Performance and Caching Issues in an Integration of Neural Net and Conventional PC

Veerachai Gosasang
Faculty of Science and Technology
Assumption University
Bangkok 10240, Thailand

Thitipong Tanprasert
Faculty of Science and Technology
Assumption University
Bangkok 10240, Thailand

Abstract

This paper presents an evaluation on the integration of Neural Network Hardware to PC [1] and a solution to the cache coherence problem. An evaluation is measured by determining clock cycles in CPU operation compare to mixed CPU-ANN mode. Cache coherence problem is resolved by hardware-based protocol executed on an additional cache consistency controller.

Keyword: Architecture, Hardware, Application

1. Introduction

Nowadays, DRAM works by sequential operation, which does not match with CPU speed. An improvement of applying Artificial Neural Network into this sequential operation has gained a lot of attention in order to relieve CPU-to-DRAM bottleneck. There are other extensive works that utilize neural networks as hardware computing unit [2][3] or integrate neural network computation on multiprocessor system [4][5].

Typical simulation of neural network computation uses sequential programming because conventional sequential computers are more suitable for some aspects, such as, large and accurate data storage and logic data processing. Besides, the conventional computer standards are already developed and well defined. However, simulating neural network by using sequential programming is limited to memory bandwidth of conventional PC architecture.

This paper is based on a previous work, *An Efficient Approach to Engage Neural Net Hardware to PC* [1]. This work is designed to solve a memory bandwidth bottleneck problem by adding ANN hardware to conventional PC as a special processor. The ANN hardware can simultaneously access multiple memory banks in parallel operation.

This work is partially supported by Thailand Research Fund (TRF) under contract No. BRG/3/2542

From previous work [1], there are two unsolved issues i.e. the model's performance evaluation and cache coherence problem. These issues are solved here as the following:

1. The model's performance: Performance evaluation is done by software simulation. A simulation program is created by using assembly language. It works by comparing Pentium clock cycles between sequential parts and mixed sequential-parallel parts. The main factors that used for performance investigation are number of input vectors, number of input nodes, number of hidden nodes and number of output nodes.

2. Cache coherence problem: Cache coherence problem occurs because DRAM use I/O bus as an alternative way to update data in parallel operation, but cache memory still use CPU/Memory bus in sequential update operation. When DRAM is updated, the cache is still unaltered then cache coherence problem occurs. A solution of this inconsistency problem is to implement a cache consistency controller.

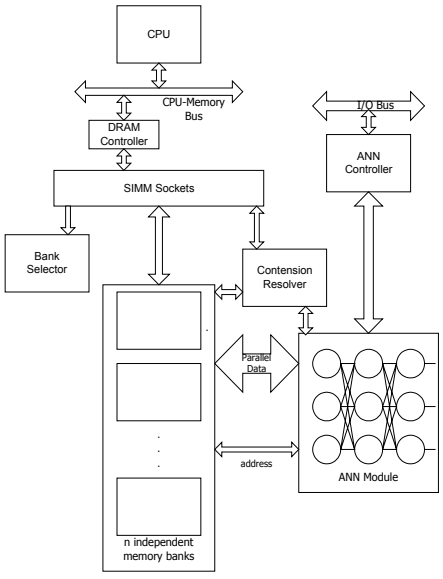

Figure 1. The Model combined DRAM/ANN architecture to evaluate performance.

Thus, this paper presents two main topics:

1. A performance evaluation on the model that presented in Thitipong's paper [1].
2. Solution of the cache coherence problem.

The rest of this work is organized as follows: Section 2, presents an evaluation performance of Neural Net Hardware. Section 3, presents a solution of cache coherence problem. Section 4, provides results and analysis. Finally, the conclusion is in Section 5.

2. Performance evaluation of Neural Net Hardware

This paper shows the evaluation of the proposed model by accumulating Pentium clock cycle usage for each instruction.

Instr. Code	Meaning	Format	Pentium Clock Cycle
add	add interger	add reg,reg	1 or 3
		add mem,reg	1 or 3
cmp	compare	cmp reg,reg	1 or 2
		cmp mem,reg	1 or 2
fadd	addition	fadd mem	1-7
fld	load real	fld data	1-3
fld	load real	fld data	1-3
imul	sign integer multiple	imul reg	10-11
jmp	jump	jump (short)	1
jge	jump cond.	jcc	1
mov	move to	mov reg,reg	1
		mov mem,reg	1
out	output data	out pt,acc	12-26
pop	pop	pop reg	1
		pop mem	3
push	push	push reg	1
		push mem	1 or 2

Table 1. The Example of instruction coding.

The following steps show the algorithm of the evaluation:

1. The test mode part of Neural Net simulator program is written in C/C^{++} language.
2. Compiles the program into an assembly language [6].
3. Simulates the instruction code executions based on Pentium clock cycles.
4. Accumulates the Pentium processor clock cycles for each program.
5. Compare Pentium processor clock cycles between sequential parts and mixed sequential-parallel parts execution. The analysis will be conducted to obtain the speedup criteria.

From this section, we will have clock cycle for CPU and mixed CPU-ANN mode, CPU is a sequential operation but mixed CPU-ANN mode is a mixed sequential- parallel execution.

$$T_{CPU} = T_{seq} + T_{comp1\_sq} + T_{comp2\_sq} \text{ ----(1)}$$

T_{CPU} is total execution and preparing time in CPU mode.

T_{seq} is preparing time for; read input, read weight1 (for input layer and hidden layer), read weight2 (for hidden layer and output layer),

T_{comp1_sq} is computation time in computation1 (for input layer and hidden layer), as a sequential operation.

T_{comp2_sq} is computation time in computation2 (for hidden layer and output layer), as a sequential operation.

$$T_{mix} = T_{seq} + T_{comp1\_par} + T_{comp2\_par} + T_{comm} \text{ (2)}$$

T_{mix} is total execution and preparing time in mixed CPU-ANN mode.

T_{seq} is preparing time for; read input, read weight1 (for input layer and hidden layer), read weight2 (for hidden layer and output layer).

T_{comp1_par} is computation time in computation1 (for input layer and hidden layer), as a parallel operation.

T_{comp2_par} is computation time in computation2 (for hidden layer and output layer), as a parallel operation.

T_{comm} is an overhead for communication between CPU and ANN. This time is about 12-26 clock cycles.(measured by OUT instruction of Pentium Clock Cycle) [9].

From equation (1) and (2)
1.T_{CPU} is a sequential operation. Four factors affects T_{CPU}:1. number of input vectors, 2. number of input nodes, 3. number of hidden nodes and 4. number of output nodes.

2.T_{mix} is a mixed sequential-parallel operation. In case of sequential operation, the execution time is same as T_{CPU}. In parallel operation (T_{comp1_par} and T_{comp2_par}),

the main factor that affects its value is only the number of input vectors.

3.T_{comm} is the overhead for communication between CPU and ANN. It is about 12-26 clock cycles.

3. A solution of cache coherence problem

The cache consistency controller that maintains cache and DRAM consistency can be implemented by a microcontroller--a single Very Large Scale Integration (VLSI) chip. A microcontroller contains devices and functions identical to a microcomputer [8].

Figure 1. illustrates how ANN controller attached on the I/O buses and shows how ANN controller controls ANN module.

The cache controller problem will be solved as follows;

1. CPU commands ANN module in the same way as any peripheral I/O equipment by OUT instruction [9].
2. When CPU transmits the addresses that are updated in DRAM by I/O bus these addresses are stored in cache consistency controller's registers [7][10][11].
3. When the addresses that are updated in DRAM are kept in a cache consistency controller. This controller will notify to cache controller to invalid or flush the address location in cache memory if this address is same as the address in cache memory [12].

The computation of invalidation time can be shown as:

Case1

When the time for invalidation in cache memory is less than the time in parallel computation. We will use partial invalidation technique--a technique for invalidation in cache memory, this technique invalids each memory location at a time [12].

$$n^* T_{inv.} < T_{ANN\_par} * i \ldots\ldots\ldots\ldots (1)$$

where

n is number of output addresses
i is number of input vectors
T_{inv} is invalidation time
T_{ANN_par} is parallel computation time

Case2

When the time for invalidation in cache memory is more than or equals the time in parallel computation. We will use flushing technique-- a technique for invalidation in cache memory, this technique is the flushing all the content of memory locations at a time [12].

$$n^* T_{inv.} \geq T_{ANN\_par} * i \ldots\ldots\ldots\ldots (2)$$

where

n is number of output address
i is number of input vectors
T_{inv} is invalidation time
T_{ANN_par} is parallel computation time

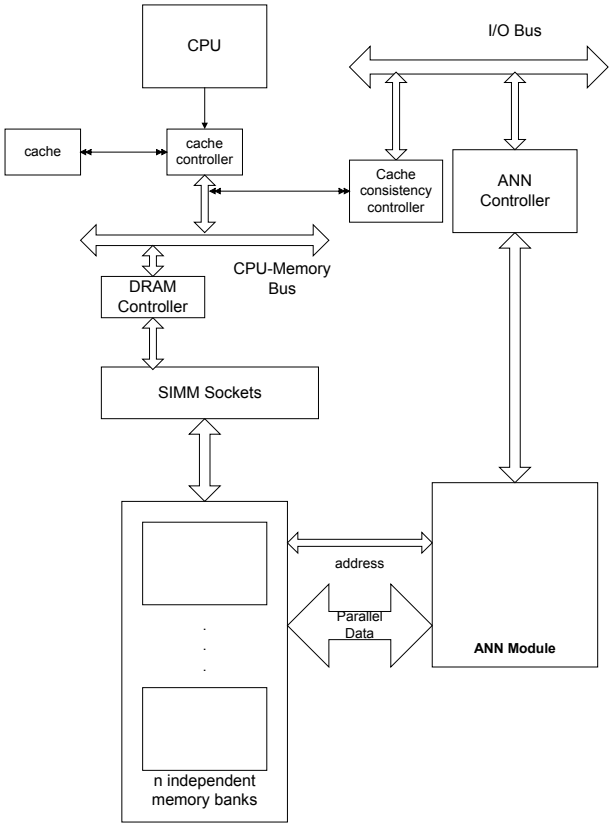

Figure 2. The Model of A solution of cache coherence problem.

Invalidation process in L_1 (internal cache)

1.Cache consistency controller transfers the address to L_2 controller.

2. L_2 controller activates to CPU's AHOLD input and the processor in response cease to drive its address bus and prepare to receive a snoop address over its address bus. The L_2 then transfers the memory address from the its register to the processor local bus.

3.L_2 controller signals CPU by EADS# signal. The EADS# signal tells the processor that a valid address is on its local bus.

4.CPU interrogates the address in L_1 cache(snoop).

5.If cache hit (snoop hit), CPU notifies to L_2 controller by HIT# signal.

6.After cache hit, L_2 controller signal CPU by INV signal, in case of invalidation or FLUSH# signal, in case of flushing.
-In each step takes one Pentium clock cycle.

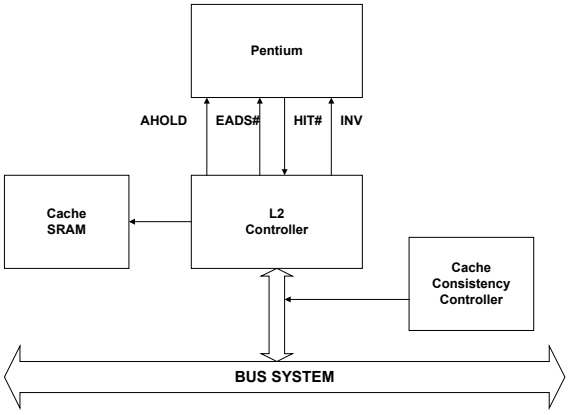

Figure 3. A solution of cache coherence problem in case of partial invalidation technique.

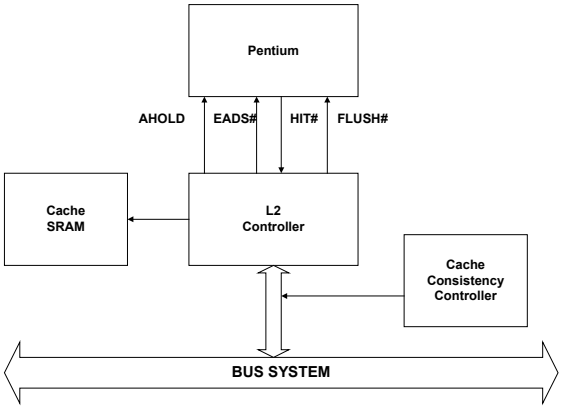

Figure 4. A solution of cache coherence problem in case of flushing technique.

Invalidation process in L_2 (external cache)

1. Cache consistency controller transfers the address to L_2 controller.

2. L_2 controller keeps the address and compare to L_2 cache. If the address in cache are the same as address in L_2 controller .Finally, L_2 controller invalidates that address in L_1.
-In each step takes one Pentium clock cycle.

Note: L_1 is internal cache
L_2 is external cache

4. Results and Analysis

This section is the part of analysis and discussions of a performance evaluation of Neural Net Hardware and a solution of cache coherence problem

Table 2,3 and Figure 5,6.
1.Increasing of number of input nodes, number of hidden nodes and number of output nodes will increase clock cycles in sequential part of CPU and sequential part of mixed CPU-ANN mode.

Sequential part of CPU includes preparing input, weight and sequential computation.

-Preparing input has two factors are: number of input vectors and number of input nodes.
-Preparing weight1 has two factors are: number of input nodes and number of hidden nodes.
-Preparing weight2 has two factors are: number of hidden nodes and number of output nodes.
-Sequential compuatation1 has three factors are: number of input vectors, number of input nodes and number of hidden nodes.
-Sequential compuatation2 has three factors are: number of input vectors, number of hidden nodes and number of output nodes.

Sequential part of mixed CPU-ANN mode is preparing input, weights in CPU.

Number of input nodes, hidden nodes and output nodes are not effected in the part of parallel execution.

2.Increasing of number of input vector affects both CPU and CPU-ANN mode. It results in longer input preparation time and longer computation time. In CPU mode, as the number of input vector increases, it takes more time compares to CPU-ANN mode.

3.Increasing of ANN size makes exponential incremental clock cycles in CPU, but in mixed CPU-ANN mode is increasing clock cycles by linear incremental. The execution time in CPU part is 3.3 to 24.5 times of mixed CPU-ANN mode, in case of non-pipelining, and 4-9 to 26.8 times, in case of pipelining.

4.Comparing between part of sequential and parallel execution of mixed CPU-ANN mode. Table 3,

A number of input vectors are 10 input vectors and parallel computation is 2,320 clock cycles, but a sequential part is longer preparation time, depends on a size of ANN module.

5.Pipelining execution takes time less than non-pipelining execution. Pipelining execution is the execution of a function of the next cycle beginning before the function of the current cycle is completed, the other is non-pipelining.

ANN Size	Tcpu	Tmix_np	Tmix_pipe	Tcomp_sq	Tseq
2x2x2	11,168	3,312	2,268	10,190	978
3x3x3	23,885	3,386	2,842	22,333	1,552
4x4x4	41,468	4,572	3,528	39,230	2,238
5x5x5	63,926	5,370	4,326	60,890	3,036
6x6x6	91,256	6,280	5,236	87,310	3,946
7x7x7	123,458	7,302	6,258	118,490	4,968
8x8x8	160,532	8,436	7,392	154,430	6,102
9x9x9	202,478	9,682	8,638	195,130	7,348
10x10x10	249,296	11,040	9,996	240,590	8,706
10x10x12	273,896	11,600	10,556	264,630	9,266
10x10x14	298,290	12,160	11,116	288,670	9,826
10x12x10	298,290	12,174	11,130	288,450	9,840
10x14x10	347,247	13,308	12,264	336,273	10,974
12x10x10	274,230	12,174	11,130	264,390	9,840
14x10x10	299,164	13,308	12,264	288,190	10,974

Table 2. Execution time of CPU and mixed CPU-ANN mode(non-pipelining and pipelining)

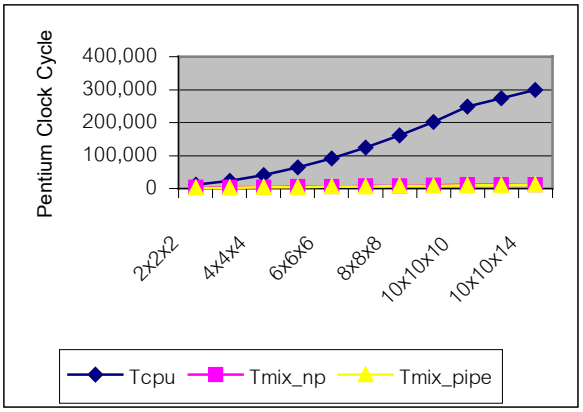

Figure 5. A Comparison execution time of CPU and mixed CPU-ANN mode (non-pipelining and pipelining)

ANN Size	Tann_seq	Tann_par
3x1x3	1,202	2,320
3x2x3	1,377	2,320
3x3x3	1,552	2,320
3x4x3	1,727	2,320
3x5x3	1,902	2,320
3x6x3	2,077	2,320
3x7x3	2,252	2,320
3x8x3	2,427	2,320
3x9x3	2,602	2,320
3x10x3	2,777	2,320

Table 3. Execution time of mixed CPU-ANN mode (compared to sequential and parallel part)

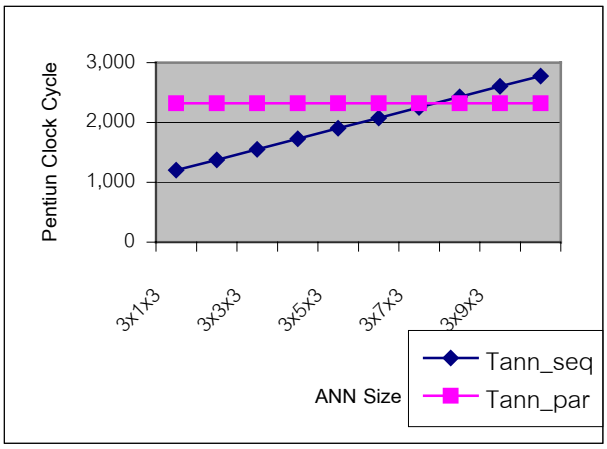

Figure 6. A Comparison of mixed CPU-ANN mode (compared to sequential and parallel part)

Input Vector	#Output address	T(L1_update)	T(L2_update)	Tann(comp_np)	Tann(comp_pipe)
5	110	660	220	1,160	696
5	116	696	232	1,160	696
10	212	1,272	424	2,320	1,276
20	1,000	6,000	2,000	4,640	2,436
20	2,300	13,800	4,600	4,640	2,436
50	980	5,880	1,960	11,600	5,916
50	1,933	11,598	3,866	11,600	5,916
100	1,950	11,700	3,900	23,200	11,716
100	2,000	12,000	4,000	23,200	11,716

Table 4.Time for invalidation in L_1 and L_2

Table 4 shows the invalidation time in L_1 and L_2 cache compares with T_{ann} in case of pipelining computation and non-pipelining computation. If number of input vectors increases, T_{ann} also increases. If number of output addresses, invalidation time increases.

Cache coherence controller provides cache and DRAM consistency. The drawback of this approach is that when the cache update takes place, the CPU must hold all cache operations.

5: Conclusions

This paper presents an evaluation of Neural Net Hardware integrated on PC and a solution to cache coherence problem.

An evaluation is measured by comparing Pentium clock cycles between CPU and mixed CPU-ANN mode. When we integrates ANN module to Conventional PC. This module takes executions time less than the part of CPU part. That means parallel computation in ANN module to is gained advantage.

Cache coherence problem is resolved by an additional cache consistency controller. This controller will notify to cache controller to invalid cache memory if the addresses are same as the address in cache memory.

The main factor is a number of input vectors will effect in parallel computation and a number of output addresses is a factor for invalidation time in cache memory.

6: Bibliography

1.T. Tanprasert, P.H.Hanh, "An Efficient Approach to Engage Neural Net Hardware to PC": IJCNN'98, Ancherage, Alaska, USA, May 1998.

2.X. Fang, P. Thole, J. Gvppert, W. Rosenstiel, " A Hardware Supported System for a Special Online Application of Self-Organizing Map", Proc. of the 1996 IEEE International Conference on Neural Networks(ICNN'96).

3.T. Hamalainen, H. Klapuri, J. Saarinen and K. Kaski, "Mapping of Multilayer Perceptron Networks to Tree Shape Parallel Neurocomputer", Proc. of the 1996 IEEE International Conference on Neural Networks(ICNN'96).

4. O. Hammani and D. Suzuki, "A Pipelined Speculative SIMD Architecture for SOM ANN", Proc. of the 1997 IEEE International Conference on Neural Networks(ICNN'97).

5.H. Abbas and M. M. Bayoumi, "On the implementation of Backpropagation on the Alex AVX-2 Parallel System", Proc. of the 1997 IEEE International Conference on Neural Networks(ICNN'97).

6. Ray Duncan, Microsoft macro assembler: Microsoft programming Series, 1992

7. W. Stalling, Computer Organization and Architecture: Prentice Hall, 1996

8. H. Way Huang, Using the MCS-51 Microcontroller: Oxford University Press, 2000

9. Barry B.Brey,The Intel Microprocessors 5th edition: Prentice Hall, 2000

10. G.Wyant and T. Hammerstrom, How microprocessors work: Ziff-Davis Press, 1994

11. Roger M. Mersey, Personal Computer Operation and Troubleshooting: Prentice-Hall, 1996

12.D. Anderson., and T. Shanley, Pentium processor System Architecture 2 nd edition: Mindshare, Inc.,1995