Arquitectura de Computadoras

TE2031

2017

Hector R. Sucar

Tema 7

SISTEMAS DUALES

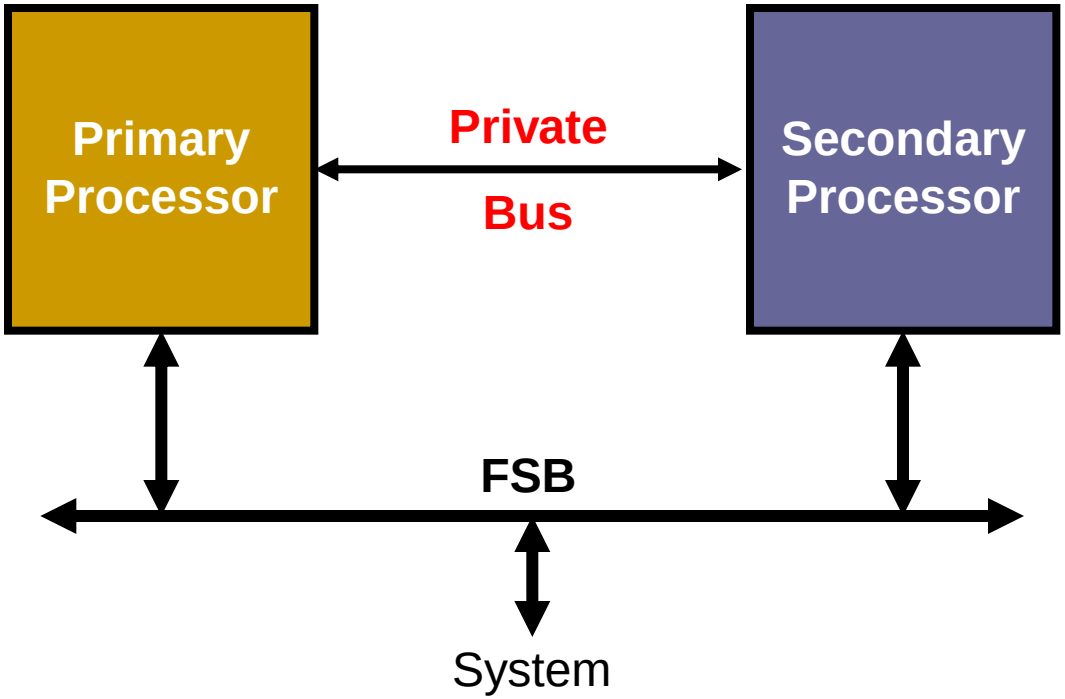
Señales Primarias de Control y Operación

Operación del Sistema

Session 13

Dual Processor System

Topology


Primary Processor
Private
Bus
Secondary Processor
FSB
System

Snoop Control/Response Signals

- input AHOLD – address hold: maintain memory consistency
- input EADS# - external address strobe: external; informs the processor of a valid address on its local bus to be snooped (internal cache lookup)
- input INV – invalidate: external; informs the processor to leave cache line valid or mark it invalid on a snoop hit
- inout HIT# - hit on internal cache – notify other processors that information is shared by another cache
- inout HITM# - hit modified line: hit to a modified line in cache, write-back required

Private Bus Control Signals

- inout PBGNT# - private bus grant: asserted to grant bus ownership
- inout PBREQ# - private bus request: asserted to request bus ownership
- inout PHIT# - private bus hit: indicates snoop result
- inout PHITM# - private bus hit modified
- output D/P# - Dual/Primary: asserted indicates bus ownership by primary processor

Dual Processor Setup

- Input pin CPUTYP is used to determine presence of a dual processor
- Primary processor has CPUTYP strapped low. Secondary processor has CPUTYP strapped high
- During reset, dual processor asserts output DPEN#. At the end of reset, primary processor samples input DPEN#; if asserted, dual processor is present, then, primary processor configures itself for dual processor operation

Multiprocessor Initialization

- Primary processor initiates program execution at the power-on restart address and sends startup inter-processor interrupt (IPI) to secondary processor with program start address. IPI message specifies start address of a 4KB page
- A symmetric multiprocessing OS is loaded and initialized by the primary processor. The OS places the startup code for the secondary processor in memory space and provides page start address

Bus Arbitration

- The two processors negotiate, directly with each other, to determine which gets ownership access.
- Assuming both processors may own access to the bus: the last processor that used the bus is the “Most Recent Master” (MRM) while the other is the “Least Recent Master” (LRM). The primary processor is the MRM first.

Arbitration Signals

- Private bus request PBREQ# - output from the LRM and input to the MRM
- Private bus grant PBGNT# - output from the MRM and input to the LRM
- MRM always monitors PBREQ# when it owns the bus. LRM requests bus ownership and MRM grants it when not in use. LRM detects PBGNT#. Then MRM and LRM switch and direction of PB signals is reversed. MRM initiates transactions after it takes ownership
- Bus remains parked on MRM while there is no request from LRM

Arbitration During Pipelined Transfers

- When MRM samples NA# asserted, then, target device supports pipelined transfers. MRM may pipeline another transfer in the next clock cycle
- However, if LRM has asserted PBREQ#, MRM will assert PBGNT# and not pipeline another transfer. LRM requests take precedence over pipelined transfers

Group Activity

- Define sequence of events for two cache line fills, each occurring on a different processor
- Assume the processors are identified as PA and PB and PA is the current MRM
- Consider appropriate bus cycle signals

Session 14

Address Snooping

Line-stored State

- Shared “S”:
  - line updated in internal cache and in system memory (write-through)
- Exclusive “E”
  - location stored in internal cache will be updated with no bus cycle transitioning to M state
- Modified “M”
  - location stored in internal cache updated with no bus cycle
- Invalid “I”
  - location stored in internal cache obsolete

Address Snooping
Single Processor System

- Access by a bus master to cacheable memory, information may exist in the processor L1 cache; then, processor must snoop the memory address and report results to external logic:

1. Asserting AHOLD instructs the processor to disconnect from the address bus
2. External address is presented to the processor on the address bus
3. EADS# asserted indicates valid address to be snooped. Then the processor latches the address and performs internal cache lookup

Single Processor Snoop (cont)

4. Snoop results are presented on HIT# and HITM# outputs
5. If snoop results in a hit on L1 and INV was deasserted, the line is presented on the data bus (shared state)
6. If snoop results in a hit on L1 and INV was asserted the processor will invalidate its copy of the line

- A hit on a modified line asserts HITM# and a snoop write back is initiated (bus master must release the bus to the processor)

Address Snooping
Dual Processor System

- Address snooping requires that the two processors maintain cache coherency between their caches and system memory
- Inter-processor cache snooping always occurs since LRM always snoops MRM transactions
- External cache snooping, another bus master is accessing system memory, processors must snoop external address

Inter-Processor Cache Snooping

- LRM monitors MRM bus transactions and snoops the transaction:
  - MRM asserts ADS# to initiate a transaction; LRM monitors ADS#
  - MRM drives address onto the bus. LRM latches address
  - MRM asserts CACHE# when it initiates cache line read or write-back of modified line. LRM latches CACHE#
  - Bus transaction definition outputs of the MRM are inputs to the LRM to determine when the MRM is accessing memory (M/IO#, D/C#, W/R#, LOCK#)
- When LRM detects bus cycle performed by MRM, snoops address to determine if it has a copy of the location being accessed. LRM action depends on type of transfer the MRM is performing and the result of the snoop

Snoop Memory Reads and Writes by MRM

- LRM detects MRM memory read or write transaction from/to a cacheable area of memory (ADS#, cycle definition signals).
- Possible snoop results:
  - Snoop miss
  - Snoop hit clean
  - Snoop hit modified

Snoop Results - Reads

- Miss: line not cached in LRM, keeps PHIT# and PHITM# deasserted. MRM completes memory read and stores line in E state
- Hit Clean: line cached in LRM in E or S state, asserts PHIT# but not PHITM#, line state transitions to S (or stays in S), MRM completes memory read and stores line in S state
- Hit Modified: line cached in LRM in M state, LRM asserts PHIT# and PHITM#, LRM and MRM switch for memory write back and switch back for memory read. Final line state transitions from M to S

Snoop Results - Writes

- Miss: line not cached in LRM, keeps PHIT# and PHITM# deasserted. MRM completes memory write
- Hit Clean: line cached in LRM in E or S state, asserts PHIT# but not PHITM#, MRM completes memory write. LRM invalidates its copy (transitions to I state)
- Hit Modified: line cached in LRM in M state, LRM asserts PHIT# and PHITM# and transitions line to I state, LRM and MRM switch for memory write back. LRM and MRM switch back for memory read to capture data from write-back. MRM initiates memory write and LRM snoops again resulting in a miss (line was invalidated). Write completes and neither processor has a copy of the target cache line

External Cache Snooping

- Both processors must snoop transfers by other masters (e.g. L2 controller, PCI)
- Other bus masters access cacheable memory
- Both processors are forced to snoop the transaction and report results (AHOLD, EADS#, INV, HIT#, HITM#) like a single processor system

Snoop Memory Reads and Writes by Other Masters

- Both processors snoop, one may have a copy of the location in S, E, or M state.
- Possible snoop results:
  - MRM miss / LRM miss
  - MRM hit clean / LRM miss
  - MRM miss / LRM hit clean
  - MRM hit clean / LRM hit clean
  - MRM hit modified / LRM miss
  - MRM miss /LRM hit modified

Group Activity

- Describe the operation of all six snoop results for external cache snooping

Solution - Reads

- MRM/LRM snoop miss: line is not cached in either processor, both report miss (HIT# and HITM# deasserted). Bus master completes memory read
- MRM snoop hit clean / LRM snoop miss: line is cached in MRM (S or E state; E transitions to S). MRM reports hit (HIT# asserted, HITM# deasserted)
- MRM snoop miss / LRM snoop hit clean: line is cached in LRM, (S or E state; E transitions to S). LRM reports hit on PB (PHIT asserted, PHITM# deasserted). Then MRM asserts HIT# to notify the system

Solution - Reads (cont)

- MRM snoop hit clean / LRM snoop hit clean: line is cached in MRM and LRM (S state). MRM and LRM report hit (HIT# & PHIT# asserted, and , HITM# and PHITM# deasserted; respectively)
- MRM snoop hit modified / LRM snoop miss: line cached in MRM in M state. MRM reports hit asserting HIT# and HITM#, taking control of the bus, and proceeds with write-back (M transitions to S). After write-back completes, MRM deasserts HIT# and HITM# and releases bus control for the bus master to complete memory read
- MRM snoop miss / LRM snoop hit modified: line cached in LRM in M state. LRM reports hit asserting PHIT# and PHITM#. Then, MRM asserts HIT# and HITM#, taking control of the bus. LRM asserts PBREQ# to request bus ownership to perform write-back (M transitions to S). MRM grants ownership asserting PBGNT#. MRM and LRM switch. New MRM writes line to memory. After write-back completes, the MRM (original LRM) deasserts PHIT# and PHITM#, causing the LRM to deassert HIT# and HITM#; releasing bus control for the bus master to complete memory read

Solution - Writes

- MRM/LRM snoop miss: line is not cached in either processor, both report miss (HIT# and HITM# deasserted). Bus master completes memory write
- MRM snoop hit clean / LRM snoop miss: line is cached in MRM (S or E state; transitions to I state). MRM reports hit (HIT# asserted, HITM# deasserted)
- MRM snoop miss / LRM snoop hit clean: line is cached in LRM, (S or E state; transitions to I state). LRM reports hit on PB (PHIT asserted, PHITM# deasserted). Then MRM asserts HIT# to notify the system

Solution – Writes (cont)

- MRM snoop hit clean / LRM snoop hit clean: line is cached in MRM and LRM (S state; both transition to I state). MRM and LRM report hit (HIT# & PHIT# asserted, and , HITM# and PHITM# deasserted; respectively)
- MRM snoop hit modified / LRM snoop miss: line cached in MRM in M state. MRM reports hit asserting HIT# and HITM#, taking control of the bus, and proceeds with write-back (M transitions to I). After write-back completes, MRM deasserts HIT# and HITM# and releases bus control for the bus master to complete memory write
- MRM snoop miss / LRM snoop hit modified: line cached in LRM in M state. LRM reports hit asserting PHIT# and PHITM#. Then, MRM asserts HIT# and HITM#, taking control of the bus. LRM asserts BPREQ# to request bus ownership to perform write-back (M transitions to I). MRM grants ownership asserting PBGNT#. MRM and LRM switch. New MRM writes line to memory. After write-back completes, the MRM (original LRM) deasserts PHIT# and PHITM#, causing the LRM to deassert HIT# and HITM#; releasing bus control for the bus master to complete memory write