버스 전력 소모 최소를 위한 통합된 데이터 압축과 인코딩 기법

정도한
과학영재학교
jdh1311@hanmail.net
김태환°
서울대학교, 전기.컴퓨터공학부
tkim@ssl.snu.ac.kr

An Integrated Technique for Data Compression and Encoding for Bus Power Minimization

Dohan Jung
Busan Science Academy
Taewhan Kim°
School of EECS, Seoul National University

요 약

딥-서버마이크론 기술에서, 버스 상에서의 전력 소모를 최소화하는 것은 가장 중요한 설계 목표들 가운데 하나이다. 전력 소모를 줄이기 위해 일반적으로 사용되고 있는 효과적인 기법들은 근본적으로 데이터 압축 또는 데이터 인코딩을 이용하고 있지만 압축과 인코딩을 모두 사용한 기법은 현재까지 알려져 있지 않다. 본 논문은 버스에서의 데이터 전송 시 발생하는 전력소모량을 크로스톡 지연을 완전히 제거함과 동시에 최대한 줄이는 통합된 데이터 압축 및 인코딩 알고리즘을 제안한다. 전력 소모를 줄이는 문제를 셀프 천이와 크로스-커플된 천이 양에 대한 가중 합을 최소화하는 문제로 형상화하여 풀었으며, 이 과정에서 자주-인용 데이터에 기반한 압축과 셀프-쉴드 인코딩이라는 개념을 활용하였다. 벤치마크를 사용한 실험에서 우리는 제안한 방법을 사용하면, 기존의 순차적인 압축 및 인코딩 적용 방식보다 7.9%-39.4% 더 적은 전력 소모를 가짐을 알 수 있었다.

1. 서 론

여러 가지 전자기기의 시장이 넓어지고 있는 현실에서, 그 중 아주 놀라운 것이 모바일 기기의 발달이다. 모바일 기기란 휴대하기 편리한 기기들을 말하는 것으로서 휴대용 전화기, PDA 등을 예로 들 수 있다. 이런 기기들은 과거에는 그렇지 않던 것이 점점 발달하여 요즈음에 들어서는 그 작고 휴대하기 편리한 기기로 엄청나게 많은 일들을 할 수 있게 되었다. 예를 들어, 휴대용 전화기로 인터넷이나 게임을 즐기는 등의 일이다. 이렇게 모바일 기기가 편리해지고 있다. 하지만 이 모바일 기기에서 중요하게 고려되어야 할 점들이 있다. 바로 메모리 용량 문제와 속도, 배터리의 문제 등이다. 본 논문에서는 그 중에서 배터리의 문제에 대해 따져보기로 하자. 배터리의 경우보통의 가전제품과 같은 경우에는 커다란 배터리를 달아 훨씬 오래 가게 할 수 있지만 모바일 기기는 그 특성상 큰 배터리를 달지 못한다. 그러므로 결국 이 문제에 대해서는 성능을 약간 떨어지게 하여 배터리의 소비전력을 줄인다든 지와 같이 근본적으로 배터리의 소비전력을 감소시키는 방향으로 애듭지어지는 것이다.

소비전력을 감소시키려는 노력은 기기의 여러 부분에서 행해지고 있지만 본 논문에서는 특히 데이터가 이동하는 길인 버스 상에서의 소비전력 감소에 관해 다뤄보겠다.

이 부분에 관해서는 이미 2가지 방향으로 연구가 되어 있다. 근본적인 데이터 압축을 통해 버스를 지나다니는 데이터의 양을 줄이는 방법[1]과 데이터 인코딩을 통해 버스 상에서 데이터 전송 시 발생하는 소비전력 외에 인접 연결선에서 데이터 전송의 간섭으로 생기는 크로스톡 지연 (crosstalk delay)을 없애는 방법[2]들이 그것이다.

데이터 압축의 경우엔 자주-인용 데이터에 기반을 하는 방법으로써 가장 빈번하게 사용되는 명령어들을 짧은 비트수의 다른 이진 비트 열로 나타내게 하여 압축의 효과를 내는 것이다.

데이터 인코딩의 경우에는 커플링(coupling) 현상을 최대한 줄이는 방법을 강구하고 있다. 커플링 현상은 인접한 두 버스에 서로 다른 신호가 전달되기 시작할 때 일어나는 현상을 말한다. 이 때 특히 다음 그림 1과 같은 경우에는 일반 커플링 현상으로 인해 소비되는 전력보다 훨씬 많은 전력이 소비된다. 그래서 데이터 인코딩의 경

우 이런 경우는 없애며 일반 커플링 현상은 최대한 줄이는 방법을 사용한다. 데이터 인코딩은 다음과 같은 방법으로 한다.[2]

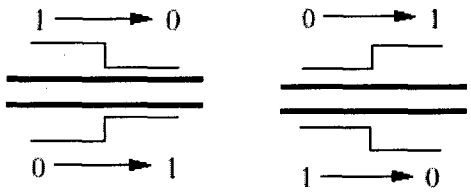
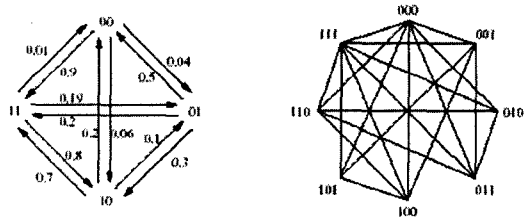


그림 1. 두 라인 모두 바뀌며 다른 신호를 나타내는 경우

(Step 1) 데이터에 대한 확률 그래프와 모든 클릭(clique)들을 구한다: 확률그래프란 어떤 신호에서 다른 어떤 신호로 변하는 확률을 말하는 그래프이며 클릭은 인코딩시킬 비트 열 중 크로스톡이 생기지 않는 것들끼리 이은 코드워드 그래프(code-word graph)에서 노드(node)들 사이에 모두 에지(edge)가 존재하는 부그래프(subgraph)들을 말한다.



(a) transition probability graph (b) codeword graph

그림 2. 확률 그래프와 코드-워드 그래프의 예시

(Step 2) Greedy하게 데이터 인코딩 후 다음 식에 의해 $\hat{Z}$를 구한다:

$$\hat{Z} = \sum_{\forall arc\ (c_i,c_j)\in G_{en}} p(c_i,c_j)\cdot w(f(c_i),f(c_j))$$

여기서 $p(c_i,c_j)$는 확률그래프에서의 가중치이고 $w(f(c_i),f(c_j))$는 다음 식에 의해 구해진다.

$$Z = X + \gamma \cdot Y.$$

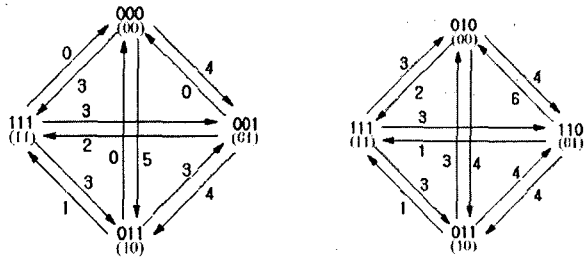
여기서 X는 셀프 천이에 관련한 값이며 Y는 크로스-커플된 천이에 관련한 값이며 γ는 capacitance ratio이다.

(Step 3) 매핑가능한 클릭들을 이용해 인코딩 시키는 비트 열을 변화시켜 주면서 $\hat{Z}$값을 줄인다. 이때 계속 반복을 하며 갱신시켜주다가 더 이상 $\hat{Z}$값이 줄지 않을 때 종료한다.

이렇게 두 가지 방향의 연구들이 있었으나 아직 압축, 인코딩을 모두 사용한 방법은 알려져 있지 않다. 본 연구에서는 그 두 가지 방법을 모두 사용한 알고리즘을 제안한다.

2. 모티베이팅(Motivating) 예제

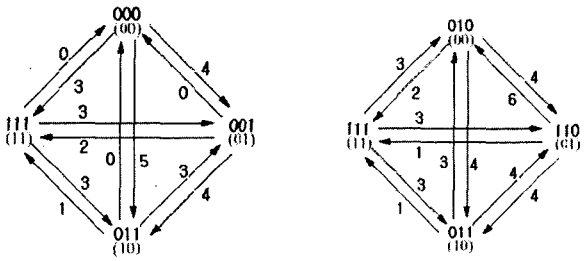
표 1은 모티베이팅 예제를 나타낸 것으로써 6개의 명령어들을 이용하여 200개의 명령어들을 입력으로 가지는 예제



$\hat{Z}$ = 9.26656 $\hat{Z}$=8.08878

ㄱ) greedy 인코딩 결과 ㄴ) 최선의 인코딩 결과

(a) I1=01, I2=11, I3=01, M=00으로 압축(compression)



$\hat{Z}$ = 8.37436 $\hat{Z}$=7.08488

ㄱ) greedy 인코딩결과 ㄴ) 최선의 인코딩 결과

(b) I1=10, I2=11, I4=01, M=00으로 압축

그림 3. 압축, 인코딩 알고리즘을 합친 여러 경우의 결과

instructions	original code	no. of accesses
I1	10011100	46
I2	10011101	44
I3	01010101	42
I4	11011011	40
I5	11101001	18
I6	11001100	10

표 1. 모티베이팅 예제

이다. 이 데이터에 대해 기존에 알려져 있던 압축, 인코딩 기법을 합친 알고리즘을 2가지 압축 방법을 통해 적용시켜 보았다. 인코딩의 경우 당연히 최선의인코딩 경우가 소비전력을 많이 줄일 수 있고 데이터 압축을 하는 경우에는 기존에 알려져 있던 가장 많이 인용되는 데이터에 대해 최선의 맵핑을 하는 방법[1] 이외에 다르게 압축을 할지라도 인코딩한 후 더욱 많은 소비전력을 줄일 수 있다. 위 예제는 이 사실을 확실히 보여준다. 그림 3-(a)-ㄴ)은 기존의 방법으로 압축하고 기존의 방법으로 인코딩한 결과 값이며 그림 3-(b)-ㄴ)은 다른 방법으로 압축을 한 후 기존의 방법

으로 인코딩한 결과이다. 확실히 차이가 남을 알 수 있다. 본 연구에서 지향하는 결과의 형태는 그림 3-(b)-ㄴ)과 같은 형태이다.

3. 제안하는 알고리즘

우리의 제안된 기법은 세 단계로 구성된다.

(Step 1) 서론에서 소개된 기존의 데이터 압축 방법으로 입력된 데이터를 압축한다.[1]

(Step 2) 서론에서 소개된 기존의 데이터 인코딩 방법으로 첫 번째 단계에서 압축된 데이터를 인코딩하여 소비전력을 구한다.[2]

(Step 3) 기존의 압축 방법에 변화를 주어 다시 압축한 후 기존의 방법으로 인코딩하여 새롭게 나온 결과 값을 이전의 결과 값과 비교해 필요할 경우 갱신한다. 이 때 압축 방법의 변화에는 여러 가지 방법이 있다. 많은 명령어들을 모두 다른 방법으로 압축을 하여 확인 해 보면 가장 좋겠으나 시간이 많이 걸리므로 본 연구에서는 월등히 많이 접근되는 명령어들을 가지고 여러 가지 형태로 압축을 한다. 이때 명령어들의 특성을 이용하는데 그 특성에는 압축하지 않았을 경우 크로스-커플된 천이가 많이 발생할 소지가 있는 것, 다른 것 보다 월등히 많이 접근되는 명령어들 등은 우선적으로 압축한다는 것이다.

세 번째 단계는 필요할 때까지 다음 그림 4와 같은 과정으로 반복한다.

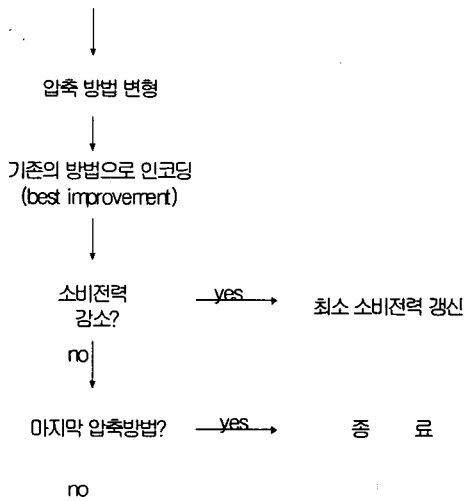


그림 4. Step 3의 흐름

위 그림 4에서 보는 바와 같이 더 이상 새로운 압축 형태가 없을 경우 종료하며 혹은 중간에 원하는 정도의 소비전력이 되었을 경우 종료한다.

4. 실험 결과 & 결론

우리는 앞서 설명한 방법을 벤치마킹 데이터를 이용하여 테스트해 보았다. capacitance ratio인 γ는 3으로 설정하

자료	기존방법		제안한 방법		소비전력 감소량
	data compression	data encoding	compression change	best improvement	
1	9.61941	8.4419	8.335	7.05363	16.4%
2	10.2578	8.71709	7.90967	6.34685	27.2%
3	9.50664	8.33581	7.84286	6.66049	20.1%
4	10.4787	9.29416	8.66073	7.21257	22.4%
5	6.81751	5.77052	6.35248	5.31515	7.9%
6	9.26656	8.08878	8.37436	7.08488	12.4%
7	9.81751	8.77052	6.32134	5.31542	39.4%
8	7.42345	5.77052	6.35248	4.31515	25.2%
9	8.32431	7.34543	6.83741	5.43234	26.0%
10	8.56321	7.45232	7.45651	6.01384	19.3%

표 2. 기존의 방법과 우리의 방법의 비교

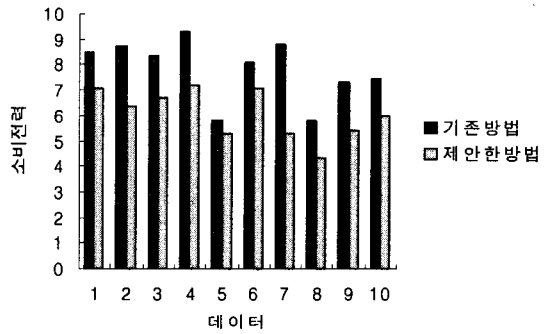


그림 5. 기존의 방법과 우리의 방법의 비교

였다. 표 2는 기존의 압축, 인코딩 기법을 이용 하였을 때의 소비전력과 제안한 알고리즘을 이용하였을 때의 소비전력을 테스트하여 결과를 낸 것이다. 진하게 표시된 부분이 그 방법을 사용하였을 때 최소 소비전력을 나타낸다. 기존의 방법을 사용하였을 때 보다 제안한 방법을 사용하였을 때 소비전력 감량은 percentage로 나타내었다.

그림 5는 기존, 제안한 방법의 best improvement를 그래프로 나타낸 것이다. 제안한 방법이 기존의 방법보다 훨씬 적은 전력을 소비 한다는 것을 알 수 있다. 비교에서 보듯이 기존의 방법 보다 7.9%-39.4% 더 적은 전력소모를 가짐을 알 수 있다.

감사의 글: 본 연구는 KAIST 과학영재 교육원의 R&E 프로그램 지원을 받았음.

참조문헌

[1] B. Lee and T. Kim, "Power Minimization for Instruction Accesses of Code Compression Environment" (internal manuscript, in preparation for publication)

[2] C.-G. Lyuh and T. Kim, "Low power bus encoding with crosstalk delay elimination", *ASIC/SOC*, 2002