(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2013-0088285
(43) 공개일자 2013년08월08일

(51) 국제특허분류(Int. Cl.)
G06F 9/455 (2006.01) *G06F 9/45* (2006.01)
(21) 출원번호 10-2012-0009426
(22) 출원일자 2012년01월31일
심사청구일자 없음

(71) 출원인
삼성전자주식회사
경기도 수원시 영통구 삼성로 129 (매탄동)
(72) 발명자
윤의선
경기도 성남시 분당구 이매동 64-3번지 201호
김범석
충청남도 천안시 동남구 신방동 성지새말아파트 203-1203
(뒷면에 계속)
(74) 대리인
윤동열

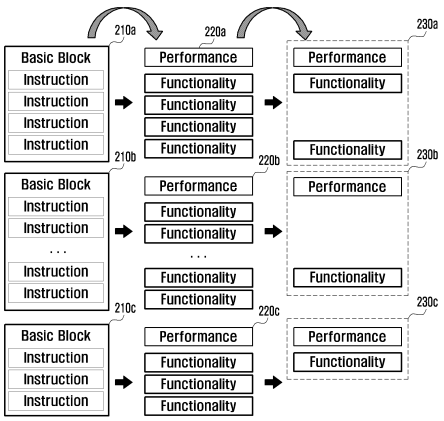
전체 청구항 수 : 총 10 항

(54) 발명의 명칭 **데이터 처리 시스템 및 그 시스템에서 데이터 시뮬레이션 방법**

(57) 요 약

본 발명은 데이터 처리 시스템 및 그 시스템에서 시뮬레이션을 수행하는 방법에 관한 것으로, 어시뮬레이션이 수행될 어플리케이션 코드를 입력하는 과정과, 상기 입력된 어플리케이션 코드를 어셈블리 데이터 구성으로 변환하는 과정과, 상기 어셈블리 데이터 구성으로부터 기본 블록을 생성하는 과정과, 상기 생성된 기본 블록을 통해 상기 시뮬레이션을 수행하는 과정으로 구성된다. 따라서 시뮬레이션이 인스트럭션 단위에서 기본 블록 단위로 수행되기 때문에 시뮬레이션 수행 시간이 단축된다.

대 표 도 - 도2



(72) 발명자

조경수

서울특별시 강서구 염창동 268번지 삼성관음아파트 101-1406

문지중

경기도 화성시 반송동 92-2번지 유진마젤란오피스텔 2703호

특허청구의 범위

청구항 1

데이터 처리 시스템의 데이터 시뮬레이션 방법에 있어서,

시뮬레이션이 수행될 어플리케이션 코드를 입력하는 과정과,

상기 입력된 어플리케이션 코드를 어셈블리 데이터 구성으로 변환하는 과정과,

상기 어셈블리 데이터 구성으로부터 기본 블록을 생성하는 과정과,

상기 생성된 기본 블록을 통해 상기 시뮬레이션을 수행하는 과정을 포함하는 데이터 시뮬레이션 방법.

청구항 2

제1항에 있어서, 상기 기본 블록을 생성하는 과정은

상기 기본 블록의 수행 요소를 예측하여 시뮬레이션 레벨을 설정하는 과정과,

상기 설정된 시뮬레이션 레벨에 따라 시뮬레이션 코드를 생성하는 과정을 포함하는 것을 특징으로 하는 데이터 시뮬레이션 방법.

청구항 3

제2항에 있어서, 상기 기본 블록을 생성하는 과정은

상기 시뮬레이션 목적에 따라 상기 시뮬레이션 레벨을 설정하는 과정을 포함하는 것을 특징으로 하는 데이터 시뮬레이션 방법.

청구항 4

제3항에 있어서, 상기 시뮬레이션 레벨을 설정하는 과정은

상기 기본 블록을 구성하는 적어도 하나의 인스트럭션에 공통으로 적용된 Performance와 각 인스트럭션 별 functionality를 추출하는 과정과,

상기 인스트럭션별로 추출된 Functionality 중에서 시뮬레이션을 수행할 Functionality를 선택하는 과정을 포함하는 것을 특징으로 하는 데이터 시뮬레이션 방법.

청구항 5

제1항에 있어서, 상기 기본 블록을 생성하는 과정은

상기 어셈블리 데이터 구성에서 기본 블록(Basic Block)을 생성하기 위한 Entry Point와 Exit Point를 검색하는 과정과,

상기 검색한 Entry Point와 Exit Point 중 동일한 Entry Point와 Exit Point를 갖는 어셈블리 데이터 구성을 그룹핑하여 기본 블록으로 생성하는 과정을 포함하는 것을 특징으로 하는 데이터 시뮬레이션 방법.

청구항 6

시뮬레이션이 수행될 어플리케이션 코드를 입력하는 타겟 컴파일 시스템과,

상기 입력된 어플리케이션 코드를 어셈블리 데이터 구성으로 변환하고, 어셈블리 데이터 구성으로부터 기본 블록을 생성하는 수행 추정 코드 생성 시스템과,

상기 생성된 기본 블록을 통해 상기 시뮬레이션을 수행하는 호스트 컴파일 시스템을 포함하는 데이터 시뮬레이션 시스템.

청구항 7

제6항에 있어서, 상기 수행 추정 코드 생성 시스템은

상기 기본 블록의 수행 요소를 예측하여 시뮬레이션 레벨을 설정하고, 상기 설정된 시뮬레이션 레벨에 따라 시뮬레이션 코드를 생성하는 것을 특징으로 하는 데이터 시뮬레이션 시스템.

청구항 8

제7항에 있어서, 상기 수행 추정 코드 생성 시스템은

상기 시뮬레이션 목적에 따라 상기 시뮬레이션 레벨을 설정하는 것을 특징으로 하는 데이터 시뮬레이션 시스템.

청구항 9

제8항에 있어서, 상기 수행 추정 코드 생성 시스템은

상기 기본 블록을 구성하는 적어도 하나의 인스트럭션에 공통으로 적용된 Performance와 각 인스트럭션 별 functionality를 추출하고, 상기 인스트럭션별로 추출된 Functionality 중에서 시뮬레이션을 수행할 Functionality를 선택하는 것을 특징으로 하는 데이터 시뮬레이션 시스템.

청구항 10

제6항에 있어서, 상기 수행 추정 코드 생성 시스템은

상기 어셈블리 데이터 구성에서 기본 블록(Basic Block)을 생성하기 위한 Entry Point와 Exit Point를 검색하고, 상기 검색한 Entry Point와 Exit Point 중 동일한 Entry Point와 Exit Point를 갖는 어셈블리 데이터 구성을 그룹핑하여 기본 블록으로 생성하는 것을 특징으로 하는 데이터 시뮬레이션 시스템.

명 세 서

기 술 분 야

[0001] 본 발명은 데이터 처리 시스템 및 그 시스템에서 데이터 시뮬레이션 방법에 관한 것으로, 다양한 Computing Resource가 있는 HMA(Heterogeneous Multi/Many Core) 환경에서 효율적인 CPU 활용을 위해 특정 CPU 상에서 소프트웨어 성능을 예측하는 방법에 관한 것이다.

배 경 기 술

[0002] 일반적으로 사용자에 의해 선택되는 소프트웨어가 하드웨어에서 수행되기 위해서는 컴파일 과정을 거쳐야 한다. 그리고 효율적인 CPU 활용을 위해 특정 CPU상에서 소프트웨어 성능을 예측하기 위해 시뮬레이션을 수행해야 한다. 그러기 위해 해당 CPU를 목적으로 하는 Assembly 코드가 수신되면, 이를 Simulation 가능한 Host Assembly-level C 코드로 변환한다.

[0003] 좀 더 상세히 설명하면, 우선 측정하고자 하는 C 코드가 Target Compiler를 이용하여 Assembly 코드로 컴파일된다. 그리고 Translator는 컴파일된 Assembly 코드를 Simulation이 가능한 C 코드로 생성한다. 생성된 Simulation C 코드는 Host Compiler를 이용하여 Simulation Object 파일로 컴파일 된다. 마지막으로 앞의 절차를 통하여 생성된 다른 Object 파일과 Simulation 등을 위한 Library와 링크되어 실행 가능한 Simulation 파일이 생성된다.

[0004] Translator는 Assembly 코드를 Main Data Structure로 변환하고, 이 데이터 구조체를 중심으로 다중의 연산을 수행한다. 이러한 연산으로 WORD EXPANSION, DEPENDENCY RESOLUTION, SYMBOL EXTRACTION 등이 대표적이다.

[0005] WORD EXPANSION은 TARGET CPU에 따라 해당 Assembly Instruction을 확장/축소하여 변환하는 연산이다. DEPENDENCY RESOLUTION은 Assembly Instruction이 수행하기 위하여 필요한 H/W Resource의 의존관계를 분석하여 시간정보를 삽입하는 연산이다. SYMBOL EXTRACTION은 Assembly File으로부터 함수 이름, 전역 변수, 정적 변수, Assembly 라벨 등의 Symbol 정보를 추출하여 SYMBOL TABLE를 생성한다. ELEMENTARY SIMULATION MODEL GENERATION, SIMULATION ASSEMBLY, SIMULATION EXPORT는 앞에서 생성한 정보를 통해 Assembly Instruction 단위로 Host Simulation C 코드를 생성하는 연산이다.

발명의 내용

해결하려는 과제

[0006] 이러한 과정들을 통해 시뮬레이션을 수행하는 경우, 실행되는 시간이 오래 걸린다는 문제점이 발생한다. 즉 시뮬레이션 수행시, Assembly Instruction 단위로 시뮬레이션이 수행되기 때문에, 소요되는 시간이 늘어난다는 문제점이 발생한다.

[0007] 따라서 본 발명에서는 데이터 처리 시스템 및 그 시스템에서 데이터 시뮬레이션 방법을 제안한다.

과제의 해결 수단

[0008] 상기와 같은 문제점을 해결하기 위해 본 발명에서 데이터 시뮬레이션 방법은 시뮬레이션이 수행될 어플리케이션 코드를 입력하는 과정과, 상기 입력된 어플리케이션 코드를 어셈블리 데이터 구성으로 변환하는 과정과, 상기 어셈블리 데이터 구성으로부터 기본 블록을 생성하는 과정과, 상기 생성된 기본 블록을 통해 상기 시뮬레이션을 수행하는 과정을 포함한다.

[0009] 또한 본 발명에서 상기 기본 블록을 생성하는 과정은 상기 기본 블록의 수행 요소를 예측하여 시뮬레이션 레벨을 설정하는 과정과, 상기 설정된 시뮬레이션 레벨에 따라 시뮬레이션 코드를 생성하는 과정을 포함하는 것을 특징으로 한다.

[0010] 그리고 본 발명에서 상기 기본 블록을 생성하는 과정은 상기 시뮬레이션 목적에 따라 상기 시뮬레이션 레벨을 설정하는 과정을 포함하는 것을 특징으로 한다.

[0011] 다음으로 본 발명에서 상기 기본 블록을 생성하는 과정은 상기 어셈블리 데이터 구성에서 기본 블록(Basic Block)을 생성하기 위한 Entry Point와 Exit Point를 검색하는 과정과, 상기 검색한 Entry Point와 Exit Point 중 동일한 Entry Point와 Exit Point를 갖는 어셈블리 데이터 구성을 그룹핑하여 기본 블록으로 생성하는 과정을 포함하는 것을 특징으로 한다.

[0012] 그리고 상기와 같은 문제점을 해결하기 위해 본 발명에서 데이터 시뮬레이션 시스템은 시뮬레이션이 수행될 어플리케이션 코드를 입력하는 타겟 컴파일 시스템과, 상기 입력된 어플리케이션 코드를 어셈블리 데이터 구성으로 변환하고, 어셈블리 데이터 구성으로부터 기본 블록을 생성하는 수행 추정 코드 생성 시스템과, 상기 생성된 기본 블록을 통해 상기 시뮬레이션을 수행하는 호스트 컴파일 시스템을 포함한다.

[0013] 그리고 본 발명에서 상기 수행 추정 코드 생성 시스템은 상기 기본 블록의 수행 요소를 예측하여 시뮬레이션 레벨을 설정하고, 상기 설정된 시뮬레이션 레벨에 따라 시뮬레이션 코드를 생성하는 것을 특징으로 한다.

[0014] 또한 본 발명에서 상기 수행 추정 코드 생성 시스템은 상기 시뮬레이션 목적에 따라 상기 시뮬레이션 레벨을 설정하는 것을 특징으로 한다.

[0015] 다음으로 본 발명에서 상기 수행 추정 코드 생성 시스템은 상기 어셈블리 데이터 구성에서 기본 블록(Basic Block)을 생성하기 위한 Entry Point와 Exit Point를 검색하고, 상기 검색한 Entry Point와 Exit Point 중 동일한 Entry Point와 Exit Point를 갖는 어셈블리 데이터 구성을 그룹핑하여 기본 블록으로 생성하는 것을 특징으로 한다.

발명의 효과

[0016] 본 발명에 따르면, 시뮬레이션이 인스트럭션 단위에서 기본 블록 단위로 수행되기 때문에 시뮬레이션 수행 시간이 단축된다. 또한 기본 블록 단위로 시뮬레이션이 수행됨으로써, 인스트럭션별로 수행되던 시뮬레이션 수행 시간이 단축된다. 그리고 기본 불록 단위에서 시뮬레이션하고자 하는 레벨에 따라 선택적인 시뮬레이션 코드가 생성됨으로써, 시뮬레이션 수행 시간이 단축된다.

도면의 간단한 설명

[0017] 도 1은 본 발명의 실시예에 따른 데이터 처리 시스템을 도시한 도면.

도 2는 본 발명의 실시예에 따라 구성된 기본 블록을 이용하여 시뮬레이션 레벨을 설정하는 과정을 도시한 도면.

도 3은 본 발명의 실시예에 따라 데이터 시뮬레이션 방법을 도시한 도면.

발명을 실시하기 위한 구체적인 내용

[0018] '데이터 처리 시스템'은 데이터를 생성하거나, 외부로부터 송수신되는 데이터를 처리할 수 있는 정보 처리 기기를 의미한다. 여기서 데이터 처리 시스템은 컴퓨터, 노트북, 테블릿 PC, 휴대 단말, 스마트 폰 등이 포함되며, 적어도 두 개의 처리 장치 다시 말해, 중앙 처리 장치 및 그래픽 처리 장치를 포함한다.

[0019] '컴파일러(Compiler) 또는 컴파일(Compile)'은 고급 언어로 작성된 어플리케이션을 그와 의미적으로 동등하며, 단말에서 즉시 실행될 수 있는 형태의 목적 어플리케이션으로 바꾸어 주는 프로그램 또는 과정을 의미한다. 즉 단말에서 특정 어플리케이션이 수행되기 위해서 단말이 직접 이해할 수 있는 언어로 바꿔야 한다. 그리고 이러한 일을 하는 프로그램이 컴파일러이다.

[0020] 컴파일을 하기 위해 입력되는 어플리케이션은 원시 어플리케이션이라 하고, 이 어플리케이션을 기술한 언어는 소스 코드(Source Code)라고 한다. 그리고 컴파일되어 출력되는 어플리케이션을 목적 어플리케이션이라 하고, 목적 어플리케이션을 기술한 언어가 목적 코드(Object Language 또는 Target Language)라 한다. 하나의 어플리케이션을 컴파일되어 목적 어플리케이션으로 변경되면, 원시 어플리케이션을 수정하지 않는 한 계속 반복해서 수행할 수 있다.

[0021] '중앙 처리 장치(CPU; Central Processing Unit)'는 단어 그대로 단말의 중앙에서 모든 데이터를 처리하는 장치라는 의미로, 사용자로부터 입력된 명령어를 해석, 연산한 다음 그 결과를 출력하는 기능을 수행한다. CPU의 가장 기본적인 역할은 연산/계산 작업이다.

[0022] '그래픽 처리 장치(GPU;Graphics Processing Unit)'는 단말의 그래픽을 담당하는 장치로, 3D 그래픽을 주로 처리한다. 이에 GPU가 함께 구성된 단말에서 그래픽과 관련된 연산이 수행될 때 CPU의 부담이 줄어들 수 있다. 또한 GPU는 대량의 행렬과 벡터를 처리할 수 있으므로, 이러한 연산을 많이 사용하는 어플리케이션들을 수행할 수 있다.

[0023] 이하 첨부된 도면을 참조하여 본 발명의 동작 원리를 상세히 설명한다. 하기에서 본 발명을 설명함에 있어 관련된 공지 기능 또는 구성에 대한 구체적인 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명을 생략할 것이다. 그리고 후술되는 용어들은 본 발명에서의 기능을 고려하여 정의된 용어들로서 이는 사용자, 운용자의 의도 또는 관례 등에 따라 달라질 수 있다. 그러므로 그 정의는 본 명세서 전반에 걸친 내용을 토대로 내려져야 할 것이다.

[0024] 도 1은 본 발명의 실시예에 따른 데이터 처리 시스템을 도시한 도면이다.

[0025] 도 1을 참조하면, 데이터 처리 시스템은 타겟 컴파일 시스템(TARGET Compile System), 수행 추정 코드 생성 시스템(Performance Estimation Code Generation System;110), 호스트 컴파일 시스템(Host Compile System)으로 구성된다.

[0026] TARGET Compile System은 성능을 예측하기 위한 S/W 코드를 TARGET Assembly Code를 생성하는데 사용된다. 다시 말해 TARGET Compile System은 Soft Ware가 실행될 Hard Ware인 CPU 또는 GPU에 맞는 Compiler를 이용하여 C 코드를 이용하여 Simulation Code 생성을 위한 Assembly Code를 생성한다. 이때 Compiler는 실제 개발에 사용될 Compiler를 사용할 수 있으며, 예로 GNU GCC(GNU Compiler Collection)를 들 수 있다.

[0027] Performance Estimation Code Generation System(110)은 TARGET Compile System에서 생성한 Assembly Code를 입력으로 받아 Simulation C 코드를 생성하는 기능을 담당한다. 그러기 위해 Performance Estimation Code Generation System(110)은 파서(Parser;115), 기본 블록 생성부(Generate Basic Block;120), 기본 블록 수행 추정부(Estimate Performance of basic block; 130), 심레벨 선택부(Sim Level;140), 기본 블록 변환부(Translate Basic Blocks Into SIM C Code;150)로 구성된다.

[0028] 파서(115)는 TARGET Compile System에서 처리된 Assembly 파일을 읽어, 시스템이 인식할 수 있는 Assembly Data Structure로 변환을 수행한다.

[0029] 기본 블록 생성부(120)는 파서(115)에서 변환된 Assembly Data Structure로부터 기본 블록(Basic Block)을 생성하기 위한 Entry Point와 Exit Point를 검색한다. 그리고 기본 블록 생성부(120)는 검색한 Entry Point와 Exit Point 중 동일한 Entry Point와 Exit Point를 갖는 Assembly Data Instruction을 그룹핑하여 기본 블록으로 생성한다. 여기서 Entry Point가 될 수 있는 곳은 함수의 시작 지점, Label, Branch 명령의 다음 Assembly Instruction 지점, Branch 명령의 목적 주소가 되는 지점이 될 수 있다. 그리고 Exit Point가 될 수 있는 곳을

Branch 명령이 있는 지점, Return 되는 지점, Program Counter Register를 조작하는 지점이 될 수 있다.

[0030] 기본 블록 수행 추정부(130)는 기본 블록 생성부(120)에서 생성된 기본 블록별로 예측 가능한 요소인 해당 Assembly Instruction의 Performance를 예측한다. 이때 예측 가능한 요소로 Cycles, Power 등이 있으며, Performance 측정 중 선택 가능한 옵션으로 CPU Execution Cycles, Cache Hit/Miss Count, Power Consumption 등 있다. 그리고 기본 블록 수행 추정부(130)는 기본 블록 별로 Instruction 실행을 위하여 필요한 Resource를 계산한다. 그리고 기본 블록 수행 추정부(130)는 계산된 Resource를 기본 블록을 구성하는 Performance_T 필드에 저장한다.

[0031] 심레벨 선택부(Sim Level;140)는 코드 분석을 통하여 Simulation의 레벨을 설정할 수 있다. 여기서 Simulation 중 선택 가능한 옵션으로 Branch에 영향을 주는 Assembly Instruction 만을 선택하여 Simulation을 하도록 하는 Control-Only 옵션, Cache Hit/Miss Count를 측정하기 위하여 Memory Access Address 계산에 영향을 주는 Assembly Instruction 만을 선택하여 Simulation을 하도록 하는 Cache-Effect 옵션, 모든 Assembly Instruction 이 수행되어 기능적으로 완전하도록 하는 ALL 옵션 등이 있을 수 있다. Control-Only 옵션은 Simulation을 위하여 기본적으로 선택되는 옵션이다. Cache-Effect 옵션은 Memory Read/Write 시에 Cache Effect를 예측하기 위하여 Cache Model에 입력을 생성하는 코드를 생성하도록 선택되는 옵션이다. Simulation의 레벨을 설정하는 방법에 대하여 도 2를 참조하여 설명한다.

[0032] 기본 블록 변환부(150)는 앞서 생성된 기본 블록을 변환하여 통해 Simulation C 코드(SIM C code)를 표 1과 같이 생성한다.

표 1

```
ldr r3, [sp, #4]
add r0, r1, r2
mul r2, r0, r1
sub r0, r0, r1
cmp r2, r3
beq .L4
```

→

```
// Performance
cycles += curBB.perf.cycles;
power += curBB.perf.power;

// Functionality
r3 = sp[ 4 ];

r2 = r0 * r1;

if( r2 == r3 )
  goto .L4
```

[0033]

[0034] 좀 더 상세히 설명하면, 기본 블록 변환부(150)는 Simulation Level에 따라 Assembly Instruction의 Functionality를 C 코드로 변환한다. 그리고 기본 블록 변환부(150)는 기본 블록이 호출될 때마다 Performance 코드를 작성한다.

[0035] Host Compiler System은 Simulation Code Generator를 통하여 생성된 C 코드를 Simulation이 수행되는 Host인 Hard Ware에서 실행 가능한 파일로 생성할 수 있다. 다시 말해 Host Compile System은 생성된 Simulation 코드와 Simulation Library, 다른 Simulation 코드를 컴파일하고 링크하여 실행 가능한 Simulation 파일을 생성한다. 이때 사용 가능한 Host Compiler로 GNU GCC를 예로 들 수 있다.

[0036] 도 2는 본 발명의 실시예에 따라 구성된 기본 블록을 이용하여 시뮬레이션 레벨을 설정하는 과정을 도시한 도면이다.

[0037] 도 2를 참조하면, 기본 블록(210a, 210b, 210c)은 적어도 하나의 Simulation으로 구성된다. 그리고 하나의 Simulation은 Entry Point와 Exit Point를 가진 연속된 코드들로 구성된다. 이에 기본 블록(210a, 210b, 210c)을 구성하는 각 instruction들은 연속적으로 실행될 수 있다. 따라서 기본 블록(210a, 210b, 210c)은 각 Assembly instruction과 마찬가지로 실행의 단위로 생각할 수 있다. 기본 블록(210a, 210b, 210c)을 이용하여 데이터 처리 시스템에서 Instruction 단위보다 큰 단위인 Basic Block Simulation 수행이 가능하다. 그리고

Simulation 시간의 단축이 가능하다.

[0038] 다음으로 기본 블록(210a, 210b, 210c)을 통해 Multi-level Simulation이 가능하다. 즉 코드 분석을 통하여 다양한 Level의 Simulation이 가능하다. 그러기 위해 각 기본 블록(210a, 210b, 210c)을 구성하는 Instruction들에 공통으로 적용된 Performance와 각 Instruction별 functionality(220a, 220b, 220c)가 Simulation을 수행하기 위해 추출된다. 이때 Simulation 목적에 따라 Basic Block의 시뮬레이션 레벨이 설정된다. 그리고 각 Instruction별로 추출된 Functionality 중에서 Simulation하고자 하는 레벨에 따라 코드 분석을 하여 Functionality(230a, 230b, 230c)가 선택된다.

[0039] 이와 같이 functionality를 선택하기 위해 각 기본 블록에 설정되는 Simulation Level은 다음과 같다.

[0040] -Level 1 : 가장 Minimal한 Simulation으로 Branch에 관련된 Instruction만 Simulation을 수행한다.

[0041] -Level 2 : Memory 접근 주소와 관련된 Instruction만 Simulation 수행하여 Cache Effect를 고려한 성능 측정이 예측을 가능하게 한다.

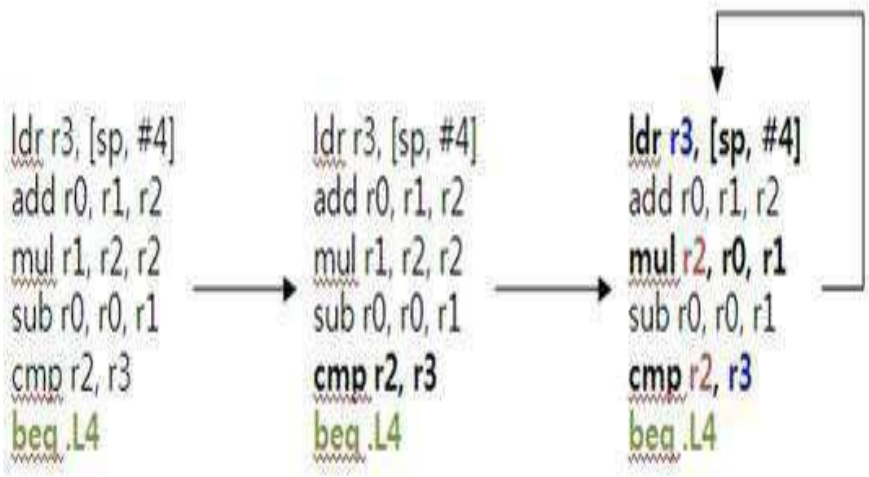
[0042] -Level 3 : Target 이 되는 코드가 다른 소스 코드에서 호출 등으로 사용될 경우, Out에 관련된 Instruction의 Simulation을 수행한다.

[0043] -Level 4 : 모든 Functionality에 대한 Simulation을 수행한다.

[0044] 예를 들어 기본 블록이 Level 1로 설정된 경우, Simulation은 표 2와 같이 구현될 수 있다.

표 2

[0045]



[0046] Performance Estimation Code Generation System은 Level 1로 설정된 Basic Block을 구성하는 Instruction 중에서 Condition을 생성하는 Instruction을 찾는다. 그리고 Performance Estimation Code Generation System은 계산을 위해 사용되는 Register를 목적 Register로 사용하는 Instruction을 찾는다. 마지막으로 Host Compiler System은 찾은 Instruction을 이용하여 Simulation을 수행한다.

[0047] 이외에도 설정된 Level에 따라 Simulation 수행시, 기본적인 CPU Execution Time 예측을 위해 Branch 조건 계산을 위한 Dependency에 해당하는 instructions가 simulation될 수 있다. 또한 Cache Effect 를 고려하기 위해 Memory Access Address 관련된 instructions이 추가로 simulation될 수 있다. 또는 다른 함수들과 함께 Simulation 등을 위하여, 모든 기능들이 C 코드로 구현되어 simulation될 수 있다. 이와 같이 Simulation 목적에 따라 Basic Block의 시뮬레이션 레벨이 설정됨으로써, Functionality가 선택적으로 Simulation될 수 있다. 따라서 Simulation 시간의 단축이 가능하다.

[0048] 도 3은 본 발명의 실시예에 따라 데이터 시뮬레이션 방법을 도시한 도면이다.

[0049] 도 3을 참조하면, 데이터 처리 시스템은 310단계에서 시뮬레이션이 수행될 어플리케이션 코드를 입력한다. 그리고 데이터 처리 시스템은 320단계에서 입력된 어플리케이션 코드를 이용하여 어셈블리 데이터 구성으로 변환한

다. 다음으로 데이터 처리 시스템은 330단계에서 변환된 데이터 구성으로부터 기본 블록을 생성한다. 좀 더 상세히 설명하면, 데이터 처리 시스템은 Assembly Data Structure로부터 기본 블록(Basic Block)을 생성하기 위한 Entry Point와 Exit Point를 검색한다. 그리고 데이터 처리 시스템은 검색한 Entry Point와 Exit Point 중 동일한 Entry Point와 Exit Point를 갖는 Assembly Data Instruction을 그룹핑하여 기본 블록으로 생성한다.

[0050] 데이터 처리 시스템은 340단계에서 기본 블록의 수행 요소를 예측한다. 다음으로 데이터 처리 시스템은 350단계에서 예측한 수행 요소로 시뮬레이션 레벨을 설정한다. 이때 데이터 처리 시스템은 Simulation을 수행하기 위해 기본 블록을 구성하는 Instruction들에 공통으로 적용된 Performance와 각 Instruction별 functionality를 추출한다. 이때 Simulation 목적에 따라 Basic Block의 시뮬레이션 레벨이 설정된다. 그리고 데이터 처리 시스템은 각 Instruction별로 추출된 Functionality 중에서 Simulation하고자 하는 레벨에 따라 코드 분석을 하여 Simulation을 수행할 Functionality를 선택한다.

[0051] 데이터 처리 시스템은 360단계에서 설정된 시뮬레이션 레벨에 따른 시뮬레이션 코드를 생성한다. 즉 데이터 처리 시스템은 선택된 Functionality를 C 코드로 변환한다. 그리고 데이터 처리 시스템은 기본 블록이 호출될 때마다 Performance 코드를 작성한다. 다음으로 데이터 처리 시스템은 변환된 C 코드를 Simulation이 수행되는 Host인 Hard Ware에서 실행 가능한 파일로 생성한다.

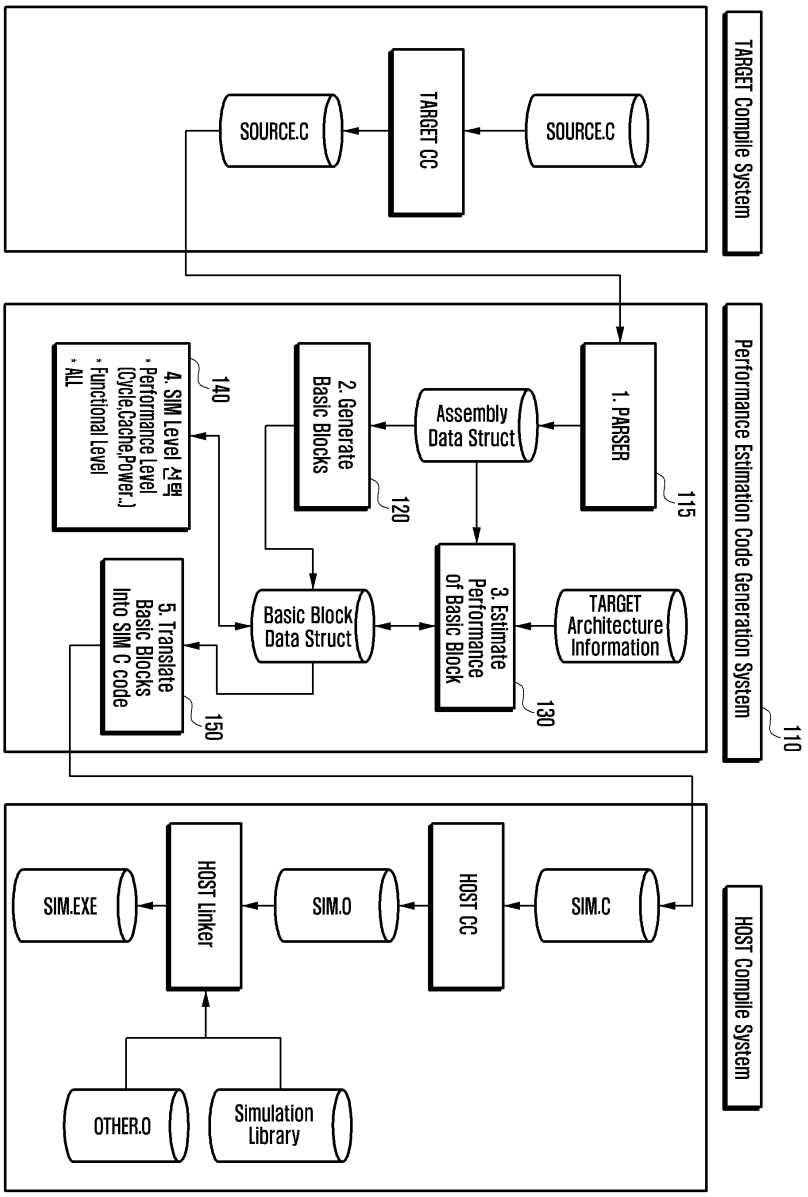
[0052] 데이터 처리 시스템은 370단계에서 시뮬레이션을 수행할 해당 하드웨어에서 처리하기 위한 컴파일을 수행한다. 마지막으로 데이터 처리 시스템은 380단계에서 해당 하드웨어를 통해 시뮬레이션을 수행한다.

[0053] 이러한 과정들을 통해 Simulation이 Instruction 단위에서 기본 블록 단위로 수행되기 때문에 시뮬레이션 수행 시간이 단축된다. 또한 기본 블록 단위별로 설정된 레벨에 따라 Simulation이 수행됨으로써, Instruction별로 수행되던 시뮬레이션 수행 시간이 단축된다. 그리고 기본 불록 단위에서 Simulation하고자 하는 레벨에 따라 선택적인 Simulation 코드가 생성됨으로써, Simulation 수행 시간이 단축된다.

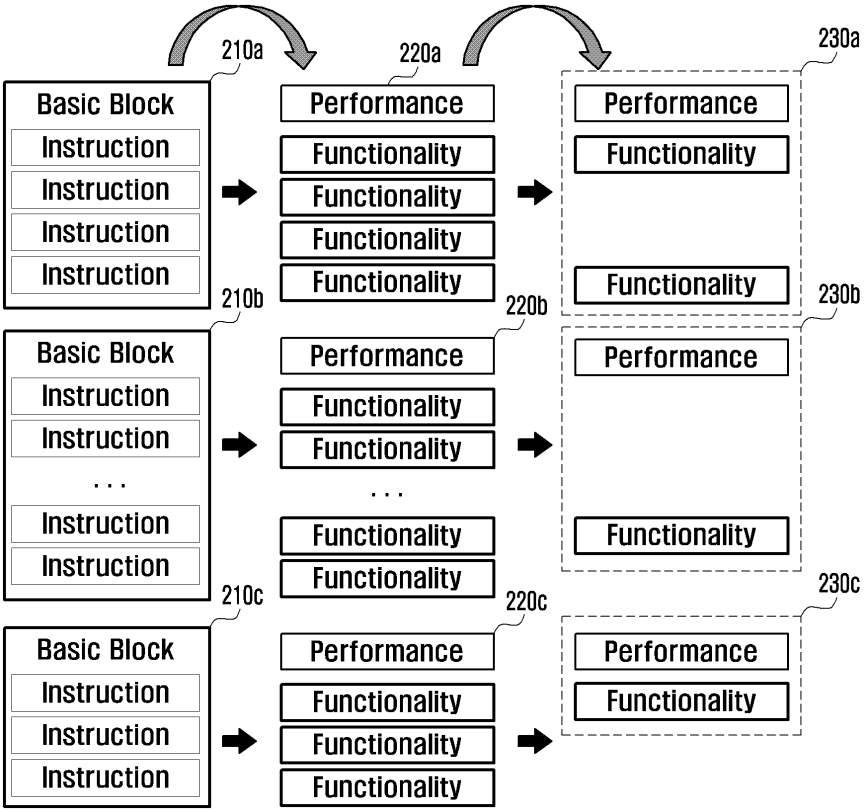
[0054] 한편 본 발명의 상세한 설명에서는 구체적인 실시예에 관해 설명하였으나, 본 발명의 범위에서 벗어나지 않는 한도 내에서 여러 가지 변형이 가능함은 물론이다. 그러므로 본 발명의 범위는 설명된 실시예에 국한되지 않으며, 후술되는 특허청구의 범위뿐만 아니라 이 특허청구의 범위와 균등한 것들에 의해 정해져야 한다.

도면

도면1


TARGET Compile System
SOURCE.C
TARGET CC
SOURCE.C
Performance Estimation Code Generation System
110
4. SIM Level 선택
* Performance Level (Cycle,Cache,Power..)
* Functional Level
* ALL
140
2. Generate Basic Blocks
120
Assembly Data Struct
1. PARSER
115
5. Translate Basic Blocks Into SIM C code
150
Basic Block Data Struct
3. Estimate Performance of Basic Block
130
TARGET Architecture Information
HOST Compile System
SIM.EXE
HOST Linker
SIM.O
HOST CC
SIM.C
OTHER.O
Simulation Library

도면2


210a
220a
230a
Basic Block
Instruction
Instruction
Instruction
Instruction
Performance
Functionality
Functionality
Functionality
Functionality
Performance
Functionality
Functionality
210b
220b
230b
Basic Block
Instruction
Instruction
. . .
Instruction
Instruction
Performance
Functionality
Functionality
. . .
Functionality
Functionality
Performance
Functionality
210c
220c
230c
Basic Block
Instruction
Instruction
Instruction
Performance
Functionality
Functionality
Functionality
Performance
Functionality

도면3


시 작
수행될 어플리케이션 코드 입력
310
어셈블리 데이터 구성 변환
320
변환된 데이터 구성으로부터 기본 블록 생성
330
기본 블록의 수행 요소 예측
340
예측한 수행 요소로 시뮬레이션 레벨 설정
350
설정된 시뮬레이션 레벨에 따른 시뮬레이션 코드 생성
360
해당 하드웨어에서 처리하기 위한 컴파일
370
시뮬레이션 수행
380
종 료