# MIPS R4000 Caches and Coherency

**Paul Ries**



mips

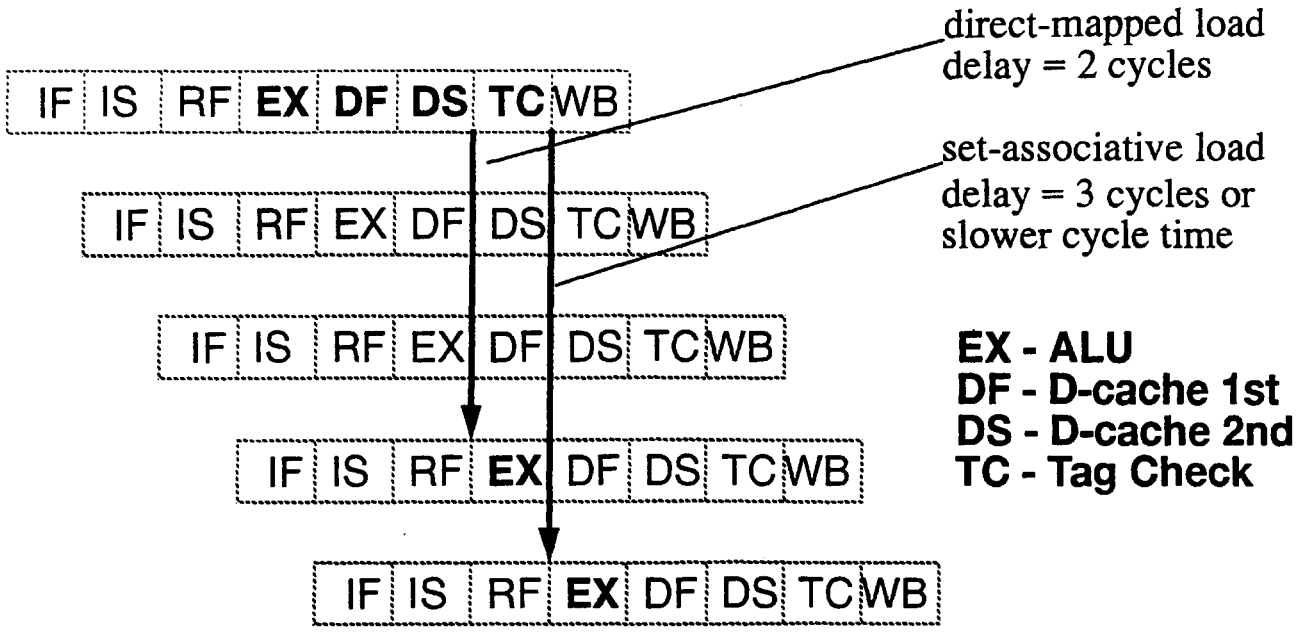
## R4000 Primary Caches

- 8KB to 32KB
- Direct-mapped
- Virtual index, Physical Tag
- Pipelined
- Write-back



mips

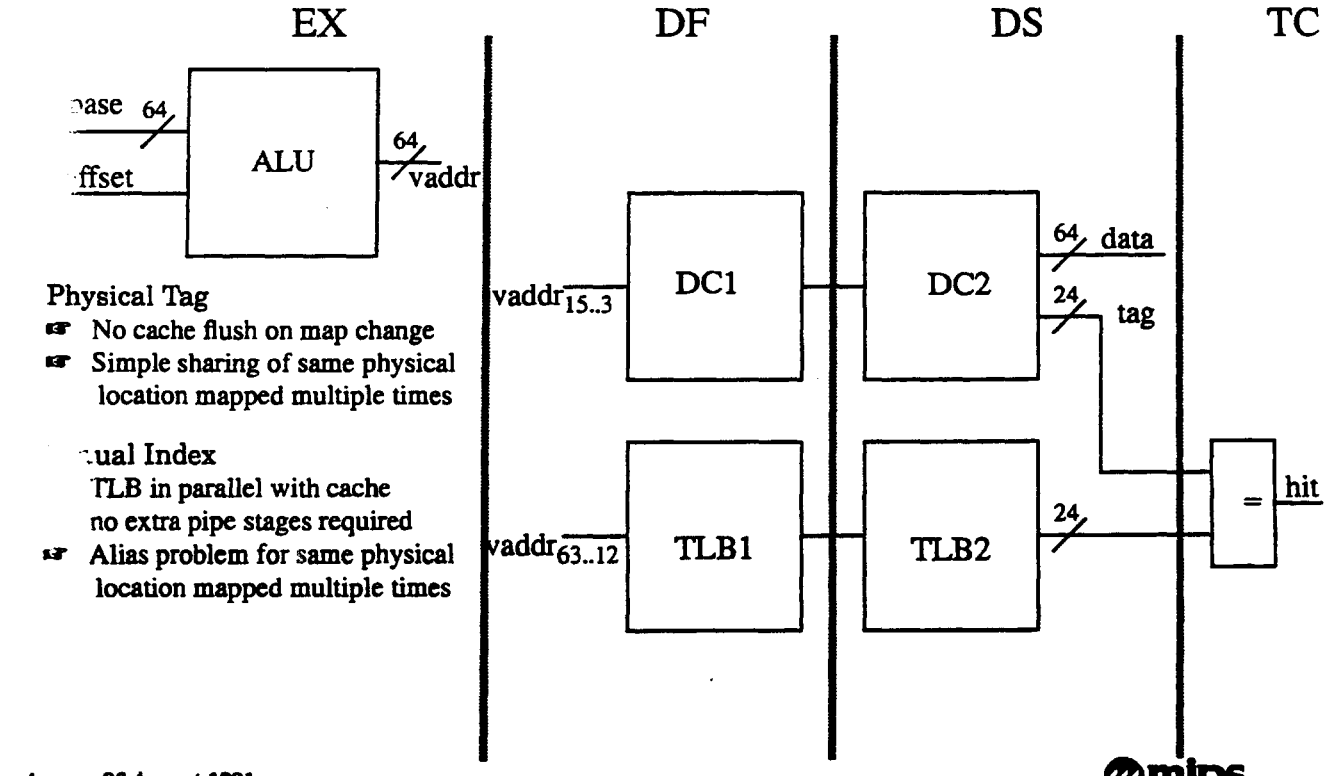
## Direct-mapped Primary Caches






mips

## Virtual Index, Physical Tag

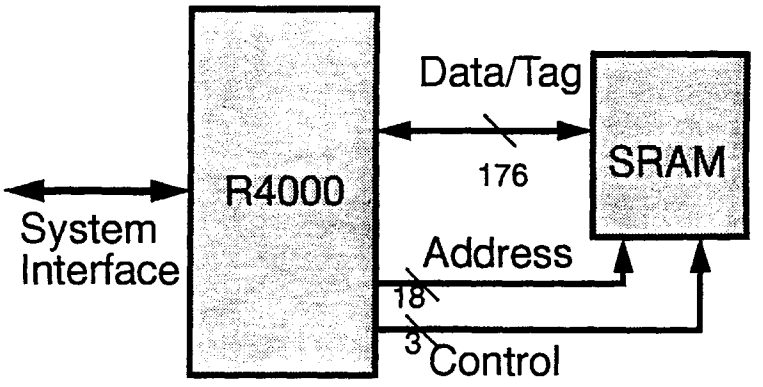





mips

## R4000 Secondary Cache

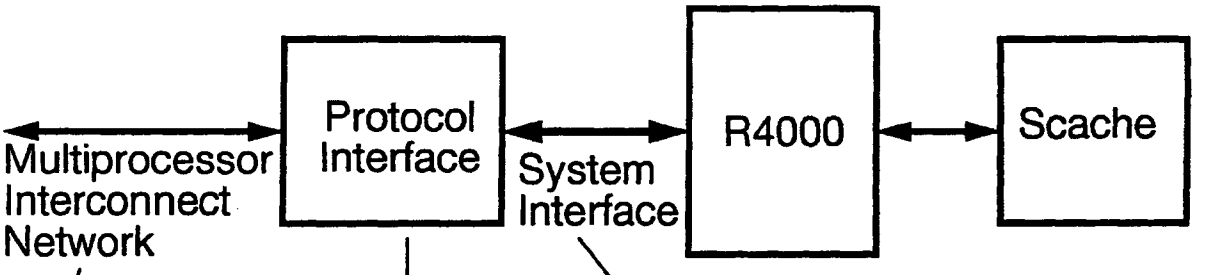
- External, standard asynchronous SRAMs
- 256KB to 4MB
- Direct-mapped
- Physical index, Physical Tag
- 128b Wide
- ECC data protection






mips

## R4000 Cache Coherency




*almost anything*

snoopy bus
cross-bar switch
butterfly switch

write-invalidate
write-update

*user ASIC*

*low-level, generic interface*

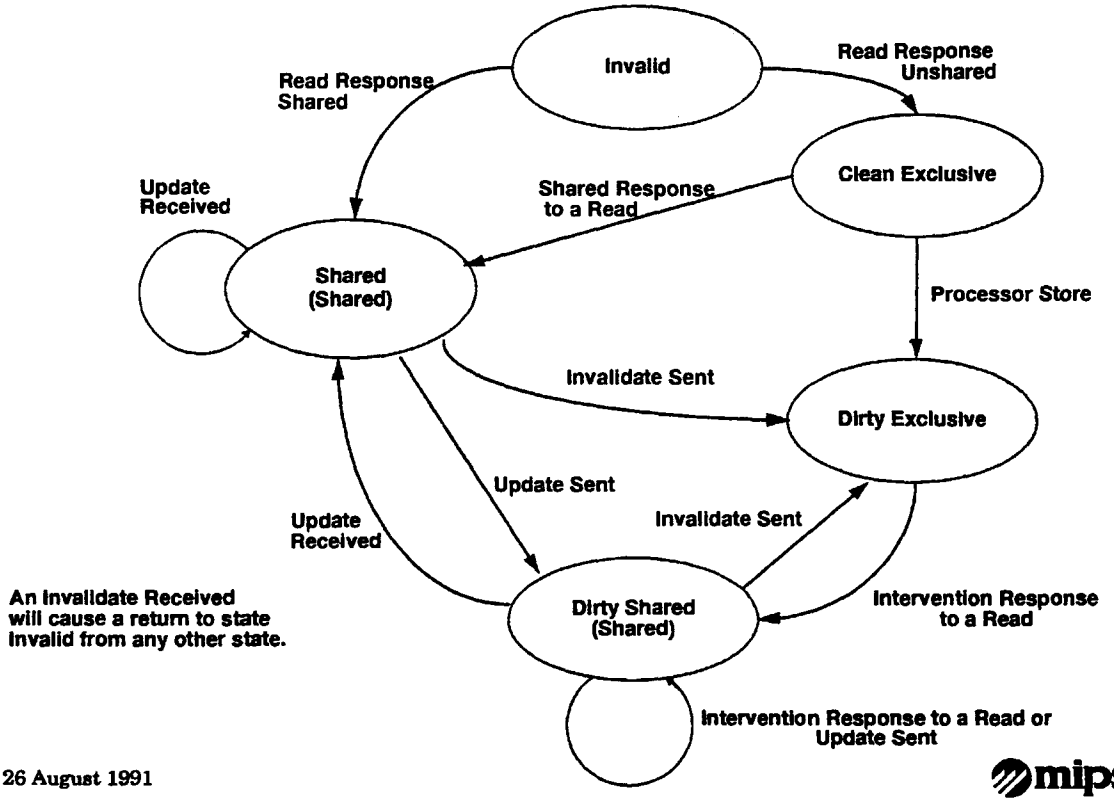
**Read Non-coherent**
**Read Coherent**
**Read Coherent Exclusive**
**Write**
**Invalidate**
**Update**

**Read Data**
**Read Data, Shared**
**Snoop Request**
**Intervention Request**
**State Change**
**Invalidate**
**Update**



mips

## R4000 Coherency States






mips

## R4000 Cache Coherency Methods

- Primary caches are forced to be a subset of Secondary cache

  *s-cache misses check and invalidate the primaries if necessary*

- Received Snoops, Interventions, Invalidates, and Updates cause R4000 to check S-cache in parallel without interrupting program execution

  *s-cache is physical index, physical tag, so snoops are trivial*

- Misses in the S-cache require no further action

  *no duplicate tags required, because s-cache serves as a filter*

- Hits may require access to the primary caches to complete the transaction, but the primary caches are virtually indexed, so we store the virtual index in the s-cache tag



mips

## R4000 Cache Coherency Options

- Page Table Entries select one of several cache algorithms

| algorithm | load miss | store miss | store hit |
|---|---|---|---|
| uncached | word read | word write | - |
| non-coherent | read non-coherent | read non-coherent | - |
| coherent exclusive | read exclusive | read exclusive | - |
| coherent write exclusive | read | read exclusive | - |
| coherent write update | read | read / update | update |

- Protocol Interface ASIC can control some state transitions, thus effecting a subset protocol



## R4000 Synchronization

LL/SC instructions provide synchronization between processes based only on cache coherency

Example (fetch-and-add):

```
Loop:
        LL      T0, 0(T1)       load counter
        ADDU    T1, T0, 1       increment
        SC      T1, 0(T1)       store back if unchanged
        BEQ     T1, 0, Loop     retry if store failed
```

SC fails if the location has been invalidated or updated since preceeding LL

Can implement semaphores, bit-locks, fetch-and-add, etc. with the LL and SC primitives