논문 2004-41SD-7-10

패킷형 데이터를 위한 저전력 전송방법

(A Low-Power Bus Transmission Scheme for Packet-Type Data)

윤 명 철*

(Myungchul Yoon)

요 약

패킷형 데이터전송은 다량의 데이터가 연속적으로 전달되는 특성이 있다. 이와 같은 데이터를 버스를 통하여 전송할 때에 데이터의 전송순서는 버스의 전력소모에 영향을 주는 하나의 요소로서 작용한다. 본 논문에서는 패킷형 데이터를 전송할 때 데이터의 전송순서를 변화시켜 버스의 전이횟수를 줄임으로써 버스의 소비전력을 감소시키는 순서변환코딩 (Sequence-Switch Coding, SSC)을 제안하였다. 또한 SSC를 구현하는 알고리즘을 개발하였으며, 실험을 통하여 이들 알고리즘으로 기존의 널리 알려진 Bus-Invert Coding보다 우수한 성능을 얻을 수 있음을 보였다. SSC는 버스의 소비전력을 줄이는 하나의 방법이며, 이를 실현하는 알고리즘은 무수히 많다. 알고리즘의 다양성은 SSC가 갖는 하나의 장점으로써 회로설계자에게 버스의 소비전력과 회로의 구현부담 사이에서 넓은 범위에 걸쳐 절충할 수 있는 자유를 제공해 준다.

Abstract

Packet-type data transmission is characterized by the continuous transmission of massive data with relatively constant rate. In such transmission, the dynamic power consumed on buses is influenced by the sequence of transmitted data. A new coding scheme called Sequence-Switch Coding (SSC) is proposed in this paper. SSC reduces the number of bus transitions in the transmission of packet-type data by changing the sending order of the data. Some simple algorithms are presented, too. The simulation results show that SSC outperforms the well-known Bus-Invert Coding with these algorithms. SSC is not a specific algorithm but a method to reduce the number of bus-transitions. There could be lots of algorithms for realizing SSC. The variety of SSC algorithms provides circuit designers a wide range of trade-off between performance and circuit complexity.

Keywords: Sequence-Switch Coding, Low-Power Design, Transition-Reduction Scheme, Lagger Algorithm, VLSI System.

I. 서 론

반도체 기술의 발달과 함께 소자 및 각 모듈들의 크기는 점점 작아지는 반면 SoC의 경향에 따라 칩의 크기와 칩 안에 들어가는 모듈의 수는 점점 증가하고 있다. 이에 따라 이들을 연결하고 상호간의 데이터를 주고받기 위한 통신수단으로써 버스의 기능과 역할이 점점 더 중요한 요소로 작용하고 있다. 버스들은 보통 그에 연결되어 있는 많은 수의 모듈들에 의해 높은 부하 커패시턴스 값을 갖게 되므로 동작 시 많은 전력을 소모한다. 또한 최근의 VLSI 칩들은 버스가 칩 전체 면적의 반 이상을 차지하고 있으므로[1] 버스에 의한 전력소모가 칩의 전체 소비전력의 상당부분을 차지하게 된다. 따라서 저전력용 칩을 설계하기 위해서는 버스의 소비전력을 줄이는 것이 매우 중요한 요소가 된다. 버스에서 소비하는 전력은 다음과 같은 식으로 표현할 수 있다[2].

$$P_{bus} = \sum_{line} C_{load} V_{DD} N_{trans} \quad (1)$$

* 정회원, 고려대학교 전자컴퓨터공학과
(Department of Electronics & Computer Engineering, Korea University)

※ This work was supported by BrainKorea21 Program in year 2003.
접수일자: 2003년6월19일, 수정완료일: 2004년6월22일

여기서 C_{load} 는 부하 캐패시턴스 값이며 V_{DD} 는 버스의 동작전압이고, N_{trans} 는 1초에 버스가 전이하는 횟수이다. 버스의 전력소모를 줄이는 방법은 크게 두 가지 방식으로 나눌 수 있다. 하나는 C_{load} 또는 V_{DD}를 줄임으로서 버스 동작 시 소모되는 전력을 줄이는 방법이며, 다른 하나는 버스가 동작하는 횟수를 줄이는 방법이다. 전자의 방법으로는 버스 동작 시 전압변화의 폭을 조절하는 Reduced-Swing Bus[3][4]와 같은 방법이 있다. 후자의 방법은 코딩을 통하여 버스의 전이 횟수를 줄이는 방법으로써, Bus-Invert Coding[5], Gray Coding[6], T0 Coding[7] 등이 이러한 방법에 속한다.

기존의 코딩 방법들은 주로 독립적이고 단발적인 데이터전송을 가정하여 개발되었다. 독립적이고 단발적인 데이터전송이란, 마이크로프로세서와 같이 한 명령어에 의해 그에 필요한 몇 바이트만의 데이터를 입출력하는 방식을 뜻하며, 이러한 입출력 양식은 그동안 대부분의 응용분야에서 사용하는 방식이었다. 그러나 근래에 널리 퍼지고 있는 인터넷 및 멀티미디어 분야 등의 데이터 입출력 패턴을 살펴보면 기존의 입출력방식과는 근본적으로 다른 형태를 갖는다. 이 분야의 응용 칩들은 기본동작에 필요한 데이터로써 파일 또는 대량의 데이터를 입출력으로 요구한다. 즉 하나의 명령어가 실행되면 작게는 수 킬로바이트에서 많게는 수 기가바이트에 달하는 파일 또는 스트리밍 데이터를 입출력으로 사용한다 (앞으로 이 논문에서는 이러한 방식을 패킷형 데이터 입출력방식이라 부르기로 한다). 이러한 패킷형 데이터 입출력방식은 인터넷, 이동컴퓨팅 및 멀티미디어의 확산에 따라 점점 더 중요하고 대중적인 전송 형식으로 자리잡아가고 있다. 패킷형 데이터전송은 단발형 데이터전송이 가질 수 없는 데이터의 전송순서라는 자유도를 하나 더 갖고 있다. 즉 패킷형 데이터를 전송할 때, 데이터 전송순서는 고정되어 있는 것이 아니라, 사용자가 임의로 바꾸어 보내고 이를 수신단 측에서 다시 원래의 순서로 복구하여 사용할 수 있는, 사용자의 의도에 의하여 가변적으로 바뀔 수 있는 하나의 파라미터가 된다.

본 논문에서는 데이터의 전송 순서를 바꾸어 줌으로써 패킷형 데이터를 전송할 때 발생하는 버스의 전이 횟수를 줄여주는 순서변환코딩 (Sequence-Switch Cod-ing, SSC)을 제안하였다. SSC는 일반목적(general purpose)의 코딩방법으로써, 기존의 방법들이 이용하지 않았던 데이터의 전송순서를 버스의 전이횟수를 줄이는데 사용한 최초의 코딩방법이다.

Ⅱ. 순서 변환 코딩

패킷형 데이터의 전송은 대량의 데이터가 비교적 일정한 속도로 연속적으로 전송 되는 특성을 지닌다. 이때, 데이터의 전송순서는 버스의 전력소모에 큰 영향을 끼친다. 예를 들어 그림1-(a)와 같은 버스파형을 만들게 되는 8개의 데이터를 그 순서를 달리하여 전송하면 그림 1-(b)와 같이 아주 다른 버스파형을 얻게 된다. 이 두 파형을 비교해 보면 본래의 순서로 전송할 때는 총 32번의 전이가 발생하는 반면 다른 순서로 전송할 때는 23번의 전이가 발생한다. 즉, 데이터의 전송 순서를 변화시키는 것만으로도 버스 전력소모의 약 28%를 절약할 수 있다.

위의 예에서 볼 수 있는 바와 같이 버스의 전이횟수는 데이터의 전송순서에 따라 변화한다. SSC는 데이터 전송 시 원래의 전송순서에 비해 버스의 전이를 가능한 적게 발생시키는 새로운 전송순서를 찾아내어, 이 순서에 따라 바꾸어 전송하고 수신단에서는 코딩정보에 따라 원래의 순서를 복원하여 사용하게 함으로써, 전송선로에서 소모되는 전력을 줄이는 방법이다. SSC를 적용할 수 있는 응용분야는 다음과 같은 조건을 만족시켜야 한다. 첫째, 대부분의 동작 시 요구되는 입력 또는 출력이 최소한 2개 이상의 데이터를 연속적으로 전송하여야 한다. 둘째, 부호화/복호화에 따르는 지연시간을 수용할 수 있어야 한다. 인터넷 또는 멀티미디어 씨스템은 위의 두 조건을 잘 만족하는 예이다. 이들은 대부분의 경우 한 번 동작에 수십 킬로바이트에서 수백 메가바이트의 데이터를 연속적으로 전송하며, 순서변환 알고리즘에 의한 속도의 지연은 초기의 지연시간을 약간 증가시킬 뿐 그 이후에는 버퍼에 의해 스트리밍 과정에서 지연시간이 감추어지므로 문제가 되지 않는다. 이와 같이 위의 두 조건을 만족하는 응용분야에는 어느 곳에나 SSC가 사용될 수 있다.

일반적으로 전송하려는 전체 데이터에 대하여 최소한의 전이 횟수를 발생시키는 전송순서를 찾아내는 것은 불가능하며 또한 불필요하다. 먼저 이것은 NP[8]에 속하는 문제이기 때문이다(이 문제는 이미 NP로 잘 알려진 Traveling Salesman Problem[8]의 일종이다). 또 다른 문제는, 만일 최적의 전송순서를 쉽게 찾아낼 수 있다 하더라도 이 순서는 실제 응용에서는 복호시의 지연시간 및 구현부담 (implementation overhead) 때문에 사용될 수 없는 부적합한 순서가 될 수 있다는 점이다. 일례로 최악의 경우를 가정하여, 전송하려는 데이터의

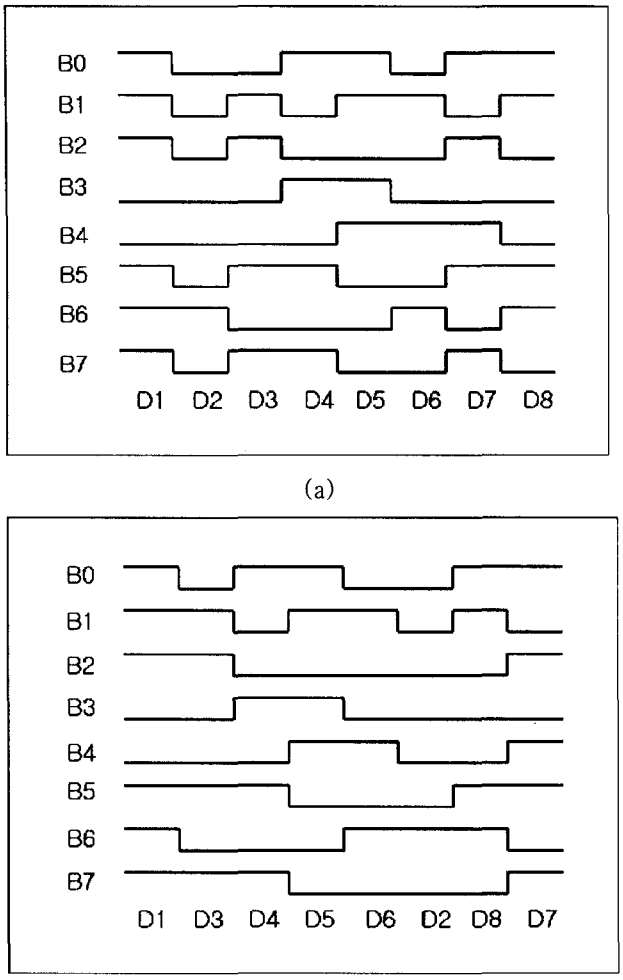

그림 1. 전송순서가 버스의 전이횟수에 미치는 영향 (a) 8개의 데이터를 원래의 순서대로 전송할 때의 버스의 전압파형 (b) 같은 데이터를 전송순서를 달리하여 보낼 때의 버스의 전압 파형

Fig. 1. Illustration of the effect of transmission sequence on bus transitions. The waveform of bus when the eight data are transmitted (a) in the original order. (b) in a different order.

첫 번째 데이터가 최적화 된 순서에는 제일 끝에 위치한다고 생각해 보자. 수신단에서 수신된 데이터를 사용하기 위해서는 일단 첫 번째 데이터가 도착해야 한다. 순서변환에 의해 첫 번째 데이터가 맨 끝에 전송되므로 복호기는 모든 데이터를 일단 버퍼에 저장하고 마지막 데이터를 수신한 후에야 복호화를 시작할 수 있다. 이 과정에서 수신데이터를 저장하기 위하여 매우 큰 버퍼가 요구되며 복호지연시간도 매우 길어지게 된다. VOD와 같이 매우 큰 데이터 전송을 요하는 응용분야에서 이것은 허용불가능한 일이다. 이런 이유로 전체데이터에 최적화된 전송순서를 구하는 것은 불필요한 일이다.

따라서, SSC는 구현부담이 적은 알고리즘을 사용하여 데이터전송 시에 원래의 전송순서에 비해 (최적은 아니더라도) 충분히 적은 수의 버스전이를 일으키는 전송순서를 빠르게 찾아내는 것을 목적으로 한다. 이러한 목적에 부합하는 전송순서는 무수히 많을 수 있고 따라서 어떻게 재배열할 것이냐에 따라 SSC 알고리즘의 수도 무수히 많을 수 있다. 원래보다 적은 전이횟수를 갖는 어떤 한 순서를 찾는 것은 어렵지 않다. 간단한 예로써, 전체의 데이터를 수많은 작은 부분으로 나누고 각 부분 부분에 대해서 최적의 배열을 찾아 나아가면 그 배열은 원래의 배열에 비해 충분히 적은 전이 횟수를 갖게 된다. 이때, 부분으로 구성하는 방법, 각 부분에 속한 데이터의 수, 각 부분에 적용되는 순서선택 방법, 등등에 따라 재배열의 결과와 그 성능이 달라지므로 무수히 많은 SSC 알고리즘이 존재하게 된다.

이러한 가능한 여러 가지 SSC 알고리즘을 분류하기 위하여, 본 논문에서는 알고리즘이 데이터의 순서를 정하기 위해 동시에 비교하는 데이터의 최대 개수를 기준으로 사용하였다. 즉, 만일 SSC 알고리즘이 어떠한 데이터의 전송 순서를 결정하기 위하여 한번에 최대한 m 개의 데이터를 비교에 사용하였다면 이것을 m-way SSC 라고 부르기로 한다. 일반적으로 m-way SSC 는 $m!$ 개의 서로 다른 순서가 존재하므로 이 중에서 최소의 버스전이를 일으키는 하나의 순서를 결정해야만 한다. m 이 커질수록 버스의 전이 회수를 더 많이 감소시킬 수 있을 것이다. 그러나 또한 그의 구현을 위하여 필요한 부담도 기하급수적으로 증가하게 된다. 따라서 $m!$ 개의 서로 다른 순서에 대해 전이횟수를 모두 구하여 비교함으로써 최적의 순서를 찾는 직접적인 방법으로 m-way SSC를 구현하는 것은 아주 작은 수의 m (즉, 2 또는 3)이 아니면 거의 불가능하며, 선험적 경험을 사용하여 적은 부담으로 구현할 수 있는 방법을 찾아야 할 것이다. 본 논문에서는 구현 부담을 고려하여 m 이 작은 경우의 SSC 알고리즘을 제안하고 그의 효율성을 실험을 통하여 검토해 보았다.

Ⅲ. 순서 변환 알고리즘

SSC 알고리즘을 고안할 때 가장 중요한 것 중의 하나는 변환된 순서를 복구하기 위해 필요한 코딩정보를 전송하는 방법이다. 본 논문에서는 Bus-Invert Coding (BI)과 같이 보조선을 이용하여 코딩정보를 전달하는 방법을 사용하였다. 필요한 보조선의 수는 알고리즘과 밀접한 관계가 있다. 따라서 칩 설계 시 허용할 수 있는

보조선의 수가 정해지면 그에 따라 선택할 수 있는 알고리즘도 제한되므로 허용 가능한 보조선의 수는 버스의 전력소모 목표치와 구현 부담 등을 고려하여 결정하여야 한다. 본 논문에서는 크게 2가지 방식 (Divide& Conquer와 Greedy 선택 방법)을 사용한 알고리즘을 제시하였다.

알고리즘들의 자세한 설명에 앞서, 알고리즘의 간략하고 명확한 기술을 위하여 몇 개의 용어 및 표현을 정의하기로 하자. 앞으로 이 논문에서는 데이터전송의 기본단위로써 워드(word)라는 용어를 사용하기로 한다. 즉 한 워드는 버스폭과 같은 비트 수로 구성된다. 한 싸이클 (cycle)은 연속적인 데이터 전송 시 하나의 워드를 전송하는데 드는 시간(주기)으로 정의한다. 두 이진 값 M 과 N 의 Hamming distance는 $H(M, N)$로 표시하기로 하자. 버스의 이진 값을 나타내는 B 를 보조선 (S-line) 까지를 포함 하도록 확대하여 $B^*=[B, S]$라 표시하기로 한다 (S 는 S-line의 이진 값). 마찬가지로 한 워드, W 를 확장한 값을 $W_X^*=[W,X]$ 라 하자, 여기서 X는 알고리즘의 선택결과에 따라 0 또는 1로 결정되는 값이다. 확장된 Hamming Distance

$$HX(W, B) = H(WX^*, B^*) \qquad (2)$$

는 한 워드 W 를 전송할 때 보조선의 전이를 포함한 버스의 전이 횟수를 나타낸다. 위의 정의를 사용하여 각 알고리즘을 제일 간단한 경우인 2-way SSC를 중심으로 하여 설명하고 m-way SSC로의 확장방법을 간략히 제시하였다.

1. Grouping Algorithm

이 알고리즘은 Divide&Conquer[8] 방식을 적용하여 입력 데이터들을 동일한 수의 워드를 갖는 소그룹으로 나누고 각 그룹마다 최소의 버스전이를 야기하는 순서를 찾는 방법이다. Grouping 알고리즘(GA)을 이용한 m-way SSC에 필요한 보조라인의 수는 다음과 같이 결정된다. m-way SSC의 경우 서로 다른 순서는 $m!$ 개 이고, m 개의 데이터를 보내는 동안 보조선 하나 당 표현 가능한 수는 2^m 이므로 n 개의 보조선을 이용한다면 보조선으로 표현 가능한 수는 $(2^m)^n = 2^{mn}$이 된다. 따라서 필요한 보조선의 수는

$$n = \log_2(m!)/m \qquad (3)$$

이 된다. 이 수식을 적용하면 1개의 보조선으로는 3-way GA 까지 가능하며 2개의 보조선으로는 8-way GA 까지 코딩 정보를 보낼 수 있다.

GA의 가장 간단한 경우인 2-way GA의 동작을 예를 들어 설명해 보면, 입력 데이터를 일단 2개씩 인접한 워드끼리 짝을 짓고 각 짝마다 보낼 순서를 결정한다. 이 결정은 바로 전 그룹의 순서 결정에 따라 나중에 보내게 되는 워드와의 Hamming distance를 계산하여, 그 값이 적은 워드를 먼저 보내도록 한다. 부호기는 2 싸이클 마다 새로운 워드짝을 받아들여 그들의 순서를 결정한다. 이 워드짝을 순서대로 X, Y 라 하면 부호기는 $H_0(X,B)$ 와 $H_1(Y,B)$ 를 비교하여 $H_0(Y,B)$ 가 작은 경우 뒤의 워드 Y 를 먼저 보내도록 하고 S 를 1로 만든다. 반대의 경우에는 원래의 순서대로 X 를 먼저 보내도록 하고 S 는 0으로 설정한다.

위의 알고리즘에 따른 복호화는 보조선의 값, S 를 이용하면 쉽게 이루어진다. 2개의 슬롯을 갖는 버퍼를 이용하여 S 가 0이면 수신되는 워드를 버퍼의 처음부터 저장하고 1 이면 반대 순서로 저장한 후 2 싸이클 마다 위 버퍼의 내용을 다음 단에 전달하여 버퍼를 비우고 다음 싸이클에 대비한다.

2-way GA는 위와 같이 매우 간단하게 구현이 되나. m-way GA (m>2) 인 경우에는 하드웨어 구현이 쉽지 않다. 하드웨어 구현이 가능한 GA는 1개의 보조선으로 코딩정보를 보낼 수 있는 3-way GA 까지가 현실적으로 가능하다고 생각되며, 4-way GA 이상의 경우에는 계산양의 기하급수적인 증가로 인하여 하드웨어에 의한 구현 보다는 소프트웨어에 의한 구현이 구현 부담을 줄이는 방법이 될 것이다. 이 알고리즘의 장점은 m 이 2 또는 3일 경우 비교적 간단하게 구현할 수 있으며 부호화나 복호화에 의한 지연시간이 최대 m 싸이클로써 적은 지연시간을 갖는 다는 점이다.

2. Lagger Algorithm

이 알고리즘은 매 싸이클 마다 m 개의 워드저장소 (word-pool)에서 Greedy 선택방식을[8] 사용하여 현재의 버스상태와 가장 적은 Hamming distance를 갖는 하나의 워드를 선택하여 전송하는 방법이다. 매 싸이클 마다 입력 데이터로부터 하나의 워드가 저장소에 입력되고 이 저장소에서 하나의 워드가 선택되어 전송되므로 저장소의 워드 수는 항상 m 으로 유지된다.

가장 간단한 2-way Lagger 알고리즘 (LA)의 부호기는 하나의 레지스터 (R_L)와 하나의 비교기, 그리고 두개의 Hamming distance 계산기로 구성된다. R_L 은 순서경합에서 탈락한 워드를 임시로 저장한다. 입력열

에서는 전송하려는 워드가 매 싸이클 마다 하나씩 입력되며 워드의 선택과 전송도 매 싸이클 마다 이루어진다. 입력되는 워드의 값을 I 라 하고 R_L 에 저장되어 있는 워드의 값을 L 이라 하면 매 싸이클 마다 다음 중 하나가 수행된다.

1. 만일 R_L 이 비어있으면 R_L 에 I 가 저장된다. 전송은 이루어지지 않으며 S 는 0으로 설정 된다.
2. 만일 R_L 이 비어 있지 않으나 입력 워드가 존재하지 않으면 R_L 값 L 이 전송되며 S 는 0으로 설정된다.
3. 위의 두 경우가 아니면, $H_1(I,B)$ 와 $H_0(L,B)$을 계산하여 비교한다. 만일 $H_1(I,B)$ 가 $H_0(L,B)$ 보다 작으면 I 가 전송되며 S 는 1 이 전송된다. 그렇지 않으면 L 이 전송되고 S 는 0 이 전송된다.

R_L 에 저장되었던 워드들을 래거(lagger)라고 부르기로 하자. S-line은 래거가 전송될 때는 항상 0의 값을 갖는다. 위 알고리즘에 의하여 변환된 데이터 열을 원래 열과 비교하여 보면 원래의 열에서 순서가 이동되어진 워드들은 모두 래거임을 발견할 수 있을 것이다. 또한 모든 래거를 바로 전 래거의 위치로 이전하면 원래의 순서가 복원됨을 발견한다. (처음의 래거는 배열의 맨 처음으로 이동한다.) 이 성질을 이용하면 2-way LA에 의해 코딩된 데이터를 버퍼를 사용하여 쉽게 복원할 수 있다. 버퍼의 첫 번째 슬롯은 래거슬롯으로써 래거들을 잠시 저장하는 장소로 이용하고, 2번째 슬롯부터는 래거가 아닌 워드들을 수신한 순서대로 저장한다. 래거가 들어오면 이 워드는 래거슬롯에 잠시 저장한 후 버퍼 내에 저장되어 있는 다른 워드들과 함께 다음단의 입력버퍼로 옮겨 버퍼를 비운 후 다음 싸이클을 준비하게 된다. 즉, S 가 1 이면 수신된 워드를 버퍼에 차례로 저장하고, S 가 0 이면 수신된 워드를 첫 번째 슬롯에 저장함과 동시에 여태까지 버퍼에 저장된 모든 워드를 다음단의 입력 버퍼로 전송하여 버퍼를 원래의 상태로 비워 놓는다. 위 과정을 반복하면 다음단의 입력버퍼에는 워드들이 원래의 순서로 복원되어 저장된다.

LA에서는 서로 경합하는 m 개의 워드들이 동적으로 결정되어 한 워드의 최대 지연시간을 예측할 수 없다. 즉 어떤 한 워드가 연속적으로 경합에서 탈락하여 입력 데이터의 끝까지 전송이 지연될 가능성이 존재하므로 최대 지연시간을 사전에 예측할 수 없다. 이것은 수신단에서 허용하기 곤란한 불확정성으로 위의 가능성에대비하여 매우 큰 버퍼를 준비하여야 한다. 이러한 단점을 없애기 위해서는 송신 시 한 워드의 송신지연(lagging)을 일정한 값 이내로 제한하여야 할 필요가 있다. 이는 부호기에 계수기를 첨가하여 해결할 수 있다. 계수기는 래거가 R_L 에 머물러 있는 싸이클 수를 저장하며 이 값이 사전에 결정된 값, k 에 도달하면 신호를 발생한다. 이 신호는 다음 싸이클에 계수기를 초기화함과 동시에 래거가 경합에서 선택되도록 만들어 한 워드가 k 번 이상 경쟁에서 지지 않도록 한다. 이와 같이 최대 송신지연이 k 를 넘지 않게 한 알고리즘을 k-bounded lagger algorithm (k-BL) 이라하고, 이러한 제한이 없는 알고리즘을 unbounded lagger algorithm (UL)이라 부르기로 한다. 두 알고리즘에 대해서 똑같은 복호기를 사용할 수 있으나, k-BL의 경우 최대 송신지연을 알고 있으므로 버퍼크기를 k+1 슬롯으로 줄일 수 있는 장점이 있다.

위에 설명한 2-way LA를 m-way로 확장하기 위해서는 약간의 변형이 필요하다. 우선 부호기에서는 m-1 개의 레지스터를 준비하여 0부터 m-2까지 차례로 번호를 붙인다 (R_0 에 가장 오래된 워드를 저장한다). 이 레지스터에 저장되어 있는 워드들과 새로이 입력되는 워드 (m-1 번째 워드)에 대하여 버스와의 Hamming distance를 구하여 가장 적은 값을 갖는 워드를 찾아내어 전송하고 보조선에는 선택된 레지스터의 번호를 전달한다. m-way k-BL을 위해서는 계수기를 각 레지스터마다 부착할 수도 있고 래거 (R_0)에만 부착할 수도 있다 전자의 경우 최대 송신지연을 k 싸이클로 제한할 수 있고 복호기의 버퍼크기를 (k+m) 슬롯으로 줄일 수 있는 장점이 있으나 부호기의 회로가 복잡해지고 구현부담이 커지는 단점이 있다. 따라서 확장성을 위해서는 구현부담이 적은 후자의 방법을 선택하는 것이 바람직하다고 생각된다. 후자의 경우 복호기는 (m-1)k+m 슬롯의 버퍼를 준비해야 하며 최대 송신지연은 (m-1)k 싸이클이 된다. 복호기는 버퍼와 m 개의 지시자(pointer)를 사용하여 수신된 S 값에 따라 해당되는 지시자가 지정하는 슬롯에 워드를 저장한다

LA은 GA에 비해 구현부담이 적으므로 m-way로의 확장성은 뛰어나나 매 싸이클 마다 선택된 워드의 번호를 전송하여야 하므로 보조선의 수가 많아지는 단점이 있다. m-way LA의 경우 $\log_2 m$ 개의 보조선이 필요하게 된다. 2개의 보조선으로는 4-way LA 까지가 가능하며 3개의 보조선으로는 8-way 까지 가능하다.

Ⅳ. 실 험

앞장에 설명한 알고리즘을 사용하여 SSC의 효율성을 씨뮬레이션을 통하여 분석해 보았다. 본 실험은 크게 두 가지의 목적으로 수행되었다. 첫째는 SSC의 실효성을 검증하기 위해서이다. SSC가 실제 설계에 적용되기 위해서는 성능이나 구현의 용이성 등등에서 최소한 기존의 방법과 비교할 만한 장점을 갖추어야한다. 이 점을 검증해 보기위하여 GA와 LA의 성능을 널리 알려진 일반목적의 코딩방법인 BI의 성능과 비교하여 그 차이점을 분석해 보았다. 둘째로는 SSC의 잠재력을 가늠해 보기 위해서이다. SSC를 실현하는 알고리즘은 무수히 많으며 그 성능은 순서변환 방법에 크게 영향을 받는다. m-way GA 나 m-way LA에서 m 값의 변화에 따른 버스전이의 감소율의 변화를 살펴봄으로써 SSC의 잠재적 성능이 어느 정도일지 예상해 본다. m의 변화에도 불구하고 성능의 변화가 미미하다면 알고리즘의 개발에 의한 SSC의 성능 개선도 한계가 있을 것이다, 반대로 그 변화가 크다면 알고리즘의 개발에 의한 SSC의 성능 개선은 그 잠재성이 매우 크다 볼 수 있다.

우선 실험을 위하여 패킷형 데이터를 주로 사용하는 응용분야인 멀티미디어와 인터넷 분야에서 많이 사용하는 형태의 파일들을 시험데이터로 선택하였다. 일단 일반적인 이진파일을 대표하여 랜덤파일을 하나의 샘플로 채택하였다. 멀티미디어 분야에 대한 데이터로는 음악파일과 영화 예고편들을 선정하였으며, 인터넷분야는 웹페이지의 주요 구성 요소인 HTML 원시파일 및 여러 가지 형식의 그림 파일들을 선택하였다. 그림 파일로써는 영상처리분야에서 시험영상으로 사용되는 그림들 중에서 선택하였는데, 이들은 MIT의 VisTex Database[9] 및 Caltech Image Database[10]와 Univ. of Washington의 Content-based Image Retrieval Data-base[11]에서 선택하였다. 각 형식에 대하여 3개씩의 대상을 선정하여 실험을 행하고 이들의 실험결과의 평균을 구하여 그 형식에 대한 최종 실험 결과로 사용하였다. 표 1에 실험을 위해 선택된 파일들과 이 파일들을 8-bit 16-bit, 32-bit, 64-bit 버스를 사용하여 원래순서로 전송할 때의 평균 전이회수를 나타내었다.

이 파일들을 사용하여 각각의 버스에서, 먼저 파일들을 원래대로 전송하고 다음 SSC의 각 알고리즘을 적용하여 전송하면서 버스의 전이횟수를 측정하여 전이횟수가 감소하는 비율을 계산하였다. 본 실험에서는 하드웨어로의 구현가능성을 고려하여 3-way GA 까지(G2, G3로 약칭함)와, 보조선 2개 이하로 구현 가능한 4-way LA (L2, L3, L4라 약칭함) 까지를 실험대상 알고리즘으로 선택하였다. 또한 기존의 방법과 비교를 위하여 동일한 파일들에 대하여 BI의 버스전이 감소율을 측정하여 그 성능을 비교, 검토하였다.

표 1. 실험용 데이터 파일 및 이 파일을 8-bit, 16-bit, 32-bit, 64-bit 버스를 통해 전송시 버스의 평균 전이회수 (전이수/cycle)

Table 1. The benchmark files for simulation and their average number of transitions per transmission when transmitted through 8-bit, 6-bit, 32-bit and 64-bit bus.

Format	Name	Data Size (Byte)	$< N_T >$			
			8	16	32	64
MP3	Fur Elise	707,076	3.9	7.9	15.5	31.7
	The Cup Of Life	4,279,212	3.8	7.8	15.7	31.7
	Under The Sea	4,691,510	3.8	7.7	15.5	31.0
MOVIE	Spiderman	9,372,379	3.8	7.6	15.3	30.6
	Beautiful Mind	5,544,025	3.7	7.5	15.0	30.0
	Treasure Planet	5,543,774	3.9	7.7	15.5	31.0
HTML	Intel	53,131	2.7	5.6	11.5	23.1
	MS	33,710	2.9	6.1	12.9	25.6
	Yahoo	36,318	2.9	5.9	11.8	24.0
GIF	Canal	244,208	4.0	8.0	15.5	31.5
	Lamp	45,479	4.0	7.9	15.7	31.7
	Paolina	243,233	4.0	7.9	15.8	31.7
PPM	Buildings	786,588	2.2	6.9	13.8	20.4
	Clouds	786,558	3.4	7.9	15.9	27.7
	Leaves	786,571	3.4	7.9	15.9	27.0
JPG	SW1	367,718	4.0	6.7	13.4	31.8
	SW2	313,866	4.0	7.8	15.7	31.8
	SW3	61,728	4.0	8.0	15.7	31.9
Binary	Random	4,096	4.0	8.1	15.9	32.2

우선, LA의 실용성을 알아보기 위하여 k 값의 변화에 따른 k-BL의 성능 변화를 UL의 성능과 비교 해 보았다. 앞서 설명한 바와 같이 UL은 복호기 구현 시 버퍼크기와 지연시간의 문제 때문에 실제적으로는 k-BL이 사용된다. 이때 발생하는 의문점은 k의 값을 얼마로 결정하여야 하는 것이다. k의 값이 작으면 충분한 성능을 얻을 수 없을 것이며 필요이상으로 크게 잡을 경우 구현시의 부담이 커지는 단점이 있다. 표 2에 L2, L3,

L4에 대하여 UL의 95% 와 99%을 성능을 갖게 되는 k의 실험값을 표시하였다. 대부분의 경우 k 가 10보다 작은 값에서 k-BL의 성능은 포화상태로 진입하게 된다. 이 사실은 매우 고무적으로써 버퍼의 크기가 크지 않더라도 k-BL로써 UL에 가까운 성능을 얻을 수 있음을 알 수 있다.

표 3은 각 알고리즘을사용하여 시험파일들을 전송하였을 때 버스의 전이 횟수가 감소하는 비율을 비교하여 나타내었다. 먼저 SSC 알고리즘간의 성능을 비교하여 보면, 동일한 m 일 경우 UL의 성능이 GA에 비해 우수한 것을 알 수 있다. 이것은, G2 또는 G3의 경우 너무 작은 수로 그룹을 나누게 되어 Divide&Conquer의 장점이 충분히 나타나지 않는 것이라 생각된다. 만일 단 하나의 보조선 만 허용되는 경우라면 L2에 비해 G3가 성능 면에서 근소한 차이로 우위에 있다고 할 수 있다. 그러나 일반적으로는 LA가 성능과 하드웨어 구현 면에서 GA에 비해 우수하다고 판명된다.

한편, SSC의 성능을 BI의 성능과 비교하여 보면 BI의 경우 응용분야에 따라 성능의 차이가 심한데 반하여 SSC의 경우에는 영향이 적음을 알 수 있다. 특히 BI는 HTML 파일 등의 문서파일 전송 시 성능이 급격히 떨어 지고 있다. 이것은 문서파일의 경우 문자, 숫자 등 문서를 구성하는 코드들이 ASCII 코드의 일부분에 국한되어 있기 때문이다. BI의 특성상 버스폭의 반 이상의 버스선에 전이가 발생할 때에 반전이 발생하여 전이를 감소시킬 수 있으나, 전송 데이터들이 코드맵의 일부분에 국한되어 있을 경우 버스의 어느 선(들)은 항상 일정한 값을 갖게 되므로, 반 이상의 버스선이 전이하게 될 확률이 훨씬 낮아지기 때문이다. 이러한 현상은 버스폭이 클수록 특히 심각하게 나타나게 되어 32-bit 이상의 버스에서 BI는 문서파일에 대하여 거의 그 효과를 보지 못한다. 반면 SSC의 경우 워드들 사이의 상대적인 Hamming distance를 비교하므로 코드가 국한되어 있는 경우에도 크게 영향을 받지 않는다. 따라서 버스폭이 커질수록 BI의 성능은 현저히 줄어드는데 반하여 SSC는 완만하게 감소함을 볼 수 있다.

이진파일이라 할 수 있는 다른 형식의 파일들에 대해서는 SSC는 BI와 비교할 만한 성능을 보인다. 표 3에서 보면, m-way GA나 LA의 성능은 m 값에 따라 크게 변함을 볼 수 있다. 낮은 m 값의 LA의 경우 이진파일을 전송할 때 버스의 전이를 감소시키는 성능은 BI에 비하여 열등하지만 m이 4이상일 경우에는 오히려 그를 능가한다. 위의 실험을 통하여 SSC가 버스의 전이 횟수를 감소시키는 효율은 순서를 결정하는 방법 (알고리즘)에 따라 크게 차이가 남을 알 수 있다. 이러한 성질은 설계자에게 넓은 범위에 걸쳐 버스의 소비전력감소율을 선택할 수 있는 융통성을 부여한다. 만일, 회로의 복잡성이 중요한 문제라면, BI가 SSC에 비해 적은 구현부담으로써 비교적 좋은 성능을 얻을 수 있다. 그러나 버스의 소비전력이 문제가 되어 BI의 성능으로 만족할 수 없다면, 설계자는 여러 SSC 알고리즘 중 필요한 정도의 성능을 갖는 알고리즘을 선택할 수 있다. 즉 회로의 구현부담이 문제가 되지 않는다면 m=4 이상의 LA나 기타 보다나은 다른 알고리즘을 사용하여 버스의 소비 전력을 BI를 사용하는 것 보다 크게 줄일 수 있다. (비공식적인 실험에서 L8의 경우 BI의 2배가 넘는 성능을 얻을 수 있었다). SSC의 경우 전송순서를 결정하는 알고리즘은 무수히 많으므로 경험을 바탕으로 하여 우수한 성능의 알고리즘을 개발하는 것은 비교적 간단한

표 2. k-BL이 UL 성능의 95% 와 99%를 갖기 위한 k의 실험 값

Table 2. The value of k for the k-BL to achieve 95% and 99% of the performance of UL.

Format	Bus width	L2		L3		L4	
		k_{95}	k_{99}	k_{95}	k_{99}	k_{95}	k_{99}
MP3	8	6.3	9.7	6.3	10.0	9.0	14.0
	16	5.3	8.0	6.0	9.0	8.0	13.0
	32	5.3	7.3	6.0	9.3	8.0	13.0
	64	5.0	7.3	6.3	10.0	7.3	12.7
Movie	8	6.0	9.0	6.0	9.7	9.0	14.0
	16	5.0	8.0	6.0	9.3	8.0	13.3
	32	5.0	7.3	6.0	9.0	8.0	13.0
	64	5.0	7.3	6.0	9.7	7.3	14.0
HTML	8	8.7	13.3	8.0	14.0	11.7	17.7
	16	5.3	9.7	6.3	10.7	9.7	17.0
	32	4.7	8.0	5.7	10.7	9.0	15.7
	64	6.7	9.3	8.0	14.0	9.7	17.7
GIF	8	5.7	8.3	5.3	8.3	8.0	12.7
	16	4.7	7.0	5.0	7.7	7.0	12.0
	32	5.0	7.7	5.0	8.0	7.0	12.3
	64	4.0	6.0	5.7	8.3	6.0	11.0
PPM	8	6.3	10.7	5.3	8.7	8.3	14.7
	16	5.3	9.3	5.7	9.3	7.7	13.7
	32	4.0	7.0	4.7	7.0	6.7	11.7
	64	3.3	6.0	6.0	9.3	6.0	10.3
JPG	8	6.0	8.0	6.0	8.7	8.0	13.0
	16	5.0	7.7	4.7	7.0	7.0	12.7
	32	4.3	7.3	4.3	7.3	7.0	11.7
	64	4.0	8.0	5.3	8.3	7.0	14.0
Rand	8	6.3	9.3	5.7	9.3	8.3	13.3
	16	4.3	7.0	5.0	9.3	7.7	13.3
	32	4.0	7.0	5.3	8.7	7.0	11.0
	64	4.0	8.3	5.7	9.3	7.3	11.3

표 3. 각 알고리즘의 버스 전이감소율
Table 3. Transition-reduction rate of the algorithms.

Format	Bus Width	Transition Reduction Rate (%)					
		BI	G2	G3	L2	L3	L4
MP3	8	17.43	4.31	9.68	9.53	13.59	16.75
	16	14.04	4.26	8.49	8.71	12.33	15.24
	32	10.50	3.31	6.46	6.88	9.87	12.22
	64	8.24	2.65	5.05	5.49	7.95	9.82
Movie	8	17.59	4.60	9.95	9.74	13.80	16.98
	16	14.08	4.27	8.62	8.70	12.42	15.37
	32	10.76	3.52	6.80	7.11	10.20	12.62
	64	8.00	2.68	5.14	5.51	8.01	9.88
HTML	8	5.13	3.25	8.39	9.60	13.21	16.66
	16	2.60	4.05	8.52	8.73	12.88	16.30
	32	1.30	3.97	8.38	8.40	12.13	15.47
	64	0.35	3.51	6.31	6.39	9.62	12.23
GIF	8	17.85	6.30	11.23	10.18	15.53	19.15
	16	14.58	5.62	9.77	9.40	13.87	17.13
	32	10.56	4.46	7.25	7.34	11.15	13.80
	64	8.22	4.25	6.52	6.53	10.13	12.60
PPM	8	15.32	11.34	15.61	13.46	20.78	25.13
	16	14.58	6.46	10.45	10.23	15.48	19.21
	32	11.15	5.73	8.76	8.58	13.19	16.37
	64	5.40	5.75	7.65	7.17	11.39	14.31
JPG	8	18.82	5.06	10.63	10.29	14.52	17.78
	16	12.40	6.41	10.17	9.43	14.56	17.99
	32	9.12	4.92	7.75	7.66	11.79	14.68
	64	8.39	2.83	5.31	5.63	8.15	10.06
RAND	8	18.48	5.02	10.10	9.93	13.98	17.16
	16	14.82	4.70	8.95	8.94	12.34	15.40
	32	10.92	3.29	6.46	7.02	9.98	12.23
	64	8.53	3.11	5.42	5.50	8.06	9.97

일이며, 오히려 이렇게 개발된 알고리즘을 적은부담으로 구현하는 방법을 찾는 것이 더욱 어려운 문제점이다.

위 실험결과를 요약하면, 먼저 SSC의 실용성이 기존의 BI와의 성능 검증을 통하여 증명되었다. SSC는 특히 문서파일의 전송이나 버스폭이 넓은 경우 BI에 비하여 우수한 성능을 보인다. 이진파일의 경우 SSC는 L4 이상의 알고리즘을 사용하여 BI이상의 성능을 얻을 수 있다. 한편, 실험결과에 따르면 LA의 성능이 GA에 비해 우수하고 또 LA의 성능도 순서선택에 사용되는 워드의 수(m 값)에 따라 큰 차이를 보인다. 이것은 SSC의 성능이 매우 넓은 범위에 걸쳐 변화하며 순서를 변환시키는 알고리즘에 크게 의존한다는 것을 의미한다. 이러한 성질은 SSC의 잠재적 가능성을 보여주는 것으로써 설계자에게 버스의 소비전력 감소와 회로의 구현부담사이에서 절충할 수 있는 범위를 넓혀 주게 된다. 즉, 순서를 변환시키는 방법은 무수히 많으므로, 여러 가지 순서변환 알고리즘들을 시험하고 그 성능들을 도표화함으로써 회로 설계자들은 제작하고자하는 칩의 규격을 고려하여 목적에 맞는 적당한 알고리즘을 선택할 수 있다.

응용분야 측면을 살펴볼 때 MP3, 동영상 파일들을 주로 다루는 멀티미디어 분야의 경우에는 BI가 적은 구현부담으로써 뛰어난 성능을 보인다, 그러나 버스의 소비전력이 주요 문제가 된다면 구현부담을 감수하고 SSC을 사용하여 그 소비전력을 크게 줄일 수 있다. 메일 서버나 인터넷 응용칩 같이 문서 파일을 많이 사용하는 분야에는 SSC가 BI에 비해 매우 우수한 성능을 갖는다.

V. 결 론

본 논문에서는 패킷형 데이터의 전송 시 데이터의 전송순서를 변화시켜 버스의 전이횟수를 감소시키는 새로운 방법인 순서변환코딩을 제안하였다. SSC의 실효성은, LA와 같은 간단한 알고리즘을 사용하여 기존의 널리 알려진 BI를 능가하는 성능을 얻음으로써 검증되었다. SSC는 데이터 전송 시 버스의 전이 횟수를 감소시키는 하나의 방법이며 구체적인 알고리즘이 아니다. SSC를 구현하는 알고리즘은 무수히 많으며, 그 성능은 순서변환 알고리즘에 크게 좌우된다. 이 점은 SSC가 갖는 하나의 장점으로써 회로 설계자는 버스의 소비전력 감소율과 회로의 복잡성에 따르는 구현 부담 사이의 절충(trade-off)을 고려하여, 다양한 알고리즘 중에서 적당한 것을 선택할 수 있다.

여러 가지 노하우를 사용하면 매우 다양한 순서변환 알고리즘을 손쉽게 개발할 수 있다. 그러나 개발된 알고리즘을 적은 부담으로써 구현할 수 있도록 하는 것은 쉬운일이 아니므로 오히려 이점이 실용적인 측면에서 더욱 중요한 문제점으로 부각되고 있다. 우리의 향후 과제는 다양한 효과적인 순서변환 알고리즘들을 제안하고 이들을 적은 부담으로 구현하는 방법을 개발하는 일이다.

참 고 문 헌

[1] R. Wilson, "Low power and paradox," Electronic Engineering Times, pp. 38. Nov. 1, 1993.

[2] N. Weste and K. Eshraghian, Principles of CMOS VLSI Design, A Systems Perspective, Reading, MA: Addison-Wesley Publishing Company, 1988.
[3] Y. Nakagome, K, Itoh, M. Isoda, K, Takeuchi, and M. Aoki, "Sub-1-V Swing Internal Bus Architecture for Future Low-Power ULSI's," IEEE Journal of Solid State Circuits, vol. 28, no.4, pp. 414-419, Apr. 1993.
[4] H. Zhang, V. George, and J. M. Rabaey, "Low swing on-chip signaling techniques: effectiveness and robustness," IEEE Trans. VLSI Syst. vol. 8, no. 3 pp. 264-272, June 2000.
[5] M. R. Stan and W. P. Burleson, "Bus-invert coding for low-power I/O," IEEE Trans. VLSI Syst., vol. 3, pp. 4958, Mar. 1995.
[6] C. L. Su, C. Y. Tsui, and A. M. Despain, "Low power architecture design and compilation technique for high-performance processors," in Proc. IEEE COMPCON, Feb. 1994, pp. 209214.
[7] L. Benini, G. De Micheli, E. Macii, D. Sciuto, and C Silvano, "Asymptotic zero-transition activity encoding for address busses in low -power microprocessor-based systems,"in Proc. Great Lakes Sysmp. VLSI, Mar. 1997, pp. 7782.
[8] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, Introduction to Algorithms, New York, McGraw-Hill Book Company, 1992.
[9] VisTex Database, http://www-white.media.mit.edu/vismod/imagery/VisionTexture/vistex.html
[10] Caltech Image Database, http://www.vision.caltech.edu/html-files/archive.html
[11] Content-based Image Retrieval Database, http://www.cs.washington.edu/research/imagedatabase/groundtruth/

저 자 소 개

윤 명 철(정회원)
1986년 서울대학교 전자공학과 학사
1988년 서울대학교 전자공학과 석사
1988년 현대전자 반도체연구소
1998년 The Univ. of Texas at Austin. ECE, Ph.D.
1999년 현대전자 씨스템 IC 연구소
2003년 고려대학교 전자컴퓨터공학과 BK21 계약교수.
<주관심분야: 반도체, VLSI 설계, Embedded System>