누설 전력 최소화를 고려한 연산 아키텍쳐 설계

원대건
과학영재학교
김태환°
서울대학교, 전기.컴퓨터공학부
{dwon, tkim}@ssl.snu.ac.kr

Design of Arithmetic Architecture Considering Leakage Power Minimization

Daegun Won
Busan Science Academy
Taewhan Kim°
School of EECS, Seoul National University

요 약

최근의 멀티미디어 시스템 설계 (예: 휴대폰, PDA) 경향에서 전력 소모를 줄이는 연구가 매우 긴요한 상황에, 본 연구는 누설 전류(leakage power)를 줄이는 연산 회로 아키텍쳐 합성 기법을 제안한다. 누설 전류를 줄이기 위한 방법으로 본 연구는 Dual threshold Voltage (Dual-V_T) 기법을 적용한다. 기존의 연구에서는, 회로 설계 단계 중 논리나 트랜지스터 수준에서Dual-V_T 를 적용한 방법과는 달리, 보다 상위 단계인 회로의 아키텍쳐 합성 단계에서의 지연시간 제약 조건을 만족하는 범위에서 최소의 누설전류 소모를 위한 합성 기법을 제안한다. 따라서, 지연 시간과 누설전류 간의 Trade-off를 이용하여 설계 조건에 맞는 융통성 있는 설계 결과를 얻을 수 있는 장점을 제공한다. 본 연구는 케리-세이브 가산기 (Carry-Save Adder) 모듈의 생성 과정에 국한된 합성 알고리즘의 적용을 보이고 있지만, 일반적인 연산 모듈을 사용한 아키텍쳐 설계 과정에서도 본 알고리즘을 쉽게 변형, 적용할 수 있다.

1. 서 론

International Technology Roadmap for Semiconductors (ITRS)[1]에 따르면, 최소 배선폭(feature size)은 2008년까지 0.07㎛에 달할 것이라고 한다. 이러한 초미세의 시대에서는 이전에 중요하지 않게 여겨졌었던 몇 가지 설계에 관한 이슈들이 중요하게 여겨진다. 이러한 이슈 중 하나가 회로에서 낭비되는 누설 전력(leakage power)이다. CMOS 회로 설계에서 전력 소비를 줄이기 위해, 동적 전력 (dynamic power) 소비를 줄이는 많은 연구가 진행되었다. 동적 전력을 주이는 방법으로 공급전압을 낮추는 기법이 많이 사용되고 있다. 그러나, 공급 전압을 줄이는 것은 줄이는 것은 회로 전체의 지연 시간을 증가시킨다. 지연시간 D는 다음과 같이 주어진다.[1]

$$D \approx \frac{CV_{dd}}{(V_{dd}-V_T)^{\alpha}}$$

여기서, V_T는 역치 전압(threshold voltage)이며, α는 1.4에서 2 사이에 존재하는 상수 값이다. 이것은 공급 전압을 역치 전압에 가깝게 줄이는 것이 기하급수적으로 지연 시간을 증가시킨다는 것을 잘 나타낸다. 하지만 역치 전압을 낮추는 것은 증가된 역치하 유도 (subthreshold conduction) 때문에 누설 전력을 기하급수적으로 증가시키게 된다. 결과적으로, 최소 배선 폭은 계속 줄어들고, 낮은 역치 전압 사용으로 인해 생기는 누설 전력의 양은 동적 전력 소모량보다 전체 전력 소비에서 더 큰 부분을 차지하게 된다. 누설 전력을 줄이기 위한 잘 알려진 방법으로는 이중(dual) 역치 전압을 사용하는 것이 있다.

누설 전력이 차지하는 비중이 증가함에 따라 이를 줄이기 위한 많은 연구가 진행되었다.[1] 그러나, CSA (carry-save) 가산기를 이용한 연산 회로 설계에 대해서는 아직 누설전류의 절감을 위해 이루어진 연구가 없다. 본 연구에서는 이중 역치 전압 (dual-threshold voltage)를 이용한 누설 전력 최소화를 고려한 CSA 가산기 할당 알고리즘을 제안한다.

2. 누설 전류 측정 모델링 및 연구 동기

n-bit CSA(Carry-Save-Adder)[2]는 n개의 FA(Full Adder)로 구성되어 있다. 그림 1-(a)은 n-bit CSA의 구조를 나타낸다. n-bit CSA는 3개의 n-bit 입력 벡터를 받아 2개의 n-bit 출력(합(Sum) 벡터와 올림(Carry) 벡터)을 만들어 낸다. 하나의 CSA는 그림 1-(b)와 같이 하나의 블록으로 표시될 수 있다. 다른 일반적인 가산기(예: ripple-carry adder)와는 달리, CSA는 올림수의 전달이 없다. 따라서, 한 개의 CSA는 비트수 n에 관계없이 한 개의 FA가 가지는 지연 시간과 같은 지연 시간을 가진다. 반면 전력은 n개의 FA가 사용하는 양과 같다. CSA는 두 개의 출력을 만들어 내기 때문에, CSA-트리의 끝에는 항상 일반 가산기가 존

재하여 하나의 결과를 만들어내게 된다.

CSA는 여러 가지 방법으로 만들 수 있으나, 여기서는 NAND 게이트를 사용한 CSA를 다룬다. 이의 구조는 그림 1 아래 부분에 나타나 있다. 게이트의 누설 전력은 입력에 큰 영향을 받는다. 표 1은 입력에 따른 NAND 게이트의 누설 전류의 양을 나타낸다.

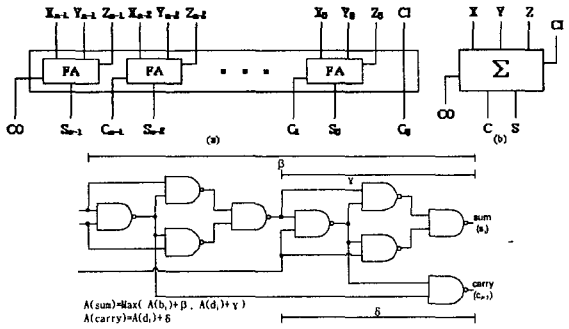


그림 1. CSA 구조와 FA 구조

입력	high-V_T	low-V_T
(0,0)	0.005	0.01
(0,1), (1,0)	0.23	0.46
(1,1)	0.45	0.90

표 1. 입력에 따른 NAND게이트의 누설 전류

FA에 들어 올 수 있는 모든 입력 패턴에 대한 누설 전류의 양을 계산해보면, 하나의 FA에서 예측할 수 있는 평균적인 누설 전력의 값이 나오게 된다. 따라서 하나의 n-bit CSA가 갖는 누설 전력을 계산해 보면 low-V_T CSA는 $(0.01\times4+0.46\times44+0.9\times24)/8\times3.3\times n = 17.276\times n$로 계산 된다. 같은 방식으로 high-$V_T$ CSA에 대한 값을 구해보면 $8.638\times n$임을 알 수 있다.

누설 전류를 줄이기 위해 이중 역치 전압을 가지는 CSA를 사용한 한 예를 보도록 하자. 우리는 8비트의 수에 대하여 산술식 K=A+B+C+D+E+F+G+H+I+J를 가지고, 이를 표현하는 회로의 타이밍 조건은 T=13.5이라 하자. 그림 3-(a)는 타이밍 조건을 만족하기 위해 low-V_T CSA만을 사용한 기존의 방법을 사용했을 때 생성되는 회로다. 그림 3-(a)의 경우속도는 제일 빠른 반면, 누설 전력의 양이 크다는 단점이 있다. 그림 3-(b)은 dual-V_T CSA를 사용했을 때 타이밍 조건인13.5를 만족하는 최적의 회로다. 이 때 회로의 전체 지연 시간은 13.4로 조건을 만족함과 동시에 누설 전력은 760.1로 기존의 방법을 썼을 때 생기는 1105.7의 약 30%가 절감된 것을 볼 수 있다.

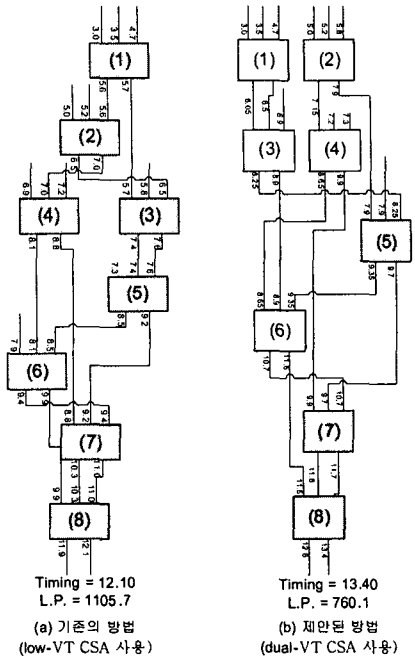


그림 3. dual-V_T CSA를 사용한 예

3. 제안 알고리즘

우리의 제안된 기법은 두 단계로 구성된다.

(Step 1) CSA-트리 생성 : high-V_T CSA만을 사용하여 CSA-트리를 생성한다. 이 때 사용되는 방법은 기존의 방법[2]을 사용한다. 따라서 생성되는 트리는 지연 시간은 가장 길고, 누설 전력의 양은 가장 적게 된다.

(Step 2) 지연시간-누설전력 Tradeoff : 가장 적절한 high-V_T CSA를 선택하여 그것을 low-V_T CSA로 바꾸게 된다. 따라서 반복 작업이 진행될 때마다 지연 시간은 줄어들고, 누설 전력의 양은 일정량씩 늘어나게 된다. 이 단계는 그림 4와 같은 과정을 반복한다.

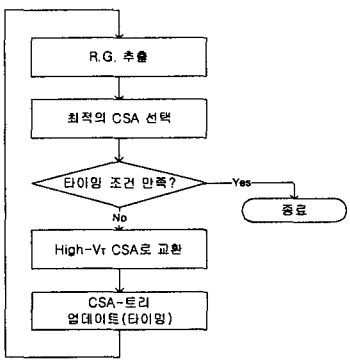


그림 4. Step 2 의 흐름

CSA-트리에서는 critical path가 존재한다. 그 경로에 의해 회로 전체의 지연 시간이 영향을 받으므로 지연 시간을 줄이기 위해서는 critical path 상에 위치하는 CSA를 high-V_T CSA로 바꿔야 함은 당연한 일이다. 게다가 critical path가 여러 개 있을 경우, 가장 많은 critical path를 거치는 CSA를 바꾸게 되면 그 효과가 가장 크다고 볼 수 있다. 그러나 critical path의 수가 적은 경우, 후보의 개수가 많아지게 된다. 우리는 이 점에 착안하여 slack이라는 개념을 도입, critical path의 수를 늘렸다.

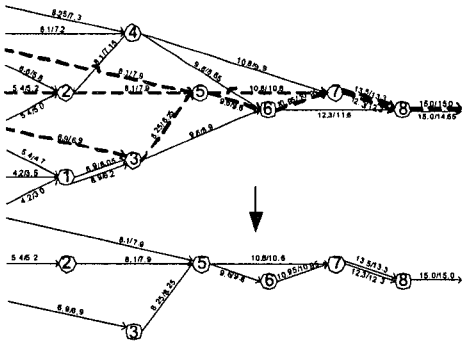


그림 5. R.G. 추출과정을 보여 주는 예

$\Theta(a_i)$를 입력벡터 a_i가 critical path가 되기 위해 도달해야 하는 시간이라 하자. 실제 도달 시간과 Θ값을 비교했을 때 그 차가 0이라면 critical path라 할 수 있다. 그러나 slack을 이용하면 차가 0이 아니어도 정해진 상수 이내의 값이면 critical path로 간주한다. 이 때 CSA-트리에서 critical path만 추출한 그래프를 얻는다. (이 그래프를 RG (reduced graph)라 부른다.) 그림 5는 RG 생성 과정의 한 예를 보여준다. 각 선분에 쓰여 있는 값은 각각 Θ, 도착시간을 나타낸다. R.G.를 추출한 후에 우리는 최적의 CSA를 선택하기 위해 각 CSA가 가지는 degree를 계산한다. 이는 CSA가 속해 있는 critical path의 개수이다. 이 값을 구하기 위해 우리는 다음과 같은 특성을 이용한다.

특성 1 : *방향성을 지니는 그래프에서 정점 A를 지나는 경로의 개수는 그 정점이 가지는 누적된 indegree와 누적된 outdegree의 곱과 같다.*

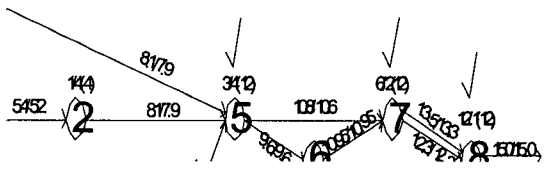


그림 6. degree의 계산

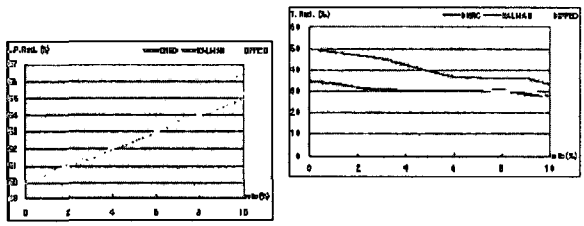
그림 6을 보도록 하자. 누적된 indegree와 outdegree란 단순히 그 노드 하나에서 계산한 indegree와 outdegree가 아닌, 그 노드와 연결된 다른 노드의 indegree와 outdegree까지 고려한 값이라 할 수 있다. 결국 여기서 구한 indegree와 outdegree의 의미는 그 노드에 도달할 수 있는 경로의 수, 뻗어나갈 수 있는 경로의 수가 된다. 이와 같은 방법으로 RG 상의 모든 정점에 대해 각각의 degree를 구하면 최적의 CSA을 선택할 수 있다. (지면 관계로 자세한 설명은 생략한다.)

4. 실험 결과

우리는 앞서 설명한 방법을 몇 개의 산술식과 벤치마킹 데이터를 이용하여 테스트해 보았다. 그림 1의 β,γ,δ와 slack을 각각 1.8, 1.0, 0.9, 0.3으로 설정하였으며 low-V_T CSA의 경우 β,γ,δ에 1.5배한 값을 설정했다. 표 2는 각각 임의의 산술식과 벤치마킹 데이터[3]를 이용하여 테스트한 결과를 나타낸다. 여기서 우리는 논리 단계에서의 CSA 지연시간은 Synopsys DC[4]를 이용하여 구하였다. (0.35u[5] 가술을 사용하였음.) 비교에서 보듯이 지연시간 제약 조건을 만족하면서 누설 전류를 40% 이상 감소함을 볼수 있다.

설계	기존방법		제안한 방법		누설전류 감소량
	지연시간	누설전류	지연시간	누설전류	
DIFFEQ	43.5	101721.1	55.8	55836.0	45.1%
KALMAN	10.0	8568.9	12.25	5113.7	40.4%
DHRC	35.4	20731.2	44.5	11609.5	44.0%
FFT	13.5	1934.9	16.8	1105.7	42.9%
FILTER	11.0	552.8	13.2	320.0	42.1%

표 2. 벤치마킹 데이터에 대한 기존의 방법과 우리의 방법의 비교



위 두 표는 본 제안한 기법을 사용하여 각기 high-V_T CSA의 비에 따른 전력 소모와 지연 시간의 증감소에 대한 비교를 나낸 것이다.

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

참조문헌

[1] K. Khouri and N. Jha, "Leakage power analysis and reduction during behavioral synthesis", *IEEE TVLSI*, pp. 876-885, Dec. 2002.

[2] T. Kim, W. Jao, and S. Tjiang, "Circuit optimization using carry-save-adder cells", *IEEE TCAD*, pp.974-984, October 1998.

[3] N. D. Dutt, "High-Level Synthesis Design Repositories", *http://www.ics.uci.edu/~dutt.*

[4] Synopsys Inc., *Design Compiler User Guide*, 2002

[5] LSI Logic Inc., *G10-p Cell-Based ASIC Products Databook*, 2000