2013년 7월 전자공학회 논문지 제 50 권 제 7 호
Journal of The Institute of Electronics Engineers of Korea Vol. 50, NO. 7, July 2013 http://dx.doi.org/10.5573/ieek.2013.50.7.159

논문 2013-50-7-20

움직임 추정 전용 프로세서를 위한 효율적인 루프 가속기

(Efficient Loop Accelerator for Motion Estimation Specific Instruction-set Processor)

하 재 명*, 정 호 선*, 선 우 명 훈**

(Jae Myung Ha, Ho Sun Jung, and Myung Hoon Sunwoo©)

요 약

본 논문은 움직임 추정 전용 프로세서를 위한 효율적인 루프 가속기를 제안한다. 실제로 움직임 추정 알고리즘은 복잡하고 다양한 순환 명령어들을 포함하고 있다. 본 논문에서는 효율적인 하드웨어 루프 명령어들을 지원하기 위해서, 네 개의 루프 명령어와 그에 따른 하드웨어 구조를 소개한다. 검증 결과 제안된 루프 가속기가 early-termination을 이용한 움직임 추정 시 비교명령어와 조건부 점프명령어를 갖고 있는 전형적인 구현 방법과 비교했을 때 평균 명령어 사이클 수를 약 29% 줄일 수 있다는 것을 보여준다. 제안된 움직임 추정 전용 프로세서 루프 가속기는 프로그램 메모리의 접근 빈도를 상당히 줄일 수 있고, 전력 소모를 많이 절약할 수 있다. 따라서, 제안된 루프 가속기는 전력 소모가 적고, 유연한 움직임 추정에 적합하다.

Abstract

This paper proposes an efficient loop accelerator for a motion estimation specific instruction-set processor. ME algorithms in nature contain complex and multiple loop operations. To support efficient hardware (HW) loop operations, this paper introduces four loop instructions and their specific HW architecture. The simulation results show that the proposed loop accelerator can reduce about 29% average instruction cycles for ME early-termination schemes compared with typical implementation having a combination of compare and conditional jump instructions. The proposed loop accelerator of the motion estimation specific instruction-set processor can significantly reduce the number of program memory accesses and greatly save power consumption. Hence, it can be quite suitable for low power and flexible ME implementation.

Keywords : motion estimation/motion compensation (ME/MC), application-specific instruction-set processor (ASIP), low-power design, loop accelerator

Ⅰ. 서 론

비디오 코딩에서 움직임 추정은 화질과 코딩 시간에 직접적으로 영향을 미치기 때문에 매우 중요한 부분이다. MPEG-4, H.264/AVC[1], HEVC[2] 등과 같은 최근의 비디오 코딩에서 움직임 추정은 기존의 비디오 코딩 표준에 비해 코딩 효율과 영상의 질을 향상시키기 위해서 여러 개의 참조 프레임을 이용하는 방법과, 움직임 벡터의 정확도를 높이는 방법과 같은 코딩 기술과 다양한 블록 사이즈를 사용한다. 그 결과, 움직임 추정은 인코더 전체의 연산시간의 60%이상을 차지한다[3].

움직임 추정을 위한 Application Specific Integrated Circuit (ASIC) 솔루션은 성능과 필요 전력을 최적으로 만족시킬 수 있지만, 프로그램을 할 수 없어 유연성 면에서는 떨어진다. ASIC은 상대적으로 설계시간이 길고,

* 학생회원, ** 평생회원, 아주대학교 전자공학부
(Department of Electrical and Computer Engineering, Ajou University)
© Corresponding Author(E-mail: sunwoo@ajou.ac.kr)
※ This work was supported by the Mid-career Researcher Program through an NRF grant funded by the MEST (2013031132) and by the framework of international cooperation program managed by the National Research Foundation of Korea (2012-0030930).
접수일자: 2013년4월18일, 수정완료일: 2013년6월26일

다양한 탐색 알고리즘, 1/4 픽셀 해상도, 여러 개의 참조 프레임, 여러 개의 파티션 크기 등과 같은 특정한 변화를 다루는 것에 대해서 어려움이 있다. 따라서, 표준과 사양이 변할 때마다 ASIC을 새로 설계하고 제조해야하므로, Non-Recurring Engineering (NRE) cost와 time-to-market은 필연적으로 늘어나게 된다. 반면에 Digital Signal Processor (DSP)는 유연성이 높지만 필요성능을 내기 위해서 높은 클록 주파수와 더 큰 사이즈가 필요하고, 그로 인해 더 많은 전력을 소모하게 된다. 따라서, 최근 ASIC과 DSP의 장점을 절충하기 위해서 Application Specific Instruction-set Processor (ASIP)이 대안[4~5]으로 각광받고 있다.

최근, 저전력 설계를 요구하는 스마트 폰, 태블릿과 같은 모바일 기기의 시장이 크게 성장하였다. 따라서, System-on-Chip (SoC) 설계에서 ASIP기반의 구현이 주요 화두로 떠오르고 있다[5]. 움직임 추정에 관한 다양한 ASIP들이 제안되고 있지만[6~9], 이러한 ASIP들은 주로 효율적인 병렬 연산 구조에 초점을 두고 있다. 복잡한 움직임 추정 알고리즘을 효율적으로 지원하기 위해서는 병렬 연산뿐만 아니라 프로그램 제어에 대해서도 연구가 필요하다.

본 논문은 움직임 추정 전용 프로세서의 효율적인 루프 가속기에 대해서 제안한다. 움직임 추정 알고리즘은 보통 여러 개의 루프 연산으로 이루어져 있다. 게다가, early-termination을 이용한 움직임 추정 알고리즘은 추가적인 비교연산과 점프연산이 필요하다. 제안된 루프 명령어들과 그에 따른 구조들은 별도의 사이클 없이 여러 개의 내포된 루프를 지원할 수 있다. 대부분의 움직임 추정 알고리즘은 계산 복잡도를 줄이기 위해서 early-termination을 사용하고 있다. 이러한 early-termination 방법은 제안된 단일 명령어를 이용하여 구현될 수 있다. 본 논문은 다음과 같이 구성되어 있다. 제 Ⅱ 장에서는 기존의 움직임 추정 ASIP과 제안하는 효율적인 루프 가속기에 대해서 소개한다. 제 Ⅲ 장에서는 제안된 방법의 검증결과에 대해서 서술하였다. 마지막으로 제 Ⅳ 장에서는 본 논문의 결론을 간략하게 서술한다.

Ⅱ. 본 론

1. 기존의 ASIP에서의 하드웨어 루프

전역 탐색 알고리즘은 단순하고 매우 효과적인 움직임 추정 알고리즘으로, 탐색 영역에서 모든 후보 군을 비교한다. 전역 탐색 알고리즘은 규칙적인 데이터흐름과 단순한 제어, 그리고 높은 하드웨어 사용효율 때문에 하드웨어 구현에 선호된다. 하드웨어 가속기는 전역 탐색 알고리즘에 주로 사용된다. 하드웨어 가속기를 이용하기 위해서는 탐색 영역의 크기와 블록의 크기, 움직임 추정의 정밀성이 고정되어야 한다. 그러나 ASIP은 프로그램을 수정하여 이러한 움직임 추정 특징들을 변경할 수 있다. 전역 탐색과 고속 탐색의 차이점은 후보 탐색 지점이다. 고속 탐색은 후보 탐색 지점의 수를 줄여서 연산의 복잡성을 감소시킨다. 고속 탐색은 다이아몬드, 육각, 크로스 모양 등과 같은 다양한 탐색 패턴들을 이용하므로, ASIC 구현은 용이치 않다. 대조적으로 ASIP은 특별한 명령어를 사용하여 다양한 탐색 패턴과 블록사이즈, 해상도를 지원할 수 있다.

움직임 추정에서 기존의 ASIP들은 효율적인 병렬 연산 구조에 초점을 두고 있다. L. Sousa등은 4 x 4 블록에 대해 Sum of Absolute Difference16 (SAD16) 명령어를 제안했다[6]. 4 x 4 블록에 포함되어 있는 모든 픽셀의 집합은 하나의 SAD16 명령어로 계산된다. SAD16 명령어는 개선된 예측 지역 탐색 (Enhanced Predictive Zonal Search) 알고리즘을 지원한다. V. Chouliaras 등은 16 x 16과 8 x 8블록을 계산할 수 있는 다른 SAD 명령어[7]를 제안했다. 여기서, 정화소 움직임 추정 하드웨어는 탐색 지점에 따라서 SAD unit의 수를 변경할 수 있도록 구조변경이 가능하다. 움직임 추정에 대한 ASIP 템플릿은 [8]에 설명되어 있다. Very Long Instruction Word (VLIW) 프로세서를 기반으로 하는 [8]의 ASIP은 하드웨어 가속기를 사용하는 데 있어 저전력을 만족시킨다. [9]에서 제안하는 MESIP은 다양한 블록 움직임 추정에서 병렬 SAD 연산을 지원한다. 탐색 패턴에 기반을 둔 SAD 명령어와 변경 가능한 하드웨어 구조는 전역 탐색 알고리즘뿐만 아니라 다양한 패턴을 갖는 고속 탐색 알고리즘들도 다룰 수 있다.

하지만, 기존의 움직임 추정 ASIP[6~9]들은 하드웨어 루프 연산 같은 효율적인 프로그램 컨트롤 방법은 지원하지 않는다. 그림 1은 전형적인 움직임 추정 흐름과 내포된 FOR 루프처럼 반복되는 구조들을 보여준다.

내부 루프의 수는 움직임 추정 알고리즘의 복잡성에 비례한다. 예를 들면, 움직임 추정 알고리즘이 여러 개의 참조 프레임을 사용하는 방법, 다양한 블록과 탐색 패턴 등과 같은 여러 가지 특징들을 지원할 때 내부 루프의 수는 증가한다. 따라서, 추가의 오버헤드가 없는 하드웨어 루프 가속기는 움직임 추정 전용 프로세서에 필수적이다. 기존의 하드웨어 루프 가속기[10~11]는 루프

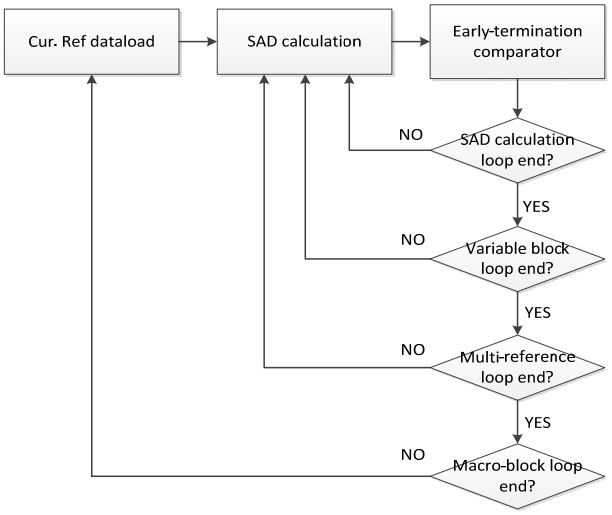

그림 1. 전형적인 움직임 추정 흐름
Fig. 1. Typical ME flow.

연산의 사이클 오버헤드에 대해 초점을 맞추고 있다. 제로-오버헤드 루프 제어 (Zero-Overhead Loop Control, ZOLC)는 [10]에 설명되어 있다. 제로-오버헤드 루프제어는 명령어 fetch 구간에서 루프의 오버헤드를 제거하기 위해서 사용되고, 복잡한 루프 구조에 적용될 수 있다. 스택 기반의 루프 제어기는 [11]에 나와 있다. 이것은 순차적인 내포된 루프를 보다 저전력으로 지원할 수 있다. 움직임 추정 알고리즘은 ASIP의 성능에 영향을 줄 수 있는 다른 중요한 요소를 지니고 있다. 대부분의 움직임 추정 알고리즘은 계산의 복잡성을 줄이기 위해서 early-termination 방법을 사용하고 있다[12~14].

Early-termination은 움직임 추정 알고리즘에서 실행속도를 높이기 위한 아주 중요한 요소이다. 일반적으로, 현재의 후보 탐색 지점의 SAD 값이 이전의 SAD 값보다 더 높아지면 탐색이 종료된다. 점프와 비교 명령어는 보통 early-termination을 위해서 사용된다. 이 방법을 사용하기 위해서, 각각의 비교 명령어는 현재의 SAD 값보다 높아지면 상태 비트를 참으로 설정한다. 점프 명령어는 이 상태 비트를 읽을 수 있고, 필요에 따라 실행 흐름을 변경할 수 있다. 따라서, 본 논문은 early-termination에 특정한 명령어를 사용함으로써 반복되는 비교, 점프 명령어들을 대체할 수 있는 효율적인 하드웨어 루프에 대해서 제안한다.

2. 제안하는 루프 가속기

표 1은 움직임 추정 프로그래밍에 보통 사용되는 루프의 두 가지 종류를 보여준다. 하나는 미리 정해진 반복횟수를 갖고 있는 FOR loop이다. 나머지 하나는 코드를 주어진 상태에 기반하여 반복적으로 수행되게 하는 DO WHOLE loop로 DO loop라고 한다. DO WHILE loop는 FOR loop의 특정한 반복 횟수 대신에 종료 상태가 있다.

표 1. 하드웨어 루프 명령어
Table 1. Hardware loop instructions.

종류	명령어	설명
FOR loop	*lset (iteration, start addr, end addr)*	FOR loop
	lstep (step)	Step size of loop index
	lcond (condition, threshold, cresult)	Early-termination condition
DO WHILE	*hrepeat (R, N, condition)*	Do while loop

이 절에서는 복잡한 움직임 추정 알고리즘을 효율적으로 수행하기 위한 MESIP의 특별한 루프 가속기에 대해서 소개한다.

(1) FOR Loop 가속기

위에서 말한 것과 같이 FOR loop는 반복 횟수를 갖고 있다. 다이아몬드, 십자, 육각모양의 탐색방법처럼 고정된 수의 후보 탐색 지점을 갖고 있는 대부분의 움직임 추정 알고리즘은 FOR loop에 의해 실행된다. 그러나, 일반적인 하드웨어 루프 구조는 early-termination을 효율적으로 지원하지 못하고, 움직임 추정 알고리즘에 따라서 다양한 조기 종료 조건들이 존재한다. UMHS[12]에서 두 개의 조기 종료 기준이 사용된다. 하나는 전역 종료 상태이고, 나머지 하나는 국부 종료 상태이다. 효율적인 참조 프레임 선택기 (Efficient Reference Frame Selector, ERFS) [13]은 조기종료에서 8 x 8 블록의 율-왜곡 비용 (Rate Distortion Cost)과 움직임 벡터의 변화를 사용한다. Adaptive and Fast Multi-Frame Selection Algorithm (AFMFSA) [14]의 탐색 과정은 Mean Absolute Deviation (MAD)에 의해 계산 된 블록 경계 조건에 따라서 종료된다. 고속탐색 알고리즘을 지원하기 위해서는 Three-Step Search (TSS)[15]와 Four-Step Search (4SS)[16]과 같이 반복횟수에 의한 루프 탈출방법과 다이아몬드 육각모양 탐색과 같이 조건에 의한 루프탈출방법이 있는데, 기존의 ASIP은 이 두 가지 방법 중 한 가지 방법만 지원하므로 이러한 다양한 비교 상태를 다룰 수 없지만 제안하는 ASIP은 두 가지 방법 모두를 지원 가능하게 하였다. 추가로, 비교 및 조건부 점프 명령어는 early-

termination 조건들이 수행되는 루프에서 반복적으로 사용되기 때문에 프로세서 효율은 급격히 감소한다. MESIP은 HW loop 명령어를 비교, 결정, 점프 같은 early-termination 스텝들과 결합하였다. 제안하는 하드웨어 루프 명령어는 다음과 같다.

lset (iteration, start addr, end addr)

*lset*명령어는 FOR loop와 *iteration*, *start addr*, *end addr*로 이루어진 *lset*을 지원한다. *iteration*은 반복되는 루프의 횟수를 나타낸다. *start addr*와 *end addr*은 각각 루프의 시작 주소와 종료 주소를 뜻한다.

lstep (step)

lcond (condition, threshold, cresult)

lstep 명령어는 루프 인덱스의 스텝 사이즈를 결정한다. 루프 인덱스의 초기 스텝 사이즈는 기본적으로 1이다. *lcond* 명령어는 early-termination 방법을 사용한다. *condition*은 early-termination 조건을 나타내고, *threshold*는 early-termination에서 미리 정해진 한계치를 나타낸다. *cresult*는 현재 처리하는 블록의 결과를 선택하는 데 사용된다. *cresult*는 현재 움직임 벡터, 현재 SAD등과 같은 여러 가지 경우를 갖고 있다.

그림 2는 FOR loop 명령어의 연산 흐름을 보여준다. 먼저 루프 인덱스가 *iteration*의 수를 넘어가게 되면 바로 루프에서 탈출하게 되고 정해진 *iteration* 숫자 내에

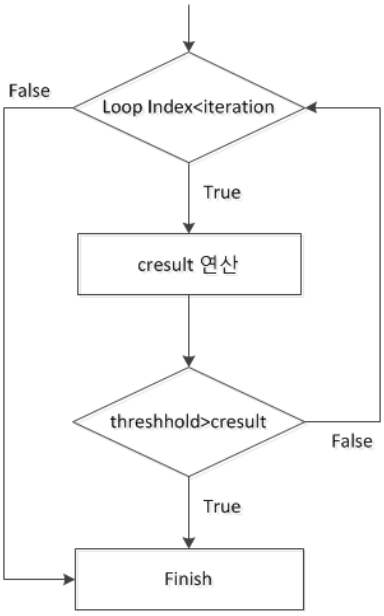

그림 2. FOR loop 명령어의 연산 흐름
Fig. 2. Operation flow of the FOR loop instruction.

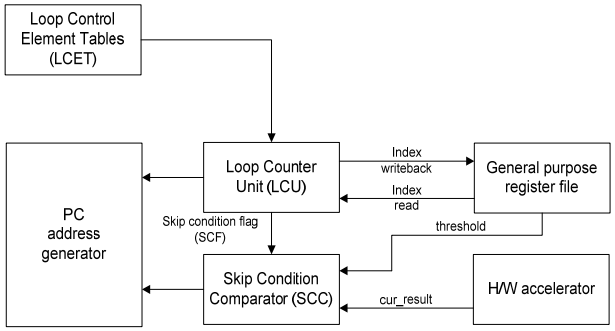

그림 3. 제안하는 하드웨어 루프 제어 구조
Fig. 3. Proposed HW loop control architecture.

서 *cresult* 값을 연산하여 이 값이 *threshold* 값보다 크면 다시 *iteration* 수를 참고하여 루프를 돌게 되고, *threshold* 값보다 작아지게 되면 *iteration* 횟수를 채우지 않더라도 바로 루프를 빠져나가게 된다.

그림 3은 제안하는 하드웨어 루프의 제어 구조에 대해서 보여준다. 먼저, lset 명령어는 시작 주소 및 종료 주소, 그리고 반복 횟수를 설정한다. 그 정보는 그림 2에서 루프 제어 요소 표 (Loop Control Element Table, LCET)에 저장된다. 루프 인덱스의 스텝 크기는 *lstep*에 의해서 제어된다. 루프 카운터 유닛 (Loop Counter Unit, LCU)은 *lstep*에서 설정된 *step*에 따라서 스텝 크기를 결정한다. *step*은 양의 정수 값으로 범위는 1부터 반복 횟수까지이다. 현재의 루프 인덱스는 범용 레지스터에 의해 결정된다. 생략 조건 플래그 (Skip Condition Flag, SCF)는 *lcond* 명령어에 의해서 1로 설정된다. 만약 선택된 *cresult*가 갱신되고, SCF가 설정되면, 생략 조건 비교기 (Skip Condition Comparator, SCC)는 범용 레지스터에 미리 정의된 한계치와 하드웨어 가속기의 결과 값을 비교한다. 만약 비교 상태가 만족되면, 현재의 LCET와 LCU는 초기화 된다. 동시에, PC의 다음 주소 값을 현재 루프의 마지막으로 설정한다. 그렇지 않으면, PC의 다음 주소 값은 프로그램 순서의 변경 없이 루프 안에서 다음 명령어로 이동한다.

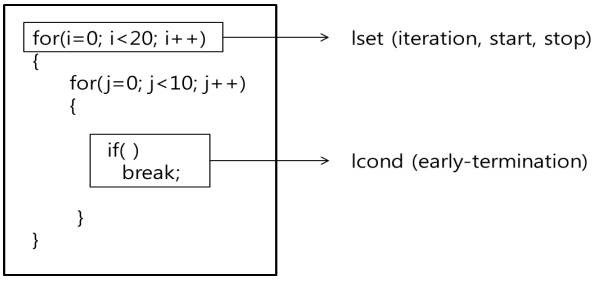

그림 4. FOR loop 명령어
Fig. 4. FOR loop instruction.

그림 4는 FOR loop 명령어의 예시를 나타낸 것으로 그림과 같이 *lset*은 여러 개의 내포된 하드웨어 루프를 지원하고 *lcond*은 다양한 early-termination을 지원한다. 기존의 명령어들이 early-termination을 지원하기 위해 점프와 비교 명령어를 여러 번 사용하지만, 제안하는 루프 명령어는 한 개의 명령어로 early-termination을 지원할 수 있어 명령어 사이클 수를 줄일 수 있는 이점이 있다.

(2) DO WHILE 루프 가속기

FOR 루프와는 대조적으로, DO WHILE 루프는 특정한 반복 횟수 대신 루프 탈출 조건이 있다. TSS, 4SS와 UMHS[5]의 local refinement와 같은 계층적 움직임 추정 알고리즘에서 루프 탈출 조건은 현재 탐색의 중간 지점이 최적의 포인트가 되는 것이므로 반복횟수는 고정되어 있지 않다. 따라서 특별한 루프 명령어 *hrepeat*을 제안한다.

hrepeat (R, N, condition)

*R*은 반복횟수를 의미하고, LCET의 *iteration*을 위해 사용된다. *N*은 반복된 명령어의 횟수를 의미하고, 루프의 시작 주소와 종료 주소는 *hrepeat*명령어에서 각각 대응 되는 주소에 의해 계산된다. 만약 *R*이 0이면, *condition*에 의해서 루프 탈출이 정해진다.

그림 5는 *hrepeat* 명령어의 연산 흐름을 보여준다. 만약 *iteration*이 0이라면, 임시 결과값과 현재 결과값이 같아질 때까지 다음 *N* 명령어들이 반복된다. 임시 결과값은 루프 탈출 상태를 계산한 후에 현재 결과값에 의해 갱신된다. *R*이 0이 아니면, 다음 *N* 명령어는 반복 횟수만큼 반복된다. 이 *hrepeat* 명령어는 TSS, 4SS등과 같은 계층적 탐색 알고리즘에 적합하다.

DO WHILE 루프에서 Infinte Loop Flag (ILF)는

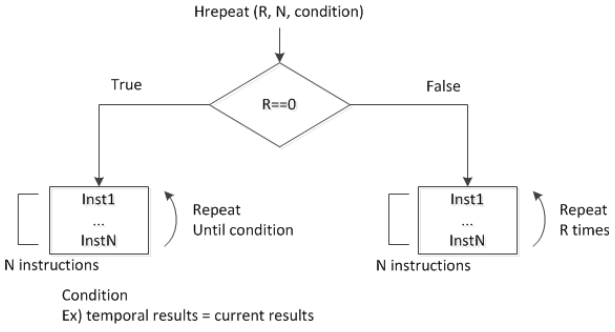

그림 5. *hrepeat* 명령어의 연산 흐름
Fig. 5. Operation flow of the *hrepeat* instruction.

```
int n=0;
if R=0
  {
     do
     {
     }while(condition)
  }
else
  {
     do
     {
        n++
     }while(n<R)
  }
```

→ hrepeat (R, N, condition)

그림 6. *hrepeat* 명령어
Fig. 6. *hrepeat* instruction.

*hrepeat*명령어에 의해서 1로 설정된다. 만약 ILF가 1이라면, LCET에서 *iteration*은 무시되고, 루프 탈출 조건은 현재의 루프 카운터에서 특수한 조건으로 변경된다. 임시 결과 값과 현재 결과 값이 같을 때 특수 한 조건은 참이 된다. 그렇지 않으면, LCET의 각각의 영역이 *hrepeat* 명령어의 오퍼랜드에 의해서 정해진다.

그림 6은 *hrepeat* 명령어의 예시를 나타낸 것으로, 그림 4의 연산흐름과 같이 R=0일 때와 0이 아닐 때로 나누어지며, R이 0일 때 조건들을 설정하여 *for loop*명령어와 같이 다양한 early-termination을 지원하고, 명령어 사이클 수도 줄일 수 있다.

Ⅲ. 실 험

제안하는 하드웨어 루프는 MESIP[9]에 포함되어 있다. 하드웨어 루프 방법을 적용한 MESIP은 Verilog HDL을 이용하여 설계되었고, LISA architecture description language[17]에 기반을 둔 Synopsys 사의 Processor Designer로 검증하였다.

전역 탐색과 다이아몬드 탐색[18], UMHS 탐색[12]에서 몇몇 중요한 블록들을 제안된 하드웨어 루프 명령어를 사용하는 경우와 사용하지 않는 경우를 각각 코딩하였다. 표 2에서 early-termination을 적용하여 각각의 알고리즘에 대해서 명령어 사이클 수를 나타내었다. 검증은 Full HD영상인 sun flower와 CIF 영상인 for man에 대해서 이루어졌다. 현재의 SAD 값이 특정한 threshold 값(이 실험에서는 500으로 설정) 이하일 때 Skip 조건은 만족하였다. 각각의 해상도에 따라 고속 전역탐색, 다이아몬드 탐색, UMHS 탐색에 대하여 각각 기존의 루프명령어와 제안하는 하드웨어 루프 명령어 사이클을 비교하였다. 제안된 하드웨어 루프 명령어는 여러 개의

표 2. 각 알고리즘에 따른 명령어 사이클
Table 2. Instruction cycles for each algorithm.

	시퀀스	해상도	고속전역 탐색	다이아몬드 탐색[16]	UMHS 탐색 [12]
기존의 루프 명령어 [10]	fore man	CIF	10960	5494	5544
제안하는 하드웨어 루프 명령어			6668	3935	3960
기존의 loop 명령어 [10]	sun flower	Full HD	96042	107798	109632
제안하는 하드웨어 루프 명령어			72501	78379	79296

루프 명령어를 하나의 명령어로 줄였기 때문에 기존의 루프 명령어[10]와 비교했을 때 평균 명령어 사이클 수를 약 29.26% 감소시킨다. 그러므로, 실시간 비디오 영상처리에서 클록 주파수를 줄일 수 있다. 게다가, 특정한 *hwloop* 명령어가 비교, 점프등과 같은 일반적인 루프 명령어들을 대신할 수 있기 때문에 프로그램 메모리의 접근 횟수 역시 감소된다. 클록 주파수와 프로그램 메모리의 접근 횟수를 감소시킴으로써 전력 소모를 상당히 줄일 수 있다.

만약 SAD16[6], SAD[7]등과 같은 다른 종류의 움직임 추정 전용 명령어들이 검증에 사용된다면, 식(1)을 사용하여 감소 된 명령어 사이클을 계산할 수 있다.

$$reduced\ instruction\ cycles = \sum_{i=1}^{\frac{SPP}{M}} (K+N_i) \cdot P_i \qquad (1)$$

여기서 SPP는 Search Points per Pattern을 의미하며, M은 명령어 당 SAD 연산의 횟수이다. K는 early-termination 조건을 수행하기 위한 명령어의 횟수를 나타낸다. N_i는 i번째 early-termination 조건에 의해 건너 뛴 명령어의 수를 나타내고 P_i는 i번째 early-termination 조건이 만족하는 경우의 수를 나타낸다. M은 보통 대부분의 움직임 추정 프로세서[6~8]에서 1로 쓰인다. 따라서, 감소된 명령어 사이클의 수는 SPP에 비례하여 증가한다. 그러므로, 제안하는 루프 가속기는 더욱 복잡한 탐색 패턴의 경우 명령어 사이클 수를 더욱 더 감소시킨다.

제안하는 루프 가속기의 효과를 명확하게 하기 위해서 IBM사의 90nm 라이브러리를 사용하여 Synopsys 사의 Design Compiler로 전력 소모 결과를 측정하였다. 실험은 early-termination 조건을 적용한 전역 탐색 알고리즘을 사용하여 Full HD영상에 대하여 실행되었다. 전압은 1.08V로 하였고, 전력 측정을 위해서 gate level switching activity를 사용하였다. 기존의 루프 제어 방법을 사용하는 MESIP의 경우 실시간 처리방식을 지원하기 위해 2.5920 mW/frame이 필요하지만 제안하는 하드웨어 루프를 적용한 MESIP의 경우에는 2.2403 mW/frame이 필요하다. 따라서 제안하는 하드웨어 루프는 저전력 모바일 기기에 아주 적합하다.

Ⅳ. 결 론

본 논문은 MESIP[9]에 대해 효율적인 루프 가속기를 제안한다. 제안하는 루프 가속기는 다양한 움직임 추정 알고리즘에 대해서 FOR 루프와 DO WHILE 루프를 모두 지원한다. 제안하는 *hwloop* 명령어는 프로그램을 제어하기 위한 오버헤드가 생기지만 추가의 사이클 없이 여러 개의 루프 연산을 지원할 수 있다. 다양한 고속탐색 알고리즘에서 사용하는 early-termination 조건들을 제안하는 명령어는 단일 명령어로 지원 가능하다. 성능 비교는 제안하는 루프 가속기가 계산하는 사이클의 횟수와 전력 소모를 상당히 줄일 수 있다는 것을 보여준다. 따라서 제안하는 루프 가속기는 저전력 움직임 추정 어플리케이션에 아주 적합하다. 저전력 MESIP은 스마트 폰, 태블릿 등과 같은 모바일 기기에 대해서 좋은 해결책이 될 것이다.

REFERENCES

[1] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, Joint Video Team (JVT) of ITU-T VCEG and ISO/IEC MPEG, Document JVT-G050, May 2003.

[2] T. Wiegand, W. J. Han, B. Bross, J. R. Ohm, and G. J. Sullivan, “WD1: Working Draft 1 of High-Efficiency Video Coding,” ITU-T SG16/WP3 Doc. JCTVC-C403, Guangzhou, China, Oct. 2010.

[3] Y. J. Wang, C. C. Cheng, and T. S. Chang, “A fast fractional pel motion estimation algorithm for H.264/MPEG-4 AVC,” in *Proc. IEEE International Symposium on Circuits and Systems*, May 2006, pp. 3974-3977.

[4] S. D. Kim and M. H. Sunwoo, “ASIP approach

for implementation of H.264/AVC, " *Journal of Signal Processing Systems*, vol. 50, no. 1, pp.53-67, Jan. 2008.

[5] V. R. Dodani, N. Kumar, U. Nanda, and K. Mahapatra, "Optimization of an application specific instruction set processor using application description language," in *Proc. IEEE Int. Conference on Industrial and Information Systems (ICIIS)*, July 2010, pp. 325-328.

[6] S. Momcilovic, N. Roma, and L. Sousa, "An ASIP approach for adaptive motion estimation on AVC," in *Proc. IEEE 3rd Conf. on Ph.D. Research in Microelectronics and Electronics*, July 2007, pp. 165 - 168.

[7] J. L. Nunez-Yanez, E. Hung, and V. A. Chouliaras, "A configurable and programmable motion estimation processor for the H.264 video codec," in *Proc. International Conference on Field Programmable Logic and Applications*, Sept. 2008, pp. 149 - 154.

[8] H. Peters, R. Sethuraman, A. Beric, P. Meuwissen, S. Balakrishnan, C. A. A. Pinto, W. Kruijtzer, F. Ernst, G. Alkadi, J. van Meerbergen, and G. de Haan, "Application specific instruction-set processor template for motion estimation in video applications, " *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 15, issue 4, pp. 508-527, April 2005.

[9] H. K. Eun, S. J. Hwang, M. H. Sunwoo, Y. H. Kim, and H. S. Kim, "Integer-pel Motion Estimation Specific Instructions and their Hardware Architecture for ASIP, " in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2011, pp.953-956.

[10] N. Kavvadias and S. Nikolaidis, "Elimination of overhead operations in complex loop structures for embedded microprocessors, " *IEEE Trans. on Computers*, vol. 57, no. 2, pp. 200-214, Feb. 2008.

[11] C. T. Wu, A. C. Hsieh, and T. T. Hwang, "Instruction buffering for nested loops in low-power design," *IEEE Trans. on VLSI Systems*, vol. 14, no. 7, pp. 780-784, July 2006.

[12] X. Xu and Y. He, "Improvements on fast motion estimation strategy for H.264/AVC, " *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 18, issue 3, pp. 285-293, March 2008

[13] T. Y. Kuo and H. J. Lu, "Efficient Reference Frame Selector for H.264," *IEEE Trans. On Circuits and Systems for Video Technology*, vol. 18, pp. 400-405, March, 2008.

[14] L. Shen, Z. Liu, Z. Zhang, and G. Wang, "An adaptive and fast multiframe selection algorithm for H.264 video coding," *IEEE Signal Processing Letters*, vol. 14, no. 11, pp. 836-839, Nov. 2007.

[15] L. Tao, Y. Su-ying, S. Zai-feng, and G. Peng, "An improved three-step search algorithm with zero detection and vector filter for motion estimation," *in Proc. IEEE International Conference on Computer Science and Software Engineering*, Vol 2, pp. 976-970, Dec. 2008.

[16] Lai-Man Po, Wing-Ching Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. On Circuits and Systems for Video Technology*, vol. 6, pp. 313-317, Jun. 1996.

[17] Synopsys Processor Designer Available: www.synopsys.com/Systems/BlockDesign/processorDev/Pages/default.aspx

[18] O. Ndili and T. Ogunfunmi, "Hardware-oriented modified diamond search for motion estimation in H.264/AVC, " in *Proc. IEEE International Conference on Image Processing (ICIP)*, Sept. 2010, pp. 749-752.

저 자 소 개

하 재 명(학생회원)

2012년 아주대학교 전자공학 학사 졸업.

2013년 현재 아주대학교 전자공학 석사 재학.

<주관심분야 : 멀티미디어 코덱, 멀티미디어 신호처리, 멀티미디어용 ASIP설계>

선 우 명 훈(평생회원)

1980년 서강대학교 전자공학 학사 졸업.

1982년 한국과학기술원 전자공학 석사 졸업.

1982년～1985년 한국전자통신연구소(ETRI) 연구원

1985년～1990년 Univ. of Texas at Austin 전자공학 박사.

1990년～1992년 Motorola, DSP Chip Division (미국)

1992년～현재 아주대학교 전자공학부 교수

2011년～현재 IEEE CASS Board of Governor

2011년～현재 IEEE Fellow Member

2012년～현재 대한전자공학회 반도체 소사이어티 회장

<주관심분야 : SoC 설계, VLSI Architecture, 통신 및 멀티미디어 ASIP 설계, 저전력 설계>

정 호 선(학생회원)

2012년 아주대학교 전자공학 학사 졸업.

2013년 현재 아주대학교 전자공학 석사 재학.

<주관심분야 : 멀티미디어 코덱, 멀티미디어 신호처리, 멀티미디어용 ASIP설계>