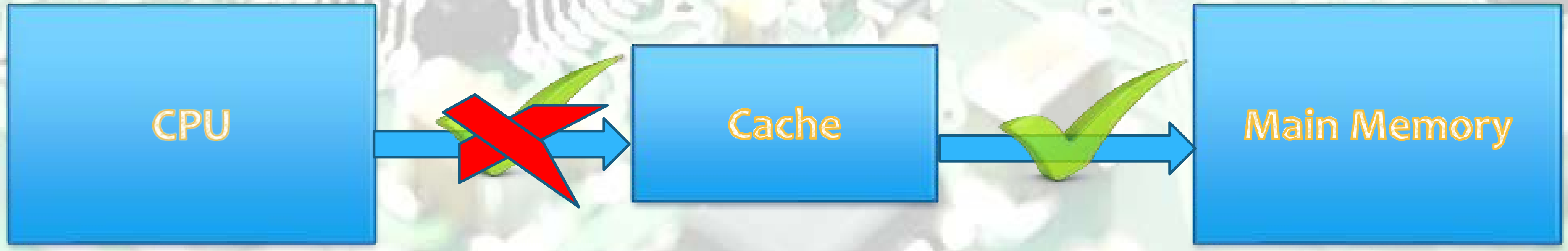


Cache Organization



Working Of Cache Memory

- The CPU initially looks in the Cache for the data it needs
- If the data is there, it will retrieve it and process it
- If the data is not there, then the CPU accesses the system memory and then puts a copy of the new data in the cache before processing it
- Next time if the CPU needs to access the same data again, it will just retrieve the data from the Cache instead of going through the whole loading process again



Levels Of Cache Memory

Level 1(L1) Cache:

- L1-cache is the fastest cache and it usually comes within the processor chip itself.
- The L1 cache typically ranges in size from 8KB to 64KB and uses the high-speed SRAM (static RAM) instead of the slower and cheaper DRAM (dynamic RAM) used for main memory.
- It is referred to as internal cache or primary cache.

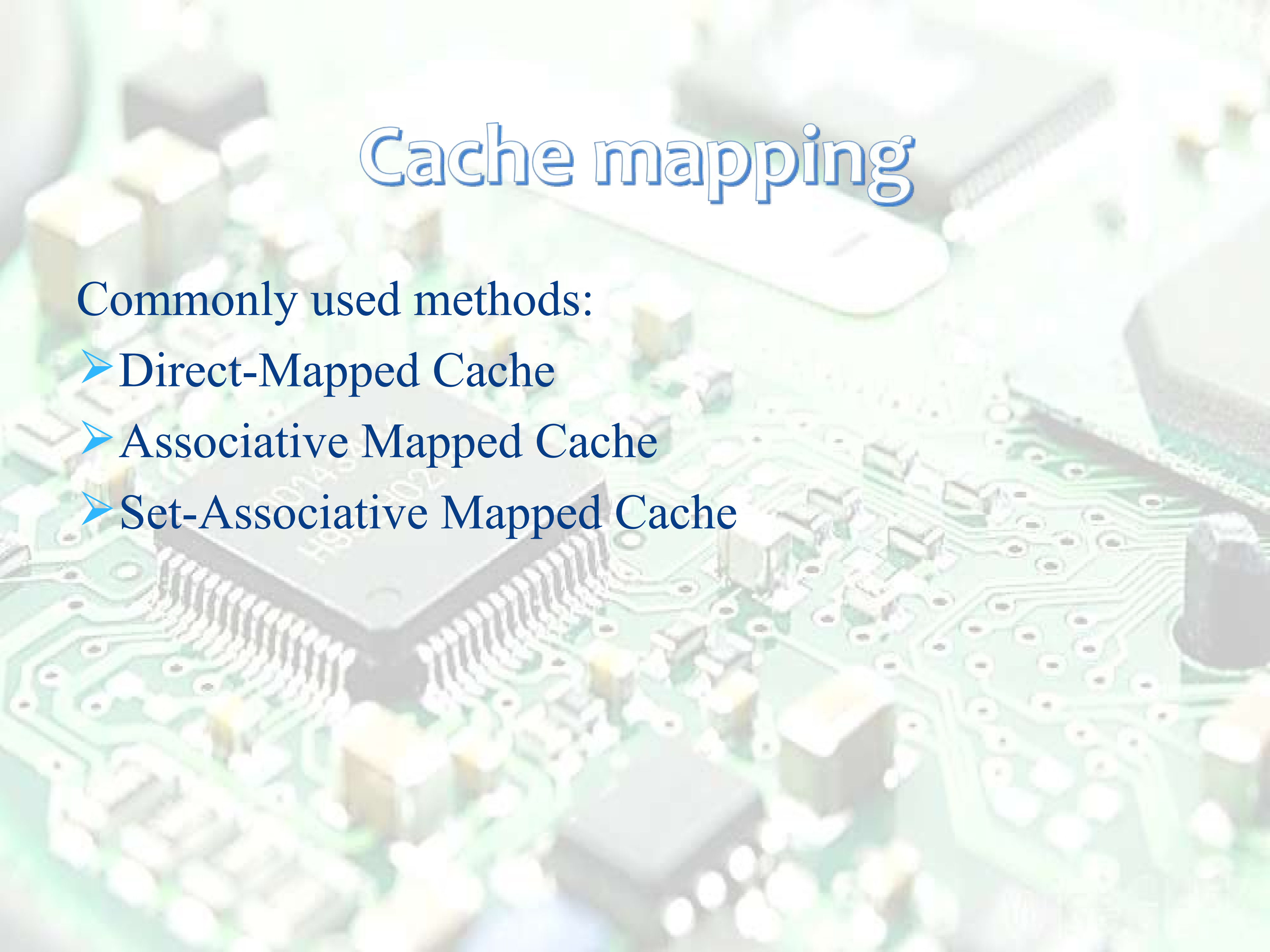
Level 2(L2) Cache:

- The L2 cache is larger but slower in speed than L1 cache.
- store recently accessed information. Also known as secondary cache, it is designed to reduce the time needed to access data in cases where data has already been accessed previously.
- L2 cache comes between L1 and RAM(processor-L1-L2-RAM) and is bigger than the primary cache (typically 64KB to 4MB).

Levels Of Cache Memory

Level 3(L3) Cache:

- L3 Cache memory is an enhanced form of memory present on the motherboard of the computer.
- L3, cache is a memory cache that is built into the motherboard. It is used to feed the L2 cache, and is typically faster than the system's main memory, but still slower than the L2 cache, having more than 3 MB of storage in it.



Cache mapping

Commonly used methods:

- Direct-Mapped Cache
- Associative Mapped Cache
- Set-Associative Mapped Cache

2. Direct Mapping Techniques

Direct Mapping

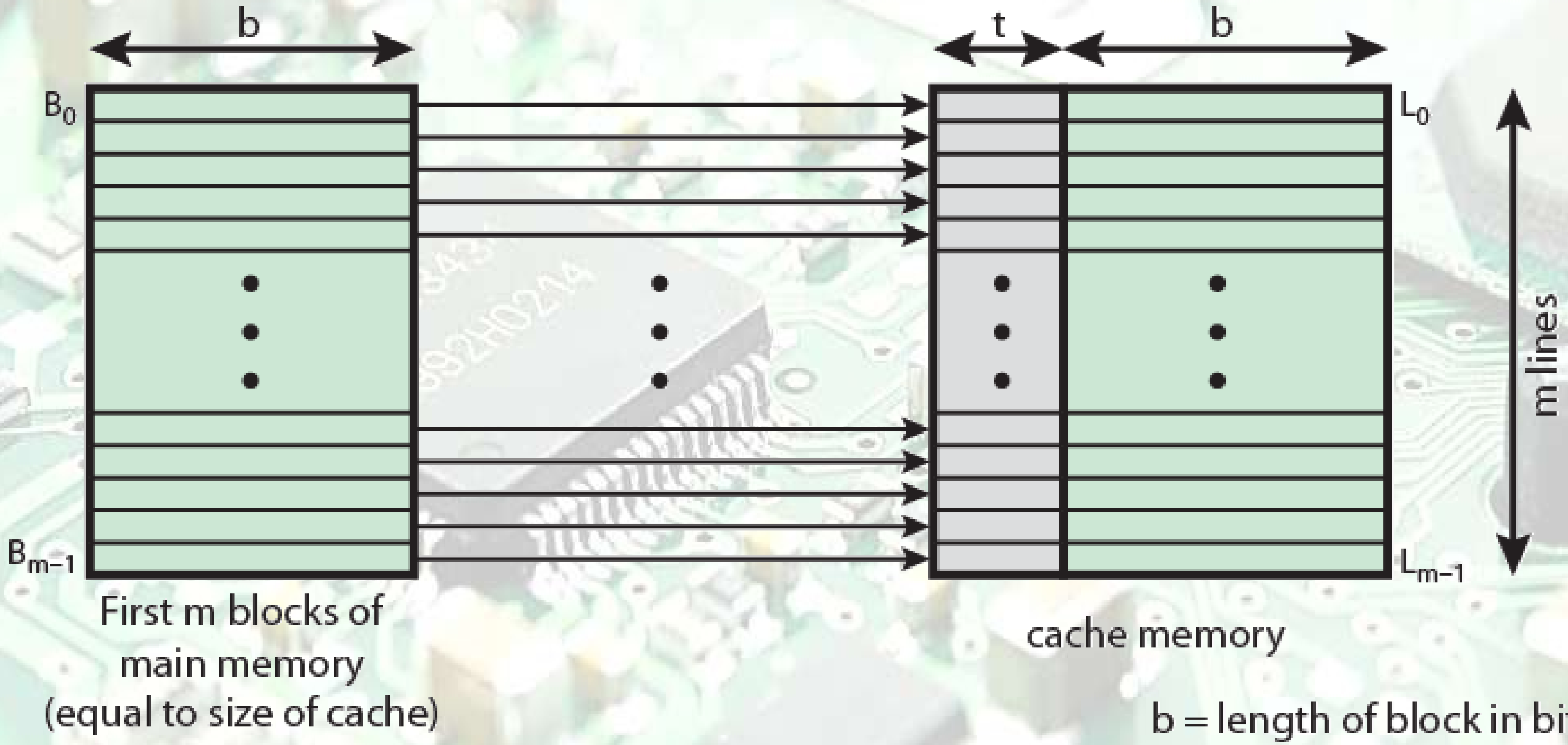
- Each block of main memory maps to only one cache line
  - i.e. if a block is in cache, it must be in one specific place
- Address is in two parts
- Least Significant w bits identify unique word
- Most Significant s bits specify one memory block
- The MSBs are split into a cache line field r and a tag of s-r (most significant)

Direct Mapping Address Structure

Tags-r	Line or Slot r	Word w
8	14	2

- 24 bit address
- 2 bit word identifier (4 byte block)
- 22 bit block identifier
  - 8 bit tag (=22-14)
  - 14 bit slot or line
- No two blocks in the same line have the same Tag field
- Check contents of cache by finding line and checking Tag

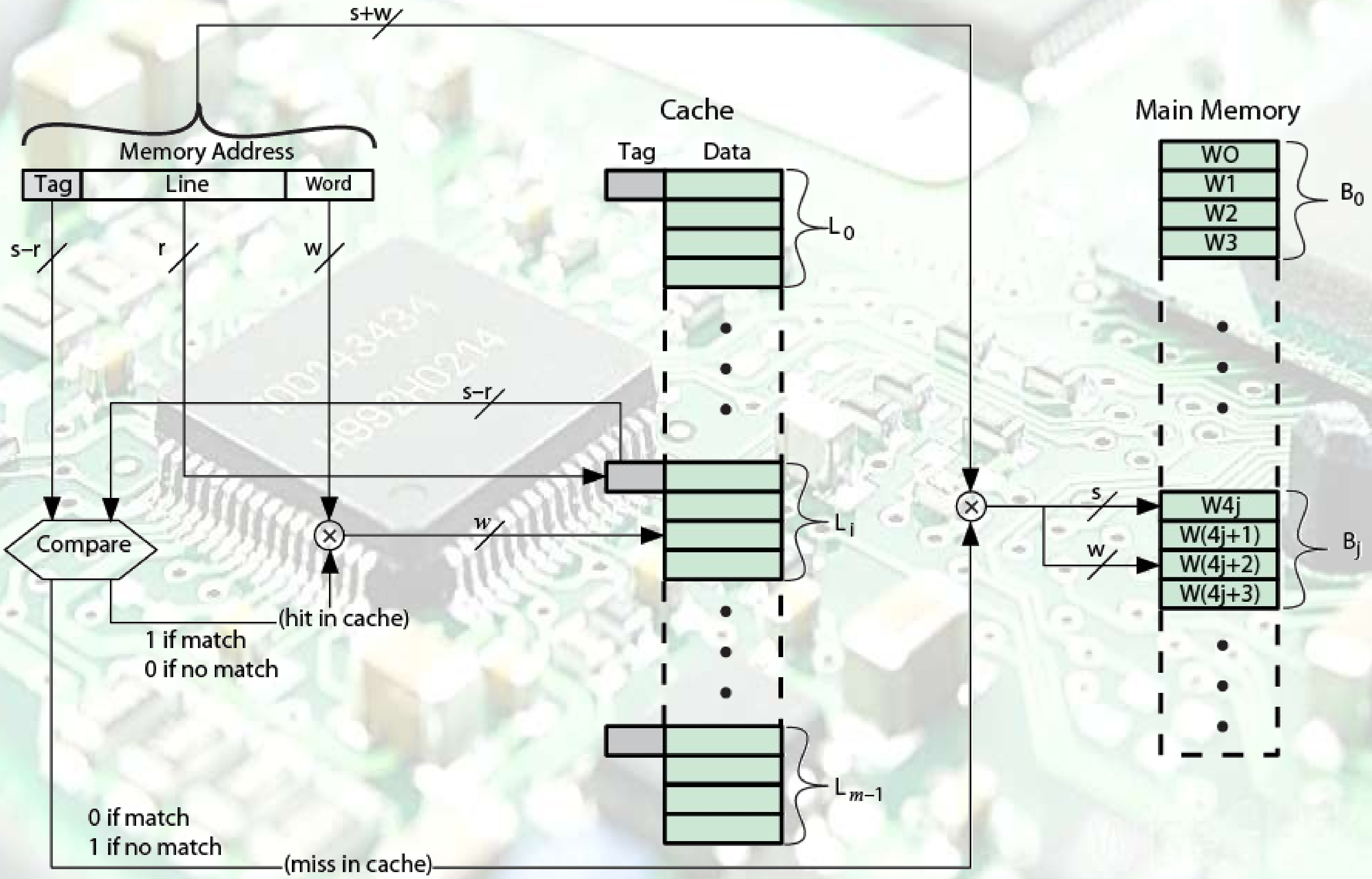
Direct Mapping from Cache to Main Memory



b = length of block in bits
t = length of tag in bits

(a) Direct mapping

Direct Mapping Cache Organization



Direct-Mapped Cache

Advantages

- The tag memory is much smaller than in associative mapped cache.
- No need for an associative search, since the slot field is used to direct the comparison to a single field.

Direct-Mapped Cache

Disadvantages

- Consider what happens when a program references locations that are 219 words apart, which is the size of the cache. Every memory reference will result in a miss, which will cause an entire block to be read into the cache even though only a single word is used.

Direct Mapping Summary

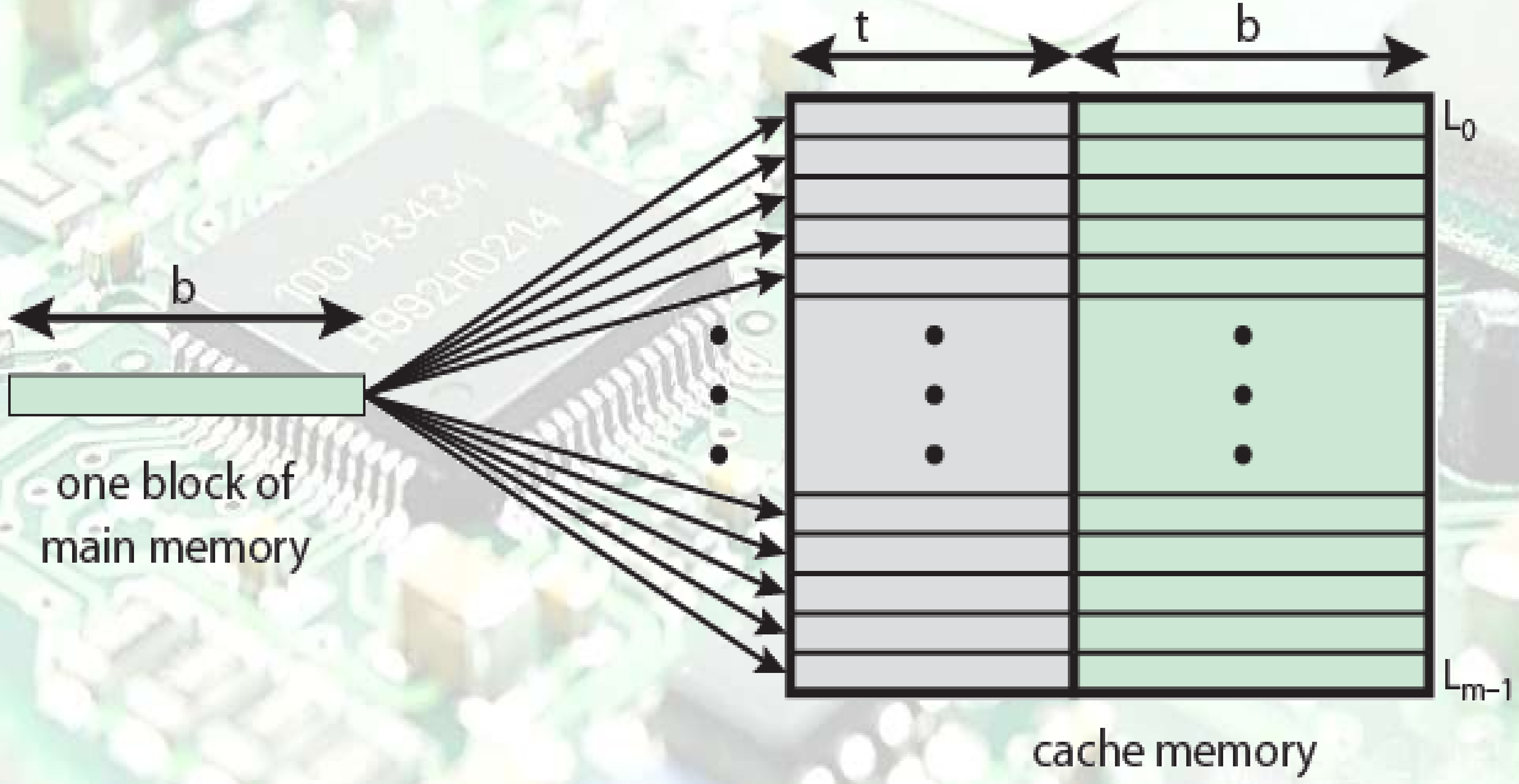
- Address length = (s + w) bits
- Number of addressable units = 2s+w words or bytes
- Block size = line size = 2w words or bytes
- Number of lines in cache = m = 2r
- Size of tag = (s – r) bits

3. Fully Associative Mapping Techniques

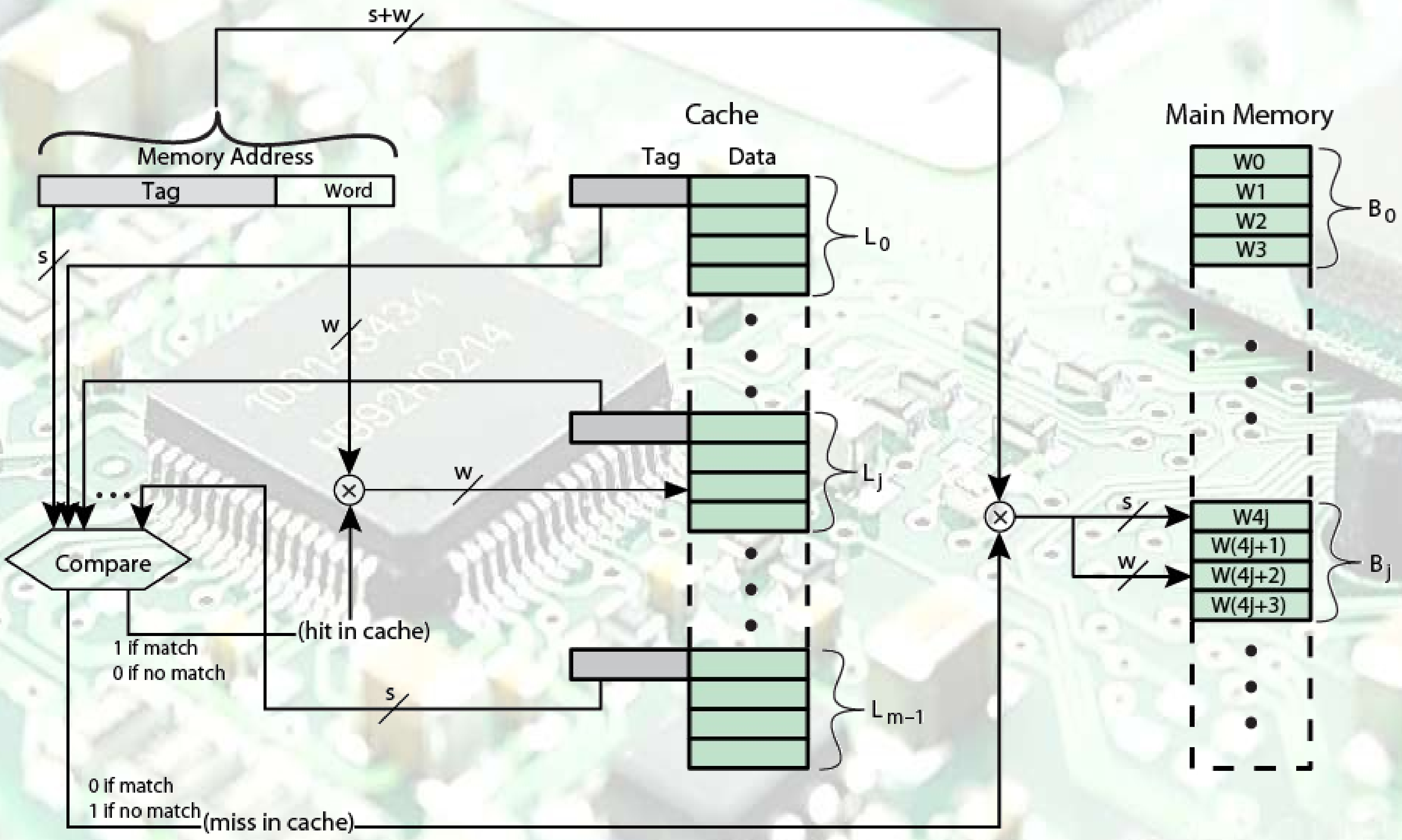
Associative Mapping

- A main memory block can load into any line of cache
- Memory address is interpreted as tag and word
- Tag uniquely identifies block of memory
- Every line's tag is examined for a match
- Cache searching gets expensive

Associative Mapping from Cache to Main Memory



Fully Associative Cache Organization



Associative Mapping Address Structure

Tag22 bit	Word 2 bit

- 22 bit tag stored with each 32 bit block of data
- Compare tag field with tag entry in cache to check for hit
- Least significant 2 bits of address identify which 16 bit word is required from 32 bit data block
- e.g.

Address	Tag	Data	Cache line
FFFFFC	FFFFFC	24682468	3FFF

Associative Mapped Cache

Advantages

- Any main memory block can be placed into any cache slot.
- Regardless of how irregular the data and program references are, if a slot is available for the block, it can be stored in the cache.

Associative Mapped Cache

Disadvantages

- Considerable hardware overhead needed for cache book keeping.
- There must be a mechanism for searching the tag memory in parallel.

Associative Mapping Summary

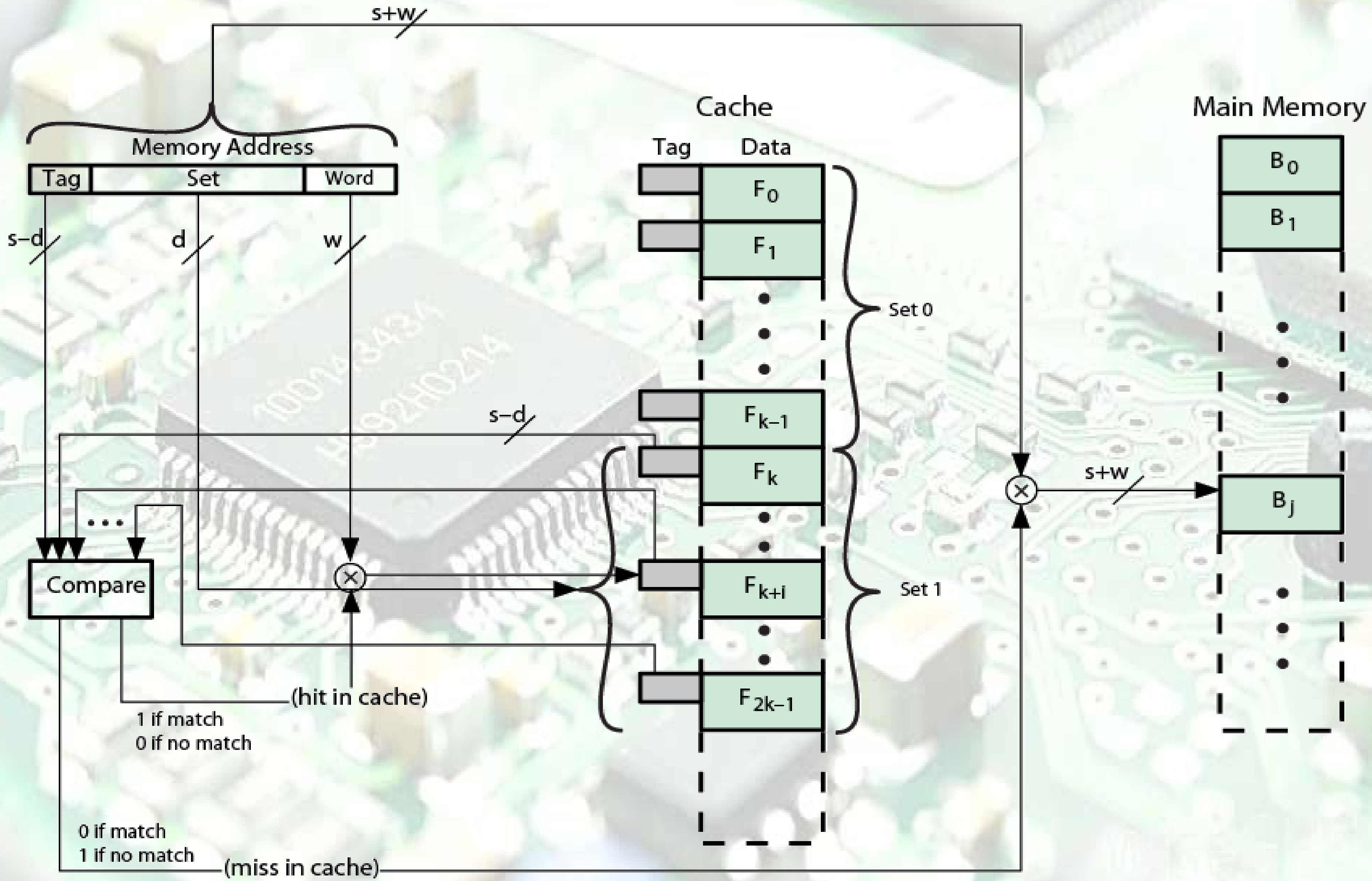
- Address length = (s + w) bits
- Number of addressable units = 2s+w words or bytes
- Block size = line size = 2w words or bytes
- Number of lines in cache = undetermined
- Size of tag = s bits

4. Set Associative Mapping Techniques

Set Associative Mapping

- Cache is divided into a number of sets
- Each set contains a number of lines
- A given block maps to any line in a given set
- e.g. Block B can be in any line of set i
- 2 way associative mapping
- A given block can be in one of 2 lines in only one set

K-Way Set Associative Cache Organization



Set-Associative Mapped Cache

Advantages

- In our example the tag memory increases only slightly from the direct mapping and only two tags need to be searched for each memory reference.
- The set-associative cache is widely used in today's microprocessors.

Set Associative Mapping Summary

- Address length = (s + w) bits
- Number of addressable units = 2s+w words or bytes
- Block size = line size = 2w words or bytes
- Number of blocks in main memory = 2d
- Number of lines in set = k
- Number of sets = v = 2d
- Size of tag = (s – d) bits

CACHE COHERENCY

- The synchronization of data in multiple caches such that reading a memory location via any cache will return the most recent data written to that location via any (other) cache.
- Some parallel processors do not provide cache accesses to shared memory to avoid the issue of cache coherency.

CACHE COHERENCY

- If caches are used with shared memory then some system is required to detect, when data in one processor's cache should be discarded or replaced, because another processor has updated that memory location. Several such schemes have been devised.