저전력 회로 설계를 위한 분할 버스-인버트 코딩 기법

김태환 홍성백 엄준형 김영태 여준기
한국과학기술원 전산학과
첨단정보기술 연구센터(AITrc)

Decomposed Bus Invert Coding Scheme For Low Power Circut Design

{tkim, hongsup, jhum, young, cglyuh}@vlsisyn.kaist.ac.kr
Taewhan Kim Sungpack Hong Junhyung Um Youngtae Kim Chungi Lyuh

Dept. of Computer Science and Advanced Information Technology Research Center(AITrc)
Korea Advanced Institute of Science and Technology

요약

버스-인버트 코딩 기법은 버스에서의 연속된 데이터 전송시 발생하는 데이터 값의 천이를 줄이는 기법이다. 기존의 방식에서는 전체 버스 라인이나 그중의 한 일부분만 에 버스-인버트 코딩을 적용했었던 것과는 달리, 우리의 기법은 버스 라인들을 몇개의 묶음으로 분할하여, 각 묶음에 대해 독립적으로 버스-인 버트 코딩을 적용하여 데이터 값의 천이를 최소화 하려고 한다. 실험을 통해서 우리의 기법은 데이터 값의 천이를 전체적으로 10 - 50 % 감소시킬수 있음이 나타났다.

1 서론

노트북이나 휴대폰 같은 휴대용 전자기기들은 제한된 배터리 용량을 가지고 있기 때문에, 이러한 회로에서는 실행 속도를 떨어뜨리지 않는 한 가능하면 적은 전력만을 소모하도록 설계되어야 한다. CMOS 회로에서의 전력 소모의 대부분은 게이트에서의 동적 전력 소모에서 기인하며, 그 값은 게이트의 출력 캐퍼시탄스와 노드 에서의 천이의 빈도에 비례한다.

[4]에서 보여지었듯 I/O 연결에 대한 캐퍼시턴스값들은 칩 내부의 게이트의 캐퍼시턴스 값보다 몇 자리 수 높은 정도의 큰 값을 가지게 된다. 그러므로, 버스 기반 시스템에서는 버스에서의 천이(transition)가 전력 소 비의 주된 원인이 된다. 칩 바깥으로(off-chip)의 신호를 전달하는 버스에서의 신호변환에 의한 전력 소모는 버스의 큰 커패시턴스 값과 큰 오프칩 드라이버 와 같은 요인 때문에, 전체 전력 소모의 70%에 이르기도 한다 [1]. 버스에서의 전력소모를 줄이기 위해서, 버스를 지나는 데이터 값의 천이의 빈도를 감소시키는 방법이 있을 수 있다. 이을 위해 버스를 통하는 일련의 신호들을 부호화 하는 기법들이 연구되어져 왔다.

연속적인 값을 가지는 성질이 있는 명령어 주소 버스에 대해서는 Gray 코드[2]와 T0 코드[3] 와 같은 기법이 존재한다. 버스-인버트(Bus-invert) 코드[4]는 임의의 값을 가질 수 있는 데이터 버스를 대상으로 고안된 것으로 버스에서의 한 번의 최대 천이의 수를 버스의 너비(width)의 절반 이하로 제한한 것이다. 버스-인버트 코드를 적용하기 위해서는 현재 버스에서의 데이터가 반전된 것인지 여부를 나타내기 위한 추가적 한 회선이 더 필요하다. 부분적 버스-인버트(Partial bus-invert) 코드 [5]는 데이터 주소 버스를 대상으로 한 것으로, 데이터 주소는 명령어 주소만큼 연속적인 값을 가지지는 않지만, 데이터만큼 임의 값을 가지지는 않는다고 가정되었다. 부분적 버스-인버트 코드는 버스의 데이터 중에 상호 연관성이 높은 한 그룹의 신호에 대해서만 버스-인버트 코드를 적용한다.

이 논문에서 우리가 제안하는 부호화 기법은 분할 버스-인버트 (Decomposed bus-invert) 코드로, 이러한 데이터 주소 버스를 대상으로 한 것이다. 이 코드에서는 일단 회로의 수행 내용이 주어지면, 정적으로 통계적 방법을 사용하여 상호 연관성 높은 버스 회선들을 몇 개의 서브 버스들로 분할한다. 이러한 서브 버스들 각각에 대해 서로 독립적으로 버스-인버트 코드를 적용한다.

2 버스-인버트 코드

버스-인버트 코드는 대상 버스 라인들을 통해 전송되는 데이터들 중에서 천이의 개수가 전체 버스 회선 너비의 1/2을 넘을 경우 데이터의 값을 0은 1로 1은 0으로 반전하여 전송한다. 단 이경우 데이터의 반전이 일어났음을 알리는 추가적인 버스 라인이 필요하다. [4] 이렇게 함으로서 한번 전송에서의 천이의 개수를 언제나 전체 버스 회선 너비의 1/2 이하로 유지한다는 수 있다는 특성을 가진다. 반면에 전송되는 의 데이터들이 시간적, 공간적으로 독립적일 경우 그 효과가 크지 못하게 된다.

부분적 버스-인버트(PBI) 코드 [5]는 이러한 점을 착안하여 버스 회선들을 중에서 서로간의 천이의 연관성이높고 천이의 정도가 큰 버스 회선들을 하나의 그룹으로 골라내어, 이 그룹에 대해서만 버스-인버트 코드를 적용하 였다. 여기서 두 버스 회선 사이에 상호 연관성이 있다는 의

미는, 두 개의 회선이 동시에 천이가 일어날 값을 가지게 될 경우가 많다는 뜻이다.

천이의 개수를 더 줄이기 위해 서로간의 상호 연관성이 높은 버스 회선들끼리, 여러 개의 그룹으로 묶는 방법을 생각할 수 있다. 만약 [?]에서 제시한 방법을 반복적으로 적용한다면, 각 단계에서 PBI 알고리즘은 현 상태의 최대 효율의 그룹을 찾으므로, 전체 버스 회선에 대한 고려가 부족할 수 있다. 게다가 PBI 알고리즘의 반복 적용은 매 단계마다, 시뮬레이션을 통한 확인 과정을 거쳐야 하므로 수행 시간이 무척 길게 된다는 단점이 있다.

우리가 제시하는 분할 버스-인버트(DBI) 코드에서는 전체 버스 회선을 몇 개의 상호 배제 부분집합들로 분할하되, 각각에 그룹에 대한 버스-인버트 코드의 효과를 전체적으로 고려하면서 한다. 이렇게 함으로서 버스에서의 천이의 개수를 더 줄일 수 있게 된다.

3 분할 알고리즘

버스 회선들의 모든 가능한 분할들 중에서, 전체 천이의 개수가 최소가 되도록 하는 최적의 분할을 찾는 일은 계산학적으로 어려운 일이므로, 휴리스틱에 기반한 검색을 사용한다.

먼저 버스 회선들과 각 버스 회선을 통하는 데이터들의 사전정보를 가지고 완전 그래프를 구성한다. 처음 그래프에서 노드는 버스 회선을 나타내며 노드 사이의 에지는 두 버스 회선을 한데 묶어서 버스-인버트 코드를 적용할 경우 그 효과가 얼마나 좋을 것인가 하는 예상치의 측도를 나타낸다. 이러한 예상치의 측정은 4장에서 설명한다.

일단 이러한 그래프를 만든 다음에는, 그래프의 에지들 중에서 가장 큰 값을 가지는 에지를 선택하여, 그 에지에 연결된 노드들을 하나의 노드로 합친다. 합쳐진 노드에 속해있던 회선들은 하나의 그룹으로 묶이게 된다. 새로 생긴 노드와 기존의 노드들 사이의 에지값들은 모두 새로 계산해 주어야 한다. 그러므로 이렇게 해서 합쳐진이후의 그래프는 원래의 그래프와 다른, 하나의 후보*구성 상태*가 된다. 이런 식으로 한번에 하나의 에지를 선택해 가면서 일련의 후보 *구성 상태*들을 만들어 낸다.

위의 방법을 모든 에지가 사라지고 하나의 노드만 남을 때까지 반복한다. 그 다음 최종적으로 시뮬레이션을 통해 모든 후보 *후보 구성 상태* 중에서 가장 좋은 *구성 상태*를 골라낸다. 우리의 분할 알고리즘은 다음과 같이 기술 될 수 있다.

Algorithm *DBI: 버스 분할 알고리즘*
/* 초기화 */
- $M = \{S_1, S_2, \cdots, S_n\}, S_i = \{b^{i-1}\}$;
- 현재 후보 구성 상태를 저장한다.

while ($|M| \geq 2$)
- 에지의 $Gain$ 값이 가장 큰 노드쌍 (S_j, S_k)를 고른다.
- S_j and S_k 를 하나의 집합으로 합치고, 현재 구성 상태를 저장한다.
- $Gain$값들을 다시 계산 한다.

endwhile
- 모든 구성 상태에 대해 그룹별로 버스 인버트 코드를 적용하여 시뮬레이션 한다..
- 천이의 개수가 가장 적은 구성 상태를 고른다.

위에서 b^i는 버스의 i번째 회선을 뜻하고, S_i는 그래프의 각 노드이다. 그래프에 각 노드에 주어지는 $Gain$값은 4절에서 정의되는 값으로 각 노드를 합쳐서 버스-인버트 코드를 적용하였을 때, 천이의 개수에 있어서 어느 정도의 이득이 있을 것인가 하는 정도를 예측한 값이다.

4 Cost 계산

4.1 천이 벡터

버스-인버트 코드의 효과를 효율적으로 설명하기 위하여 천이 부호 와 천이 벡터라는 개념을 도입하겠다. j번 버스 회선에서 i번째 시간 에 대한 천이 부호t_i^j를 도입한다. 이 값은 $b_{i-1}^j \neq b_i^j$ 이면 $t_i^j = 1$ 이고, 아니면 $t_i^j = 0$ 이다.

특정 시간 인덱스 i에 있어서 몇개의 버스 회선들의 천이 부호들을 모은 벡터를 천이 벡터라고 정의 한다. 그림 1은 이러한 천이벡터를 보여준다. 이 천이 벡터들만을 고려해도 원래 데이터 순열이 버스-인버트(BI) 코드에 의해 부호화 되어 될 때의 실제 버스에서의 천이의 개수를 세어 볼 수 있다. 즉, 천이 벡터의 순열에 대해 버스-인버트 코드를 적용하여 1의 개수는 원래 데이터 순열에 대해 BI 코드를 적용한후 천이의 개수와 같게 된다. 결과적으로 어떤 천이 벡터에 버스-인버트 코드가 적용되었을 때, 천이 벡터의 1의 개수가 (벡터의 길이)/2 보다 컸다면, 1의 값은 0으로 바뀌고 0의 값은 1로 바뀌는 반전이 일어나게 됨을 알 수 있다. (그림 2)

b0	10100110	t0	11110101
b1	10100011	t1	11110010
b2	10100110	t2	11110100
b3	10100111	t3	11110101
b4	00001100	t4	00001010
b5	00001001	t5	00001101
b6	00001000	t6	00001100
b7	00001101	t7	00001011

그림 1: 천이 벡터로의 변환

4.2 Cost의 결정

이제 우리의 목표는 천이 벡터의 순열들을 그룹으로 나누어 각 그룹에 버스-인버트 코드를 적용하였을때, 결과로 변환된 천이 벡터의 순열들에 1의 값이 최소가 되는 그룹 분할을 찾아 내는 것이다.

한쌍의 버스 회선 (b^i,b^j)의 연관성을 측정하기 위해 다음의 값들을 정의 한다.

$$P_{xy}(b^i, b^j) = |\{k|\ (t_k^i, t_k^j) = (x,y)\ or\ (y,x)\}|$$

다시말해서 t_i,t_j가 (x,y) 또는 (x,y) 를 가지는 경우의 개수를 나타낸다. 이 개념을 확장해서 버스 라인들의 집합 S_i에 속한 임의의 두 버스 라인에 대한 천이 벡터가 (x,y) 또는 (x,y)를 가지는 경우의 수를 $P_{xy}(S_i)$라고 정의한다. 이제 버스 라인들의 집합 S_i에서의 천이의 개수는 천이 벡터에서의 1의 개수의 총합과 같으므로 다음의 식으로 표현될 수 있다.

$$\mathcal{F}(S_j) = \frac{(2 \cdot P_{11}(S_j) + P_{01}(S_j))}{(|S_j| - 1)} \quad (1)$$

버스 인버트 코드가 적용되고 나면, 천이 벡터들에 앞절에 끝에서 설명했던것과 같은 반전이 일어나게 된다. 이러한 반전이 일어났을때, P_{xy}를 통해 표시했던 (x,y) 쌍의 개수가 변하게 됨을 알 수 있다. 이를테면 일부의 (0,0) 쌍은 (1,1)로 바뀌게 되고 (0,0)쌍은 (1,1)로 바뀌게 된다. 이를테면 그림 2에서 $S_j = t_1, t_2, t_3, t_4$ 라고 할때, 버스-인버트 코드 를 적용하기 전에 $P_{11}(S_j)$의 값은 28이지만 버스-인

버트 코드가 적용된 다음에는 1이된다. $P(0,1)$의 값은 버스-인버트 코드의 적용에도 변하지 않음을 주목하라.

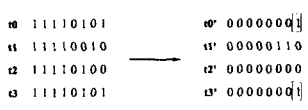

그림 2: BI 코드를 적용한 다음의 천이 벡터

집합 Sj의 천이 벡터들에 있어서 모든 (1,1)쌍들중 (0,0)쌍으로 변하는 비율과, 그 반대로 변하는 비율을 알게된다면, 버스-인버트 코드가 적용된 다음의 (1,1)쌍의 개수를 알수 있다. 각각의 비율을 $prob_{11}$,$prob_{00}$라고 한다면, 버스-인버트 코드가 적용된 다음의 (1,1)쌍의 개 수는 $P_{00} \cdot prob00 + P_{11} \cdot (1 - prob_{11})$ 이 된다. (0,0) 쌍의 개수도 비슷하게 생각할수 있다.

이러한 비율은 실제로 버스 인버트 코드를 적용시켜 보기전 까지는 정확하게 구할 수는 없다. 그렇다고 모든 분할 가능성에 대해 버스-인버트 코드를 적용해 볼 수는 없 으므로 이러한 비율에 대한 예측값으로 우리는 다음과 같은 식을 사용했다.

$$
\begin{aligned}
prob_{11} &= (P_{11}(S_j) + \frac{1}{2}P_{01}(S_j))/P_{all} \\
prob_{00} &= max\{(-P_{00}(S_j) + \frac{1}{2}P_{01}(S_j))/P_{all}, 0\} \\
, where P_{all} &= P_{11}(S_j) + P_{01}(S_j) + P_{00}(S_j) \qquad (2)
\end{aligned}
$$

위의 식은 원래 천이 벡터에서 P_{11}의 비율이 높을 수록 버스-인버트에 의한 반전이 일어날 확률이 높고, P_{00}의 비율이 클수록 그 확률이 작다는 관찰에서 비롯되었다. 위의 식 2를 사용하여, 앞에서 설명했던것과 같이 버스-인버트 코드를 적용하고 난 다음의 변한 P_{11}과 P_{00}을 예측할 수 있다. P_{01}의 값은 변하지 않으므로, 버스-인버트 코드를 적용하고 난 다음의 천이 벡터에서의 1의 개수를 식 1에 의해 예측 할 수 있다. 이 값을 $\hat{\mathcal{F}}$라고 하자. 결국 버스 인버트 코드의 적용으로 인해 줄어드는 천이의 개수는 원래 천이 벡터에서의 1의 개수에서 버스 인버트 코드 적용 이후의 1의 개수의 차이므로, 다음의 식으로 표현된다.

$$
\begin{aligned}
Save(S_j) &= \mathcal{F}(S_j) - \hat{\mathcal{F}}(S_j) \\
&= \frac{2 \cdot (P\_11(S_j) \cdot prob_{11} - P\_00(S_j) \cdot prob_{00})}{(|S_j| - 1)}
\end{aligned}
$$

마지막으로 두 집합 S_1, S_2에 독립적으로 버스-인버트 코드를 적용했을 때와 하나의 그룹으로 적용했을 때의 천이 개수 감소의 차이는 다음의 식으로 표현된다.

$$Gain(S_j, S_k) = Save(S_j \cup S_k) - Save(S_j) - Save(S_k) \quad (3)$$

5 실험 결과

우리는 [6]에서 제시된 일련의 벤치마크 설계들을 가지고, 어떤 특정 수행에 대한 데이터 패턴을 추출해서 제시된 알고리즘을 적용해 보았다.우리는 우리의 결과를 버스-인버트 (BI)와 부분적 버스-인버트 (PBI) 코드에 의한 결과와 비교하여 표 ??에 요약하였다. 결과에 따르면 DBI는 BI와 PBI에 비해 천이의 개수를 각각 평균적으로 47.2%와 11.9% 정도 감소시켰다. 표 3은 최대 분할의 개수를 2,3으로 제약했을 때, 수행된 DBI의 결과를 나타낸다. 최대 분할의 개수가 2개인 경우에도 PBI에 비해 약간의 증가를 보였다. 결국 DBI는 수행시간이나 회로 면적의 커다란 추가적인 증가 없이 버스에서의 천이의 개수를 줄일수 있다는 점을 보여주고 있다.

design	BI:PBI:DBI (# Trans.)	Red. wrt BI/PBI (%)
Compress	1066266 : 722260 : 588150	44.8/18.6
Laplace	2377233 : 1603476 : 1349402	43.2/15.8
Linear	2420802 : 1227402 : 1109002	54.2/9.65
Lowpass	656133 : 399690 : 331462	49.5/17.1
SOR	1900735 : 1343694 : 1210218	36.3/9.93
Wavelet	1406 : 626 : 626	55.4/0
Average		47.2/11.9

표 1: 벤치마크 프로그램에 대한 코드 적용 결과

design	DBI 2-part:3-part	Red. wrt BI (%)	Red. wrt PBI (%)
Compress	663466:611370	37.8/42.7	8.1/15.4
Laplace	1460284:1448032	38.6/39.1	8.9/9.7
Linear	1207402:1144692	50.1/52.7	1.6/6.7
Lowpass	364422:340757	44.5/48.0	8.8/14.7
SOR	1347235:1277983	29.1/32.8	-0.2/4.9
Wavelet	626:626	55.4/55.4	0/0
Average		42.6/45.1	4.5/8.6

표 2: 분할의 최대 개수를 제한하였을 때의 천이의 개수

6 결론

이 논문에서는 버스에서의 연속된 데이터 전송 시 발생하는 데이터 값의 천이를 줄이는 새로운 버스-인버트 코딩 기법을 소개하였다. 우리의 알고리즘은 버스를 여러개의 그룹으로 나누어 각각의 그룹에 버스-인버트 코드를 적용하였다. 기존의 기법에 비해 적은 추가 하드웨어 부담만을 가지고 우수한 결과를 나타내었다. 또 하나의 장점은 주어진 설계의 목표와 제약조건에 맞는 버스 회선의 분할의 개수를 설계자가 선택할 수 있도록 해 줄 수 있다는 점이다.

감사의 글

본 논문은 첨단정보기술 연구센터(AITrc)를 통하여 과학재단의 지원을 받았음.

참조 서적

[1] D. Liu and C. Svensson, "Power consumption estimation in CMOS VLSI chips," *IEEE Journal of Solid State Circuits*, vol 29, no. 6 ,pp.663-670, 1994

[2] C.L. Su, C.Y. Tsui, and A.M. Despain, "Saving power in the control path of embedded processors," *IEEE Design and Test of Computers*, vol 11, no. 4, pp.24-30, 1994

[3] L. Benni, G. De Micheli, E. Macii, D. Scivto, and C. Silvano, "Asymptotic zero-transition activity encoding for address busses in low-power microprocessor-based systems," *Proceedings of Grate Lakes Symposium on VLSI*, pp.77-82, 1997

[4] M.R. Stan and W.P. Burleson, "Bus-invert coding for low-power I/O," *IEEE Transactions on VLSI Systems*, vol 3, no.1, pp.49-58,1995

[5] Y. Shin, S. Chae and K. Choi, "Reduction of bus transitions with partial bus-invert coding," *IEE Electronics Letters*, vol 34, no. 7, pp.642-643, 1998

[6] P.R. Panda and N.D. Dutt, "1995 high level synthesis design repository," *Proceedings of International Symposium on System Synthesis*, 1995