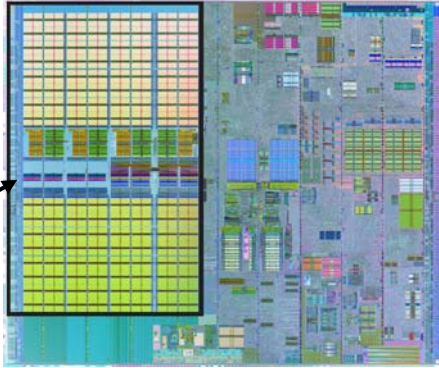


## A Typical Die Photo

Pentium4 Prescott chip with 2MB L2\$

L2 Cache



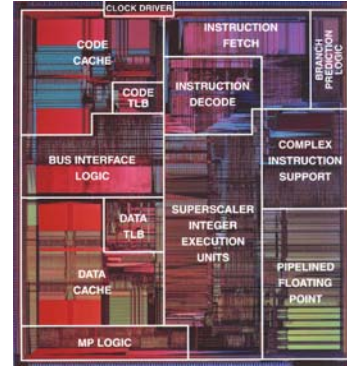
© 2008 Daniel J. Sorin from Roth and Lebeck

ECE 152

17

## A Closer Look at that Die Photo

Pentium4 Prescott chip with 2MB L2\$



© 2008 Daniel J. Sorin from Roth and Lebeck

ECE 152

18

## A Multicore Die Photo from IBM

IBM's Xenon chip with 3 PowerPC cores

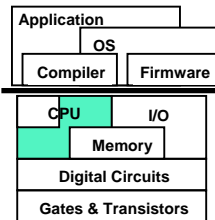


© 2008 Daniel J. Sorin from Roth and Lebeck

ECE 152

19

## This Unit: Caches and Memory Hierarchies



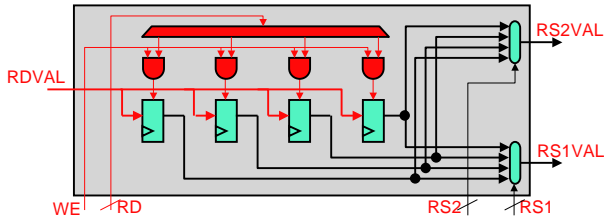
- Memory hierarchy
  - Basic concepts
- **SRAM technology**
  - [Transistors and circuits](#)
- Cache organization
  - ABC
  - CAM (content associative memory)
  - Classifying misses
  - Two optimizations
  - Writing into a cache
- Some example calculations

© 2008 Daniel J. Sorin from Roth and Lebeck

ECE 152

20

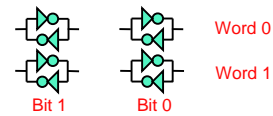
## Implementing Big Storage Arrays



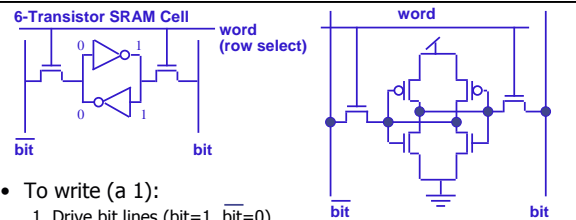
- Register file: bits as flip-flops, read ports as muxes
  - Not realistic, even if we replace muxes with tri-state buffers
  - MIPS register file: each read port is a 32-bit 32-to-1 mux?
    - Just routing the wires would be a nightmare
  - What about a **cache**? each read port is a 1024-to-1 mux? Yuck!

## SRAM

- Reality: large storage arrays implemented in “analog” way
  - Bits as cross-coupled inverters, not flip-flops
    - Inverters: 2 gates = 4 transistors per bit
    - Flip-flops: 8 gates = ~32 transistors per bit
  - Ports implemented as shared buses called bitlines (next slide)
  - Called **SRAM (static random access memory)**
    - “Static” → a written bit maintains its value (but still volatile)
- Example: storage array with two 2-bit words

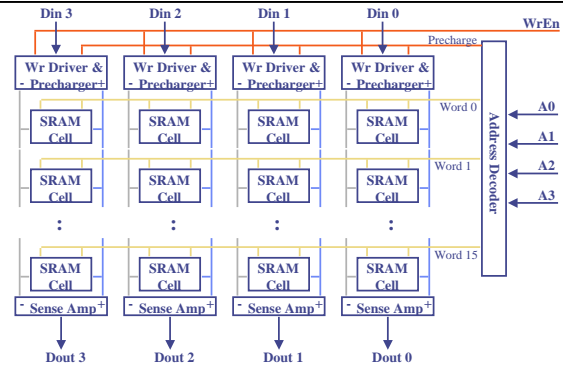


## Static RAM Cell

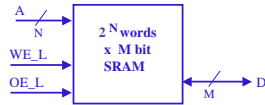


- To write (a 1):
  1. Drive bit lines (bit=1, bit=0)
  2. Select row
- To read:
  1. Pre-charge bit and bit to Vdd (set to 1)
  2. Select row
  3. Cell pulls one line lower (pulls towards 0)
  4. Sense amp on column detects difference between bit and bit

## Typical SRAM Organization: 16-word x 4-bit

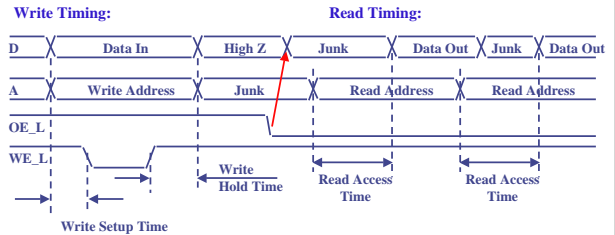
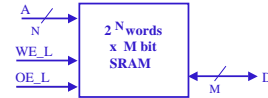


## Logic Diagram of a Typical SRAM



- Write Enable is usually active low (WE\_L)
- Din and Dout are combined (D) to save pins:
  - A new control signal, output enable (OE\_L) is needed
  - WE\_L is asserted (Low), OE\_L is de-asserted (High)
    - D serves as the data input pin
  - WE\_L is de-asserted (High), OE\_L is asserted (Low)
    - D is now the data output pin
  - Both WE\_L and OE\_L are asserted:
    - Result is unknown. **Don't do that!!!**

## Typical SRAM Timing: Write then 2 Reads



## SRAM Summary

- Large storage arrays cannot be implemented "digitally"
  - Muxing and wire routing become impractical
- SRAM implementation exploits analog transistor properties
  - Inverter pair bits much smaller than flip-flop bits
  - Wordline/bitline arrangement makes for simple "grid-like" routing
  - Basic understanding of reading and writing
    - Wordlines select words
    - Overwhelm inverter-pair to write
    - Drain pre-charged line or swing voltage to read
  - Access latency proportional to  $\sqrt{\text{\#bits} * \text{\#ports}}$
- You must understand important properties of SRAM
  - Will help when we talk about DRAM (next unit)