정수 선형 프로그래밍을 이용한 혼합 가산기 구조의 최적 설계

(Optimal Design for Heterogeneous Adder Organization Using Integer Linear Programming)

이 덕 영 † 이 정 근 †† 이 정 아 ††† 이 상 민 ††††
(Deok-Young Lee) (Jeong-Gun Lee) (Jeong-A Lee) (Sang-Min Rhee)

요 약 비용 효과가 좋은 디지털 시스템을 설계하기 위하여, 트랜지스터 수준부터 RTL 수준까지 최적화를 위한 다양한 설계 방법이 연구되어 왔다. 가산기는 디지털 시스템에서 가장 기본적인 산술연산을 수행하는 필수 회로로서, 전체 시스템의 성능에 영향을 줄 수 있다. 본 논문에서는 최적의 가산기를 설계하기 위하여 상위수준에서 연구하였다. 결과로 혼합 가산기 구조를 제안하고 이를 정수 선형 프로그래밍(ILP: integer liner programming)을 이용해 수학적으로 모델링한다. 혼합 가산기 구조는 다양한 캐리 전달 방식을 가진 가산기 블록을 선형적으로 연결한 구조로서, 사용된 가산기 블록의 종류와 개수에 따라 다양한 가산기 조합이 발생한다. 이러한 조합에 의해 확장된 가산기의 설계공간을 탐색함으로써, 단일 타입의 가산기만을 고려한 것보다 나은 최적의 가산기를 설계할 수 있다. 제안한 혼합 가산기 구조와 ILP를 이용한 최적화 기법은 연산시간과 회로면적 등의 특성이 다른 가산기 IP(intellectual property)들을 비트 수준에서 재합성하기 때문에, 보다 미세한 수준에서 최적화를 수행할 수 있다.

키워드 : 가산기, 캐리 전달 방식, 설계 공간, 최적화, 정수 선형 프로그래밍

Abstract Lots of effort toward design optimizations have been paid for a cost-effective system design in various ways from a transistor level to RTL designs. In this paper, we propose a bit level optimization of an adder design for expanding its design space. For the bit-level optimization, a heterogeneous adder organization utilizing a mixture of carry propagation schemes is proposed to design a delay-area efficient adder which were not available in an ordinary design space. Then, we develop an optimization method based on Integer Linear Programming to search the expanded design space of the heterogeneous adder. The novelty of the proposed architecture and optimization method is introducing a bit level reconstruction/recombination of IPs which have same functionality but different speed and area characteristics for producing more find-grained delay-area optimization.

Key words : Adder, Carry propagation scheme, Design Space, Optimization, Integer Linear Programming

1. 서 론

가산연산은 디지털 시스템의 데이타 처리에서 가장 많이 사용되며, 가산기는 대부분의 ASIC(application specific integrated circuit) 및 프로세서 설계에 있어 필수 요소이다. 어떤 종류의 가산기를 사용하느냐에 따라 시스템의 성능과 구현비용이 달라지므로, 가산기 설계에 대한 다양한 분석과 연구가 지난 수십 년간 진행되었다. 가산기를 설계할 때 고려해야 할 요소에는 속도와 회로 면적 그리고 소모 전력량이 있다[1]. 수십 년간의 연구에도 불구하고, 가산기에 대한 연구는 지금도 지속적으로 이루어지고 있으며, 이러한 연구들에는 하위 수준의 설계에서부터 상위 수준의 설계까지 다양하게 연구되고 있다. 하위 수준의 가산기 설계에는 새로운 반도체 공정기술을 이용한 설계와 트랜지스터의 구성 방식을 달리하는 설계 등이 있고, 상위 수준의 가산기 설계에는 새로운 데이타 표현 방식을 이용한 설계 등이

† 정 회 원 : 한림대학교 기초교육대학
riverlike@hallym.ac.kr
†† 정 회 원 : 캠브리지 대학교 컴퓨터랩
Jeonggun.Lee@gmail.com
††† 종신회원 : 조선대학교 컴퓨터공학과 교수
jalee@chosun.ac.kr
†††† 정 회 원 : 강원대학교 컴퓨터과학과 교수
smrhee@cc.kangwon.ac.kr
논문접수 : 2006년 12월 27일
심사완료 : 2007년 5월 19일

있다. 가산기의 연산 속도 및 구현 비용은 캐리의 전달 지연을 처리하는 방식에 영향을 받기 때문에, 캐리의 전달 속도를 향상시키기 위한 방식들이 다양하게 연구되어 왔다. 대표적인 방식들로는 CLA(Carry Look-ahead Adder) 방식, CSKA(Carry SKip Adder) 방식, CSEA (Carry SElect Adder) 방식 등이 있다. CLA 방식은 캐리 생성식에 의해 캐리를 계산하는 것으로, 입력 값이 결정되면 바로 캐리를 계산한다. CSKA 방식은 캐리의 전달을 스킵하는 것으로, 작은 블록으로 나누고 블록 수준에서 캐리를 스킵한다. CSEA 방식은 캐리 값에 따라 결과값을 미리 계산하는 것으로, 캐리가 전달되었을 때 결과값을 선택한다. 이외에도 다양한 방식들이 있다[2,3].

최근의 디지털 시스템들은 핸드폰 및 멀티미디어 단말을 포함하는 이동통신 기기와 개인용 컴퓨터에서 대형 컴퓨터에 이르기까지 설계상의 요구사항이 매우 다양화되고 있다. 이동통신 기기들은 휴대하기 편해야 하기 때문에 성능은 좀 떨어지더라도 크기가 작아야 하며, 은행이나 정부기관에서 사용되는 대형컴퓨터는 시스템의 크기보다는 성능이 우수하여야 한다. 또한 개인용 컴퓨터도 수 기가헤르츠(GHz) 대의 동작주파수를 갖는 프로세서를 사용하는 등 점점 고성능화되고 있다. 이처럼 디지털 회로 설계의 필수 요소인 가산기는 다양한 요구에 알맞은 시스템을 설계하는 데에 많은 영향을 준다. 그러므로 시스템의 요구사항에 부합되는 최적의 가산기를 설계할 필요가 있다.

최적의 가산기 설계에 대한 기존 연구들은 대부분 하위 수준에서 이루어져 왔으며, 이들 연구에는 새로운 반도체 공정기술을 이용한 회로 설계와 트랜지스터 수준에서의 설계에 대한 것 등이 있다. 이러한 설계를 통해 회로를 최소화하고 속도를 향상시키는 가산기를 설계할 수 있다. 그러나 레지스터 수준(RTL: register transfer level)에서의 가산기 최적 설계에 대한 연구는 이루어지고 있지 않다. 따라서 본 논문에서는 RTL 수준에서 가산기의 최적화에 대하여 연구하고자 한다.

시스템의 요구사항에는 가산기의 연산시간에 대한 제약조건을 나타내는 시간제약이나 가산기가 차지하는 면적에 대한 제약조건을 나타내는 면적제약이 있다. 가산기는 사용하는 캐리 전달 방식에 따라 연산시간과 회로면적이 달라진다. 연산시간과 회로면적은 절충(trade-off) 관계에 있으며, 이 관계는 가산기 설계 시에 고려되어야 하며 가산기의 특성에 따라 서로 다른 연산시간-회로면적(delay-area) 곡선으로 표현된다[1,4]. 이러한 곡선들로 구성된 공간을 가산기의 설계공간(design space)라고 하고, 목적 시스템의 요구사항에 부합되는 가산기를 찾는 것을 가산기의 설계공간 탐색(design space exploration)이라 한다[5]. 그러나 단일 타입의 캐리 전달 방식만을 고려한 설계공간에서 목적 시스템의 요구사항을 최대한 수용하는 최적의 가산기를 탐색하기는 어렵다.

이러한 문제를 해결하기 위하여 서로 다른 캐리 전달 방식을 하나의 가산기에 혼합한 가산기를 제안한다. 혼합 가산기는 이종의 캐리 전달 방식을 가진 단일 가산기들을 선형으로 연결한 구조이기 때문에, 사용하는 가산기의 종류와 개수에 따라 다양한 가산기 조합이 존재한다. 이러한 조합에 의해 가산기의 설계공간이 확장되는 것이다. 확장된 설계공간을 탐색함으로써, 단일 타입의 가산기만을 고려한 것보다 더 좋은 최적의 가산기를 찾을 수 있다. 그러나 설계공간이 확장된 만큼 최적의 가산기를 찾는 데 많은 시간이 소요된다. 그러므로 최적의 가산기를 자동으로 검색하기 위한 방법이 필요하다. 설계공간을 효과적으로 검색하기 위하여 최적화 기법인 ILP(integer liner programming)를 사용하였다. ILP 기반의 최적화를 위하여, 혼합 가산기의 연산시간을 수학적으로 모델링하고 ILP solver를 이용하여 실험하였다.

본 논문에서는 목적 시스템의 요구사항 즉, 주어진 설계의 제약조건에 최대한 부합되는 최적의 가산기를 설계하기 위하여 혼합 가산기 구조를 제안한다. 또한 이를 더욱 효과적으로 사용하기 위하여 ILP 기반의 최적화 설계 방법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문의 배경이 되는 관련연구로 가산기의 종류와 정수 선형 프로그래밍을 설명하고 가산기 설계상의 문제점을 살펴본다. 3장에서는 새롭게 제안하는 혼합 가산기의 구조에 대해서 기술한다. 4장에서는 제안하는 혼합 가산기 구조를 효과적으로 사용하기 위한 최적화 방법에 대하여 설명한다. 이후 5장에서는 제안된 최적화 방법에 기초하여 최적화를 수행한 결과를 보이고, 그 결과에 대하여 해석한다. 마지막으로 6장에서는 결론과 향후과제에 대해서 기술한다.

2. 관련 연구

2.1 가산기의 종류의 특성

사용 빈도가 가장 높은 연산기인 가산기는 캐리 전달 방식에 따라 다양하게 구분될 수 있지만, 구조적 측면에서 몇 가지 기본적인 캐리 전달 방식으로 구분될 수 있다. 대표적인 가산기 구조로 다음 세 가지의 가산기를 간략히 기술하고자 한다.

- Ripple Carry Adder (RCA) : 가산기 종류 중 가장 간단한 구조로, 전가산기의 캐리 입출력을 선형으로 연결한 것이다. n비트인 경우, 지연시간 T와 회로면적 A는 아래와 같다. 즉, 연산에 필요한 시간이 비트 수에 선형적으로 증가하므로 고성능 하드웨어에 이용할 수 없다.

$$T = n\,T_{FA}$$
$$A = n\,A_{FA}$$

- Carry SKip Adder (CSKA) : 특정 블록의 비트 패턴 정보를 이용하여, 그 블록으로 캐리 입력이 인가되었을 경우, 블록 내에서 비트별 캐리 전파 없이 다음 블록으로 캐리 신호를 전파하는 캐리 전달 방식을 갖는다. 이를 위하여 각 블록은 캐리를 전파 시킬 수 있는 입력 패턴을 검출하기 위한 부가 회로가 필요하다. 지연시간과 회로면적은 아래의 식과 같다. RCA보다 향상된 가산 처리 속도를 갖고 있으나, 부가 회로의 추가로 인하여 회로면적이 증가한다.

$$T = (2\sqrt{2n} - 3.5)\,T_{FA}$$
$$A = (n + \frac{n}{2m})\,A_{FA} \approx n\,A_{FA}$$

- Carry Look-ahead Adder (CLA) : 하위 비트의 캐리 출력을 미리 고려함으로써, 입력이 결정되면 캐리 생성 식을 이용하여 빠른 시간에 연산 결과를 얻을 수 있다. 캐리 생성 식은 Lookahead 블록 회로로 구현되며, 입력 비트 수가 많아질수록 팬-인(fan-in)이 많아지고 많은 회로 영역을 필요로 한다. 이러한 단점을 극복하기 위하여, 가산기를 여러 그룹으로 나눈 후 그 그룹을 더 작은 그룹으로 나누는 다층 레벨로 구현하는 방법이 많이 이용되고 있다. 다층 레벨로 구현되는 경우, CSKA에 비해 회로면적이 크게 증가하며 지연시간과 회로면적은 아래와 같다[6].

$$T = \log_2 n \cdot T_{AND}$$
$$A \approx n \log_2 n \cdot A_{AND}$$

이외에도 Prefix Adder, Carry Select Adder, Carry Save Adder와 하이브리드 구조를 가진 가산기들이 있으며, 다양한 수의 체계에 따라 많은 가산기 구조가 있다.

2.2 정수 선형 프로그래밍

선형 프로그래밍의 분야는 최적화 문제를 해결하기 위해 많은 연구가 진행되어 온 분야이다[7]. 특히 선형 시스템의 해가 정수라는 제약을 갖는 정수 선형 프로그래밍(integer linear programming, ILP)은 전산학, 산업공학 및 전자공학 등 많은 분야에서 최적화 문제를 해결하기 위한 방법으로 사용되고 있다.

선형 프로그래밍을 이용한 최적화 문제는 유한개의 제약조건인 선형부등식 또는 선형방정식을 만족하면서, 선형 함수 $f(x_1, x_2, \cdots, x_n)$가 최소 또는 최대를 갖도록 하는 $x_1, x_2, \cdots, x_n$을 결정하는 문제로 정의할 수 있다. 이때의 선형함수 $f(x_1, x_2, \cdots, x_n)$을 목적함수(object function)라 하고, 조건을 나타내는 선형부등식 또는 선형방정식을 제약조건(constraints)이라 한다. 모든 제약조건을 만족하는 해의 집합을 실행가능 영역(feasible region)이라 하고, 실행가능 영역 내에서 목적함수의 값이 최대이거나 최소인 해를 최적해(optimal solution)라 한다[11].

일반적인 LP 문제는 다음과 같이 나타낼 수 있다.

$$\begin{pmatrix} Min \\ Max \end{pmatrix} f(x) = c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$$
$$\text{제약조건} \quad \begin{array}{l} a_{11} x_1 + a_{12} x_2 + \cdots + a_{1n} x_n \geq (,=,\leq)\ b_1 \\ a_{21} x_1 + a_{22} x_2 + \cdots + a_{2n} x_n \geq (,=,\leq)\ b_2 \\ \quad\vdots\ \ \vdots\ \ \vdots\ \ \vdots \\ a_{m1} x_1 + a_{m2} x_2 + \cdots + a_{mn} x_n \geq (,=,\leq)\ b_m \end{array}$$

특히, 모든 변수가 정수로 제한되는 선형 프로그래밍을 정수 선형 프로그래밍(ILP)이라 하며, 아래와 같이 행렬과 벡터로 간단히 나타낼 수 있다.

$$\begin{pmatrix} Min \\ Max \end{pmatrix} f(x) = Cx$$
$$\text{제약조건} \quad Ax \leq B,\ x \geq 0$$

C는 비용 벡터(cost vector)이고, A는 제약조건 행렬(constraint matrix)이고, B는 제약조건의 컬럼 벡터(column vector)이며, x는 정수 변수들의 벡터이다.

2.3 가산기 설계상의 문제점

그림 1은 단일 타입의 가산기 구조에 대한 설계공간이다. 바나나 모양의 각 곡선 상에 있는 (Delay, Area) 좌표 점에는 그에 상응하는 실제 가산기가 존재한다. 이 좌표점은 연산시간 x축과 회로면적 y축으로 표현된 가산기의 설계공간에서 파레토 최적해(Pareto optimal point)가 된다[8]. 연산시간과 회로면적은 서로 상반관계를 갖고 있으므로 Trade-off가 필요하다[1,4]. 그러나 전체 시간의 측면에서 볼 때, 계단형의 Trade-off 곡선을 형성한다. 이는 연산시간과 회로면적의 관계가 반비례 관계를 형성하지 못하는 비이상적인 결과를 초래한다. 즉, 지연시간에 대한 제약조건이 주어진 경우에 조건에 부합하는 가산기를 선택해야 함으로 회로면적을 최소화할 수 없다. 예를 들어, 그림 1의 ①과 ②로 마크된 영역에 시간제약이 주어졌다고 가정하자. 이 영역에서는 연산시간의 제약조건이 약화되고 있음에도 불구하고 회로면적이 큰 가산기를 선택한다. 그러나 가산기의 선택 범위가 넓어진다면 즉, 가산기의 설계공간이 확대된다면 보다 작은 회로면적을 가진 가산기를 선택할 수 있다.

그러므로 그림 1에서 점선으로 표현된 것과 같이 전체 시간 축 상에서 보다 반비례에 가까운 최적의 가산기 설계공간을 구성해야할 필요가 있다.

3. 혼합 가산기의 구조 및 특징

본 논문에서는 목적 시스템의 요구사항을 최대한 만족하는 가산기를 탐색하기 위하여, 서로 다른 캐리 전달

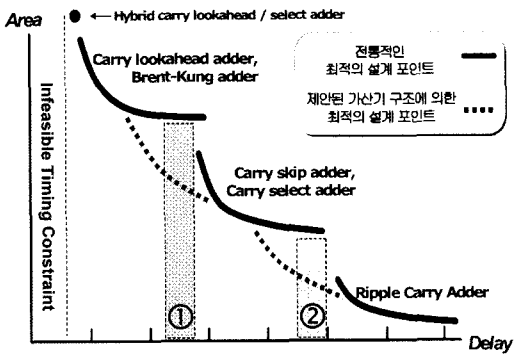

그림 1 가산기의 설계공간 (연산시간-회로면적 절충 관계)

방식을 결합한 혼합 가산기 구조를 제안한다.

3.1 혼합 가산기의 구조

혼합 가산기는 캐리 전달 방식이 서로 다른 가산기 블록들이 선형으로 연결된 구조를 갖는다. 설계하고자 하는 "n비트 가산기"는 다수의 가산기 블록들로 구성될 수 있다. 각 가산기 블록(CA)은 서로 다른 비트 폭을 가지며, 가산기 블록들의 비트 폭의 합은 n이다. 일반화하여 설명하면 다음과 같다. 가산기 블록 $CA_i(n_i)$는 캐리 전달 방식이 CA_i 형태인 n_i비트 가산기로 정의한다. 고려하고 있는 캐리 전달 방식의 종류가 I 가지 일 경우, 제안된 n비트 가산기는 I 개의 $CA_i(n_i)$가 선형적으로 연결된 n비트 가산기로 정의 된다 ($1 \le i \le I$). 그림 2와 같이 $CA_i(n_i)$의 캐리 입력은 $CA_{i-1}(n_{i-1})$의 캐리 출력 신호이고, n_i의 모든 합은 n이다. 즉, $I=3$이고 $n_1+n_2+n_3=n$이다. $n_i=0$ 일 경우 $CA_i(n_i)$는 $CA_{i-1}(n_{i-1})$에서 $CA_{i+1}(n_{i+1})$로 캐리를 전달하는 도선이 된다.

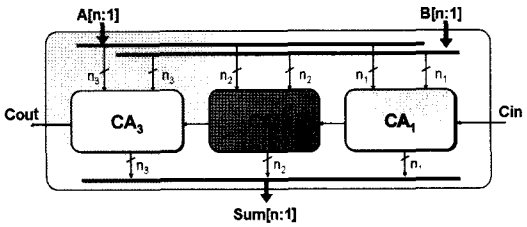

그림 2 제안한 혼합 가산기 구조

3.2 혼합 가산기의 특징

그림 3은 시스템 설계자의 가산기 선택 과정을 나타낸다. (a)에서는 단일 캐리 전달 방식만을 고려하고 있다. 그러므로 회로면적이 작은 가산기를 설계하기 위해 RCA구조가 선택되고, 고속 연산을 위한 가산기를 설계하기 위해 CLA구조가 선택된다. (b)에서는 혼합 가산기

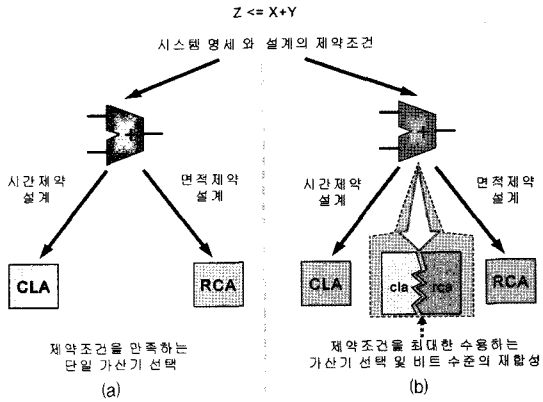

그림 3 제약조건을 만족하는 가산기의 선택 방법

구조를 고려하고 있다. 그러므로 주어진 제약조건을 만족하면서 다른 제약조건까지 최적화할 수 있는 가산기가 선택된다. 예를 들어, 임의 시간제약을 가진 64비트 가산기를 설계한다고 가정하자. 단일 타입의 가산기만을 고려한 설계공간에서는 64비트 RCA 또는 64비트 CSKA, 또는 64비트 CLA만이 탐색된다. 그러나 확장된 설계공간에서는 주어진 설계 조건에 따라 다양한 조합이 가능하며, 그림 3(b)와 같이 "캐리 신호에 의해 선형으로 연결된 RCA[63:33](31비트 RCA) CLA[32:0](33비트 CLA)" 혼합 가산기가 탐색될 수 있다. 이와 같이 혼합 가산기는 이종의 캐리 전달 방식을 비트 수준에서 조합함으로써 보다 세밀한 연산시간-회로면적 절충 관계를 갖는다.

그림 4는 그림 3(a)의 구현 방식(implementation selection)과 (b)의 구현 및 재합성 방식(implementation selection and bit-level recombination)을 나타내고 있다. 기존에는 제약조건을 만족시키기 위해, 라이브러리 내에서 가장 적합한 가산기를 검색한다. 그러나 제안하는 혼합 가산기 구조는 최적의 가산기를 검색하기 위하여 단일 캐리 전달 방식의 가산기들을 하나의 가산기로 조합한다. 따라서 주어진 제약조건을 만족시킬 뿐 아니라 다른

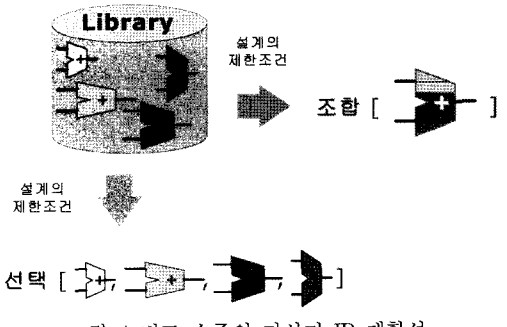

그림 4 비트 수준의 가산기 IP 재합성

제약조건까지 최적화할 수 있다. 이는 혼합가산기가 비트 수준에서 재합성되기 때문에 보다 미세한 최적화가 가능하다는 것을 나타낸다.

4. ILP 기반의 최적화

제안하는 혼합 가산기는 가산기 블록에 할당되는 비트 폭에 따라 다양하게 구성될 수 있다. 혼합 가산기 구조에 의해 생성되는 모든 조합을 탐색하는 데는 많은 시간이 소모되므로, 제약조건을 최대한 만족하는 최적의 가산기를 효과적으로 탐색하기 위하여 최적화 기법인 ILP를 이용한다.

4.1 최적화 문제 정의

혼합 가산기의 최적화 문제는 "n비트 가산기에 대하여, 각 가산기 블록들의 최적 비트 폭, $n_1, n_2,...,n_I$ 를 찾는 것"이다. 각 가산기 블록들에 대한 비트 폭은 제약조건에 따라 최적화된다. 제약조건의 대상은 연산시간과 회로면적이다. 연산시간과 회로면적은 절충 관계에 있으므로, 제약조건의 대상이 연산시간이면 최적화 대상은 회로면적이고 제약조건의 대상이 회로면적이면 최적화 대상은 연산시간이 된다.

1) 연산시간

최적화 대상을 연산시간으로 할 때 최적화 문제는 다음과 같이 정의된다.

[Problem 1]

n비트 가산기에 대하여, 회로면적에 대한 제약조건을 만족하는 가산기 회로의 연산시간을 최소화하는 가산기 블록들의 비트 폭, $n_1, n_2,...,n_I$ 를 찾는 것

최적화 문제를 정확하게 정의하기 위해 먼저, 제안하는 가산기 구조의 연산시간에 대해 살펴보고자 한다. 연산시간은 합의 생성시간과 캐리 전달시간과 관련된다.

제안하는 n비트 혼합 가산기는 그림 5와 같이 각 가산기 블록 $CA_i(n_i)$들이 선형적으로 연결된 것이며,

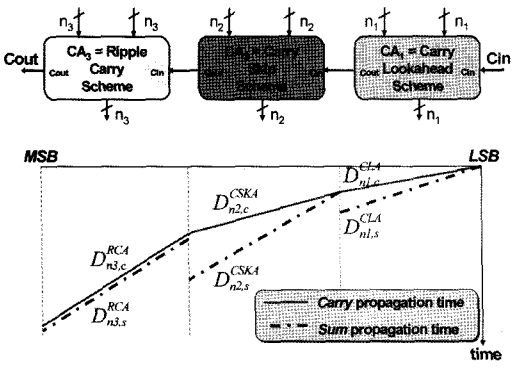

그림 5 각 가산기 블록에서 캐리 전달과 합의 생성 시간

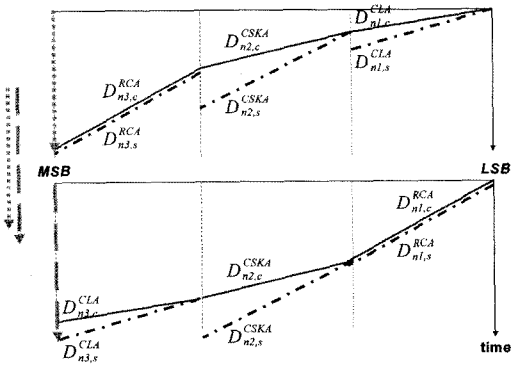

그림 6 가산기 블록의 위치조합에 따른 연산시간 비교

$CA_i(n_i)$의 캐리 입력은 $CA_{i-1}(n_{i-1})$의 캐리 출력 신호이다. 캐리의 전달지연은 가산기의 성능에 영향을 미친다. 일반적으로 가산기의 캐리는 LSB에서 MSB로 전달되며, 전달 지연시간은 비트 폭뿐만 아니라 캐리 전달 방식에 따라 다르다.

n비트 가산기의 연산시간은 첫 번째 비트에서 마지막 비트까지의 캐리 전달시간과 중간 중간 발생하는 각 $CA_i(n_i)$의 합 생성 시간 중 최대 시간을 합한 것으로 기술한다. $D_{n_i,s}^{CA_i}$와 $D_{n_i,c}^{CA_i}$는 비트 폭이 n_i인 i번째 가산기 블록에서의 합 생성 시간과 캐리 전달시간이다.

그림 5에서 연산시간은, 첫 번째 가산기 블록에서 마지막 가산기 블록까지 전달된 캐리 전달시간($D_{n_1,c}^{CLA} + D_{n_2,c}^{CSKA}$)과 마지막 가산기 블록의 합 생성시간 $D_{n_3,s}^{RCA}$의 합으로 계산된다. 이러한 연산시간의 계산 방법은 제안하는 혼합 가산기 구조에서 가능한 모든 가산기 조합에 적용되지 않는다. 왜냐하면, 가산기 블록에서 사용하는 캐리 전달 방식과 비트 폭 및 위치가 결정적이지 않기 때문이다. 가산기 블록의 캐리 전달 방식이나 할당된 비트 폭에 따라 캐리 전달 시간과 합 생성시간이 다르기 때문에 어떤 가산기 블록에서 합 생성이 가장 늦는지 알 수 없다.

그림 6의 하단 그림에서 연산시간은 두 번째 가산기 블록의 합 생성 시간에 의해 결정된다. 각 가산기 블록의 캐리전달 방식과 할당된 비트 폭과 위치에 따라 합 생성시간 $D_{n_i,s}^{CA_i}$와 캐리 전달시간 $D_{n_i,c}^{CA_i}$가 달라지므로, 가산기의 연산시간을 정확히 계산하기 어렵다. 예를 들어 CLA와 CSKA를 살펴보면, 비트 폭이 큰 경우에는 CLA의 캐리 전달시간이 CSKA보다 빠르지만 그렇지 않은 경우에는 CLA의 회로 오버헤드로 인하여 CSKA의 캐리 전달 시간이 빠를 수 있다. 그림 6에서 연산시간은 첫 번째 가산기 블록의 캐리 전달시간 $D_{n_1,c}^{RCA}$과

두 번째 가산기 블록의 합 생성시간 $D_{n_2 s}^{CSKA}$의 합으로 계산된다. 그러므로 최종 연산시간을 계산하기 위한 일반화 된 방법이 필요하다. 가산기의 연산시간을 정확하게 계산하기 위해, 각 가산기 블록마다 합 생성시간을 검사해야 한다. I 개의 가산기 블록들에 대하여, 합 생성은 병렬적으로 발생할 수 있다.

그림 7은 최종 연산시간을 계산하는 알고리즘을 나타낸 것이다. I 는 가산기 블록의 개수를 나타내며, 각 가산기 블록마다 부분적으로 연산시간을 계산한다. 부분적인 연산시간 중 최대값이 혼합 가산기의 최종적인 연산시간이 된다. i번째 가산기 블록에서의 부분 연산시간은 식 (1)과 같이 첫 번째 가산기 블록부터 i번째 가산기 블록까지 전달되어온 캐리 지연시간과 i번째 가산기 블록에서의 합 생성시간의 합으로 나타낼 수 있다.

```
Sum[0] = Carry[0] = 0;
For i = 1 to I
Begin
        Sum[i] = Carry[i-1] + D^{CA_i}_{n_i s};
        Carry[i] = Carry[i-1] + D^{CA_i}_{n_i c};
End

CompletionTime = Max_i { Sum[i] };
```

그림 7 최종 연산시간에 대한 알고리즘

$$Sum[i] = S_i = D_{n_1 c}^{CA_1} + D_{n_2 c}^{CA_2} + \cdots + D_{n_{i-1} c}^{ACi-1} + D_{n_i s}^{ACi}$$

$$= S_i = \sum_{k=1}^{i-1} D_{n_k c}^{CA_k} + D_{n_i s}^{CA_i} \cdots \quad (1)$$

2) 회로면적

최적화 대상을 회로면적으로 할 때 최적화 문제는 다음과 같이 정의된다.

[Problem 2]

n비트 가산기에 대하여, 연산시간에 대한 제약조건을 만족하는 가산기의 회로면적을 최소화하는 가산기 블록들의 비트 폭, $n_1, n_2, \ldots, n_I$ 를 찾는 것

회로면적은 IC의 설계과정의 마지막 단계인 배치 및 배선 과정에 따라 달라질 수 있다. 두 개의 회로를 합성한 회로는 각 회로를 더한 크기가 될 수도 있고, 또는 작거나 클 수 있다. 본 논문에서는 선형적으로 최적화 문제를 해결하기 위하여, 회로면적이 선형적 증가한다고 가정하였다. 따라서 가산기 블록 A와 B의 면적은 식 (2)와 같이 정의할 수 있다.

$$Area(A) + Area(B) \approx Area(A+B) \quad (2)$$

4.1 ILP 형식화

각 가산기 블록에 비트 폭을 할당하는 문제는 가산기의 연산시간을 최소화하거나 회로면적을 최소화함으로써 최적화할 수 있다. 이때, 연산시간과 회로면적은 각각 면적과 시간에 대한 제약조건을 갖는다. 혼합 가산기는 다양한 캐리 전달 방식을 갖는 가산기 블록들로 구성될 수 있기 때문에, 회로면적과 최종 연산시간은 단순히 선형적이지 않으며 그 값을 구하는 문제는 NP-hard하다. 이와 같이 NP-hard한 최적화 문제를 정수 선형 프로그래밍(integer linear programming)을 사용하여 해결하기 위해 ILP 모델을 제시한다.

1) 연산시간 최적화

연산시간을 최적화하는 [Problem 1]은 식 (3)과 같이 형식화 될 수 있다.

목적함수 : Minimize

$$\underset{i}{Max}\{DELAY(CA_i)\}$$

제약조건 :

$$\sum_{i=1}^{I} AREA(CA_i) < UB$$

$$\sum_{i=1}^{I} n_i = n \quad (3)$$

식 (3)의 $AREA$와 $DELAY$는 i번째 가산기 블록 CA_i에서의 연산시간과 회로면적을 계산하는 함수이다. RCA를 제외한 CSKA와 CLA의 연산시간과 회로면적에 대한 함수는 가산기 블록에 주어진 비트 폭에 대해 비선형적이다. 비선형 함수의 문제는 설계공간의 탐색을 어렵게 한다.

이처럼 복잡한 ILP 문제를 해결하기 위하여, 가능한 캐리 전달 방식과 가능한 비트 폭에 대해 연산시간과 회로면적을 계산한 후 이 값을 새로 도입한 변수와 조합하여 비선형 함수를 선형화하였다. 새로 도입한 변수는 특정 캐리 전달 방식과 특정 비트 폭이 할당되었는지를 나타낸다.

연산시간에 대한 비선형 문제를 식 (4)와 같이 선형식으로 나타내었다. 먼저, 가산기 블록에 할당 가능한 비트 폭에 따라 합 생성시간 $D_{n_i s}^{CA_i}$ 및 캐리 전달시간 $D_{n_i c}^{CA_i}$을 미리 계산하였다. 후에 계산된 값과 새로 도입한 변수 $x_{n_i}^{CA_i}$를 조합하여 전체 가산기의 연산시간을 기술하였다. 변수 $x_{n_i}^{CA_i}$는 i번째 가산기 블록에 비트 폭 n_i이 할당되었는지를 나타낸다. 할당되지 않으면 0, 할당되면 1 이상의 값을 가진다. 비트 폭 n_i는 0비트부터 n비트까지 가능하다($1 \le i \le I$). 그러므로, i번째 가산기 블록에서 연산시간은 식 (4)와 같이 기술된다.

$$S_i = \sum_{k=1}^{i-1}\sum_{n_k=0}^{n} D_{n_k c}^{CA_k} \cdot x_{n_k}^{CA_k} + \sum_{n_i=0}^{n} D_{n_i s}^{CA_i} \cdot x_{n_i}^{CA_i} \quad (4)$$

가산기의 최종 연산시간은 $Max_i\{S_i\}$이다.

회로면적에 대한 비선형 문제는 식 (5)와 같이 선형식으로 나타내었다. 각 가산기 블록에 할당 가능한 비트 폭에 따라 회로면적 $A_{n_i}^{CA_i}$를 계산한 후 변수 $x_{n_i}^{CA_i}$와 조합하여 전체 가산기의 회로면적을 식 (5)와 같이 기술하였다.

$$AREA = \sum_{k=1}^{I}\sum_{n_k=0}^{n} A_{n_k}^{CA_k} \cdot x_{n_k}^{CA_k} \quad (5)$$

식 (3)의 목적 함수는 Min-Max 비선형 문제이다. Min-Max 문제를 해결하기 위해 새로운 변수와 제약조건을 추가하였다. 새로운 변수 z는 식 (4)의 연산시간 S_i의 최대값을 의미한다. 그러므로 목적함수는 식 (6)과 같이 나타낼 수 있고, z를 최소화하는 함수이다.

Minimize z

$$z = \max_i\{S_i\} \quad (6)$$

식 (6)이 의미하는 것처럼, 모든 i에 대해 $S_i \leqq z$이 성립해야 한다. 그러므로 식 (3)의 제약조건 식에 $S_i \leqq z$이 추가되어야 한다. 가산기의 전체 비트 폭 n은 변수 $x_{n_i}^{CA_i}$와 조합하여 식 (7)과 같이 나타낸다.

$$\sum_{i=1}^{I} n_i x_{n_i}^{CA_i} = n \quad (7)$$

결과적으로, 연산시간을 최적화하기 위한 ILP 모델은 그림 8과 같이 기술할 수 있다.

목적함수 : Minimize CompletionTime
= Minimize $Max_i\{S_i\}$
= Minimize z

제약조건 :

$$\sum_{k=1}^{i-1}\sum_{n_k=0}^{n} D_{n_k c}^{CA_k} \cdot x_{n_k}^{CA_k} + \sum_{n_i=0}^{n} D_{n_i s}^{CA_i} \cdot x_{n_i}^{CA_i} \leqq z$$
$$for\ all\quad 1 \leqq i \leqq I$$

$$AREA = \sum_{k=1}^{I}\sum_{n_k=0}^{n} A_{n_k}^{CA_k} \cdot x_{n_k}^{CA_k} < UB$$

$$\sum_{i=1}^{I} n_i x_{n_i}^{CA_i} = n$$

$$x_{n_i}^{CA_i} \geqq 0 \quad and \quad Integer$$
$$for\ all\quad 1 \leqq i \leqq I,\quad 1 \leqq n_i \leqq n$$

그림 8 연산시간 최적화를 위한 ILP 모델

2) 회로면적 최적화

회로면적을 최적화하는 [Problem 2]는 식 (8)과 같이 형식화 될 수 있다.

목적함수 : Minimize

$$\sum_{i=1}^{I} AREA(\ CA_i\ )$$

제약조건 :

$$\max_i\{DELAY(\ CA_i\ )\} < UB$$

$$\sum_{i=1}^{I} n_i = n \quad (8)$$

식 (8)의 회로면적은 식 (5)와 같고, 연산시간은 식 (4), 식 (6)과 같다. 따라서 회로면적을 최적화하기 위한 ILP 모델은 그림 9와 같이 기술할 수 있다.

목적함수 : Minimize Area
= Minimize $\sum_{k=1}^{I}\sum_{n_k=0}^{n} A_{n_k}^{CA_k} \cdot x_{n_k}^{CA_k}$

제약조건 :

$$\sum_{k=1}^{i-1}\sum_{n_k=0}^{n} D_{n_k c}^{CA_k} \cdot x_{n_k}^{CA_k} + \sum_{n_i=0}^{n} D_{n_i s}^{CA_i} \cdot x_{n_i}^{CA_i} \leqq UB$$
$$for\ all\quad 1 \leqq i \leqq I$$

$$\sum_{i=1}^{I} n_i x_{n_i}^{CA_i} = n$$

$$x_{n_i}^{CA_i} \geqq 0 \quad and \quad Integer$$
$$for\ all\quad 1 \leqq i \leqq I,\quad 1 \leqq n_i \leqq n$$

그림 9 회로면적 최적화를 위한 ILP 모델

5. 실험 및 결과 분석

혼합 가산기의 타당성과 유용성을 평가하기 위하여, 128비트 혼합가산기의 설계공간을 탐색하는 실험을 하였다. 이 실험은 4장에서 형식화한 ILP 문제를 해결하는 것으로, Eindhoven University of Technology에서 개발한 public-domain ILP 소프트웨어인 lp_solve 4.0을 사용하였다[10]. 실험에 사용된 128비트 혼합가산기는 세 개의 가산기 블록 RCA, CSKA, CLA 순으로 구성하였고, 각 가산기 블록에 할당될 수 있는 비트 폭은 4, 8, 16, 32, 64, 128로 제한하였다. 표 1은 실험에 사용된 상수값으로 [9]에서 제안한 모델을 프로그래밍하여 계산한 값이다. 각 상수는 가산기 블록에 할당 가능한 비트 폭에 따른 합 생성시간 $D_{n_i s}^{CA_i}$와 캐리 전달시간 $D_{n_i c}^{CA_i}$ 그리고 회로면적을 나타내는 $A_{n_i}^{CA_i}$이다.

앞에서 정의한 ILP 문제의 최적해는 그림 10과 같이 연산시간 Delay와 회로면적 Area로 표현되는 설계공간에서 하나의 포인트로 표시된다. '■'로 표시된 포인트는

표 1 계산된 $D_{n_i s}^{CA_i}$, $D_{n_i c}^{CA_i}$ 및 $A_{n_i}^{CA_i}$

가산기 타입	비트 크기	합 생성시간 $D_{n_i s}^{CA_i}$	캐리 전달시간 $D_{n_i c}^{CA_i}$	회로면적 $A_{n_i}^{CA_i}$
RCA	4	11.933	10.5533	442.23
	8	22.333	20.9533	883.29
	16	43.133	41.7533	1765.41
	32	84.733	83.3533	3529.65
	64	167.933	166.553	7058.13
	128	334.333	332.953	14115.10
CSKA	4	11.9333	10.5333	442.23
	8	22.4867	13.5933	956.38
	16	28.5667	19.6733	1984.70
	32	40.7267	31.833	4041.32
	64	65.0467	56.1533	8154.56
	128	113.687	104.793	16381.00
CLA	4	11.1467	9.88	687.31
	8	16.50	9.88	1434.16
	16	16.586	14.3	2949.97
	32	21.94	14.3	5959.48
	64	22.0267	18.92	12000.60
	128	27.38	18.92	24060.80

그림 1의 점선 상의 포인트에 해당되고, '+'로 표시된 포인트는 그림 1의 바나나 모양 곡선 상의 포인트에 해당된다. '■'로 표시된 최적의 포인트들은 설계비용을 최적화하는데 도움을 준다. 그림 10의 ①과 ②의 설계 포인트는 제안한 혼합 가산기 구조에 의해 새로 생성된 최적 포인트의 일부이다. 64비트 CLA와 64비트 CSKA로 구성된 가산기는 단일 타입 가산기 CLA만으로 구성된 가산기보다 회로면적이 작다. 또한 Delay 축의 범위가 약 150에서 340사이의 범위 내에 있을 때, 혼합 가산기 구조에 의해 생성된 ②에 해당하는 혼합 가산기는 단일 타입 가산기 CSKA보다 작은 크기의 회로로 구현될 수 있다. 단일 타입 가산기와 혼합 가산기가 모두 연산시간에 대한 제약조건을 만족하는 경우, 혼합 가산기를 선택함으로써 회로면적이 단일 타입 가산기보다 작은 가산기를 구현할 수 있음을 나타낸다. 그림 11은 최적의 설계포인트가 어떤 형태로 조합되는지를 나타낸다. 이 실험을 통하여 혼합 가산기 구조가, 단일 타입 가산기만을 고려한 기존의 설계공간에서는 볼 수 없었던 설계 포인트인 파레토 최적해를 생성한다는 것을 확인하였다. 이러한 설계 포인트들은 연산시간과 회로면적에 대해 최적의 Trade-off 관계를 갖는다.

그림 12는 회로면적이 제약조건으로 주어졌을 때 연산시간을 최소화 하는 최적해를 찾아 연산시간-회로면적 곡선으로 나타낸 것이다. 이는 회로면적에 대한 제약조건을 수용하며 연산시간이 최소인 저비용 고성능 가산기의 구현이 가능하다는 것을 나타낸다.

이상으로 RCA-CSKA-CLA와 CLA-CSKA-RCA 순서조합으로 구성한 혼합가산기의 설계공간에 대해 살펴보았다. 다른 순서조합을 가진 혼합가산기의 설계공간과 위에서 실험한 결과를 비교하기 위하여, RCA와 CSKA와 CLA로 가능한 모든 순서조합을 고려하여 실험하였다. 구성 가능한 순서조합은 모두 6가지이다. 그림 13은 6가지 혼합가산기의 설계공간을 나타낸다. 이는 6가지 순서조합 모두 단일 타입 가산기보다 최적화된 혼합가산기를 설계할 수 있다는 것을 나타낸다.

6. 결론 및 향후과제

디지털 시스템 설계자들의 목표는 최소의 개발비용으로 최고 성능을 발휘하는 시스템을 설계하는데 있다. 그리고 시스템의 성능은 가산기의 성능에 의해 좌우된다. 이것은 가산기가 디지털 시스템에서 가장 기본적인 연산회로이기 때문이다. 이러한 가산기의 성능은 연산 속도와 구현 비용으로 나타낼 수 있으며, 캐리의 전달 지연을 처리하는 방식에 영향을 받는다. 그러므로 캐리의 전달 속도를 향상시키기 위한 설계 방법들이 다양하게 연구되어 왔다[2,3].

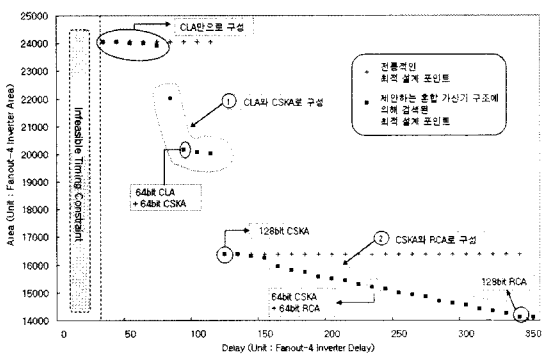

그림 10 128비트 혼합가산기의 설계공간

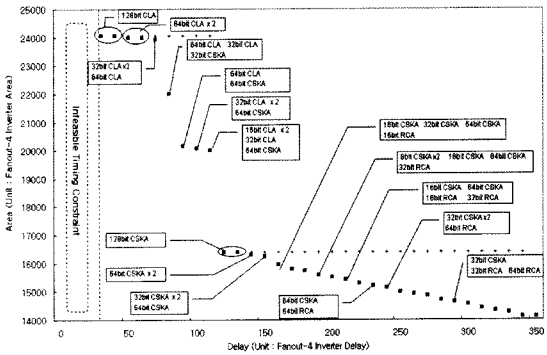

그림 11 설계공간 상의 혼합 가산기 조합

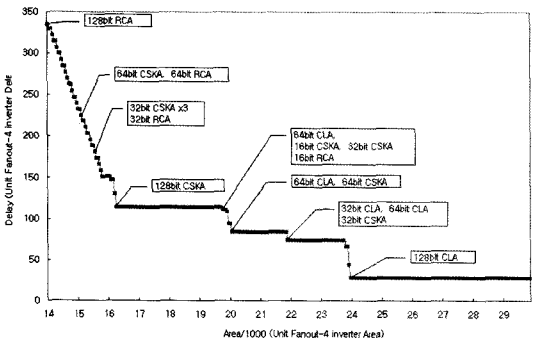

그림 12 제약조건이 회로면적인 연산시간의 최적화

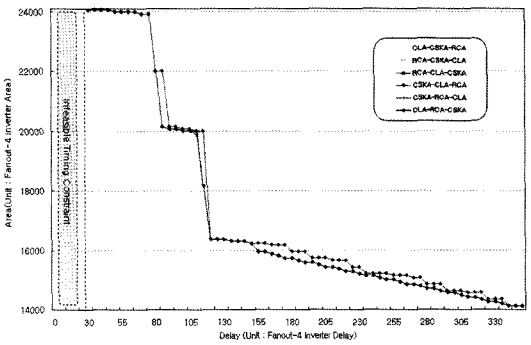

그림 13 모든 순서조합의 연산시간-회로면적 곡선

가산기에 대한 기존 연구들은 대부분 캐리를 고속으로 전달함으로써 연산 속도를 향상시키는데 목적을 두고 있기 때문에, 회로면적에 대한 고려가 상대적으로 적다. 그러나 회로면적은 개발비용과 밀접한 관계를 갖고 있기 때문에, 시스템 설계 시 고려해야 할 주요 요소 중의 하나이다. 동일한 연산속도를 가진 가산기가 있다고 할 때, 설계자는 설계비용을 최소화하기 위하여 회로면적이 작은 가산기를 선택해야 한다. 그러므로 연산속도뿐 아니라 회로면적도 고려하여 최적의 가산기를 설계해야 할 필요가 있다.

본 논문에서는 목적하는 시스템의 요구사항에 최대한 부합되는 가산기를 설계하기 위하여, 혼합 가산기 구조를 제안하였다. 혼합 가산기 구조는 다양한 설계 포인트들을 만들어낸다. 이러한 설계 포인트들은 구현될 수 있는 혼합 가산기들을 의미한다. 혼합 가산기는 다양한 캐리 전달 방식을 가진 가산기 블록을 선형으로 연결한 구조이다. 서로 다른 Delay-Area 특성을 갖는 이종의 가산기 블록들은 서로 다른 비트 폭으로 구성되며, 캐리 입력 및 캐리 출력 신호를 통해 선형적으로 연결됨으로써 비트 수준에서 재합성된다. 혼합 가산기에 사용된 가산기 블록의 종류와 개수 그리고 가산기 블록의 순서조합에 따라 매우 다양한 가산기 조합이 존재한다. 이러한 가산기 조합 중에서 목적시스템의 연산속도와 회로면적에 대한 제약조건을 만족하는 최적의 가산기 조합을 선택함으로써, 시스템의 요구사항을 만족시킬 수 있다.

혼합 가산기 구조는 단일 가산기의 종류와 개수 그리고 순서조합에 따라 매우 다양한 가산기 조합이 존재하므로, 최적의 가산기를 검색하는데 많은 시간이 소모된다. 이러한 문제를 해결하기 위하여 ILP(integer liner programming)를 사용함으로써, 혼합 가산기 구조에 의해 생성되는 가산기 조합들 중 최적의 가산기 조합을 효과적으로 검색할 수 있다. ILP는 최적화 문제를 해결하기 위해 수학적인 모델링을 이용하는 기법이다. 그러므로 최적화 문제를 혼합 가산기의 각 가산기 블록에 할당되는 비트 폭을 결정하는 문제로 정의하여 혼합 가산기를 수학적으로 모델링하였다. 모델링된 혼합 가산기의 최적해를 찾기 위하여 ILP solver를 이용하여 실험하였다. 실험에 사용된 혼합 가산기는 128비트이고, 사용된 가산기 블록은 RCA, CSKA, CLA 세 가지이다. ILP는 다양한 혼합 가산기를 생성한다. 이때 세 가산기 블록의 비트 폭은 다양하게 구성될 수 있다.

ILP를 이용하여 연산시간과 회로면적을 대상으로 혼합 가산기를 최적화였다. 실험 결과, 다양한 가산기의 설계 포인트들이 설계공간에 나타났다. 이러한 설계 포인트들은 단일 타입 가산기만을 고려한 기존의 설계공간에서 볼 수 없었던 포인트들로서 파레토 최적해(Pareto optimal point)를 의미한다. 혼합 가산기 구조에 의해 확장된 설계공간의 설계 포인트들은 최적의 연산시간과 회로면적에 대해 Trade-off 관계를 갖는다. 이와같이 ILP를 이용하여 확장된 가산기의 설계공간을 효과적으로 탐색함으로써 주어진 설계의 제약조건 즉, 목적 시스템의 요구사항에 부합되는 최적의 가산기를 설계할 수 있다.

최근 디지털 시스템의 설계 동향은 보다 빠르고 작은 회로를 설계하고, 보다 적은 전력량을 소모하도록 설계하는 것이다. 이러한 설계 동향을 반영하는 시스템을 구현하기 위해, 해당 시스템의 요구사항에 부합되는 최적의 가산기가 반드시 필요하다. 본 논문에서는 최적의 가산기를 설계하기 위하여 혼합 가산기 구조를 제안하였다. 이 구조는 단일 타입의 가산기들을 비트 수준에서 재합성하기 때문에 미세한 최적화를 수행할 수 있다. 비트 폭을 파라미터로 하는 Soft Adder IP에 기반을 두고 있으므로 상위수준 합성에 효과적으로 사용될 수 있을 것이다. 또한 이동통신 기기뿐 아니라 개인용 컴퓨터에서 대형컴퓨터에 이르기까지 다양한 분야에서 혼합 가산기를 사용하여, 최소의 개발비용으로 최고의 성능을

발휘하는 시스템을 설계할 수 있을 것으로 판단된다. 그러므로 본 논문에서 제안한 혼합 가산기는 실용적인 측면에서 기여도가 높다고 할 수 있다.

더불어 본 논문에서 제안한 수학적 모델링 기법은 전력소모를 최적화하기 위한 방법으로도 사용될 수 있으며, 향후 연산시간-회로면적-소모전력을 동시에 고려하는 3차원적인 최적화에 대한 연구도 계속되어야 할 것이다.

참 고 문 헌

[1] C. Nagendra, M.J. Irwin, R.M. Owens, "Area - time - power tradeoffs in parallel adders," In IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing, vol. 43, pp. 689-702, Oct. 1996.

[2] M.D. Ercegovac and T. Lang, "Digital Arithmetic," Morgan Kaufmann Publishers, 2004.

[3] James E. Stine, "Digital Computer Arithmetic Datapath Design Using Verilog HDL," Kluwer Academic Publishers, Nov. 2003.

[4] J. Zhu and R. Kelly, "Architectural Diversity - The Key to Design Compiler Optimization," In DESIGN WARE, Synopsys Technical Bulletin, vol. 3, Q2 1998.

[5] Anu Gupta, "Programmatic design space exploration through validity filtering and quality filtering"

[6] Y. Wang, C. Pai, X. Song, "The design of hybrid carry-lookahead/carry-select adders," In IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing, vol. 49, Jan. 2002.

[7] H.P. Williams, "Model Building in Mathematical Programming," 4th Ed., John Wiley, New York, 1999.

[8] Giovanni De Micheli, "Synthesis and Optimization of Digital Circuits," McGraw-Hill Jan., 1994.

[9] D.E. Williams, E.E. Jr. Swartzlander, "Parametric delay and area models for adders," In Proc. of the 36th Midwest Symposium on Circuits and Systems, pp. 863 - 870, Aug. 1993.

[10] M. Berkelaar, "lp_solve - version 4.0," Eindhoven University of Technology, "ftp://ftp.ics.ele.tue.nl/pub/lp_ solve/," 2003.

[11] 오세영, "최적화이론", 교우사

이 덕 영

1993년 강원대학교 전자계산학과 이학사 1999년 강원대학교 컴퓨터과학과 이학석사. 2006년 강원대학교 컴퓨터과학과 이학박사. 2004년~현재 한림대학교 기초교육대학 강의전임강사. 관심분야는 digital arithmetic, secure system

이 정 근

1996년 한림대학교 전자계산학과 공학사 1998년 GIST 정보통신공학과 공학석사 2005년 GIST 정보통신공학과 공학박사 2005년~현재 캠브리지대학교 컴퓨터랩 박사후과정. 관심분야는 concurrent systems, asynchronous circuit design and CAD

이 정 아

1982년 서울대학교 컴퓨터공학과 공학사 1985년 인디애나 주립대학교 컴퓨터학과 공학석사. 1990년 캘리포니아 주립대학교 (UCLA) 컴퓨터공학과 공학박사. 1990년~1995년 Assistant Professor, University of Houston USA. 1995년~현재 조선대학교 컴퓨터공학과 교수. 관심분야는 computer architecture, fast digital arithmetic, application specific VLSI architectures

이 상 민

1976년 경북대학교 전자공학과 공학사 1979년 경북대학교 전자공학과 공학석사 1993년 경북대학교 전자공학과 공학박사 1984년~현재 강원대학교 컴퓨터과학과 교수. 관심분야는 컴퓨터구조, 디지탈시스템설계