



CHƯƠNG 4: KỸ THUẬT PIPELINE

- Tìm hiểu chung về pipeline.
- Xung đột (Hazard).
 - Xung đột cấu trúc (Structural Hazard).
 - Xung đột dữ liệu (Data Hazard).
 - Xung đột điều khiển (Control Hazard).
- Giải quyết xung đột.
- Ngoại lệ
- Kỹ thuật pipeline mới.
 - Super pipelining 8 tầng.
 - Super Scalar – Dual pipeline.
- Patterson, D. A., and J. L. Hennessy. [Computer Organization and Design: The Hardware/Software Interface](#), 3rd ed. San Mateo, CA: Morgan Kaufman, 2004, chapter 3. Pipeline

Khoa KTMT

Vũ Đức Lung

1



KHÁI NIỆM PIPELINE

- Tuần tự Von Neumann và Pipeline?
- Cấu trúc tuần tự :
 - Thực hiện các lệnh một cách tuần tự.
 - Theo 5 khâu :
 - IF (Instruction Fetch) : Nhận lệnh
 - ID (Instruction Decode) : Giải mã lệnh
 - DF (Data Fetch) : Nhận dữ liệu
 - EX (Execution) : Thực hiện lệnh
 - DS (Data Save) : Lưu kết quả

Khoa KTMT

Vũ Đức Lung

2



Cấu trúc tuần tự

Giả sử...

- Mỗi lệnh thực hiện trong 1 chu kì τ , mỗi khâu thực hiện trong thời gian $\tau/5$...
- Với n lệnh :

$$T_{\text{tuần tự}} = \tau * n$$

Khoa KTMT

Vũ Đức Lung

3



Hạn chế của cấu trúc tuần tự

- Các lệnh được thực hiện liên tiếp nhau.
- Lệnh trước thực hiện xong mới đến lệnh sau.
- Vì vậy xuất hiện khoảng thời gian rỗi (stall) giữa các khâu.
- Kỹ thuật pipeline được đưa ra để tận dụng những stall này, từ đó tăng tốc độ cho vi xử lý.

Khoa KTMT

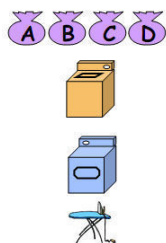
Vũ Đức Lung

4



VD Pipelining

- Tại tiệm giặt ủi: An, Bình, Chinh, Diệu mỗi người có thể sử dụng 1 lần gồm: giặt, sấy khô và ủi
- Giặt mất 30 phút
- Xấy khô mất 40 phút
- Ủi mất 20 phút



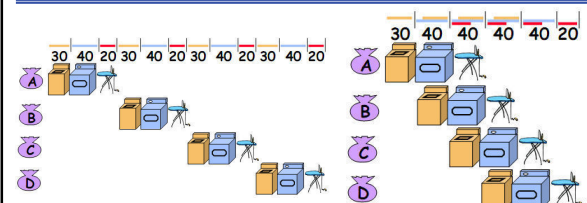
Khoa KTMT

Vũ Đức Lung

5



VD Pipelining



Khoa KTMT

Vũ Đức Lung

6



Định nghĩa Pipelining

- Pipelining là một kỹ thuật thực hiện lệnh trong đó các lệnh được thực hiện theo kiểu gổ đầu nhằm tận dụng những khoảng thời gian rỗi (stalls) giữa các công đoạn (stages), qua đó làm tăng tốc độ thực hiện lệnh của VXL.

Khoa KTMT

Vũ Đức Lung

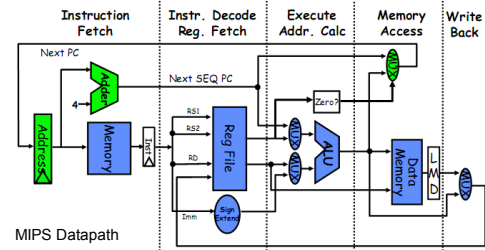
7



Cấu trúc Pipeline

5 khâu của một lệnh trong MIPS:

F (Fetch) : Nhận lệnh. D (Decode) : Giải mã lệnh. X (Execution) : Thực hiện lệnh. M (Memory Access) : Truy nhập bộ nhớ. W (Result Write Back) : Ghi kết quả.



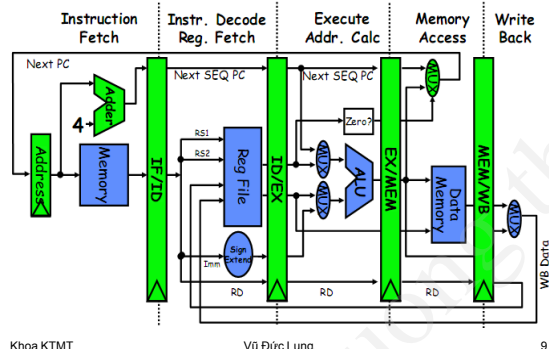
Khoa KTMT

Vũ Đức Lung

8



5 tầng pipeline trong MIPS với buffer



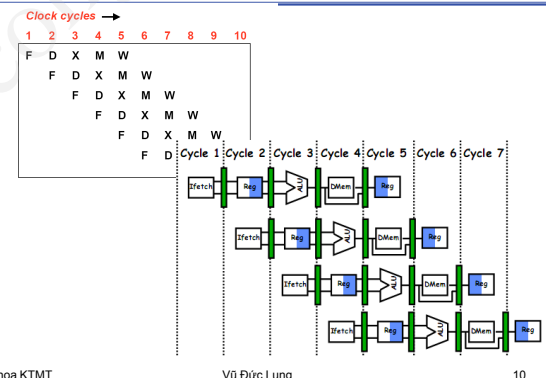
Khoa KTMT

Vũ Đức Lung

9



Mô hình pipeline lý tưởng...



Khoa KTMT

Vũ Đức Lung

10



Thông lượng trung bình của mô hình pipeline lý tưởng là 1CPI (clock cycle per instruction)...

- Trong trường hợp không có xung đột có thể tăng tốc độ vi xử lý lên 400%.
- Tính toán:
- Thời gian để thực hiện 1 công đoạn là $\tau/5$.
 - Thời gian để thực hiện 1 lệnh là τ
 - Thời gian để thực hiện 2 lệnh là $\tau + \tau/5$
 - Thời gian để thực hiện 3 lệnh là $\tau + \tau * 2/5$
 - ...
 - Thời gian để thực hiện n lệnh là $\tau + \tau * (n-1)/5$
- Tổng quát : $T_{\text{pipeline}} = \tau + \tau * (n-1)/m$

Khoa KTMT

Vũ Đức Lung

11



Xung đột (Hazard)

- Trên lý thuyết, việc sử dụng kỹ thuật pipeline sẽ làm tăng tốc độ VXL lên gần 400% nhưng thực tế, việc tăng tốc độ được bao nhiêu còn phụ thuộc vào các kiểu xung đột khác nhau dưới đây.
- Xung đột cấu trúc (Structural Hazard)
 - Xung đột dữ liệu (Data Hazard)
 - Xung đột điều khiển (Control Hazard)

Khoa KTMT

Vũ Đức Lung

12



Xung đột cấu trúc (Structural Hazard)

- Xung đột cấu trúc xảy ra khi có 2 lệnh cùng cố gắng sử dụng cùng 1 nguồn tại cùng 1 thời điểm.
 - Khi 2 lệnh cùng ghi kết quả vào 1 thanh ghi:
 - ADD R1, R2, R3
 - SUB R1, R4, R5
 - Khi cả 2 lệnh cùng truy cập vào 1 ô nhớ tại cùng một thời điểm.
 - Khi cả 2 lệnh cùng yêu cầu một bộ tính toán số học (bộ cộng, bộ nhân, bộ chia).

Khoa KTMT

Vũ Đức Lung

13



Xung đột cấu trúc (Structural Hazard)

	1	2	3	4	5	6	7	8	9	10
1	F	D	X	M	W					
2		F	D	X	M	W				
3			F	D	X	M	W			
4				o	o	o	F	D	X	M
5					o	o	o	F	D	X

bubbles

- Xung đột xảy ra khi việc nạp lệnh và đọc dữ liệu từ bộ nhớ diễn ra cùng lúc
- Những kí hiệu "o" chèn vào tượng trưng cho chu kì trễ (stall cycles) sẽ được sử dụng nếu ta sử dụng bộ nhớ đơn lưu trữ cả lệnh vào dữ liệu.

Khoa KTMT

Vũ Đức Lung

14

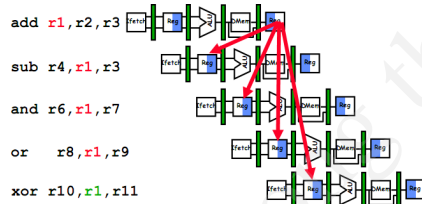


Xung đột cấu trúc (Structural Hazard)

- Khi cả stage X và D đều yêu cầu bộ cộng, mà chỉ có 1 bộ cộng trong VXL.

1	F	D	X	M	W
2		F	D	X	M

- Cùng yêu cầu r1



Khoa KTMT

Vũ Đức Lung

15



Xung đột dữ liệu (Data Hazard)

RAW (Read after Write)

- Instruction 1: ADD R2, R1, R3 $R2 \leftarrow R1 + R3$
- Instruction 2: ADD R4, R2, R3 $R4 \leftarrow R2 + R3$

I1	F	D	X	M	W
I2		F	D	X	M

Khoa KTMT

Vũ Đức Lung

16



Xung đột dữ liệu (Data Hazard)

WAR (Write after Read)

- Instruction 1: ADD R1, R2, R3 $R1 \leftarrow R2 + R3$
- Instruction 2: ADD R3, R4, R5 $R3 \leftarrow R4 + R5$
- I2 ghi vào R3 trước khi I1 đọc R3

I1	F	D	X	M	W
I2		F	D	X	M

Không có xung đột

- Chỉ xuất hiện trong trường hợp M chiếm 2 pipe stages
- Write back trong các lệnh ALU nằm ở M stage
- VD:

SW 0(R1), R2	IF	ID	EX	MEM1	MEM2	WB
ADD R2, R3, R4						WB

Khoa KTMT

Vũ Đức Lung

17



Xung đột dữ liệu (Data Hazard)

WAW (Write after Write)

- Instruction 1: ADD R2, R1, R3 $R2 \leftarrow R1 + R3$
- Instruction 2: ADD R2, R4, R7 $R2 \leftarrow R4 + R7$

I1	F	D	X	M	W
I2		F	D	X	M

Không có xung đột

- Chỉ xuất hiện trong trường hợp M chiếm 2 pipe stages
- Write back trong các lệnh ALU nằm ở M stage
- VD:

LW R1, 0(R2)	IF	ID	EX	MEM1	MEM2	WB
ADD R1, R2, R3						WB

Khoa KTMT

Vũ Đức Lung

18



Xung đột điều khiển (Control Hazard)

Control Hazard xảy ra khi có lệnh rẽ nhánh, do đó còn gọi là Branch Hazard. Khi lệnh rẽ nhánh được yêu cầu thực hiện, con trỏ bộ đếm chương trình (PC) sẽ chuyển tới địa chỉ đích bằng cách cộng thêm 4. Nếu con trỏ chương trình nhảy tới đúng địa chỉ đích của nó, thì rẽ nhánh này gọi là nhánh **Taken**; trong trường hợp ngược lại gọi là nhánh **Untaken**.

Khi lệnh i có nhánh taken thì PC sẽ không thay đổi như bình thường tới hết khâu M (memory access), sau khi tính toán và so sánh địa chỉ. Phương pháp đơn giản nhất để khắc phục control hazard là gây trễ kịp thời trên pipeline để phát hiện nhánh cho đến khâu M, sử dụng giá trị mới của PC.

Khoa KTMT

Vũ Đức Lung

19



Xung đột điều khiển (Control Hazard)

- Ta không mong muốn gây trễ trên pipeline khi lệnh chỉ có một nhánh, vì thế trễ không xuất hiện tới sau khâu D và sẽ thực hiện tiếp.

Lệnh nhánh	F	D	X	M	W						
Nhánh kế thừa		F	Sta II	Sta II	F	D	X	M	W		
Nhánh kế thừa +1						F	D	X	M	W	
Nhánh kế thừa +2							F	D	X	M	W
Nhánh kế thừa +3								F	D	X	M
Nhánh kế thừa +4									F	D	X
Nhánh kế thừa +5										F	D

- Ba chu kỳ bỏ phí ở mỗi nhánh là hao phí có nghĩa với tần số nhánh là 30% và 1 CPI lý tưởng là 1, máy tính có trễ rẽ nhánh sẽ đạt khoảng $\frac{1}{2}$ lượng tăng tốc lý tưởng của pipeline.

Khoa KTMT

Vũ Đức Lung

20



Giải quyết xung đột (Resolving Hazard)

Xung đột (Hazard) là một yếu tố quan trọng ảnh hưởng trực tiếp tới tốc độ của VXL trong kỹ thuật Pipeline. Do vậy, việc giải quyết xung đột (Resolving Hazard) là rất cần thiết. Có một số kỹ thuật giải quyết xung đột chủ yếu sau đây:

- Chèn trễ.
- Tổ chức lại các lệnh
- Sử dụng đường dữ liệu nội đặc biệt.
- Tomasulo.
- Định biểu.

Khoa KTMT

Vũ Đức Lung

21



Chèn trễ

- Kỹ thuật chèn trễ được sử dụng khá hữu hiệu để giải quyết các xung đột về cấu trúc cũng như về dữ liệu:

- Ví dụ:

LW R1, 0(R2)	F	D	X	M	W
SUB R4, R1, R5		F	D	X	M
AND R6, R1, R7			F	D	X
OR R6, R1, R9			F	D	X

- Ta nhận thấy xung đột dữ liệu xảy ra khi lệnh 1 chưa lưu kết quả vào R1 thì lệnh 2 đã thực hiện trừ R1 cho R5, đây là xung đột dữ liệu RAW (Đọc sau đó mới ghi).

Khoa KTMT

Vũ Đức Lung

22



Chèn trễ

- Chèn trễ:

LW R1, 0(R2)	F	D	X	M	W
SUB R4, R1, R5		F	D	X	M
AND R6, R1, R7			F	D	X
OR R6, R1, R9				F	D

- Kỹ thuật chèn trễ này hoạt động khá ổn định, tuy nhiên vẫn còn chưa tận dụng được nhiều chu kỳ nhàn rỗi của máy, do đó hiệu suất chưa cao.

Khoa KTMT

Vũ Đức Lung

23



Tổ chức lại các lệnh (Instruction Reorganization)

- Kỹ thuật này đòi hỏi trình dịch phải dự đoán được sự phụ thuộc dữ liệu giữa các lệnh, qua đó thay đổi trật tự thực hiện lệnh.

- Xét ví dụ: Tính $a := b + c;$ $d := e - f$

Slow code:	Fast code:
LW Rb, b	LW Rb, b
LW Rc, c	LW Rc, c
STALL	LW Re, e
ADD Ra, Rb, Rc	ADD Ra, Rb, Rc
SW a, Ra	
LW Re, e	
LW Rf, f	LW Rf, f
STALL	SW a, Ra
SUB Rd, Re, Rf	SUB Rd, Re, Rf
SW d, Rd	SW d, Rd

10 cycles (2 stalls) 8 cycles (0 stalls)

- Tuy nhiên phương pháp này đòi hỏi 1 trình dịch thông minh, chi phí cho trình dịch vì thế sẽ tăng.

Khoa KTMT

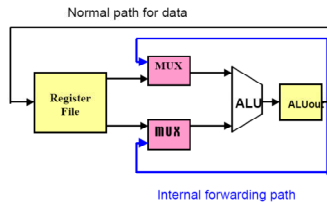
Vũ Đức Lung

24



Sử dụng đường dữ liệu nội đặc biệt

- Giá trị của biến sẽ được cập nhập rất sớm nhờ sử dụng đường dữ liệu nội đặc biệt. Điều này làm giảm số chu kỳ nhân rồi trong pipeline và tăng tốc độ VXL
- Internal Data Forwarding:**



Khoa KTMT

Vũ Đức Lung

25



Sử dụng đường dữ liệu nội đặc biệt

- Dữ liệu sẽ được lưu ở bộ đệm ALUout sau pha X, và sẽ được ghi lên thanh ghi trong pha M (hoặc là trong pha W đối với lệnh LW). Nhờ việc sử dụng bộ đệm ALUout để cất dữ liệu và sử dụng -> giải quyết được phần lớn các xung đột về dữ liệu cũng như cấu trúc -> sẽ làm giảm đáng kể số chu kỳ nhân rồi của máy -> tăng tốc độ xung nhịp.

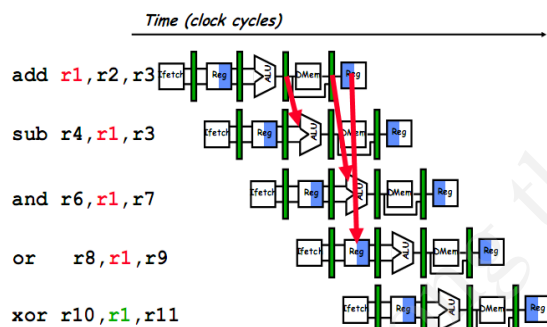
Khoa KTMT

Vũ Đức Lung

26



Sử dụng đường dữ liệu nội đặc biệt



Khoa KTMT

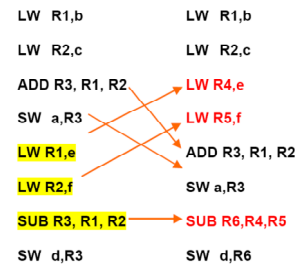
Vũ Đức Lung

27



Sử dụng đường dữ liệu nội đặc biệt

- Phương án tối ưu cho ví dụ trên kết hợp sử dụng cả Tổ chức lại các lệnh và đường dữ liệu nội đặc biệt



Khoa KTMT

Vũ Đức Lung

28



Tomasulo

- Thuật toán Tomasulo được đưa ra vào năm 1967 bởi Robert Tomasulo làm việc cho hãng IBM. Ý tưởng của thuật toán là sử dụng địa chỉ Tag và bit trạng thái để đánh dấu các thanh ghi, qua đó biết được thanh ghi đó đang bận hay rồi để sử dụng cho phù hợp.

Khoa KTMT

Vũ Đức Lung

29



Định biểu (scheduling)

- Sử dụng các vector trạng thái và bảng đặt chỗ (reservation table) để đánh dấu các trạng thái của các tầng pipeline.
- Từ đó xác định được số chu kỳ cần đưa dữ liệu vào để tránh xung đột.

Khoa KTMT

Vũ Đức Lung

30



Ngoại lệ

- Định nghĩa
- Các kiểu ngoại lệ và các yêu cầu
- Dừng và khởi động lại quá trình thực thi
- Ngoại lệ trong DLX pipeline

Khoa KTMT

Vũ Đức Lung

31



Định nghĩa

- Ngoại lệ xảy ra khi thứ tự thực hiện các câu lệnh bị thay đổi không mong muốn.
- Trong kỹ thuật pipeline, câu lệnh được thực hiện từng phần và không thể hoàn thành trong vài chu kỳ đồng hồ. Các lệnh có thể gây ra các ngoại lệ mà khiến bộ máy phải từ bỏ câu lệnh trong pipeline trước khi nó được hoàn thành.

Khoa KTMT

Vũ Đức Lung

32



Các kiểu ngoại lệ và các yêu cầu

Các kiểu ngoại lệ :

Chúng ta sử dụng khái niệm ngoại lệ để nói về các trường hợp sau:

- I/O device request (yêu cầu thiết bị I/O).
- Invoking an operating system service from user program (gọi một dịch vụ hệ điều hành từ người sử dụng chương trình).
- Tracing instruction execution (theo dấu sự thực hiện câu lệnh).
- Breakpoint (điểm ngắt – khi lập trình viên yêu cầu ngắt).
- Integer arithmetic overflow (tràn số học).
- FP arithmetic anomaly (dị thường số học dấu phẩy động).
- Page fault (not in memory) - đứt đoạn trang (không trong bộ nhớ).
- Misaligned memory access (không căn hàng khi truy cập bộ nhớ).
- Memory – protection violation (vi phạm vùng nhớ bảo vệ).
- Using an undefined or unimplemented instruction (sử dụng câu lệnh chưa định nghĩa hoặc chưa bổ sung).
- Hardware malfunctions (sự cố phần cứng).
- Power failure (thiếu nguồn).

Khoa KTMT

Vũ Đức Lung

33



Các kiểu ngoại lệ và các yêu cầu

Các yêu cầu:

Các luồng yêu cầu có thể gây ra ngoại lệ:

- Synchronous and asynchronous (đồng bộ và không đồng bộ).
- User requested and coerced (người sử dụng yêu cầu và ép buộc).
- User maskable and user nonmaskable (có thể che giấu và không thể che giấu).
- Within and between instructions (bên trong và giữa các câu lệnh).
- Resume and terminate (phục hồi lại và chấm dứt).

Khoa KTMT

Vũ Đức Lung

34



Dừng và khởi động lại quá trình thực thi

❑ Đặt vấn đề:

Ví Dụ:

Sự đứt đoạn trang trong đường ống dạng DLX làm kết quả của dữ liệu cần đưa ra không thể xuất hiện trong bước MEM của quá trình thực hiện câu lệnh. Khi có sự đứt đoạn thì một vài câu lệnh khác vẫn thực hiện nhưng chúng không thể cho ra kết quả.

Do đó, phần đứt đoạn trang phải được khởi động lại và yêu cầu sự can thiệp của các xử lý (như hệ điều hành). Chính vì thế, pipeline phải được tắt một cách an toàn, công đoạn thực hiện nhớ lại, và sau đó câu lệnh phải được thực khởi động lại tại đúng vùng mà nó thực hiện.

Khoa KTMT

Vũ Đức Lung

35



Dừng và khởi động lại quá trình thực thi

❑ Chính vì thế hầu hết các ngoại lệ khó có 2 thuộc tính chúng ta đáng quan tâm là:

- Chúng xuất hiện ở bước nào trong các câu lệnh (vd: giữa giai đoạn EX và MEM).
- Chúng có khả năng khởi động lại hay không.

Các bước lưu các tầng pipeline một cách an toàn:

1. Ép các câu lệnh có vấn đề trong pipeline vào cờ ngắt (IF) kế tiếp.
2. Khi xuất hiện vấn đề, tắt tất cả việc ghi cho các câu lệnh đứt đoạn và các câu lệnh trong pipeline. Việc đó có thể thực hiện bằng cách đặt giá trị 0 tại các chốt đường ống của tất cả các câu lệnh trong pipeline. Khởi động với các câu lệnh gây ra ngoại lệ; không áp dụng cho các pipeline trước câu lệnh đó. Việc này sẽ bảo vệ bất kỳ công đoạn bị thay đổi của các câu lệnh chưa được hoàn thành trước khi ngoại lệ được kiểm soát.
3. Sau khi những ngoại lệ được kiểm soát, thường là khi Hệ điều hành nhận được điều khiển, các ngoại lệ và các câu lệnh sai sẽ được đánh dấu, nó ngay lập tức lưu trữ trên bộ đếm chương trình (PC). Giá trị lưu trữ này sẽ được sử dụng để đáp lại (giải quyết) những ngoại lệ sau đó.

Khoa KTMT

Vũ Đức Lung

36

Ngoại lệ trong DLX pipeline

□ Trong kỹ thuật pipeline, nhiều ngoại lệ có thể xuất hiện tại cùng một chu kỳ đồng hồ do nhiều lệnh được thực hiện đồng thời.

Pipeline stage	Problem exceptions occurring
IF	Page fault on Instruction fetch; Misaligned memory access; Memory – protection violation
ID	Undefined or illegal opcode
EX	Arithmetic exception
MEM	Page fault on Instruction fetch; Misaligned memory access; Memory – protection violation
WB	None

Khoa KTMT Vũ Đức Lung 37

Ngoại lệ trong DLX pipeline

□ Ví dụ:

LW	IF	ID	EX	MEM	WB	
ADD		IF	ID	EX	MEM	WB

▪ Khi lệnh LW trong công đoạn MEM và trong khi lệnh ADD đang thực hiện trong công đoạn EX, các ngoại lệ page fault và arithmetic có thể xuất hiện cùng một lúc. Trường hợp này có thể được kiểm soát bằng cách chỉ giải quyết với ngoại lệ page fault và sau đó khởi động lại ngoại lệ. Ngoại lệ thứ hai sẽ lại xuất hiện sau khi khởi động lại và nó sẽ được kiểm soát một cách độc lập.

Khoa KTMT Vũ Đức Lung 38

Ngoại lệ trong DLX pipeline

- Trong thực tế, tình huống không đơn giản như ví dụ trên. Các ngoại lệ sẽ xuất hiện không theo thứ tự nào; khi đó, một câu lệnh ở sau có thể gây ra các ngoại lệ trước khi các câu lệnh trước đó gây ra các ngoại lệ. Xem xét lại ví dụ trên, ADD thực hiện sau LW. LW gặp ngoại lệ page fault nếu chúng trong công đoạn MEM, nhưng lệnh ADD cũng có thể gặp page fault khi chúng trong công đoạn IF.
- Kết luận:** Pipeline không thể kiểm soát được các ngoại lệ khi chúng xuất hiện cùng 1 lúc. Do đó chúng sẽ dẫn tới các ngoại lệ xuất hiện không theo thứ tự pipeline
- Yêu cầu từ ví dụ trên:** pipeline yêu cầu phải kiểm soát ngoại lệ gây ra bởi lệnh LW trước lệnh ADD.

Khoa KTMT Vũ Đức Lung 39

Ngoại lệ trong DLX pipeline

□ Giải pháp:

- Phần cứng sẽ đưa tất cả các ngoại lệ được gây ra bởi các câu lệnh vào trong một vector trạng thái ngoại lệ kết hợp với các câu lệnh đó.
- Vector này tồn tại xuyên suốt trong pipeline cho tới khi các câu lệnh kết thúc.
- Nếu có dấu hiệu xuất hiện của ngoại lệ nào, thì ngoại lệ đó sẽ được thiết lập trong vector trạng thái.
- Bất kỳ một tín hiệu điều khiển nào mà có thể tiến hành việc ghi dữ liệu đều bị tắt (bao gồm cả việc ghi vào thanh ghi và ghi vào bộ nhớ). Bởi vì, bước MEM có thể gây ra ngoại lệ, và phần cứng phải chuẩn bị để bảo vệ vùng thực hiện khỏi sự hoàn thành nếu xuất hiện ngoại lệ.
- Khi câu lệnh sang công đoạn WB (hoặc ra khỏi MEM), vector trạng thái được đánh dấu. Nếu có bất kỳ một ngoại lệ nào được đặt vào, chúng sẽ được kiểm soát theo thứ tự mà chúng xuất hiện. Với sự đảm bảo này, tất cả những ngoại lệ được gây ra bởi câu lệnh thứ i sẽ được xem xét trước câu lệnh thứ i + 1.

Khoa KTMT Vũ Đức Lung 40

Mở rộng DLX Pipeline để kiểm soát toán tử đa chu kỳ

□ Tại sao phải nghiên cứu (kiểm soát) toán tử đầu phẩy động?

→ Các toán tử đầu phẩy động khó có thể được hoàn thành trong 1 hoặc 2 chu kỳ đồng hồ. Do đó, sự thực hiện của pipeline đối với toán tử đầu phẩy động (nếu có) có những điểm “đặc biệt” so với dạng DLX thông thường (IF, ID, EX, MEM, WB)

□ Giải quyết: giả sử, các câu lệnh đầu phẩy động cũng nằm trong pipeline giống như các câu lệnh nguyên, nhưng có 2 sự thay đổi quan trọng

- Bước EX có thể lặp lại nhiều lần bằng số lần cần thiết cho việc hoàn thành sự thực hiện của toán tử.
- Sẽ có thêm các khối chức năng đầu phẩy động:

Khoa KTMT Vũ Đức Lung 41

Mở rộng DLX Pipeline để kiểm soát toán tử đa chu kỳ

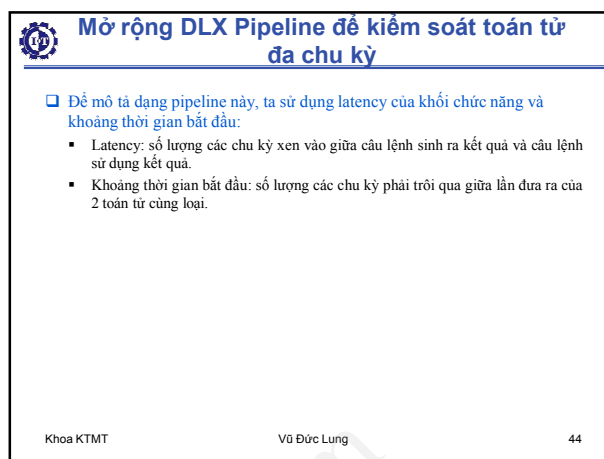
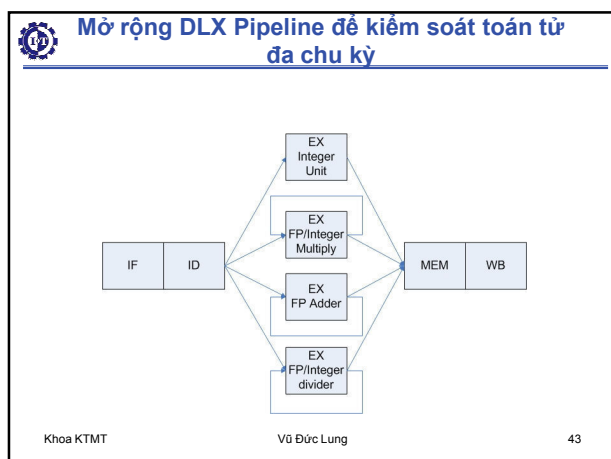
□ Thừa nhận có 4 khối chức năng riêng biệt trong sự hoạt động của DLX

- Khối nguyên chính, kiểm soát việc tải và lưu các toán tử nguyên ALU và việc rẽ nhánh.
- Bộ nhân số nguyên và số FP.
- Bộ cộng số FP để kiểm soát việc cộng, trừ và đảo chỗ số FP.
- Bộ chia số nguyên và số FP.

□ FP sẽ lập khi đến bước EX. Sau khi Kết thúc bước EX, chúng sẽ xử lý đến MEM và WB để hoàn thành quá trình thực thi.

□ Khâu pipeline EX có một số trễ đồng hồ lớn hơn 1.

Khoa KTMT Vũ Đức Lung 42

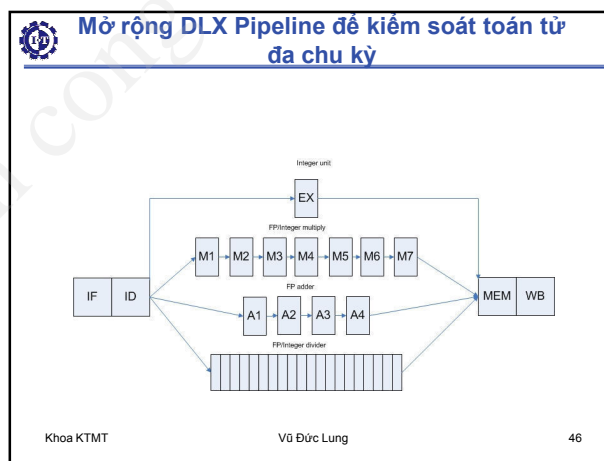


Mở rộng DLX Pipeline để kiểm soát toán tử đa chu kỳ

□ Độ trễ và thời gian khởi tạo của các khối chức năng:

Khối chức năng	Độ trễ	Thời gian khởi tạo
Integer ALU	0	1
Data memory (integer & FP loads)	1	1
FP add	3	1
FP multiply (also integer multiply)	6	1
FP divide (also integer divide)	24	25

Khoa KTMT Vũ Đức Lung 45

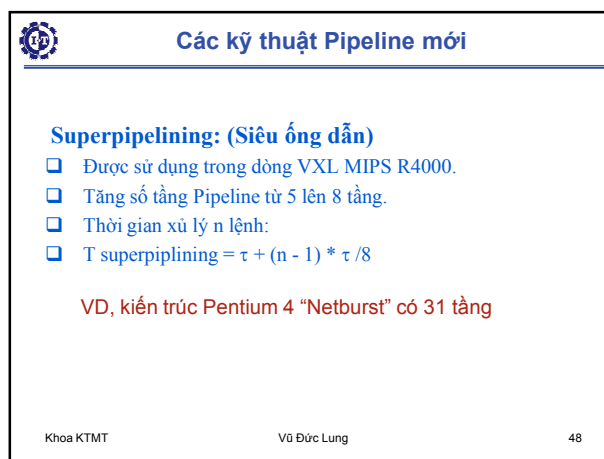


Mở rộng DLX Pipeline để kiểm soát toán tử đa chu kỳ

MULTD	IF	ID	M1	M2	M3	M4	M5	M6	M7	MEM	WB
ADD		IF	ID	A1	A2	A3	A4	MEM	WB		

Bước được in nghiêng chỉ đến nơi dữ liệu được yêu cầu.
Bước được in đậm chỉ đến nơi kết quả tồn tại.

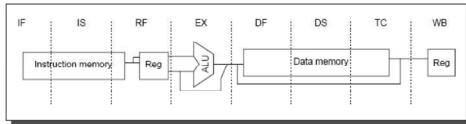
Khoa KTMT Vũ Đức Lung 47





Các kỹ thuật Pipeline mới

- Tốc độ vi xử lý tăng đáng kể so với kỹ thuật Pipeline 5 tầng, nhưng cũng vì số tầng tăng mà khả năng xảy ra xung đột cũng cao hơn
 - IF: Nửa đầu của IF cũ; PC cùng với khởi tạo truy cập instruction cache.
 - IS: Nửa còn lại của IF, hoàn thành truy cập instruction cache.
 - RF: Instruction decode and register fetch, hazard checking, instruction cache hit detection.
 - EX: Execution
 - DF: Data fetch, first half of data cache access.
 - DS: Second half of data fetch, completion of data cache access.
 - TC: Tag check, determine whether the data cache access hit.



Khoa KTMT

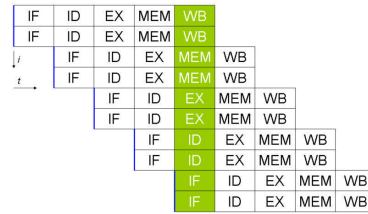
Vũ Đức Lung

49



Vi xử lý siêu tích hợp (Super Scalar Processor)

- Các VXL hiện đại bắt đầu từ dòng Pen IV sử dụng kỹ thuật ống dẫn đôi (Dual Pipelining) để tăng tốc độ lên gấp nhiều lần so với kỹ thuật Pipeline thông thường.



Khoa KTMT

Vũ Đức Lung

50



Vi xử lý siêu tích hợp (Super Scalar Processor)

- Có thể lấy ví dụ đơn giản khi ta tính toán phép tính $14 * 47 + 51 * 22$, nếu là kỹ thuật pipeline thông thường vẫn phải mất 3 bước là tính $51 * 22$, sau đó tính $14 * 27$, rồi cuối cùng cộng 2 kết quả đó lại. Nhưng với kỹ thuật ống dẫn đôi, 2 phép tính $51 * 22$ và $14 * 47$ được thực hiện cùng 1 lúc trên 2 pipeline khác nhau \Rightarrow giảm được 1 công đoạn thực hiện tính toán

Khoa KTMT

Vũ Đức Lung

51



CHƯƠNG 5: KIẾN TRÚC MÁY TÍNH SONG SONG

Nội dung

- ❑ Phân loại kiến trúc máy tính song song
- ❑ Mạng kết nối nội
- ❑ Kiến trúc chia sẻ bộ nhớ
- ❑ Kiến trúc truyền thông báo
- ❑ Mô hình trừu tượng
- ❑ Lập trình song song

Khoa KTMT

Vũ Đức Lung

1



CHƯƠNG 5: KIẾN TRÚC MÁY TÍNH SONG SONG

- ❑ PHÂN LOẠI KIẾN TRÚC MÁY TÍNH THEO FLYNN
 - SISD
 - SIMD
 - MISD
 - MIMD
- ❑ Mạng kết nối các thành phần của máy tính song song
 - Bus Base
 - Switch Base
- ❑ Topology

Khoa KTMT

Vũ Đức Lung

2



GIỚI THIỆU

- ❑ Tại sao phải sử dụng song song?
 - Nhu cầu mô phỏng thế giới thực, xử lý ảnh 3D, dự báo thời tiết,...
 - ⇒ Đòi hỏi tốc độ cao với khối lượng dữ liệu lớn
 - ⇒ Tốc độ của các CPU đã gần đạt tới giới hạn
 - Xử lý song song là quá trình xử lý gồm nhiều tiến trình hoạt động đồng thời cùng một lúc và cùng tham gia giải quyết một vấn đề*
- ❑ Xử lý song song vs xử lý tuần tự
- ❑ Yếu tố dùng xử lý song song:
 - Giá CPU rẻ đi
 - Công nghệ VLSI
 - Tốc độ CPU đạt giới hạn

Khoa KTMT

Vũ Đức Lung

3



PHÂN LOẠI KIẾN TRÚC MÁY TÍNH THEO FLYNN

- ❑ Đơn luồng lệnh – đơn luồng dữ liệu (Single-Instruction Single-Data Streams – SISD)
- ❑ Đơn luồng lệnh – đa luồng dữ liệu (Single-Instruction Multiple-Data Streams – SIMD)
- ❑ Đa luồng lệnh – đơn luồng dữ liệu (Multiple-Instruction Single-Data Streams – MISD)
- ❑ Đa luồng lệnh – đa luồng dữ liệu (Multiple-Instruction Multiple-Data Streams – MIMD)

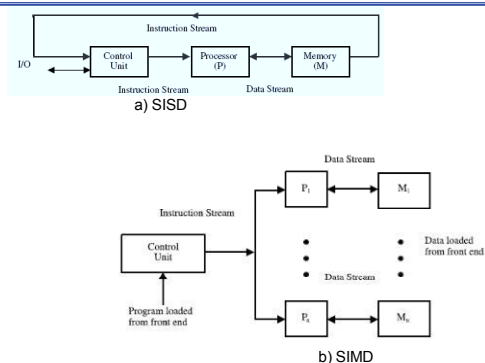
Khoa KTMT

Vũ Đức Lung

4



SISD & SIMD



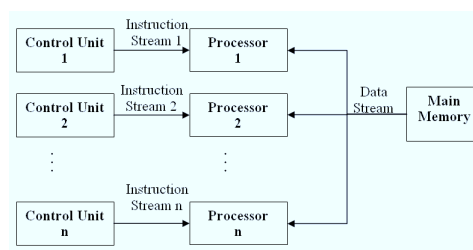
Khoa KTMT

Vũ Đức Lung

5



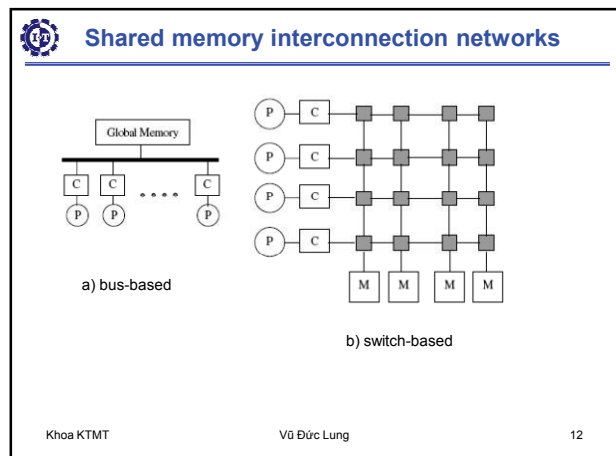
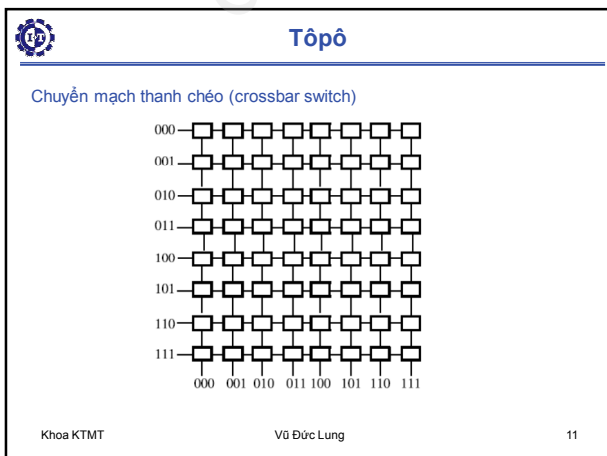
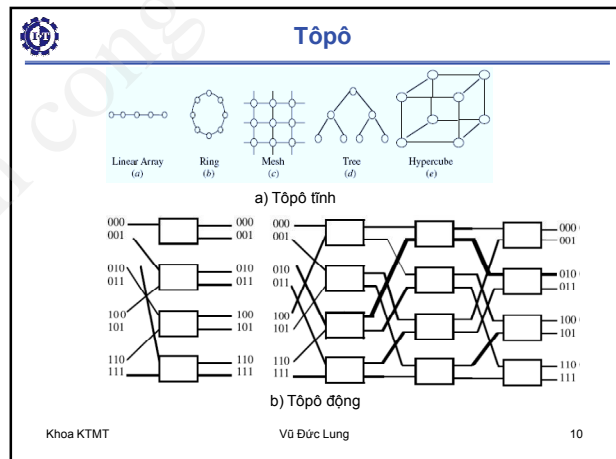
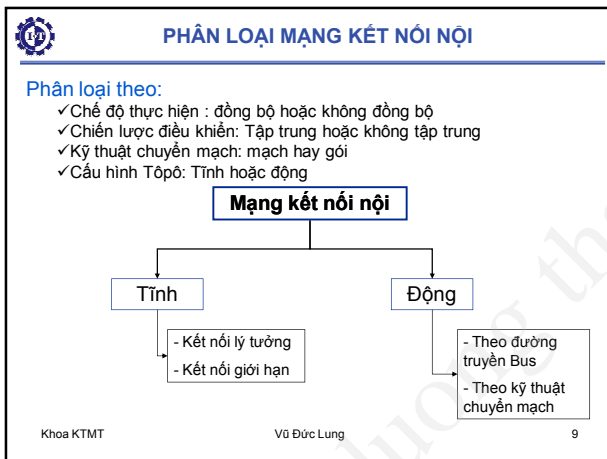
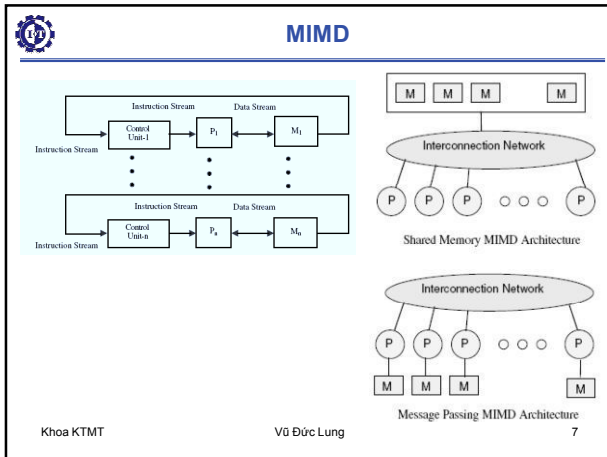
MISD



Khoa KTMT

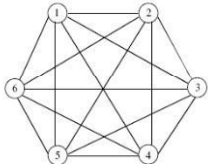
Vũ Đức Lung

6



Mạng kết nối tĩnh (0)

- Các mối liên kết cố định, không thay đổi được
- Mạng kết nối lý tưởng:** Tất cả các node mạng được kết nối với nhau trực tiếp
 - Ưu điểm: nhanh
 - Nhược điểm: quá phức tạp
- VD 1: mạng kết nối lý tưởng với 6 node



Khoa KTMT Vũ Đức Lung 13

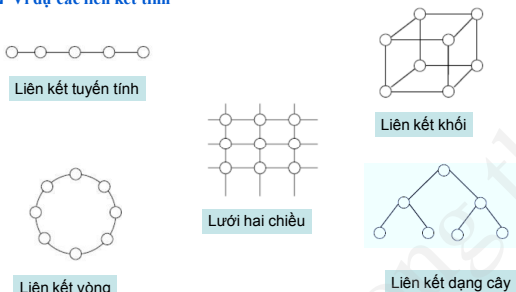
Mạng kết nối tĩnh (1)

- Mạng kết nối giới hạn:** không cho phép nối trực tiếp từ một node bất kỳ đến node bất kỳ khác
 - Liên kết tuyến tính
 - Liên kết vòng và vòng Chordal (CDC Cyberplus multiprocessor 1985, KSR-1 computer system 1992)
 - Liên kết dạng cây và hình sao (DADO multiprocessor at Columbia University 1987)
 - Liên kết lưới hai chiều không quay vòng và quay vòng tròn: được sử dụng rộng rãi nhất như Illiac IV, MPP (Massively Parallel Processor), DAP (Distributed Array Processor), CM-2, Intel Paragon,...
 - Liên kết siêu khối (iPSC, nCUBE)

Khoa KTMT Vũ Đức Lung 14

Mạng kết nối tĩnh (2)

Ví dụ các liên kết tĩnh



- Liên kết tuyến tính
- Liên kết khối
- Liên kết vòng
- Lưới hai chiều
- Liên kết dạng cây

Khoa KTMT Vũ Đức Lung 15

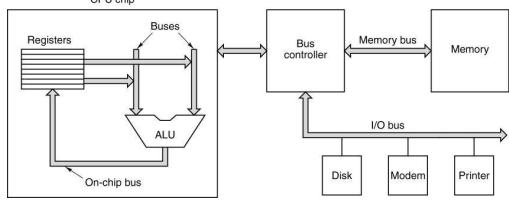
Mạng kết nối động (0)

- Cấu trúc đường truyền Bus**
 - Cấu trúc đường truyền chung dạng đơn (Single Bus Systems): HP 9000 K640, IBM RS/6000 R40, sun Enterprise 6000
 - Cấu trúc đa đường truyền chung (Multiple Bus Systems): Univac 1100/94
- Cấu trúc chuyển mạch**
 - Chuyển mạch một tầng (Single-stage)
 - Chuyển mạch nhiều tầng (Multistage)
 - Liên kết thanh chéo (crossbar network)

Khoa KTMT Vũ Đức Lung 16

Các bus của máy tính (1)

A computer system with multiple buses.



Khoa KTMT Vũ Đức Lung 17

Các bus của máy tính(2)

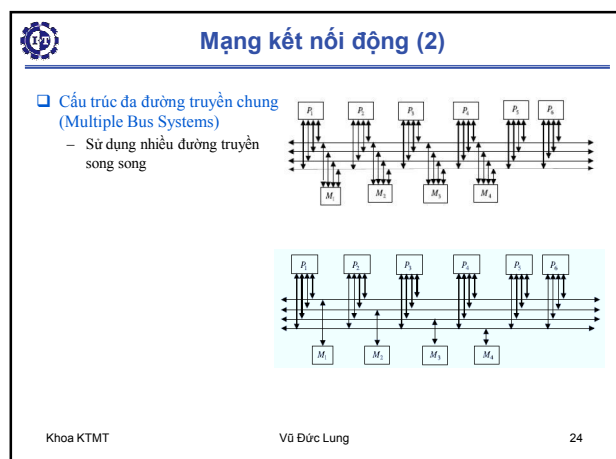
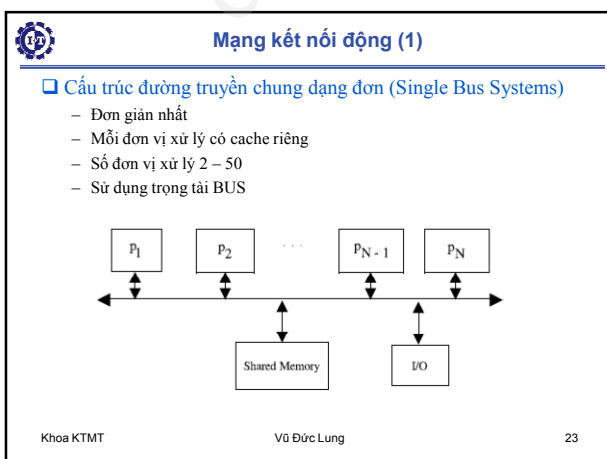
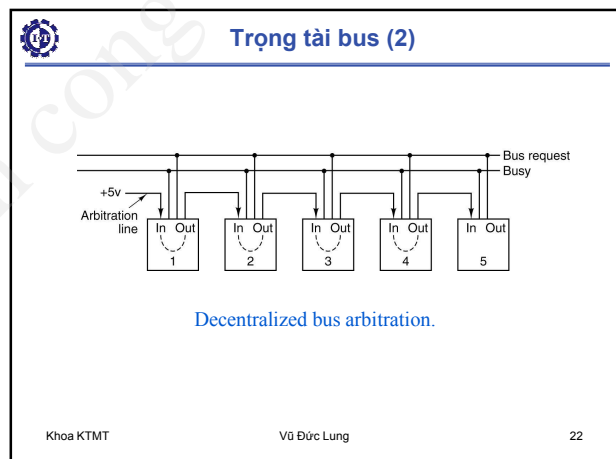
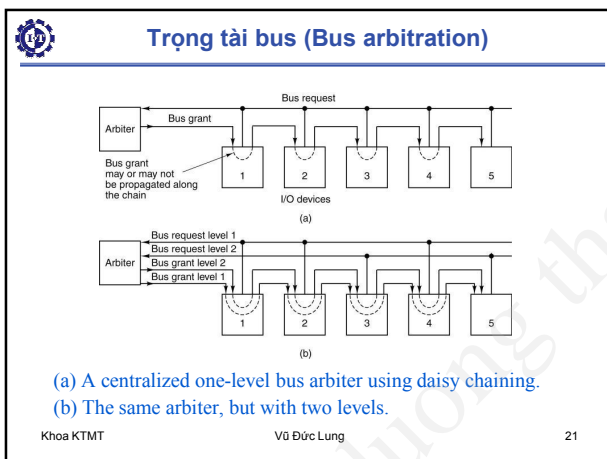
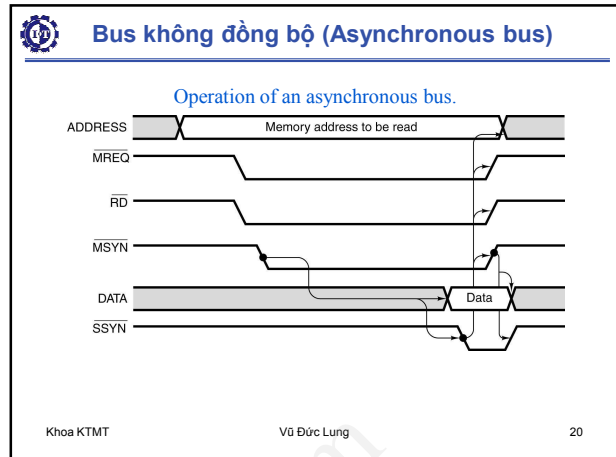
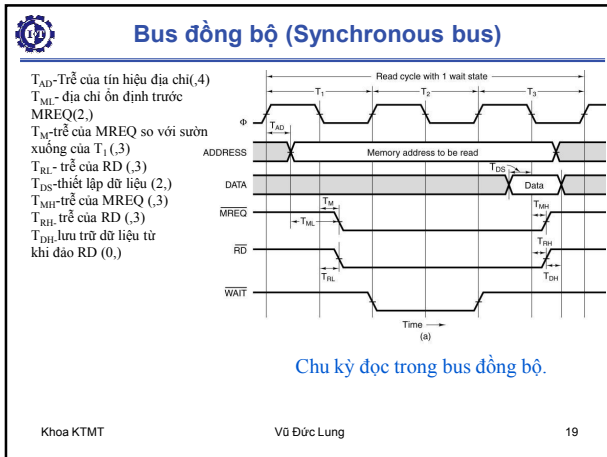
Sự làm việc của các bus:

- chủ bus (bus master): thiết bị tích cực, đòi hỏi truyền thông tin trên bus
- tớ bus (bus slave): thiết bị thụ động, chờ yêu cầu từ các thiết bị khác
- bus driver, bus receiver và transceiver

Master	Slave	Example
CPU	Memory	Fetching instructions and data
CPU	I/O device	Initiating data transfer
CPU	Coprocessor	CPU handing instruction off to coprocessor
I/O	Memory	DMA (Direct Memory Access)
Coprocessor	CPU	Coprocessor fetching operands from CPU

Một số cặp master-slave điển hình.

Khoa KTMT Vũ Đức Lung 18



Mạng kết nối động (3)

Cấu trúc chuyển mạch (switch topology)

a) b)

Khoa KTMT Vũ Đức Lung 25

Mạng kết nối động (4)

Cấu trúc chuyển mạch:

- phần tử chuyển mạch: thông thường 2 x 2, 4 x 4, 8 x 8, n x n

0 0
1 1
Straight Exchange Upper-broadcast Lower-broadcast

Chuyển mạch một tầng (Single-stage): thực hiện kết nối thông qua 2 phép toán Shuffle-Exchange theo công thức:

$$S(p_{m-1}p_{m-2} \dots p_1p_0) = p_{m-2}p_{m-3} \dots p_1p_0p_{m-1}$$

$$E(p_{m-1}p_{m-2} \dots p_1p_0) = p_{m-1}p_{m-2} \dots p_1\bar{p}_0$$

Khoa KTMT Vũ Đức Lung 26

Mạng kết nối động (5)

VD 2: Mạng 1 tầng 8 đầu vào, 8 đầu ra. Thứ tự thực hiện liên kết nguồn 0 đến đích 6.

$S(p_{m-1}p_{m-2} \dots p_1p_0) = p_{m-2}p_{m-3} \dots p_1p_0p_{m-1}$
 $E(p_{m-1}p_{m-2} \dots p_1p_0) = p_{m-1}p_{m-2} \dots p_1\bar{p}_0$

E(000) → 1(001) → S(001) → 2(010) →
 E(010) → 3(011) → S(011) → 6(110)

1 → 7 ?
 S(001) → 010 → E(010) → 011
 → S(011) → 110 → E(110) → 111

Omega network

Khoa KTMT Vũ Đức Lung 27

Mạng kết nối động (6)

Chuyển mạch một tầng (Single-stage) dạng hình khối (The Cube network)

$$C_i(p_{m-1}p_{m-2} \dots p_{i+1}p_i p_{i-1} \dots p_1p_0) = p_{m-1}p_{m-2} \dots p_{i+1}\bar{p}_i p_{i-1} \dots p_1p_0$$

VD 3: Khối 3 bit địa chỉ (N=8), a) C_0 , b) C_1 , c) C_2

$C_0(4) = 5$
 $C_1(7) = 5$
 $C_2(6) = 2$

(a) (b) (c)

Khoa KTMT Vũ Đức Lung 28

Mạng kết nối động (7)

Chuyển mạch nhiều tầng (Multistage):

- Khắc phục nhược điểm của single bus system, cho phép kết nối cùng lúc nhiều cặp P-M
- Số tầng $\log_2 N$ = số bit của địa chỉ nguồn-đích

Các hàm kết nối:

- S-E
- Cube
- Plus-Minus 2: $PM_{2+i}(P) = P + 2^i \bmod N (0 \leq i < k)$
- Butterfly: $B(p_{m-1}p_{m-2} \dots p_1p_0) = p_0p_{m-2} \dots p_1p_{m-1}$

Số component: $N/2 \cdot \log_2 N$


VD: 8x8, có 3 đường kết nối đồng thời

Khoa KTMT Vũ Đức Lung 29

Mạng kết nối động (8)

VD chuyển mạch nhiều tầng: Mạng Omega (TRAC – Texas Reconfigurable Array Computer của ĐH Texas, Cedar của ĐH Illinois, RP3 của IBM, Butterfly của BBN Laboratories và NYU Ultracomputer của NYU New York)

Khoa KTMT Vũ Đức Lung 30

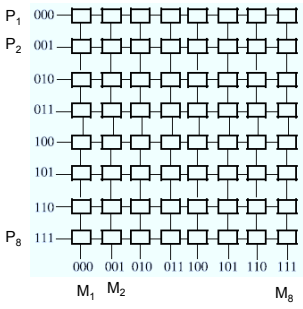


Mạng kết nối động (9)

□ Cấu trúc chuyển mạch- Liên kết thành chéo:

- cung cấp khả năng kết nối nội và băng thông cao nhất
- Tại một thời điểm mỗi cột chỉ có 1 kết nối

Số component:
 N^2

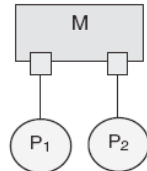


Khoa KTMT
Vũ Đức Lung
31



CHƯƠNG 6: Hệ thống bộ nhớ chia sẻ (share memory architecture)

- Các vi xử lý dùng chung một bộ nhớ thông qua mạng nội kết nối
- Các vấn đề đặt ra:
 - Kiến trúc hệ thống
 - Ràng buộc dữ liệu
 - Đồng bộ giữa các vi xử lý



Shared memory via two ports.

Khoa KTMT

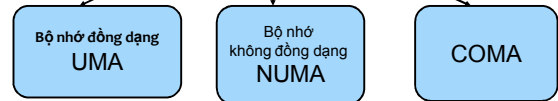
Vũ Đức Lung

1



CHƯƠNG 6: Hệ thống bộ nhớ chia sẻ

Phân loại kiến trúc



Khoa KTMT

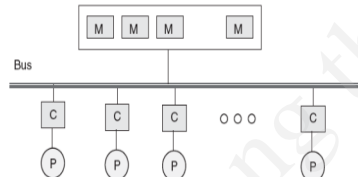
Vũ Đức Lung

2



Truy cập bộ nhớ đồng dạng (UMA)

- Các bộ P truy cập đến M như nhau và như truy cập đến bộ nhớ riêng của chính nó
- Thời gian truy cập bộ nhớ như nhau
- Mạng kết nối nội: đơn bus, đa bus hoặc chuyển mạch
- Có quyền và cơ hội như nhau
- VD: Sun Microsystems multiprocessor servers, Silicon Graphics Inc. multiprocessor servers



Khoa KTMT

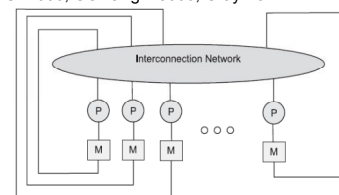
Vũ Đức Lung

3



Truy cập bộ nhớ không đồng dạng (NUMA)

- Mỗi vi xử lý được gắn với một phần của bộ nhớ chia sẻ.
- Mỗi bộ nhớ nhỏ có một không gian địa chỉ riêng.
- Bất kỳ một vi xử lý nào cũng có thể truy cập tức thời đến bộ nhớ nội sử dụng địa chỉ trực tiếp.
- Thời gian truy cập tới module bộ nhớ phụ thuộc vào khoảng cách tới vi xử lý. VD: BBN TC- 2000, SGI Origin 3000, Cray T3E.



Tổ chức bộ nhớ NUMA
Vũ Đức Lung

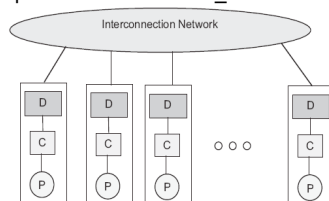
Khoa KTMT

4



Kiến trúc chia sẻ bộ nhớ chỉ cache (COMA)

- COMA có kiến trúc tương tự như NUMA
- Bộ nhớ chia sẻ bao gồm bộ nhớ cache.
- Yêu cầu dữ liệu phải đi tới vi xử lý yêu cầu nó.
- VD: Kendall Square Research's KSR_1



sơ đồ tổ chức kiến trúc COMA

Khoa KTMT

Vũ Đức Lung

5



Bus-based symmetric multiprocessors

- Số P maximum có thể:

$$N \leq \frac{BI}{(1-h)V}$$

- h - cache hit rate
- B - Băng thông của bus (chu kỳ/s)
- I - chu kỳ P (số lệnh/chu kỳ)
- V - Tốc độ của P (số lệnh/s)

- VD: $V=107$, $I=1$, $h=97\%$, $B=106$
 $\Rightarrow N=3$

- Nếu cần $N=30$ thì $h=?$

Khoa KTMT

Vũ Đức Lung

6

Ràng buộc dữ liệu

- Các phương pháp ràng buộc cache cơ bản:
 - Ràng buộc Cache – memory:**
 - Ghi đồng thời (write-through):** Thông tin được ghi đồng thời vào khối của cache và khối của bộ nhớ trong.
 - Ghi lại (write-back):** Thông tin cần ghi chỉ được ghi vào khối trong cache.

Khoa KTMT Vũ Đức Lung 7

Ràng buộc dữ liệu

- Ràng buộc Cache – Cache**
 - Write-Invalidate:** sử dụng dirty bit để thông báo cho các P là vùng nhớ X mất hiệu lực, ghi dữ liệu và xóa bit dirty
 - Write-update:** duy trì tính nhất quán bằng cách ngay lập tức cập nhật các bản copy lên tất cả các cache đang sử dụng vùng nhớ đó

Serial	Event	Write-Update		Write-Invalidate	
		P's Cache	Q's Cache	P's Cache	Q's Cache
1	P reads X	X		X	
2	Q reads X	X	X	X	X
3	Q updates X	X'	X'	INV	X'
4	Q updates X'	X''	X''	INV	X''

Khoa KTMT Vũ Đức Lung 8

Ràng buộc dữ liệu

- Giao thức snooping : là một giao thức cơ bản theo dõi hoạt động của bus và mang theo các lệnh ràng buộc khi cần thiết
- Giao thức Directory _based : là giao thức cơ bản để xác định đường dẫn tới các cache sử dụng một khối nhớ nào đó trong vùng nhớ

Khoa KTMT Vũ Đức Lung 9

Snooping protocol

- Ràng buộc dữ liệu trong kiến trúc chia sẻ bộ nhớ:**
 - Write-invalidate and write-through**
 - Write-invalidate and write-back**
 - Write-Once**
 - Write-update and write-through**
 - Write-update and write-back**

Khoa KTMT Vũ Đức Lung 10

Snooping protocol

Khoa KTMT Vũ Đức Lung 11

1. Ghi mất hiệu lực và ghi đồng thời (Write-invalidate and write-through)

Bộ nhớ luôn luôn phù hợp với bản copy cập nhật gần nhất

State	Description
Valid [VALID]	The copy is consistent with global memory.
Invalid [INV]	The copy is inconsistent.

Event	Actions
Read-Hit	Use the local copy from the cache.
Read-Miss	Fetch a copy from global memory. Set the state of this copy to Valid.
Write-Hit	Perform the write locally. Broadcast an Invalid command to all caches. Update the global memory.
Write-Miss	Get a copy from global memory. Broadcast an invalid command to all caches. Update the global memory. Update the local copy and set its state to Valid.
Block replacement	Since memory is always consistent, no write-back is needed when a block is replaced.

Khoa KTMT Vũ Đức Lung 12



1. Ghi mất hiệu lực và ghi đồng thời (Write-invalidate and write-through)

Giả sử vùng nhớ X trong bộ nhớ được đặt giá trị ban đầu là 5, và hoạt động trên X được thực thi theo các bước sau: (1) P reads X; (2) Q reads X; (3) Q updates X; (4) Q reads X; (5) Q updates X; (6) P updates X; (7) Q reads X.

Serial	Event	Memory Location X	P's Cache		Q's Cache	
			Location X	State	Location X	State
0	Original value	5				
1	P reads X (Read-Miss)	5	5	VALID		
2	Q reads X (Read-Miss)	5	5	VALID	5	VALID
3	Q updates X (Write-Hit)	10	5	INV	10	VALID
4	Q reads X (Read-Hit)	10	5	INV	10	VALID
5	Q updates X (Write-Hit)	15	5	INV	15	VALID
6	P updates X (Write-Miss)	20	20	VALID	15	INV
7	Q reads X (Read-Miss)	20	20	VALID	20	VALID

13



2. Ghi mất hiệu lực và ghi lại (Write-invalidate and write-back)

TABLE 4.5 Write-Invalidate Write-Back Protocol

State	Description
Shared (Read-Only) [RO]	Data is valid and can be read safely. Multiple copies can be in this state.
Exclusive (Read-Write) [RW]	Only one valid cache copy exists and can be read from and written to safely. Copies in other caches are invalid. The copy is inconsistent.
Invalid [INV]	
Event	Action
Read-Hit	Use the local copy from the cache.
Read-Miss	If no Exclusive (Read-Write) copy exists, then supply a copy from global memory. Set the state of this copy to Shared (Read-Only). If an Exclusive (Read-Write) copy exists, make a copy from the cache that set the state to Exclusive (Read-Write), update global memory and local cache with the copy. Set the state to Shared (Read-Only) in both caches.
Write-Hit	If the copy is Exclusive (Read-Write), perform the write locally. If the state is Shared (Read-Only), then broadcast an Invalid to all caches. Set the state to Exclusive (Read-Write).
Write-Miss	Get a copy from either a cache with an Exclusive (Read-Write) copy, or from global memory itself. Broadcast an Invalid command to all caches. Update the local copy and set its state to Exclusive (Read-Write).
Block replacement	If a copy is in an Exclusive (Read-Write) state, it has to be written back to main memory if the block is being replaced. If the copy is in Invalid or Shared (Read-Only) states, no write-back is needed when a block is replaced.

Khoa KTMT

14



2. Ghi mất hiệu lực và ghi lại (Write-invalidate and write-back)

Bộ nhớ chỉ cập nhật trong trường hợp Read-Miss và có bản copy dạng RW

TABLE 4.6 Example 3 (Write-Invalidate Write-Back)

Serial	Event	Memory Location X	P's Cache		Q's Cache	
			Location X	State	Location X	State
0	Original value	5				
1	P reads X (Read-Miss)	5	5	RO		
2	Q reads X (Read-Miss)	5	5	RO	5	RO
3	Q updates X (Write-Hit)	5	5	INV	10	RW
4	Q reads X (Read-Hit)	5	5	INV	10	RW
5	Q updates X (Write-Hit)	5	5	INV	15	RW
6	P updates X (Write-Miss)	5	20	RW	15	INV
7	Q reads X (Read-Miss)	20	20	RO	20	RO

Khoa KTMT

Vũ Đức Lung

15



3. Ghi một lần (Write-Once)

TABLE 4.7 Write-Once Protocol

State	Description
Invalid [INV]	The copy is inconsistent.
Valid [VALID]	The copy is consistent with global memory.
Reserved [RES]	Data have been written exactly once and the copy is consistent with global memory. There is only one copy of the global memory block in one local cache.
Dirty [DIRTY]	Data have been updated more than once and there is only one copy in one local cache. When a copy is dirty, it must be written back to global memory.
Event	Actions
Read-Hit	Use the local copy from the cache.
Read-Miss	If no Dirty copy exists, then supply a copy from global memory. Set the state of this copy to Valid. If a dirty copy exists, make a copy from the cache that set the state to Dirty, update global memory and local cache with the copy. Set the state to VALID in both caches.
Write-Hit	If the copy is Dirty or Reserved, perform the write locally, and set the state to Dirty. If the state is Valid, then broadcast an Invalid command to all caches. Update the global memory and set the state to Reserved.
Write-Miss	Get a copy from either a cache with a Dirty copy or from global memory itself. Broadcast an Invalid command to all caches. Update the local copy and set its state to Dirty.
Block replacement	If a copy is in a Dirty state, it has to be written back to main memory if the block is being replaced. If the copy is in Valid, Reserved, or Invalid states, no write-back is needed when a block is replaced.

Khoa KTMT

16



3. Ghi một lần (Write-Once)

RES báo dữ liệu được copy chính xác 1 lần, DIRTY báo dữ liệu được cập nhật nhiều lần

TABLE 4.8 Example 4 (Write-Once Protocol)

Serial	Event	Memory Location X	P's Cache		Q's Cache	
			Location X	State	Location X	State
0	Original value	5				
1	P reads X (Read-Miss)	5	5	VALID		
2	Q reads X (Read-Miss)	5	5	VALID	5	VALID
3	Q updates X (Write-Hit)	10	5	INV	10	RES
4	Q reads X (Read-Hit)	10	5	INV	10	RES
5	Q updates X (Write-Hit)	10	5	INV	15	DIRTY
6	P updates X (Write-Miss)	10	20	DIRTY	15	INV
7	Q reads X (Read-Miss)	20	20	VALID	20	VALID

Khoa KTMT

Vũ Đức Lung

17



4. Ghi cập nhật và ghi đồng thời cục bộ (Write-update and Partial write-through)

TABLE 4.9 Write-Update Partial Write-Through Protocol

State	Description
Valid Exclusive [VAL-X]	This is the only cache copy and is consistent with global memory.
Shared [SHARE]	There are multiple cache copies shared. All copies are consistent with memory.
Dirty [DIRTY]	This copy is not shared by other caches and has been updated. It is not consistent with global memory. (Copy ownership.)
Event	Action
Read-Hit	Use the local copy from the cache. State does not change.
Read-Miss	If no other cache copy exists, then supply a copy from global memory. Set the state of this copy to Valid Exclusive. If a cache copy exists, make a copy from the cache. Set the state to Shared in both caches. If the cache copy was in a Dirty state, the value must also be written to memory.
Write-Hit	Perform the write locally and set the state to Dirty. If the state is Shared, then broadcast data to memory and to all caches and set the state to Shared. If other caches no longer share the block, the state changes from Shared to Valid Exclusive.
Write-Miss	The block copy comes from either another cache or from global memory. If the block comes from another cache, perform the update and update all other caches that share the block and global memory. Set the state to Shared. If the copy comes from memory, perform the write and set the state to Dirty.
Block replacement	If a copy is in a Dirty state, it has to be written back to main memory if the block is being replaced. If the copy is in Valid Exclusive or Shared states, no write-back is needed when a block is replaced.

Khoa KTMT

18



4. Ghi cập nhật và ghi đồng thời cục bộ (Write-update and Partial write-through)

TABLE 4.10 Example 5 (Write-Update Partial Write-Through)

Serial	Event	Memory Location X	P's Cache		Q's Cache	
			Location X	State	Location X	State
0	Original value	5				
1	P reads X (Read-Miss)	5	5	VAL-X		
2	P updates X (Write-Hit)	5	10	DIRTY		
3	Q reads X (Read-Miss)	10	10	SHARE	10	SHARE
4	Q updates X (Write-Hit)	15	15	SHARE	15	SHARE
5	Q reads X (Read-Hit)	15	15	SHARE	15	SHARE
6	Block X is replaced in P's cache (Replace)	15	—	—	15	VAL-X
7	Q updates X (Write-Hit)	15	—	—	20	DIRTY
8	P updates X (Write-Miss)	25	25	SHARE	25	SHARE

Khoa KTMT

Vũ Đức Lung

19



5. Ghi cập nhật và ghi lại (Write-update and write-back)

TABLE 4.11 Write-Update Write-Back Protocol

State	Description
Valid Exclusive (VAL-X)	This is the only cache copy and is consistent with global memory.
Shared Clean (SH-CLN)	There are multiple cache copies shared.
Shared Dirty (SH-DRT)	There are multiple shared cache copies. This is the last one being updated. (Ownership.)
Dirty (DIRTY)	This copy is not shared by other caches and has been updated. It is not consistent with global memory. (Ownership.)
Event	Action
Read-Hit	Use the local copy from the cache. State does not change.
Read-Miss	If no other cache copy exists, then supply a copy from global memory. Set the state of this copy to Valid Exclusive. If a cache copy exists, make a copy from the cache. Set the state to Shared Clean. If the supplying cache copy was in a Valid Exclusive or Shared Clean, its new state becomes Shared Clean. If the supplying cache copy was in a Dirty or Shared Dirty state, its new state becomes Shared Dirty.
Write-Hit	If the state was Valid Exclusive or Dirty, perform the write locally and set the state to Dirty. If the state is Shared Clean or Shared Dirty, perform update and change state to Shared Dirty. Broadcast the updated block to all other caches. These caches snoop the bus and update their copies and set their state to Shared Clean.
Write-Miss	The block copy comes from either another cache or from global memory. If the block comes from another cache, perform the update, set the state to Shared Dirty, and broadcast the updated block to all other caches. Other caches snoop the bus, update their copies, and change their state to Shared Clean. If the copy comes from memory, perform the write and set the state to Dirty. If a copy is in a Dirty or Shared Dirty state, it has to be written back to main memory if the block is being replaced. If the copy is in Valid Exclusive, no write back is needed when a block is replaced.
Block replacement	

Khoa KTMT

20



5. Ghi cập nhật và ghi lại (Write-update and write-back)

TABLE 4.12 Example 6 (Write-Update Write-Back)

Serial	Event	Memory Location X	P's Cache		Q's Cache	
			Location X	State	Location X	State
0	Original value	5				
1	P reads X (Read-Miss)	5	5	VAL-X		
2	P updates X (Write-Hit)	5	10	DIRTY		
3	Q reads X (Read-Miss)	5	10	SH-DRT	10	SH-CLN
4	Q updates X (Write-Hit)	5	15	SH-CLN	15	SH-DRT
5	Q reads X (Read-Hit)	5	15	SH-CLN	15	SH-DRT
6	Block X is replaced in Q's cache (Replace)	15	15	VAL-X	—	—
7	P updates X (Write-Hit)	15	20	DIRTY	—	—
8	Q updates X (Write-Miss)	15	25	SH-CLN	25	SH-DRT

Khoa KTMT

Vũ Đức Lung

21



Directory- based protocol

- ☐ Giao thức cơ bản để xác định đường dẫn tới các cache sử dụng một khối nhớ nào đó trong vùng nhớ cơ bản
- ☐ Xác định chính xác cache nào sẽ được cập nhật khi mà một cache sử dụng vùng nhớ đó thực hiện tác vụ ghi
- ☐ Giao thức hỗ trợ cho giao thức snooping.
- ☐ Có hai cơ chế là tập trung và phân tán.
- ☐ Trong cơ chế tập trung:
 - một vùng nhớ chung làm nhiệm vụ lưu tất cả các đường dẫn
 - P cần truy cập vào vùng nhớ chung này khi muốn cập nhật=> thất cổ chai.
- ☐ Cơ chế phân tán: khối nhớ chỉ chứa một con trỏ tới một cache sử dụng nó, trong cache sử dụng khối nhớ đó lại có một con trỏ tới một cache cùng sử dụng khối nhớ đó, ...

Khoa KTMT

Vũ Đức Lung

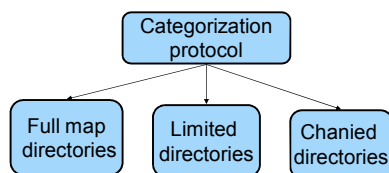
22



Directory- based protocol

Categorization protocol :

Mỗi khối nhớ sẽ có một vùng để lưu các con trỏ của các cache sử dụng nó, và nó sẽ giữ các dirty bit cho các cache mà nó chứa con trỏ tới khi nó được một cache nào đó write lên



Khoa KTMT

Vũ Đức Lung

23



Full map directories

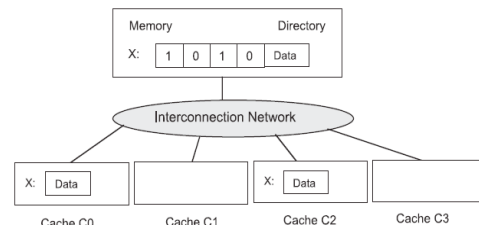
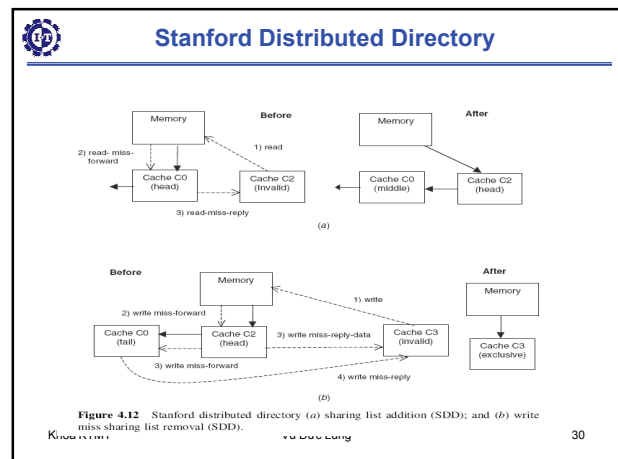
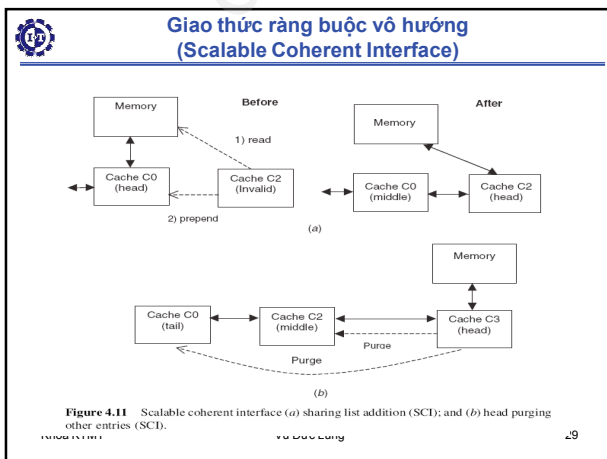
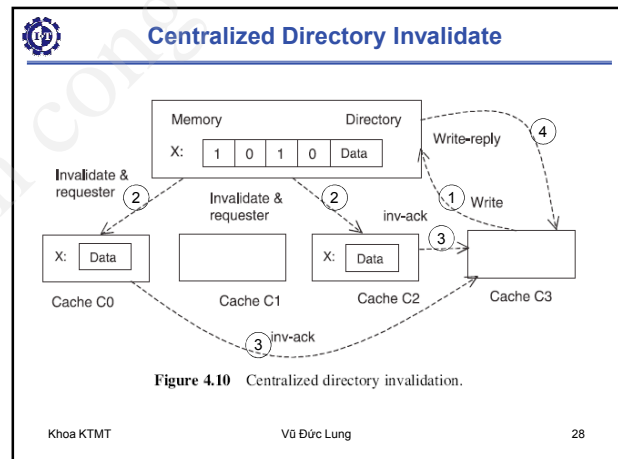
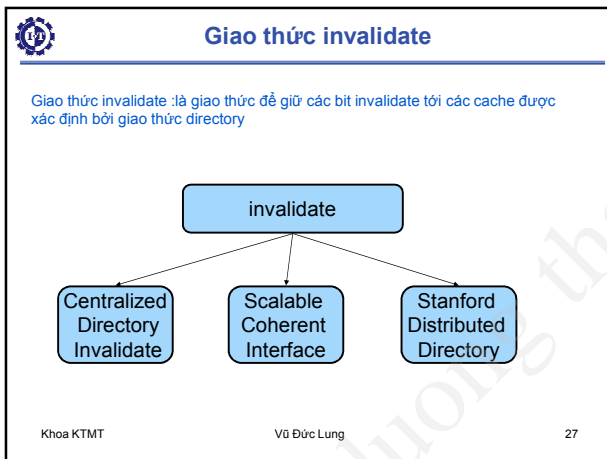
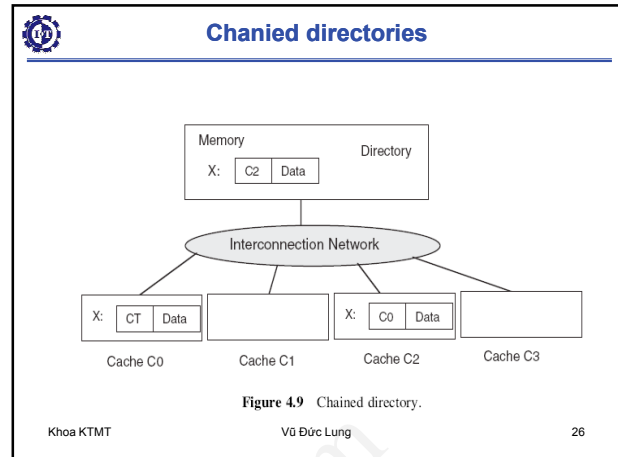
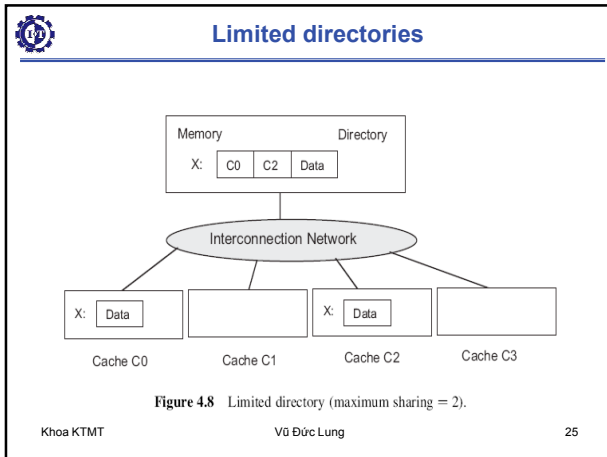


Figure 4.7 Fully mapped directory.

Khoa KTMT

Vũ Đức Lung

24





ĐỌC THÊM

- ☐ Đồng bộ giữa các vi xử lý trong KT bộ nhớ chia sẻ
- ☐ Kiến trúc truyền thông điệp (Message passing architecture)

cuu duong than cong . com

CHƯƠNG 07: MESSAGE PASSING

NỘI DUNG

- Giới thiệu
- Kỹ thuật định tuyến
- Các cơ chế chuyển mạch
- Mô hình lập trình
- Xử lý trong MPA

Khoa KTMT Vũ Đức Lung 1

GIỚI THIỆU

□ Kiến trúc Message Passing là gì? Nó hoạt động như thế nào?

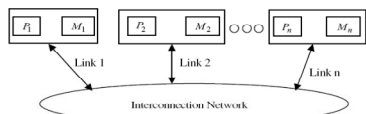


Figure 5.1 Message passing systems.

Khoa KTMT Vũ Đức Lung 2

GIỚI THIỆU

□ Quá trình thực thi các chương trình ứng dụng trong hệ thống Message Passing như thế nào?

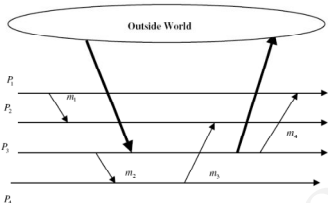


Figure 5.2 An example of a message passing system.

Khoa KTMT Vũ Đức Lung 3

Kỹ thuật định tuyến trong mạng truyền thông điệp (Routing in message passing networks)

□ Routing là kỹ thuật dùng để lựa chọn đường dẫn chính xác cho các tin nhắn trong mạng. Nó bao gồm các tập đường dẫn khả thi có thể được thông điệp sử dụng để đến đích trong quá trình truyền tải và một chức năng chọn ra một trong số các đường dẫn đó.

□ Phân loại kỹ thuật routing theo cách mà nó chọn đường dẫn:

- Adaptive
- Deterministic.

□ Ngoài ra kỹ thuật routing cũng có thể được phân biệt theo cách thức đưa ra quyết định là tập trung hay phân tán.

Khoa KTMT Vũ Đức Lung 4

Kỹ thuật định tuyến

□ Ví dụ điển hình về Deterministic :

- Xem $S = S_5 \dots S_0$ là địa chỉ của node nguồn (6 bit) và $D = D_5 \dots D_0$ là địa chỉ node đích.
- Lấy $R = S \text{ XOR } D$, kết quả thu được sẽ quyết định chiều mà thông điệp được chuyển để đến đích.
- Giả sử $S = 10(001010)$ và $D = 39(100111) \Rightarrow R = (101101)$.
- \Rightarrow Thông điệp được chuyển theo chiều 0,2,3,5. Thứ tự của các chiều này không quan trọng. Giả sử thông điệp truyền theo chiều 5,3,2,0.
- \Rightarrow kết quả:

$10(001010) \rightarrow 42(101010) \rightarrow 34(100010) \rightarrow 38(100110) \rightarrow 39(100111)$

$$C_i(p_{m-1}p_{m-2} \dots p_{i+1}p_i p_{i-1} \dots p_1 p_0) = p_{m-1}p_{m-2} \dots p_{i+1} \bar{p}_i p_{i-1} \dots p_1 p_0$$

Khoa KTMT Vũ Đức Lung 5

Kỹ thuật định tuyến

□ Có 2 loại thao tác liên lạc trong hệ thống message passing là one-to-one(point-to-point hay unicast) và collective

□ Routing cho Broadcasting và Multicasting:

- Broadcast và Multicasting là gì?
- Chúng được dùng trong trường hợp nào?
- Các vấn đề cần quan tâm?

Khoa KTMT Vũ Đức Lung 6

Kỹ thuật định tuyến

Figure 5.3 Hypercube broadcast tree-based communication.

Khoa KTMT Vũ Đức Lung 7

Kỹ thuật định tuyến

Routing Potential Problems (Các vấn đề tiềm tàng). Trong vấn đề định tuyến chúng ta sẽ gặp phải các trường hợp sau:

1. Deadlock
2. Livelock
3. Starvation

Khoa KTMT Vũ Đức Lung 8

Kỹ thuật định tuyến

Deadlock: xuất hiện mỗi khi tồn tại một vòng phụ thuộc các tài nguyên (đội vô hạn).

Các phương pháp có thể áp dụng để tránh deadlock là gì?

Khoa KTMT Vũ Đức Lung 9

Kỹ thuật định tuyến

Figure 5.4 A 4-node network and its CDGs (a) a 4-node network; (b) channel dependency graph; and (c) CDG for a deadlock-free version of the network.

Khoa KTMT Vũ Đức Lung 10

Kỹ thuật định tuyến

Livelock: livelock diễn tả 1 trường hợp mà thông điệp di chuyển quanh mạng mà không bao giờ đến được điểm đích.

Hiện tượng xảy ra khi sử dụng thuật toán thích nghi (adaptive) khi mà chúng được reroute với mong muốn tìm được đích đến.

Khi một node muốn giao tiếp chúng sẽ phóng thông điệp ra ngoài mạng. Có 2 loại mô hình đưa thông điệp ra mạng cơ bản là:

- static injection
- dynamic injection.

Khoa KTMT Vũ Đức Lung 11

Kỹ thuật định tuyến

Starvation: Một node được xem là Starvation khi mà nó có thông điệp cần chuyển ra mạng nhưng không thể được phép làm chuyển đó.

Các phương pháp để tránh starvation

Đơn giản nhất là cho phép mỗi node có một hàng đợi được chuyển dữ liệu ra ngoài mạng. Nhược điểm của phương pháp này là về tốc độ, nó sẽ làm chậm toàn bộ các node trong mạng.

Khoa KTMT Vũ Đức Lung 12



CÁC CƠ CHẾ CHUYỂN MẠCH

SWITCHING MECHANISMS IN MESSAGE PASSING

Khoa KTMT

Vũ Đức Lung

13



Các cơ chế chuyển mạch

□ Switching mechanism là cơ chế dùng để chuyển dữ liệu từ kênh input sang kênh output. Độ trễ bắt buộc của mạng tùy thuộc vào cơ chế switching được sử dụng.

□ Các loại thường được sử dụng:

- store-and-forward,
- circuit-switching,
- virtual-cut-through,
- wormhole.

Khoa KTMT

Vũ Đức Lung

14



Các cơ chế chuyển mạch

□ Cơ chế circuit-switching:

- Xác định đường dẫn giữa đích và nguồn mà không cần có bộ đệm giữa các node.
- Kỹ thuật này phù hợp với các thông điệp dài và tránh được deadlock.
- Giảm thiểu tối đa độ trễ bắt buộc của mạng, do tổng phí delay chỉ xuất hiện khi cài đặt đường dẫn và còn lại rất ít hầu như không có độ trễ nào khác.
- Tiêu tốn rất nhiều băng thông của mạng khi mà tất cả các node trung gian phải thiết lập đường dẫn riêng cho quá trình truyền tải thông điệp đó.

Khoa KTMT

Vũ Đức Lung

15



Các cơ chế chuyển mạch

□ Cơ chế store-and-forward:

- Cấp phát một băng thông động cho thông điệp khi nó được chuyển trong mạng, điều này tránh được các hạn chế trong cơ chế circuit-switching.
- Có 2 loại chính trong cơ chế này:
 - packet-switched
 - virtual cut-through.

Khoa KTMT

Vũ Đức Lung

16



Các cơ chế chuyển mạch

□ Cơ chế wormhole:

- Để giảm kích thước của bộ đệm và giảm độ trễ
- Các packet được chia nhỏ thành các flits (flow control bits). Các flits này di chuyển trong một mô hình ống dẫn, đầu là header flit.
- Khi mà header flit bị block do mạng tắc thì các flit còn lại cũng bị block.
- Cần trang bị thêm 1 bộ đệm có thể lưu flit
- Độ trễ tùy thuộc vào chiều dài của thông điệp và nó cần ít chỗ lưu trữ hơn tại các node so với 2 kỹ thuật store-and-forward trên.

Khoa KTMT

Vũ Đức Lung

17



Các cơ chế chuyển mạch

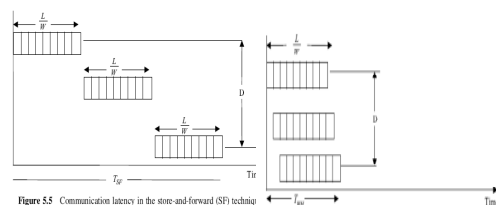


Figure 5.6 Communication latency in the wormhole (WH) technique.

Khoa KTMT

Vũ Đức Lung

18

So sánh các cơ chế chuyển mạch

Switching Mechanism	Advantage	Disadvantage
Circuit switching	<ol style="list-style-type: none"> 1. Thích hợp cho thông điệp dài 2. Tránh được deadlock 	<ol style="list-style-type: none"> 1. Lãng phí băng thông
Store and forward	<ol style="list-style-type: none"> 1. Đơn giản 2. Thích hợp khi gặp các vấn đề về lưu thông mạng 3. Băng thông động 	<ol style="list-style-type: none"> 1. Đòi hỏi bộ đệm 2. Có thể gây ra độ trễ lớn 3. Có thể gặp deadlock
Virtual cut-through	<ol style="list-style-type: none"> 1. Tốt cho các thông điệp dài 2. Có thể tránh deadlock 3. Loại bỏ phương thức data-link 	<ol style="list-style-type: none"> 1. Cần có nhiều bộ đệm cho thông điệp 2. Lãng phí băng thông 3. Chịu yếu đáng với các thuật toán routing tốt
Wormhole	<ol style="list-style-type: none"> 1. Tốt cho các thông điệp dài 2. Giảm kích thước bộ đệm 3. Giảm ảnh hưởng chiều dài đường dẫn 	<ol style="list-style-type: none"> 1. Có thể bị deadlock 2. Không có khả năng "quay lui"

Khoa KTMT

Các cơ chế chuyển mạch

□ Thuật toán routing cho kỹ thuật wormhole: Dimension-Ordered routing

Figure 5.7 Dimension-ordered (X-Y) routing in an 8×8 mesh network.

Khoa KTMT Vũ Đức Lung 20

Các cơ chế chuyển mạch

□ Virtual Channels:

Figure 5.8 Path multiplexing through the same link.

Khoa KTMT Vũ Đức Lung 21

MÔ HÌNH LẬP TRÌNH TRONG KIẾN TRÚC TRUYỀN THÔNG ĐIỆN

MESSAGE PASSING PROGRAMMING MODELS

Khoa KTMT Vũ Đức Lung 22

Mô hình lập trình

□ Kiến trúc message passing sử dụng các thành phần cơ bản để các vi xử lý tại các node khác nhau liên lạc được với nhau, đó là các lệnh *send, receive, broadcast*.

□ Lệnh *send* lấy một vùng nhớ đệm lưu thông điệp vào đó và chuyển cho node đích.

□ Lệnh *receive* chấp nhận thông điệp từ node nguồn và lưu nó vào một vùng nhớ đệm xác định.


Khoa KTMT Vũ Đức Lung 23

Mô hình lập trình

□ Có 2 cơ chế là block và không bị block.

Figure 5.9 Blocking send/receive handshaking protocol.


Khoa KTMT Vũ Đức Lung 24



CÁC VI XỬ LÝ TRONG KIẾN TRÚC TRUYỀN THÔNG ĐIỆN

PROCESSOR SUPPORT FOR MESSAGE PASSING

Khoa KTMTVũ Đức Lung25




Vi xử lý trong MPA

☐ Các vi xử lý hỗ trợ cho cơ chế truyền thông điệp đòi hỏi cần phải có những lệnh đặc biệt hỗ trợ cho việc truyền thông giữa các vi xử lý, và chúng cần có những đặc tính như sau:

1. Một port là một kênh truyền thông giao tiếp, là đối tượng tham chiếu đến cho các tác vụ và luồng. Các thao tác chính có thể thực hiện ở port là *send* và *receive*
2. Thông điệp được sử dụng như là đối tượng. Thông điệp được chia ra thành 2 phần header và body. Thông điệp giữ các thông tin cần trao đổi giữa các tiến trình
3. Các tập port (Port sets).

☐ Cơ chế chống xung đột của các port

Khoa KTMTVũ Đức Lung26




Vi xử lý trong MPA

☐ Máy iPAX 432 của Intel sử dụng giao tiếp chuyển thông điệp và nó hoàn toàn hỗ trợ kiến trúc này. Nó có thể sử dụng các đối tượng port như là đường dẫn cho các thông điệp. Vi xử lý chứa một hàng đợi thông điệp. Một giao tiếp thông điệp có thể sắp xếp để giao tiếp dựa theo các tiêu chí sau đây:

1. Thời điểm đến (như là cấu trúc FIFO chẳng hạn)
2. Độ ưu tiên
3. Thời hạn trong vòng ưu tiên

Khoa KTMTVũ Đức Lung27



CÂU HỎI & BÀI TẬP

Khoa KTMTVũ Đức Lung28