Computer Science 703

# Advance Computer Architecture

2010 Semester 1

## Lecture Notes

## 1Apr10

## Atomic Actions

James Goodman

Department of Computer Science



# Challenge of Sharing Memory

The ability to read and/or write multiple memory locations in an atomic transaction.

***"Stop the world!"***



# Atomic Swap

- Operation: atomically exchange a register value and a memory value
- Might be as little as a single bit
- Test for failure: register indicates bit was already set

- Useful for: Acquiring a lock

- Reference: ???



# Test & Set

- Operation: Set memory value (single bit) to 1; report previous value of memory location
- Test for failure: memory bit was already 1
- Variant of Atomic Swap
- Also: Test & Clear

- Useful for: Acquiring a lock

- Reference: IBM System/360 (1959)

# The Critical Section

```
CriticalSection() {
    acquire(lock);
    read(data1);
    read(data2);
    write(data1);
    release(lock);
}
```

```
acquire(lock) {
    while (swap(lock,HELD) != FREE)
            ;
     MemBar();
}

release(lock) {
     MemBar();
     lock = FREE;
}
```



# Why is this slow?

- Lock is truly shared
  - contended lock is actively shared during spinning
- Ordering constraints dictate that data cannot (should not) be pre-fetched
  - unless sharing is unlikely



# Acquire()

```
acquire(lock) {
    while (swap(lock,HELD) != FREE)
            ;
     MemBar();
}
```



# T&S

| P1: | P2: | P3: | P4: |
|---|---|---|---|
| Swap(lock) | | | |
| Read(data1) | Swap(lock) | Swap(lock) | Swap(lock) |
| Read(data2) | Swap(lock) | Swap(lock) | Swap(lock) |
| Write(data1) | Swap(lock) | Swap(lock) | Swap(lock) |
| Write(lock) | Swap(lock) | Swap(lock) | Swap(lock) |
| | Swap(lock) | Swap(lock) | Swap(lock) |
| | ... | ... | ... |

# Test & Test & Set

- Operation: Two-stage test: don't attempt to set bit until it is clear
- Software implementation: Test + Test&Set
- Test for failure: after second test, same as Test & Set
- No guaratee after first test, but avoids spinning on bus

- Useful for: Acquiring a lock, reduced contention

- Reference: L. Rudolph and Z. Segall, "Dynamic decentralized cache schemes for MIMD parallel processors." In ISCA-11, pages 340-347, June 1984.

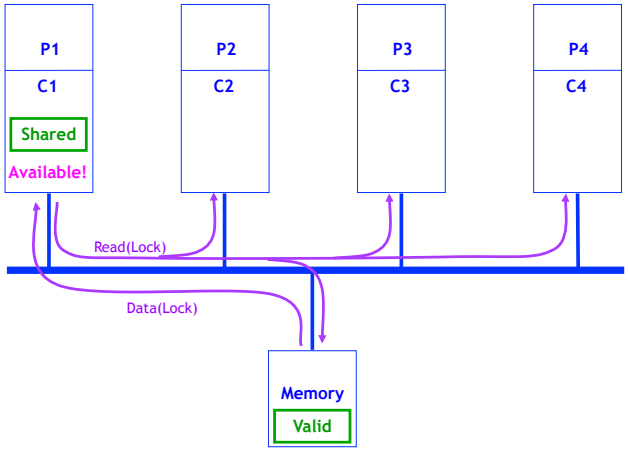


# Acquire() using T&T&S

```
acquire(lock) {
    do {
        while (lock == HELD) {
            ;
        }
    } while (swap(lock, HELD) != FREE);
    MemBar();
}
```

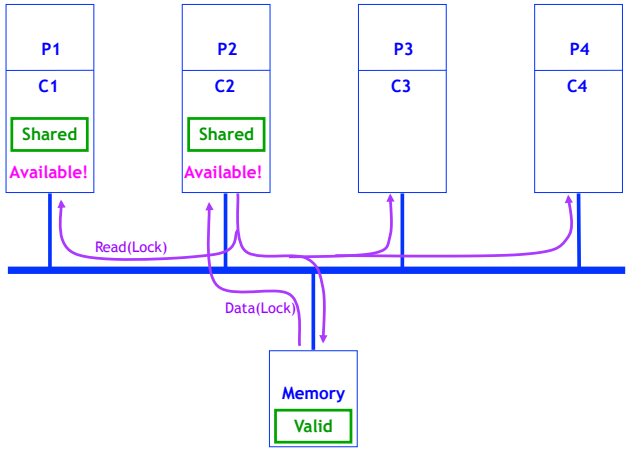


# Test&Test&Set

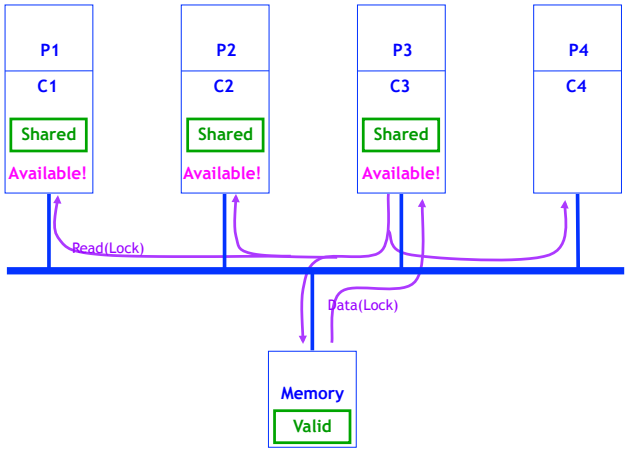





# Test&Test&Set

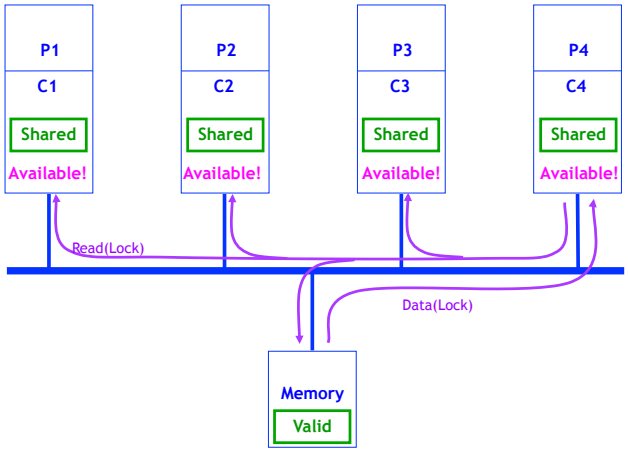


# Test&Test&Set



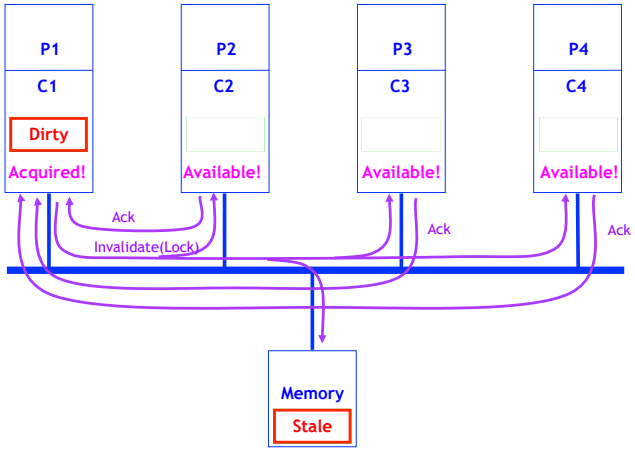



# Test&Test&Set



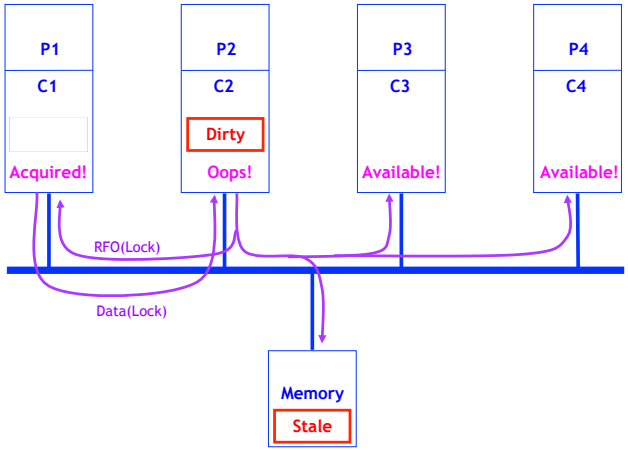



# Test&Test&Set

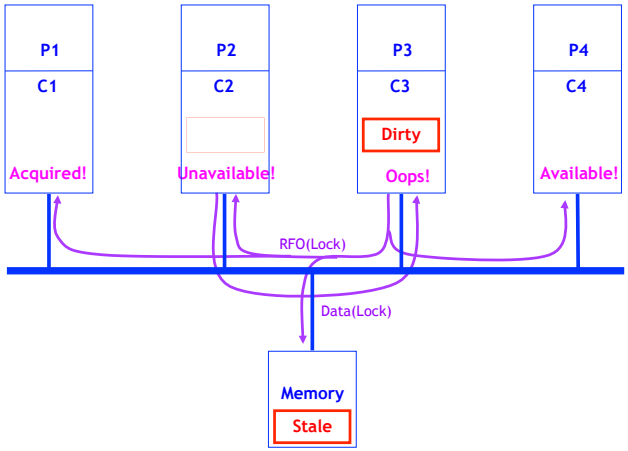





# Test&Test&Set

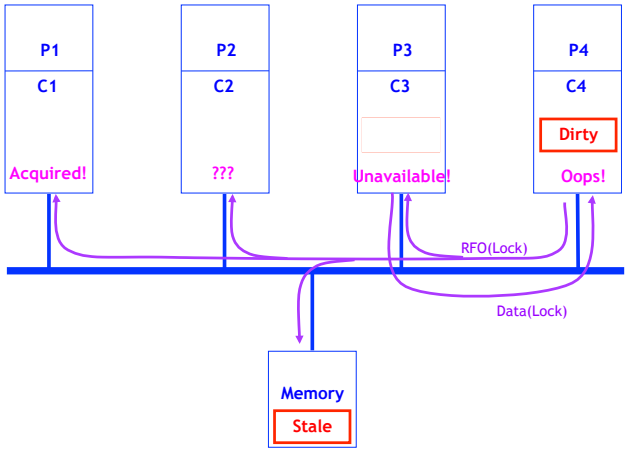


# Test&Test&Set



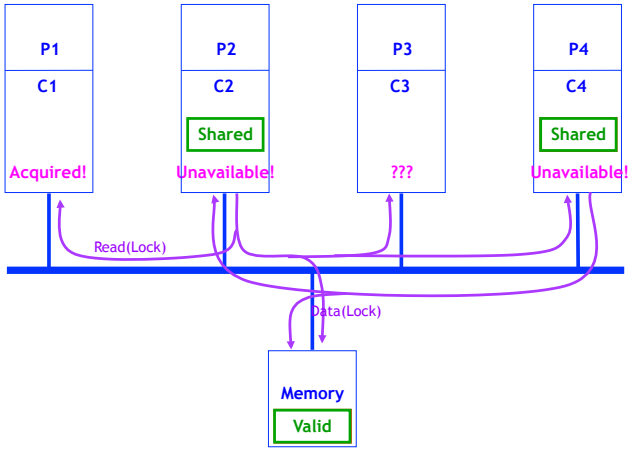



# Test&Test&Set



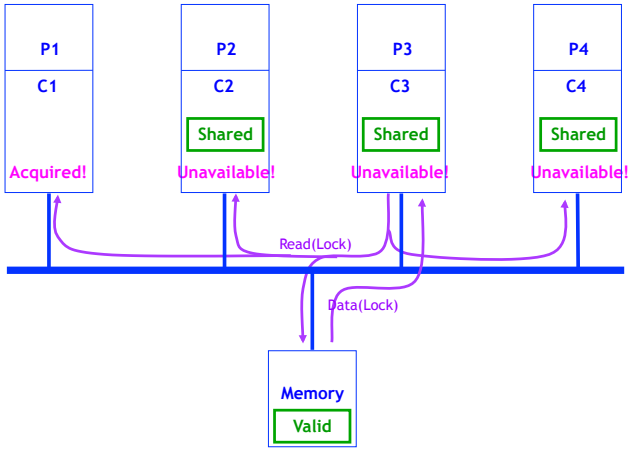



# Test&Test&Set

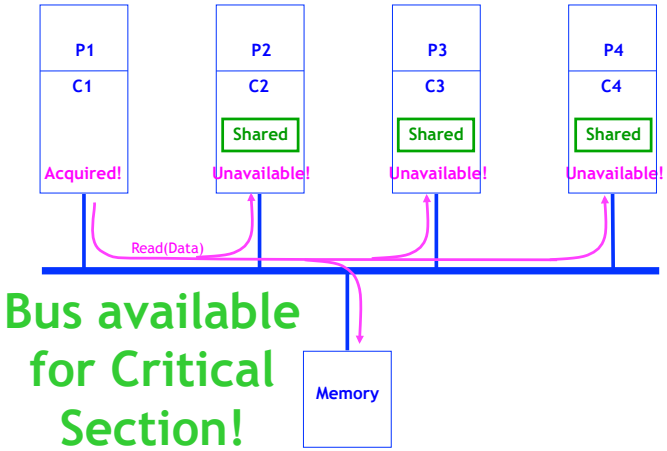





# Test&Test&Set

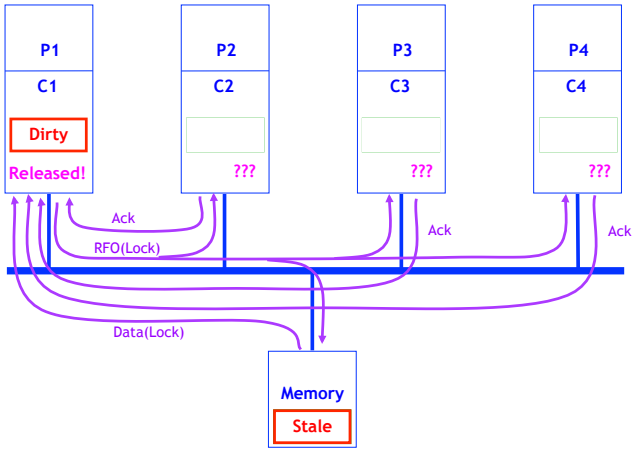


# Test&Test&Set






# Test&Test&Set






# Essence of TM

*The ability to access multiple memory locations in an atomic transaction, without specifying how atomicity is achieved.*

-- Mark Moir