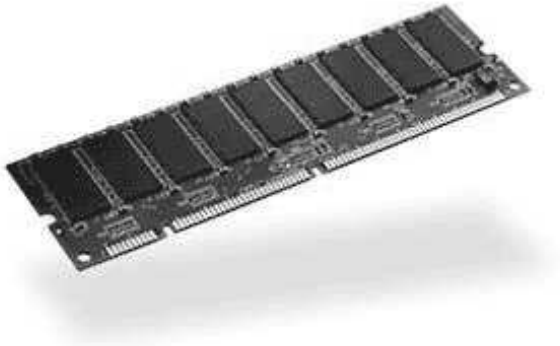
PHILIPS

An introduction to SDRAM and memory controllers

Benny Åkesson

Presentation outline

- DRAM history and evolution
- SDRAM scheduling basics
- Memory efficiency
- Memory controller overview



DRAM history

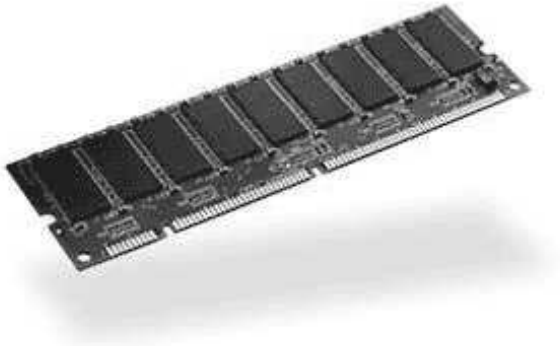
- DRAM was patented in 1968 by Dennard.
- Significantly cheaper than SRAM .
  - 1 transistor and 1 capacitor vs. 6 transistors.
  - A bit is represented by a high or low charge on the capacitor.
- Significantly slower than SRAM.
  - SRAM used for on-chip memory like caches and scratchpads.
  - DRAM is off-chip.

The DRAM evolution

- There has been multiple improvements to the DRAM design in the past ten years.
  - A clock signal was added making the design synchronous (SDRAM).
  - The data bus transfers data on both rising and falling edge of the clock (DDR SDRAM).
  - Second generation of DDR memory (DDR2) scales to higher clock frequencies.
  - DDR3 is currently being standardized by JEDEC.

Presentation outline

- DRAM history and evolution
- SDRAM scheduling basics
- Memory efficiency
- Memory controller overview



Multi-bank architecture

- SDRAMs have a multi-bank architecture and is organized in banks, rows and columns.

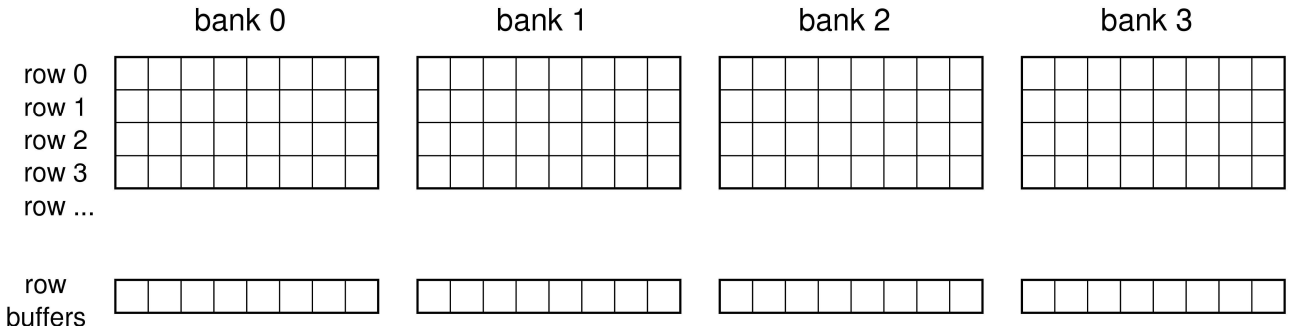
Typical figures:

4 – 8 banks

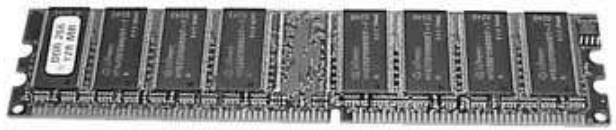
16K rows / bank

1024 columns / row

4 – 16 bits / column

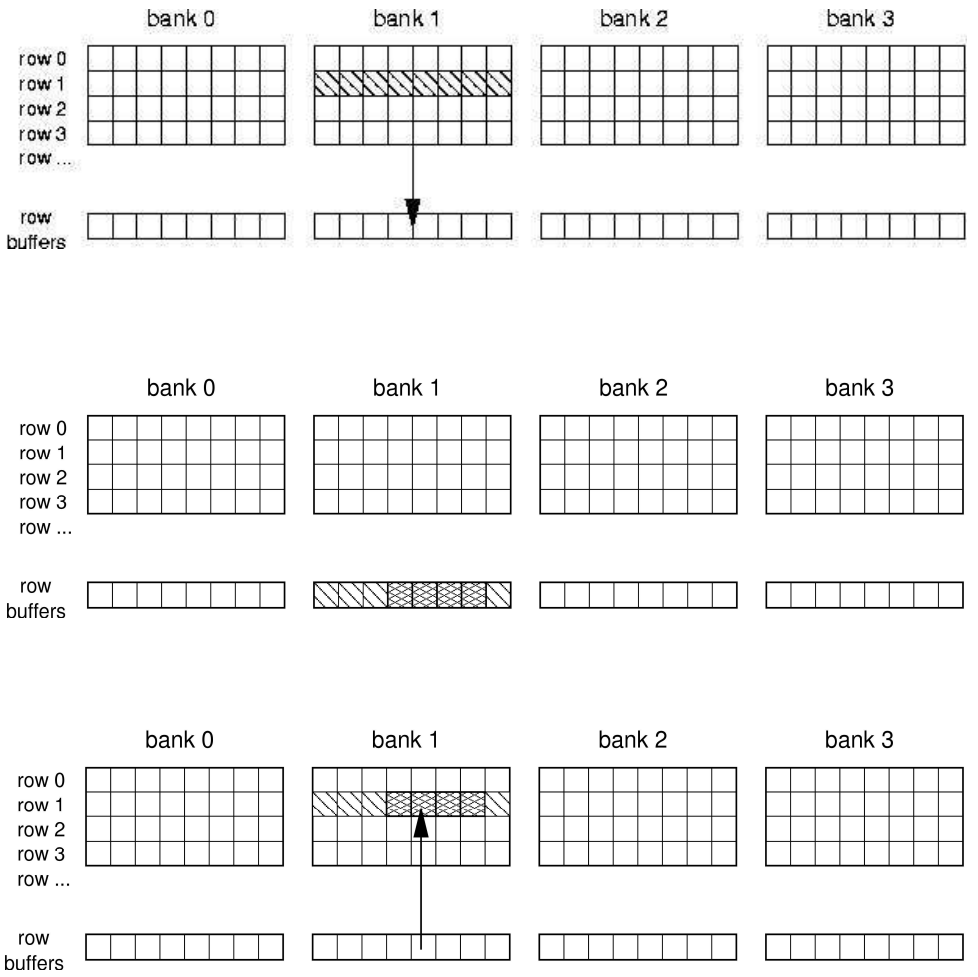


- Many chips are combined on a memory module to increase the word width. This is called the *memory configuration.*



Naïve memory access

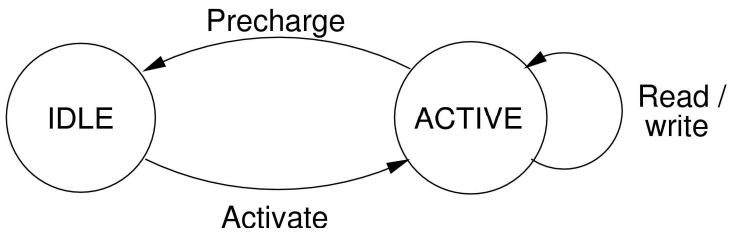
- The requested row is *activated* and copied to the row buffer of the corresponding bank.
- *Read* and/or *write* bursts are issued to the active row.
- The row is *precharged* and stored back into the memory array.



Refresh

- The capacitor is leaking and needs to be periodically refreshed in order not to loose its data.
- All banks must be precharged when a refresh command is issued.
- A DDR2 memory needes to be refreshed once every 64 ms.

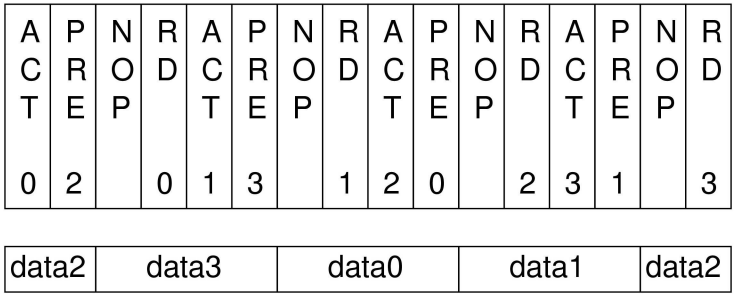
SDRAM command summary



No operation	NOP	Ignores all inputs
Activate	ACT	Activate a row in a particular bank
Read	RD	Initiate a read burst to an active row
Write	WR	Initiate a write burst to an active row
Precharge	PRE	Close a row in a particular bank
Refresh	REF	Start a refresh operation

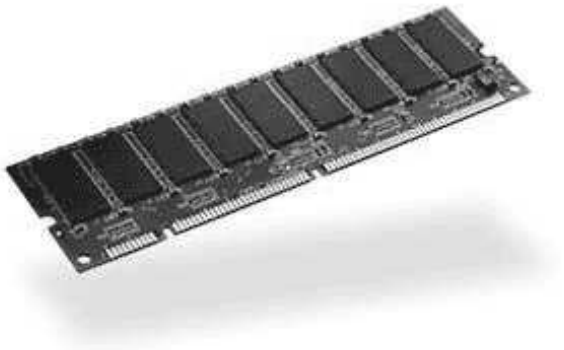
Pipelined memory access

- The SDRAM interface has a separate data and command bus.
- When data is transferred to or from a bank other banks are activated and precharged (*bank preparation*).
- Pipelining memory accesses increases efficiency and throughput.



Presentation outline

- DRAM history and evolution
- SDRAM scheduling basics
- Memory efficiency
- Memory controller overview



Memory efficiency

- Memory efficiency is the fraction between the amount of clock cycles when data is transferred and the total amount of clock cycles.

$$efficiency = \frac{transfer\_cycles}{total\_cycles}$$

- Five contributions to memory efficiency are:
  - Bank conflict efficiency
  - Refresh efficiency
  - Data efficiency
  - Read/write efficiency
  - Command conflict efficiency

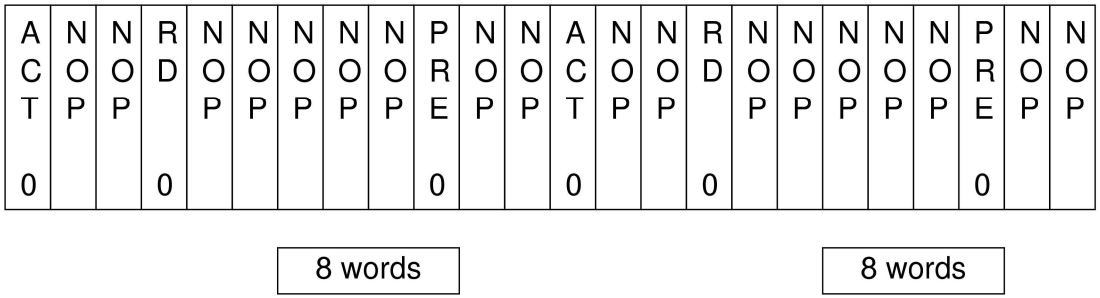
Timing behaviour

- Delays are required between SDRAM commands.
- This limits the efficiency of pipelined accesses.

Command delay	Cycles
ACT to PRE	9
ACT to ACT (same bank)	12
ACT to ACT (diff. bank)	2
ACT to RD/WR	3
WR to RD turn-around	2

Bank conflict efficiency

- Bank conflicts occur when bank preparation fails to open the requested row in a bank.
- This occurs due to the timing constraints of the memory.
  - Consider two consecutive bursts to different rows in the same bank. The ACT to ACT delay for a bank is 12 cycles, which causes very low efficiency.
  - Bank conflict efficiency is highly traffic dependent.

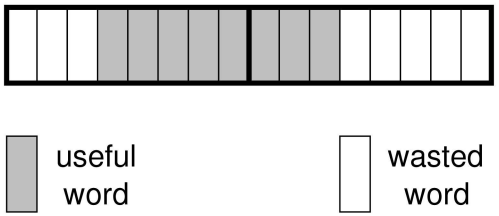


Refresh efficiency

- No data can be transfered when the memory is being refreshed.
- Recall that memory needs to be refreshed every 64 ms. With 8192 rows this means that a row should be refreshed, on average, every 7.8 µs.
- A refresh command executes in 75 ns on a DDR2-400 256 Mb device. This corresponds to roughly 1% of the time.
- Refresh efficiency is independent of traffic.

Data efficiency

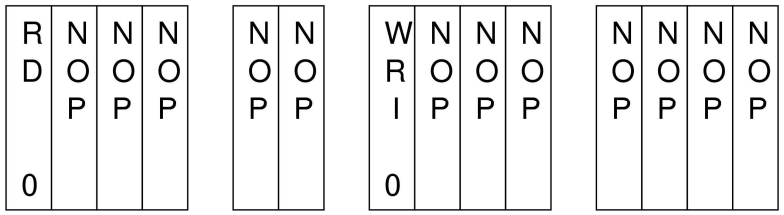
- A memory burst can access segments of the programmed burst size. This causes problem with *alignment.*
- If the requested data is poorly aligned an extra segment, and thus unrequested data, have to be fetched.



- The efficiency loss grows with smaller requests and bigger burst sizes.
- Alignment depends on data format, compiler technology etc.

Read / write efficiency

- It takes time to reverse the direction of the data bus.
  - Read to write switch costs 2 cycles.
  - Write to read switch costs 4 cycles.
  - Extra NOPs inserted between requests.
- Switch frequency dependent on traffic.

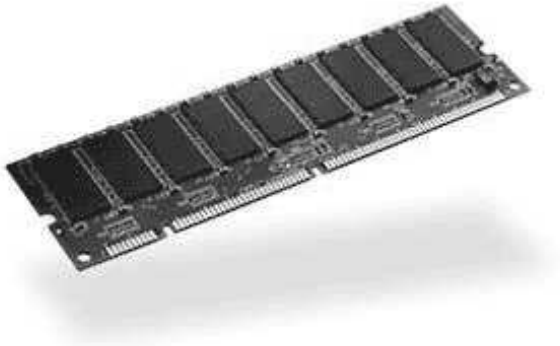


Command conflict efficiency

- Command bus uses single data rate.
- Congestion can occur on command bus.
  - Precharge and activate commands may need to be issued simultaneously.
- Problem depends on traffic and grows with smaller burst sizes.
- Command efficiency is generally quite high.

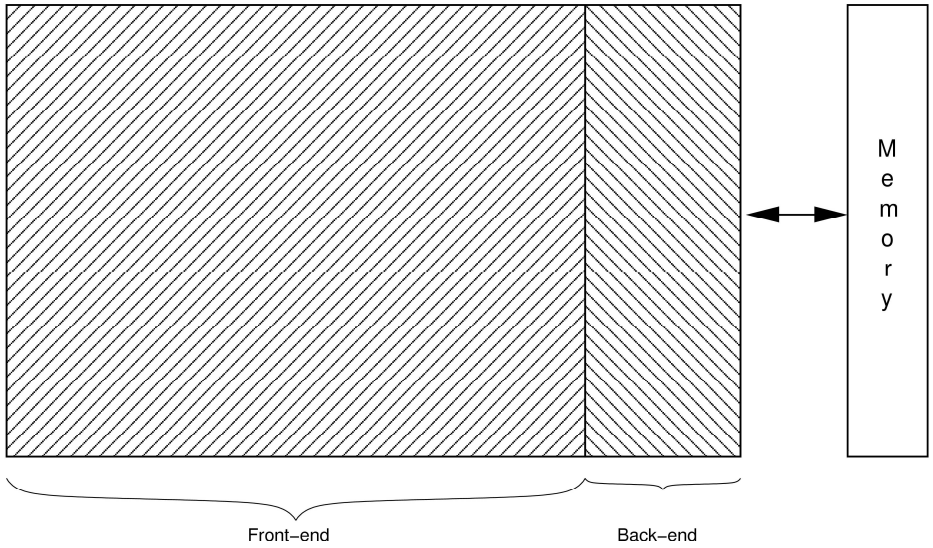
Presentation outline

- DRAM history and evolution
- SDRAM scheduling basics
- Memory efficiency
- Memory controller overview



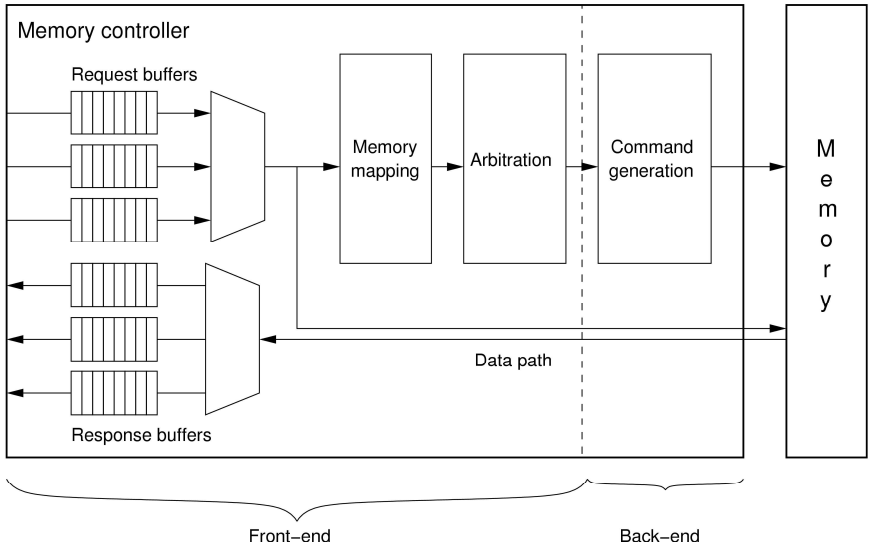
A general memory controller

- A general memory controller consists of two parts.
- The front-end:
  - buffers requests and responses.
  - provides an interface to the rest of the system.
  - is independent of the memory type.
- The back-end:
  - provides an interface towards the target memory.
  - is dependent on the memory type.



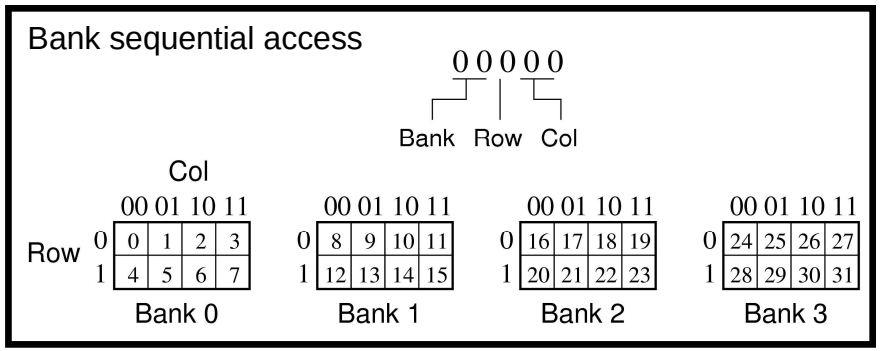
Functional blocks

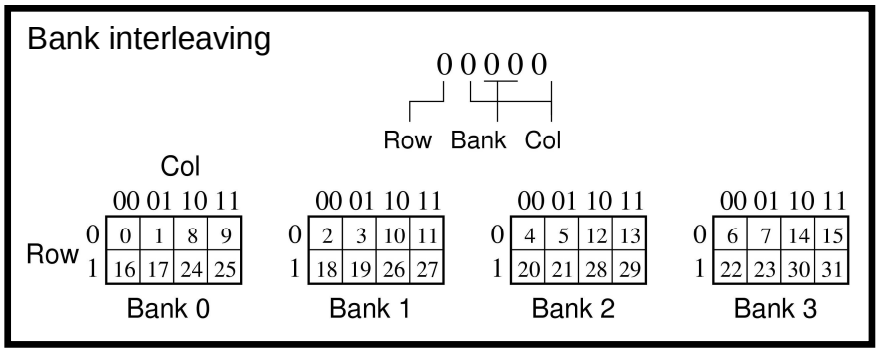
- A general memory controller consists of four functional blocks
  - Memory mapping
  - Arbiter
  - Command generator
  - Data path



Memory mapping

- The memory map decodes a memory address into (bank, row, column).
  - Decoding is done by slicing the address.
- Different maps affect the memory access pattern.
  - Bank sequential access
  - Bank interleaving
- Impacts bank conflict efficiency.



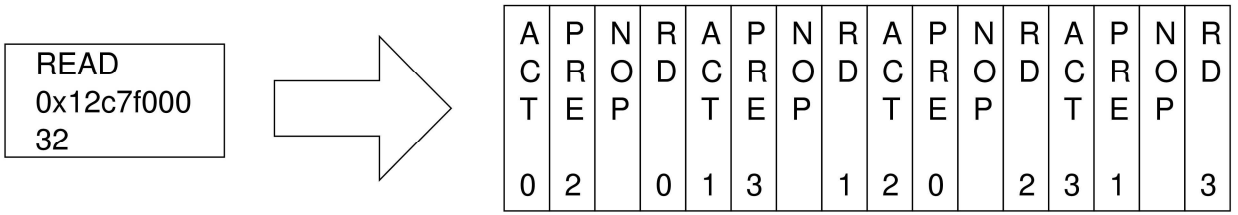


Arbiter

- The arbiter chooses the order in which requests access memory.
  - Potentially multiple layers of arbitration.
- An arbiter can have many different properties:
  - High memory efficiency
  - Predictable
  - Fast
  - Fair
  - Flexible
- Some properties are contradictive and are being traded in arbiter design.

Command generator

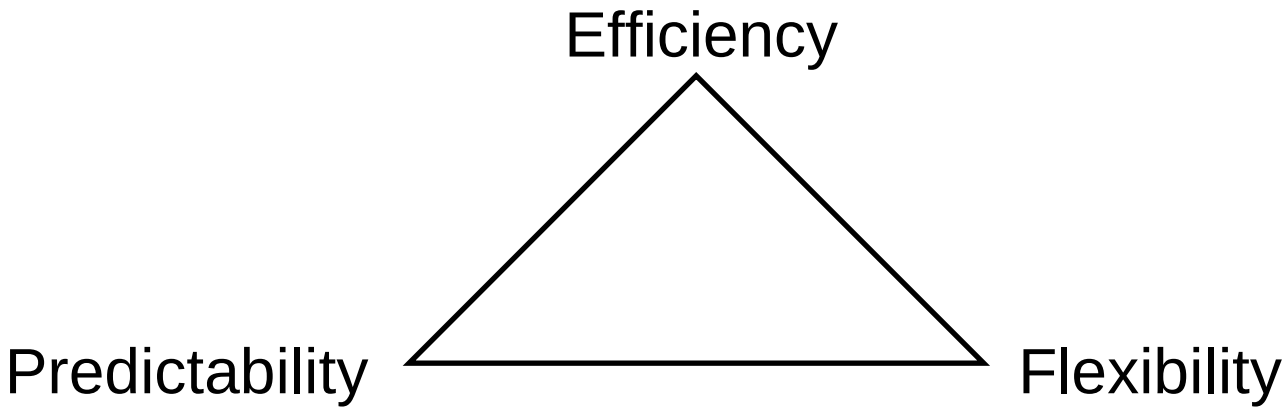
- Generates the commands for the target memory.
  - Customized for a particular memory generation.
  - Parameterized to handle different timings.



Controller designs

- Two directions in controller design:
  - Static memory controllers
  - Dynamic memory controllers

We will look into how these address three interesting memory controller properties.



Static memory controllers

- Schedule is created at design-time.
  - Traffic must be well-known and specified.
  - A fixed schedule is not flexible.
  - Computing schedules is not possible online for large systems.
  - Allocated bandwidth, worst-case latency and memory efficiency can be derived from the schedule.
- Static controllers are predictable!

Dynamic memory controllers

- Dynamic memory controllers
  - schedule requests in run-time.
  - are flexible.
- Clever tricks:
  - Schedule refresh when it does not interfere.
  - Reorder bursts to minimize bank conflicts.
  - Prefer read after read and write after write.

Dynamic memory controllers

- Dynamic arbitration
  - allows diverse service to unpredictable traffic.
  - provides good average-cases.
  - are very difficult to predict.
- The schedule is not known in advance
  - can provide statistical guarantees based on simulation.
  - memory efficiency is difficult to calculate.

Memory controller summary

Static memory controllers are found in critical systems with well-known traffic.

Dynamic memory controllers are found in general purpose and soft-real time systems with unpredictable traffic.