가변적 하드웨어 구성에 대한 수퍼스칼라 프로세서의 성능 예측 모델

이종복
jongbok@lgsemicon.co.kr
현대반도체 시스템 IC 사업부 신사업담당 프로세서팀

An Analytical Performance Model for Superscalar Processors with Various Hardware Configurations

Jongbok Lee
New Biz. Processor Team, System IC Division, HYUNDAI MicroElectronics

요 약

본 논문에서는 주어진 윈도우에 대하여 수퍼스칼라 프로세서의 하드웨어를 구성하는 기본 요소인 인출율과 연산유닛의 개수로 표현되는 성능 예측 모델을 제시하였다. 이 때, 수퍼스칼라 프로세서에서 실행되는 벤치마크 프로그램은 매 싸이클당 각 명령어 개수가 실행되는 확률과 분기 예측 정확도에 의하여 특성화된다. 초기의 실험으로 각종 파라미터를 획득한 후에는 다양한 연산유닛과 인출율을 갖는 수퍼스칼라 프로세서의 성능을 본 논문에서 제안하는 모델에 의하여 간단하게 구할 수 있다. 명령어 자취 모의실험 (trace-driven simulation)으로 측정한 성능과 본 논문에서 제안하는 성능 예측 모델에 의한 성능을 비교한 결과, 3.8 %의 평균오차를 기록하였다.

1. 서론

수퍼스칼라 프로세서는 다중 연산유닛을 이용하여 명령어 단위 병렬성을 높이므로, 각 연산유닛의 개수를 정하는 것이 설계상의 중요한 문제이다 [1] [2]. 각 연산유닛의 사용 빈번도에 따라 특정한 연산유닛은 부족하여 성능을 저하시킬 수도 있고, 또 다른 연산유닛은 과잉으로 존재하여 성능 향상에 도움을 주지 못하고 하드웨어 비용만을 상승시킬 수 있다. 따라서 연산유닛의 개수의 함수로서 수퍼스칼라 프로세서의 성능 모델을 나타낼 수 있다면, 설계의 초기 단계에서 특정한 하드웨어 환경 하에서의 성능을 예측할 수 있으며 동시에 그 프로세서의 적용에 필요한 가장 적절한 개수의 연산유닛을 정할 수 있다.

2. 성능 예측 모델

2.1 무한한 연산유닛하에서의 성능 예측 모델

본 절에서는 연산유닛의 개수가 무한할 때, 윈도우의 크기와 인출율을 함수로 하는 수퍼스칼라 프로세서의 성능 예측 모델을 제시한다. 윈도우의 크기가 W인 수퍼스칼라 프로세서에서 매 싸이클 당 k개의 명령어를 실행할 확률을 $P_W(k)$라고 하자. 만일 인출율에 제한이 없다면 ipc는

$$ipc = 1 * P_W(1) + 2 * P_W(2) + 3 * P_W(3) + \cdots + W * P_W(W) \quad (1)$$

로 나타낼 수 있다. 그러나 일반적으로 W가 k보다 크므로

$$ipc = 1 * P_W(1) + 2 * P_W(2) + \cdots + (k-1) * P_W(k-1) + k * \{P_W(k) + P_W(k+1) + \cdots + P_W(W)\} \quad (2)$$

이다. 따라서, 매 싸이클 마다 실제로 인출되는 명령어의 개수가 i이고 그 최대값이 인출율인 k이며, 분기예측 정확도가 B, 분기 미스 페날티가 T 싸이클인 수퍼스칼라 프로세서의 ipc는 다음과 같이 나타낼 수 있다. 단, 각종 캐쉬 미스에 의한 지연은 무시한다.

$$ipc = \frac{\{\sum_{i=1}^{k-1} i * P_W(i) + \sum_{i=k}^{W} k * P_W(i)\}}{1 + (1-B) * T} \quad (3)$$

2.2 유한한 연산유닛하에서의 성능 예측 모델

본 논문에서는 연산유닛의 개수의 함수로서 수퍼스칼라 프로세서의 성능 모델을 나타내기 위하여, 각 연산유닛별로 매 싸이클 당 실행되는 명령어의 평균 개수를 구한 후에 이것들의 합으로 전체 프로세서의 성능을 나타내는 방법을 제안하였다. 각 연산유닛의 개수가 제한이 없을 때 매 싸이클 당 실행되는 연산유닛별 명령어의 평균 개수를 각각 $ipc_{alu}, ipc_{ld}, ipc_{st}, ipc_{sh}, ipc_{br}$라고 할 때 프로세서의 성능은 이들의 합으로 다음과 같이 나타난다.

$$ipc = ipc_{alu} + ipc_{ld} + ipc_{st} + ipc_{sh} + ipc_{br} \quad (4)$$

윗식에서 $ipc_{alu}, ipc_{ld}, ipc_{st}, ipc_{sh}, ipc_{br}$은 각각 ALU, 로드, 스토어, 쉬프터, 분기 명령어가 매 싸이클당 각각 평균적으로 실행되는 개수이다.

이제 각 연산유닛의 개수에 제한이 있을 때 매 싸이클 당 실행되는 연산유닛별 명령어의 평균 개수를 각각 $Ripc_{alu}$, $Ripc_{ld}$, $Ripc_{st}$, $Ripc_{sh}$, $Ripc_{br}$이라 하고 이 때의 성능을 $Ripc$라 하면 역시 다음과 같이 나타낼 수 있다.

$$Ripc = Ripc_{alu} + Ripc_{ld} + Ripc_{st} + Ripc_{sh} + Ripc_{br} \quad (5)$$

이 때 각 연산유닛별 명령어의 평균 개수와 그 때의 전체 성능에 대한 관계를 살펴보면 다음과 같다. 임의의 정수형 프

로그램의 인출된 명령어 중에서 가장 빈도율이 높으며 실행에 영향을 끼치는 명령어는 ALU 명령어와 로드 명령어이다. 그것은 ALU와 로드 명령어가 기타 명령어에 대한 실제 종속성(true-dependency)이 가장 크기 때문이다.

임의의 프로세서에서 설계에 의하여 주어진 각 연산유닛의 개수를 각각 N_{alu}, N_{ld}, N_{st}, N_{sh}, N_{br}라 하자. 주어진 연산유닛의 개수가 이 프로그램의 각 연산유닛의 평균 개수보다 크면, 해당 연산유닛에서 그 평균 개수 만큼의 명령어가 실행된다고 가정한다. 그러나 주어진 연산유닛의 개수가 프로그램의 각 연산유닛의 평균 개수보다 작으면, 해당 연산유닛에서 주어진 연산유닛의 개수만큼만 실행이 된다고 가정한다. ALU의 예를 들면 다음 식 6과 같이 나타난다.

$$\begin{aligned} ipc_{alu} &< N_{alu}\text{일 때 } Ripc_{alu} = ipc_{alu} \\ ipc_{alu} &\geq N_{alu}\text{일 때 } Ripc_{alu} = N_{alu} \end{aligned} \quad (6)$$

이 때 ALU 연산유닛의 부족으로 인한 한계 상수를 L_{alu}라고 할 때 다음 식 7과 같이 정의된다.

$$L_{alu} = \frac{N_{alu}}{ipc_{alu}} \quad (7)$$

임의의 프로그램에서 ALU만 평균 연산유닛의 명령어 개수에 비하여 부족하게 지원되고 나머지 연산유닛은 모두 충분하다고 가정을 할 때 부족한 ALU의 개수에 비례하여 로드, 스토어, 쉬프트, 분기 명령어의 실행율이 감소한다. 따라서 제한된 각 연산유닛에서 실행되는 명령어의 평균 개수는 아래 식 8과 같이 나타날 수 있으며,

$$\begin{aligned} Ripc_{alu} &= N_{alu} \\ Ripc_{ld} &= ipc_{ld} \times L_{alu} \\ Ripc_{st} &= ipc_{st} \times L_{alu} \\ Ripc_{br} &= ipc_{br} \times L_{alu} \end{aligned} \quad (8)$$

이 때의 성능을 정리하면 결국 다음 식 9와 같다.

$$\begin{aligned} Ripc &= Ripc_{alu} + Ripc_{ld} + Ripc_{st} + Ripc_{sh} + Ripc_{br} \\ &= ipc \times L_{alu} \end{aligned} \quad (9)$$

마찬가지로, 임의의 프로그램에서 로드유닛만이 평균 연산유닛 개수에 비하여 부족하다면 위와 동일하게 L_{ld}를 식 10과 같이 정의할 수 있으며,

$$\begin{aligned} ipc_{ld} &< N_{ld}\text{일 때 } Lipc_{ld} = ipc_{ld} \\ ipc_{ld} &\geq N_{ld}\text{일 때 } Lipc_{ld} = N_{ld} \end{aligned} \quad (10)$$

그 때의 성능은 다음 식 11과 같다.

$$Ripc = ipc \times L_{ld} \quad (11)$$

임의의 프로그램에서 ALU와 로드유닛이 동시에 부족한 경우에는 식 12와 같이 각각의 한계 상수를 구하여 그 값이 더 작은 쪽을 택한다. 그 이유는 프로그램의 병렬도는 필요한 연산유닛의 개수가 더 작은 쪽에 의하여 그 실행에 더 큰 영향을 받기 때문이다.

$$\begin{aligned} L_{alu} &< L_{ld}\text{일 때 } Ripc = ipc \times L_{alu} \\ L_{alu} &\geq L_{ld}\text{일 때 } Ripc = ipc \times L_{ld} \end{aligned} \quad (12)$$

윗 식에서 ipc는 3절의 식 3에 의해서 나타낸 바 있다. 따라서, 초기 모의실험에서 주어진 윈도우 크기에 대하여 각 벤치마크 프로그램의 각 명령어 개수당 실행 확률과 분기 예측 정확도 및 각 연산유닛에서 평균적으로 요구되는 명령어의 개수를 측정하면, 임의의 인출율에 대하여 각 연산유닛의 개수가 다양하게 한정될 때의 수퍼스칼라 프로세서의 성능을 예측할 수 있다.

3. 모의실험 환경

본 논문에서는 기본구조로서, 그림 1에 보인 바와 같은 수퍼스칼라 프로세서를 채택하였다. Tomasulo 알고리즘에 의하여 명령어들이 비순차적으로 연산유닛으로 이슈 및 실행될 수 있다[3]. 이 때, 발행된 명령어는 정수형이므로 단위 시간만에 실행이 완료된다고 가정하였다. 본 프로세서는 윈도우의 크기가 32이고 인출율이 16인 것, 윈도우의 크기가 16이고 인출율이 8인것, 마지막으로 윈도우의 크기가 8이며 인출율이 4인 3 가지 유형에 대하여 모의실험을 수행하였다.

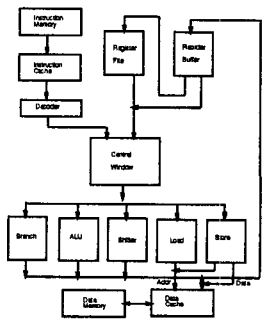

그림 1: 수퍼스칼라 프로세서의 기본구조.

표 1은 본 논문에서 모의실험에 사용한 3 가지 연산유닛에 대한 사양이다. 연산유닛의 개수를 정하기 위한 초기 모의실험에 의하여 ALU와 로드유닛의 순서로 그 요구량이 가장 크기 때문에 이와 같은 사양을 채택하였다.

표 1: 연산유닛의 개수별 사양

연산유닛 사양	ALU	로드	스토어	분기	쉬프트
1	1	1	1	1	1
2	2	1	1	1	1
3	4	2	1	1	1

명령어 자취 모의실험은 SPARC를 기반으로 하여 수행하였으며, 표 2에 실험에 사용된 8 개의 정수형 SPEC 벤치마크를 수록하였다. 이 프로그램들을 C 컴파일러에 의하여 오브젝트

표 2: SPEC 정수형 벤치마크.

벤치마크	기 술
eqntott(eq)	진리표 생성기
espresso(es)	불리언 함수 최소화기
lisp(li)	lisp 해석기
gcc1(gc)	GNU C 컴파일러의 패스 1
go(go)	바둑 게임
ijpeg(ij)	jpeg 영상 압축
mk88sim(mk)	Motorolla 88100 모의실험기
perl(pe)	문자 및 숫자 처리기

코드를 얻은 후에 Shadow를 통과시켜 천만 개의 명령어 자취를 발생시켰다. 이렇게 하여 발생된 명령어 자취는 수퍼스칼라

모의실험기로 입력되었다. 본 실험에서 편의상 분기 미스에 의한 페날티는 0 싸이클로 간주하였다.

결과적으로, 8 개의 벤치마크 프로그램에 대하여, 3 가지의 윈도우의 크기와 인출율 쌍을 갖는 수퍼스칼라 프로세서의 각 3 가지 사양의 연산유닛 하에서 각각 모의실험을 수행하여 성능을 측정한 후에, 연산유닛의 개수의 함수로서 표현되는 성능 예측 모델에 의한 결과와 비교하였다.

4. 모의실험 결과 및 분석

그림 2는 윈도우의 크기가 32이고 인출율이 16일 때 초기 모의실험으로 측정한 각 벤치마크의 각 명령어 개수에 대한 실행 확률을 나타낸다. 한 싸이클에 11개 이상의 명령어가 동시 실행되는 확률은 0.001보다 작으므로 그 이하는 생략하였다. 한 싸이클에 2 개에서 6개 사이의 명령어에 대한 확률이 0.1과 0.3사이에 위치하여 집중적으로 실행됨을 알 수 있다.

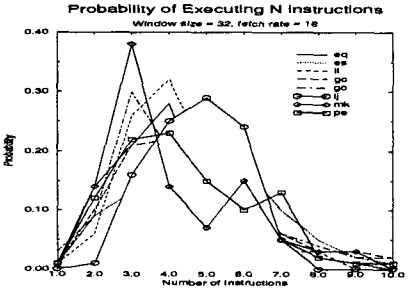

그림 2: 각 명령어 개수에 대한 실행 확률.

표 3은 윈도우의 크기가 32이고 인출율이 16일 때 각 벤치마크의 연산유닛에 대한 평균 요구 개수를 나타낸 것이다. 역시 ALU에 대한 요구 개수가 평균 2.12개로 가장 많으며, 그 다음으로는 로드유닛이 0.85를 기록하였다.

표 3: 각 연산유닛의 평균 소요 개수

benchmark	ALU	로드	스토어	쉬프트	분기
eq	2.23	0.81	0.14	0.11	1.00
es	2.42	1.11	0.27	0.22	0.95
li	1.81	1.00	0.37	0.05	1.00
gc	2.27	0.82	0.29	0.30	0.94
go	2.18	0.49	0.27	0.57	0.88
ij	2.24	0.93	0.56	0.02	0.99
mk	1.65	0.75	0.31	0.40	0.98
pe	2.15	0.85	0.29	0.13	1.00

그림 3과 그림 4는 윈도우의 크기가 32, 인출율이 16일 때 명령어 자취 모의실험에 의하여 실제로 각 연산유닛을 제한한 상태에서 측정한 성능과, 본 논문에서 제안하는 예측 성능 모델에 의하여 얻은 성능을 비교한 것이다. 그림에서 각 벤치마크에 대하여 모의실험에 의하여 직접 측정한 것은 (m)으로, 예측 성능 모델에 의하여 계산한 결과는 (p)로 나타내었으며, 평균 오차는 3.2 %를 기록하였다. 윈도우의 크기가 16, 인출율이 8인 경우에는 4.3 %, 마지막으로 윈도우의 크기가 8, 인출율이 4인 경우에는 3.9 %의 평균오차를 나타내었다.

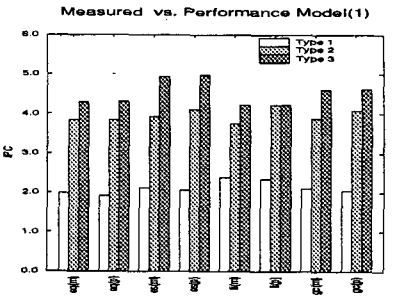

그림 3: IPC의 측정값과 모델값의 비교(a).

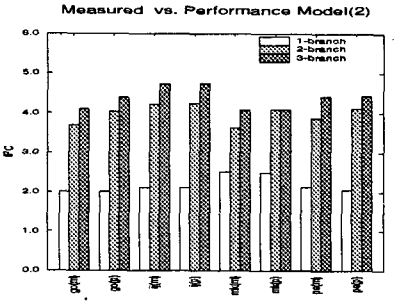

그림 4: IPC의 측정값과 모델값의 비교(b).

5. 결론

본 논문에서는 윈도우의 크기가 주어졌을 때 가변적인 인출율과 연산유닛의 개수를 갖는 수퍼스칼라 프로세서의 성능 예측 모델을 제안하였다. 3 가지의 서로 다른 윈도우의 크기, 인출율의 쌍에 대하여 각각 3 가지 유형의 연산유닛을 갖는 수퍼스칼라 프로세서를 8 개의 정수형 벤치마크에 대하여 명령어 자취 모의실험으로 측정한 성능값과, 예측 성능 모델에 의하여 계산한 값을 비교한 결과, 3.8 %의 평균오차를 기록하였다. 추후의 연구과제는 윈도우의 크기와 관계없으며 실수형 벤치마크를 포함하는 일반적인 수퍼스칼라의 성능 예측 모델을 제안하는 것이다.

참고 문헌

[1] E.S.T. Fernandes and F.M.B. Barbosa, "Effects of Buliding Blocks on the Performance of Super-Scalar Architectures," in *Proceedings of the 19th Annual Symposium on Computer Architecture*, 1992, pp. 36–45.

[2] A. Inoue and K. Takeda, "Performance Evaluation for Various Configuration of Superscalar Processors," *Computer Architecture News*, pp. 4–11, 1993.

[3] R.M. Tomasulo, "An Efficient Algorithm for Explotiting Multiple Arithmetic Units," *IBM Journal*, vol. 11, pp. 25–33, Jan. 1967.