Discussion Session 10

CS/ECE 552

Ramkumar Ravi

16 Apr 2012

Mark your calendar

IMPORTANT DATES

- **04/23** – Project Demo 2 (Submission instructions would be similar to demo1. Will be posted on course website and e-mailed)
- **05/02** – HW6 due
- **05/11** – Project Demo 3
- **05/13** – Final exam

TODAY

- HW6 discussion (Problems 1, 2 and 4. Spend some time on Problem 3)
- Demo2 requirements in brief

Problem 1 - ECC

- This problem consists of two parts:
  (1) Given the dataword, find the codeword using the code-table
  (2) Assuming that there can be a 1 bit error, find the dataword for a given code-word

- Please go through the ECC handout posted in the course webpage (Not discussing the SECDED theory today – Important for exam)
  http://pages.cs.wisc.edu/~david/courses/cs552/S12/handouts/ecc1.pdf

- Important results
  - Hamming distance (HD) – Number of bits that are different between two words
  - A code is capable of ***t*** error detection, iff minimum HD of the code is ***t+1***
  - A code is capable of ***t*** error correction, iff minimum HD of the code is ***2t+1***
  - A code is capable of correcting ***t*** errors and detecting ***d*** errors (d>=t), iff minimum HD of the code is at least ***t+d+1***

Problem 1 – ECC (contd..)

	7	6	5	4	3	2	1	0
Check bit 1	0	1	0	1	1	0	1	1
Check bit 2	0	1	1	0	1	1	0	1
Check bit 3	1	0	0	0	1	1	1	0
Check bit 4	1	1	1	1	0	0	0	0

- We will discuss the procedure for obtaining the code-word for a data-word using the table above

STEP 1: Obtain check bits from table (See location of 1's)

c[0] = d[6] ^ d[4] ^ d[3] ^ d[1] ^ d[0]
c[1] = d[6] ^ d[5] ^ d[3] ^ d[2] ^ d[0]
c[2] = d[7] ^ d[3] ^ d[2] ^ d[1]
c[3] = d[7] ^ d[6] ^ d[5] ^ d[4]

STEP 2: Take a dataword (Showing example for 10101001)

c[0] = 0 ^ 0 ^ 1 ^ 0 ^ 1 = 0
c[1] = 0 ^ 1 ^ 1 ^ 0 ^ 1 = 1
c[2] = 1 ^ 1 ^ 0 ^ 0 = 0
c[3] = 1 ^ 0 ^ 1 ^ 0 = 0

STEP 3: Form the codeword by concatenating with data word -> Result: **0010 10101001**

Problem 1 – ECC (contd..)

	7	6	5	4	3	2	1	0
Check bit 1	0	1	0	1	1	0	1	1
Check bit 2	0	1	1	0	1	1	0	1
Check bit 3	1	0	0	0	1	1	1	0
Check bit 4	1	1	1	1	0	0	0	0

- We will discuss the procedure for obtaining the data-word from a code-word using the table above and given the fact that there can be a 1 bit error

STEP 1: Obtain the check bits from the given codeword
1011 10000111 -> Check bits = 1011

STEP 2: Using the table above, calculate the check bits for the dataword 10000111
c[0] = 0 ^ 0 ^ 0 ^ 1 ^ 1 = 0
c[1] = 0 ^ 0 ^ 0 ^ 1 ^ 1 = 0
c[2] = 1 ^ 0 ^ 1 ^ 1 = 1
c[3] = 1 ^ 0 ^ 0 ^ 0 = 1

STEP 3: XOR given check bits and actual check bits -> Result = 1011 ^ 1100 = 0111 (This is called syndrome. Non-zero syndrome indicates error. 0111 from the table above indicates there is an error in name3)

Problem 2 – Booth's algorithm

- For this problem, understand Booth's encoding (see the table below)

i	i - 1	Recoded bit
0	0	0
0	1	1
1	0	-1
1	1	0

- Example: Show 1-bit booth recoding for the multiplier $(-47)_{10}$

  STEP 1: Convert to binary -> 1101 0001

  STEP 2: Group in pairs starting from LSB (Assume to the right of LSB there is a 0) -> **Answer is (0 -1 1 -1 0 0 1 -1)**

Current bit	Bit to right	1-bit coding
1	0	-1
0	1	1
0	0	0
0	0	0
1	0	-1
0	1	1
1	0	-1
1	1	0

Problem 2 – Booth's algorithm (contd..)

- From previous answer, we can obtain the 2-bit Booth's encoding easily (It is possible to obtain the 2-bit encoding directly -> discuss later).
  STEP 1: 1-bit Booth's encoding was **(0 -1 1 -1 0 0 1 -1)**
  STEP 2: Group in pairs and use the formula shown below -> Answer is **[-1 1 0 1]**

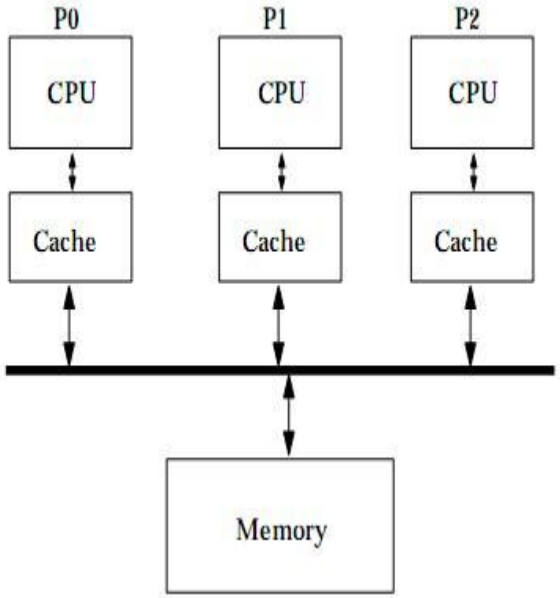
Pair (i+1, i)	[2*(i+1) + i]	2-bit Booth
[1, -1]	2*1 + (-1)	1
[0, 0]	2*0 + 0	0
[1, -1]	2*1 + (-1)	1
[0, -1]	2*0 + (-1)	-1

- In the problem given in HW6, multiplier is 0101 0101.
  Here is a step by-step description of what you need to do

STEP 1: Obtain 2-bit Booth encoding for the multiplier [Follow procedure explained] -> Result is [1, 1, 1, 1] (Note: In your assignment, you will have to show how you got the encoding. If not, **NO CREDIT** will be given)
STEP 2: Actual Multiplication (sign extend the multiplicand first and multiply with 2-bit booth encoding to get partial products)
(a) 1111 1111 1100 0101 * 1 = 1111 1111 1100 0101 (PP1)
(b) Shift the multiplicand left by 2 and repeat
1111 1111 0001 0100 * 1 = ? (PP2)
(c) Shift left (b) again by 2 and repeat
1111 1100 0101 0000 * 1 = ? (PP3)
(d) Shift left (c) again by 2 and repeat
1111 XXXX YYYY ZZZZ * 1 = ? (PP4)
STEP 3: Add all partial products obtained above (PP1 + PP2 + PP3 + PP4) to obtain final result. The expected result is $(-5015)_{10}$

Problem 4 - Coherence

- Understanding a 3-state write-invalidate cache coherence protocol (M, S and I states)
- Initially, all processors are in the Invalid state (as shown below) [Memory block at address 100 contains only one word]

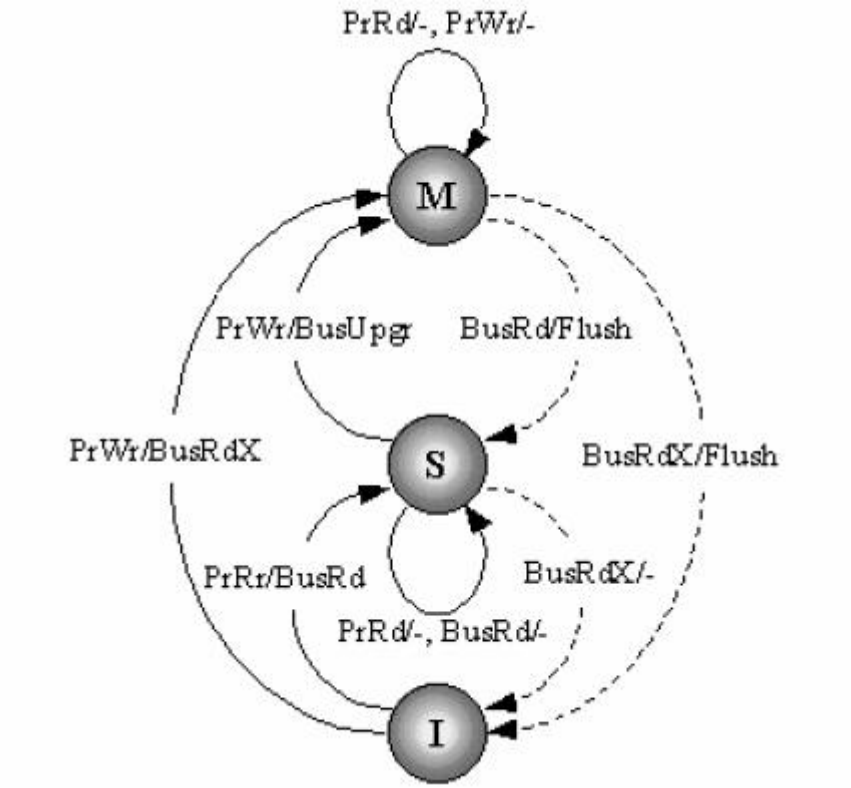


Initial state:

P0		P1		P2		Memory
State	Data	State	Data	State	Data	Data
I	0	I	0	I	0	7

- Processor P0 executes **lw $1, 100($0)**
  Action: Because the block is in I state in its cache, P0 misses and reads the block from memory (Let us show this in the table)

Problem 4 – MSI state diagram



- **INVALID** means the cache line is either not present or is in invalid state.
- If the cache line is clean and is shared by more than one processor, it is marked as **SHARED**.
- If a cache line is dirty and the processor has *exclusive* ownership of it, it is in the **MODIFIED** state.
- *BusRdX* causes other processors to invalidate (demote) its cache block to the **INVALID** state

Problem 4 – Coherence (contd..)

- P0 accessed data from memory and moves to S state (indicating it shares the data with memory). The other 2 processors still do not have valid copies of the block

P0		P1		P2		Memory
State	Data	State	Data	State	Data	Data
S	7	I	0	I	0	7

- Now P1 executes **addi $1, $0, 13** and **sw $1, 100 ($0)**. Let us see the contents of the table now (The key point to note is that only 1 processor can have the block in M state)

P0		P1		P2		Memory
State	Data	State	Data	State	Data	Data
I	7	M	13	I	0	7

- Now P2 executes **lw $1, 100 ($0)**. Note that P1 has the most valid copy of the block. So it will supply the data and update the memory as well

P0		P1		P2		Memory
State	Data	State	Data	State	Data	Data
I	7	S	13	?	?	?