論文 91-28A-5-8

Fanout 제약 조건하의 논리 회로 합성

(Fanout Constrained Logic Synthesis)

李 在 亨* 黃 善 泳**

(Jae Hyeong Lee and Sun Young Hwang)

要 約

본 논문에서는 주어진 시간 제약 조건과 fanout 제약 조건을 만족하는 회로를 최소의 면적으로 자동 생성하는 성능 구동 자동 논리 설계 시스템의 설계와 구현에 대하여 기술한다. 기술 독립적인 최적화 및 기술 의존적인 최적화가 수행된 후, 시스템은 과다한 fanout으로 큰 부하 지연시간을 가지는 게이트를 선택하여 임계경로를 제거하기 위한 재설계를 수행한다.

MCNC 벤취마크 회로에 대한 실험 결과에서 제안된 시스템은 최대 20%에 이르는 지연시간이 감소된 회로를 생성함을 볼 수 있다.

Abstract

This paper presents the design and implementation of a performance-driven logic synthesis system that automatically generates circuits satisfying the given timing and fanout constraints in minimal silicon area. After performing technology independent and dependent optimization, the system identifies and resynthesizes the gates with large loading delay due to excessive fanouts to eliminate the critical path.

Experimental results for MCNC benchmark circuits show that proposed system generates the circuits with less delay by up to 20%.

I. 서 론

논리 설계는 주어진 기능의 동작을 만족하여야 하며 최소의 면적 상에서 구현이 가능하여야 칩의 제조 비용을 감소시킬 수 있다. 최소 면적으로 구현된 회로의 동작속도가 시간 제약을 만족하지 못할 경우 그 회로가 사용되는 전체 시스템의 동작속도를 저하시키거나 오동작을 유발할 수 있으므로, 회로 면적과 동작 속도 제약을 모두 고려하여 논리회로를 설계하여야 한다. 그러나 면적과 동작 속도를 모두 고려하여 설계할 때는 회로 설계의 복잡성이 증가하여, 종래의 수작업에 의한 회로 설계 방식으로는 설계시간에 있어서 뿐만 아니라 설계된 회로 자체의 성능면에서도 비효율적인 결과를 가져오게 되었다.

이러한 문제점을 극복하기 위하여 최근 넓은 탐색 영역에서 최적의 회로를 단시간내에 설계할 수 있는 CAD tool의 개발이 활발히 진행되고 있으며, 특히 논리 회로의 설계에 있어서 최소의 면적에서 빠른 동작 속도를 가지는 회로를 얻을 수 있는 시스템들이 연구 개발되었다.[1][2][3][4][5] 회로의 동작이 동작 시간 제약 내에 이루어지는 최소 면적의 회로를 설계하는 방식으로 rule-based 방식과 DAG를 이용한 테크놀로지 매핑 방식이 주로 사용되고 있다. SOCRATES,[4] LSS[2]에서 사용된 rule-based 방식은 특정한

*準會員, **正會員, 西江大學校 電子工學科
(Dept. of Elec. Eng., Sogang Univ.)

接受日字 : 1991年 1月 23日

패턴에 대하여 변환 가능한 패턴을 rule로 설정하고 대상 회로에 대하여 향상된 성능의 회로를 구성할 수 있는 rule을 찾아내어 적용함으로써 최적화된 회로를 생성한다. Rule의 선택및 적용이 회로의 일부분에 한정되므로 국소 범위의 최적화를 방지하기 위하여 rule의 선택을 관리하는 meta-rule을 사용한다. 이 방식의 시스템은 사용자에 의한 rule의 정의가 가능하도록 유연성을 제공하며 시스템 설계가 용이하게 되나 rule의 설정에 어려움이 따르며 라이브러리의 변화에 따라 재설정 되어야하는 문제점을 가진다. MIS,[3] DAGON[4]는 DAG를 이용한 그래프 커버링 방식으로 rule-based 방식의 한계성이 되는 국소 범위의 최적화와 rule 설정의 라이브러리의 의존성을 제거하였다. 이 방식은 몇개의 기본적인 게이트 타입을 설정하고, 라이브러리에서 제공되는 게이트들을 기본적인 게이트로 재구성하여 DAG형태의 탐색 패턴을 이루며, 대상 회로를 동일한 기본 게이트로 재구성하여 대상 패턴을 이룬다. 대상 패턴에 대하여 매핑 가능한 모든 탐색 패턴에서 최적화된 회로의 구현이 가능한 적용 패턴을 선택함으로써 이론적으로 전체 범위의 최적화가 가능하다. 적용 패턴 선택은 simulated annealing 기법[6]이 주로 이용되며 많은 계산 시간이 요구된다.

그러나 MIS, SOCRATES 등의 시스템에서는 게이트의 지연시간을 게이트의 타입에 따라 부하의 영향을 고려하지 않고 고정된 시간 정보로부터 추출하여 사용함으로써, 칩으로 구현된 회로가 설계된 회로의 동작 특성과 차이를 보일 수 있다. 또한 한 게이트가 여러개의 게이트를 구동하는 경로는 과부하로 인해 긴 지연시간을 가질 수 있으나 이 지연시간의 정보를 얻을 수 없으므로 이에 대한 재설계 과정의 지원이 불가능하다. 따라서 설계된 회로가 동작 시간 제약 조건을 만족한다고 예상되더라도 칩으로 구현하였을 경우 동작 시간 제약을 만족하지 못하는 임계 경로[8]가 발생할 수 있다. 그러므로 게이트가 구동하는 부하의 크기에 비례하는 지연시간 정보를 사용하여 그 게이트의 지연시간을 나타냄으로써, 설계한 회로에서 예상할 수 있는 동작 특성을 칩으로 구현된 회로의 특성에 보다 근접시키고 과부하에 의하여 생기는 임계 경로의 추출이 가능하게 하여야한다. 추출된 임계 경로는 재설계하여 동작 시간 제약 조건을 만족하도록 하며, 아울러 fanout 제약 조건의 정보를 사용하여 이를 만족하지 못하는 게이트가 제거된 회로를 재생성함으로써 회로의 오동작을 방지하도록 하여야 한다.

이러한 단점을 보완하는 fanout 제약 조건하의 논리 재생성은 기술 의존적인 회로의 최적화와 동시에 수행하는 방식[9]이 제안 되었으나, 기술 의존적인 제약 조건과 fanout 제약 조건을 동시에 고려하여야 하는 이유로 탐색 영역이 지나치게 증가하고 모든 기술 의존적인 탐색 영역에 대하여 fanout 제약 조건을 검사해야 하므로 불필요한 탐색 영역을 가질 수도 있으며 시스템의 구현이 어려운 단점을 가진다. 따라서 참고문헌[9]에서는 버퍼 또는 인버터를 트리 구조로 사용하는 방법만이 제안되었다. 이러한 방식은 논리의 최적화와 기술 매핑을 동시에 이루어지는 rule-based 시스템에서도 사용이 가능하지만 각 rule의 기술에 대하여 fanout 제약 조건이 첨가되어야 하므로 rule의 작성에 많은 어려움이 따르며 rule의 탐색과 적용을 관리하는 meta-rule의 설정도 복잡하게 된다.

한편 기술 의존적인 최적화와 fanout 제약 조건을 고려한 회로 재생성을 독립된 패스로 수행하는 방식은 일단 최소의 면적상에서 동작 속도 제약조건을 만족하는 회로를 구현한 후 fanout 제약조건을 만족하지 못하는 일부의 게이트에 대하여 재설계하는 방법을 취할 수 있으므로 fanout 제약조건을 만족시킬 수 있는 넓은 설계 영역을 탐색하여도 전체 회로의 재생성 시간은 그다지 많은 시간이 소요되지 않는 장점을 가지며 시스템의 구현이 용이하다. 본 논문에서는 과부하에 의한 임계 경로와 fanout 제약조건을 만족하지 못하는 경로에 대한 재생성 알고리듬을 제안하고 그 실험 결과를 보이고자 한다. 이를 구현한 SiLOS (sogang intelligent logic optimization system)의 구성도를 그림 1에 보였다. SiLOS는 VHDL[7][12]을 입력으로 받아 동작 시간제약을 만족하지 못하는 임계 경로를 추출하여 사용자에게 그 정보를 제공하는 consultant mode와 임계 경로를 재설계하여 동작 시간 제약조건을 만족하는 최소 면적의 회로를 찾는 design mode로 구성되며, 재설계된 회로를 SunView 상에서의 그래픽 출력 혹은 VHDL 기술로 출력한다. Design mode는 기술 독립적인 회로 최소화를 거친후 기술 의존적인 회로의 최적화를 이루도록 한다. 기술 독립적인 회로의 최소화는 회로내의 일부에서 Boolean 식을 추출하여 불필요한 부분을 제거하는 방식으로 이루어 지며,[14] 기술 의존적인 회로의 최적화는 fanout 제약 조건을 고려하지 않으며 회로에서 사용되는 게이트의 지연시간을 평균적인 지연 시간으로 산출하여 사용된다. 또한 최적화를 이루는 게이트는 라이브러리에서 제공되는 여러가지 크기의 동일한 기능을 가지는 게이트 모듈 중에서 최소의 면적을 가지는 모듈로만 회로를 구성한다.[15][16]

Fanout 제약 조건하의 논리 재생성은 최소 크기의 모듈로만 이루어진 회로에 대하여 더 큰 크기의 게이트로 대체하거나 일부의 게이트를 첨가하는 방식으로 약간의 면적의 증가를 통하여 fanout 제약 조건을 만족시키며 회로의 동작 속도를 향상시켜 준다.

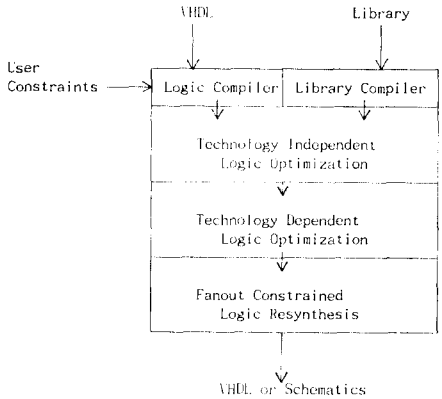

그림 1. SiLOS에서의 논리 회로 합성, 최적화 과정
Fig. 1. Logic synthesis and optimization processes in SiLOS.

Ⅱ. Fanout 제약하의 논리 재생성

한 게이트에 신호가 인가되어 다음단의 게이트에 출력이 전달되기까지의 시간을 논리 소자의 지연 시간이라 하며 고유 지연 시간(intrinsic delay), 부하지연 시간(loading delay), 연결선의 지연 시간(wire delay)으로 정의된다. 본 논문에서는 연결선에 의한 지연시간은 논외로 한다. 연결선에 의한 지연시간은 게이트의 정상적인 지연시간의 10% 이내가 되도록 레이 아웃하는 것이 보통의 design rule이 되기 때문이다.[10]

고유 지연시간은 부하 용량이 없을 경우 출력단에 신호가 형성되는 시간으로 내부 회로의 정전 용량을 충·방전하기 위하여 필요한 시간이 추가 된다. 그림 2에 부하 용량에 따른 게이트의 지연 시간 그래프를 보였다. 고유 지연시간은 X, Y점에 해당하는 지연시간이며 출력에 영향을 주는 신호가 어느 입력 단자에서 형성 되었는가에 따라 각기 다른 값을 가질 수 있다. 임계 경로상의 단순 게이트는 임계 경로가 되는 입력단의 위치를 고유 지연시간이 가장 작은 위치로 바꾸어 줌으로써 동작 시간 제약조건을 만족시킬 수 있다. 복합 게이트의 경우는 입력단이 서로 교환 가능한 쌍에 대하여 동일한 작업이 가능하다. 부하 용량의 크기는 게이트가 구동하는 게이트의 입력 용량들 합이다. 부하 용량을 이루는 각 게이트의 입력 단자들의 입력 용량은 단순 게이트의 경우 입력단의 위치에 따라 큰 차이가 없으나 복합 게이트의 경우는 입력단에 따라 많은 차이가 있을 수 있다. 게이트의 부하 지연시간은 게이트의 크기(channel width)에 따른 기울기를 가진다. 이때의 기울기를 '부하 저항'이라 정의하면 부하 지연시간은 부하 저항과 부하 용량의 곱으로 표현된다. 게이트 크기에 따라 부하 저항의 크기는 각각 다른 특성을 가지며, 그림 2에서와 같이 게이트의 크기가 클수록 작은 값을 가지고 게이트의 크기가 작을수록 큰 값을 가지게 된다. 부하 용량이 큰 경로에서는 부하 저항이 작은 게이트를 사용하여 부하 지연시간을 줄일 수 있으나 게이트의 크기가 클 경우는 고유 지연시간의 값이 커지게 되므로, 논리 회로내에서 임계 경로상의 게이트를 재설계 할때는 이들을 모두 고려하여 게이트의 크기를 조정한다. 그림2의 Z점은 게이트 크기 선택의 결정점이 된다.

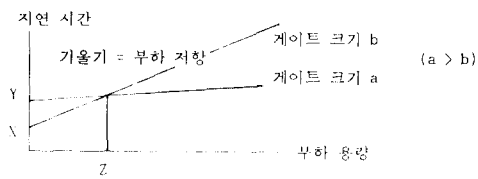

그림 2. 부하 용량에 따른 게이트의 지연 시간
Fig. 2. Delay vs. loading capacitance.

지연시간의 정보를 정확히 추출하기 위해서는 보다 세밀한 동작 정보가 기술된 라이브러리의 지원이 필요하다. 각 게이트의 동작 기술을 제공하는 라이브러리는 Synopsys사의 라이브러리 서술양식[13]을 사용하였으며 그 예를 그림 3에 보였다. 하나의 게이트 타입에 대하여 라이브러리는 여러가지 크기의 모듈을 제공할 수도 있으므로 라이브러리 컴파일러는 그림 3의 서술 양식으로부터 각 게이트 타입에 제공되는 다른 크기의 모듈을 면적의 크기에 따라 분류하여 linked list로 저장한다. SiLOS의 기술 의존적인 재설계 과정은 최소 크기의 게이트 종류를 사용하며, 본 논문에서 제시한 fanout 제약 조건하의 논리 재생성 알고리듬에서는 제공되는 모든 크기의 게이트 타입을 사용하여 회로를 최적화 한다.

```
library( example ) {
    default_max_fanout : 3 ;
    default_fanout_load : 1 ;
    cell( AND2 ) {
        area : 2 ;
        pin( A, B ) {
            direction : input ;
            capacitance : 10 pF ;
            fanout_load : 1 ;
            }
        pin( out ) {
            direction : output ;
            time() {
                intrinsic_rise_time : 2560 ps ;
                intrinsic_fall_time : 2240 ps ;
                rise_resistance : 12 ;
                fall_resistance : 13 ;
                related_pin : "A B" ;
                }
            function : "A B" ;
            max_fanout : 4 ;
            }
        }
}
```

그림 3. 라이브러리 기술 양식의 예
Fig. 3. An example library description.

게이트가 최대 fanout을 초과하는 게이트를 구동하는 경우 과부하로 인한 구동 속도 및 구동 신호 레벨에서의 손상을 유발할 수 있으므로 논리 회로의 각 논리 소자에 대해 fanout 제약 조건을 만족시키는 과정이 필요하게 된다. 특히 이러한 제약은 bipolar 트랜지스터에서 더욱 크게 나타난다. Fanout 제약조건은 대상 게이트가 구동하는 게이트들의 입력단에 명시된 fanout_load의 합이 max_fanout을 넘지 않도록 하여야한다. 그러나 시스템 구현의 편의성을 위하여 모든 입력단의 fanout_load를 동일한 기본단위로 가정하여 max_fanout 이내의 게이트가 구동되는 것을 fanout 제약 조건으로 한다.

Fanout의 제약조건을 만족시키기 위한 방법으로 게이트 복사(replication)와 인버터 또는 버퍼를 추가하는 버퍼링(buffering)을 사용할 수 있으며, 동일한 기능을 가지고 크기가 서로 다른 여러개의 게이트가 라이브러리에 존재할 경우 fanout 제약조건을 만족하는 최소 면적의 게이트를 찾아 대치하는 크기 조정(resizing)의 방법이 있다.[8] 이와 같은 방법을 조합하여 회로내의 대상 게이트가 가지는 조건을 참조하여 최대의 효과를 얻은 방법을 선택한다. 한 게이트가 변환되었을 경우 변환 효과를 측정할 수 있는 회로변수를 다음과 같이 설정한다.

Sum_slack : 최종 출력 단자들에서 나타나는 slack 값들의 합
Min_slack : 최종 출력 단자들의 slack값 중 최소치
Gate_slack : 대상 게이트의 slack 값
Area : 회로의 면적

Gate_slack은 대상 게이트가 임계 경로 상에 존재하는가를 판단하며, Min_slack은 회로의 동작 성능이 어느 정도 향상되었는가를 측정하는데 사용된다. Sum_slack은 회로 동작 성능을 측정하는 보조적인 변수로 사용되며, Area는 앞서의 세가지 변수가 동작속도에 관한 성능 측정만을 다루므로 회로를 최소의 면적 상에서 구현하기 위한 trade-off를 취하기 위하여 사용된다. 이와 같은 변수들은 변환 대상 회로에서의 값을 저장하고 변환된 회로에 대하여 재측정한 후 그 감소분을 이용하여 다음과 같은 효과치의 계산을 이룬다.

가. 임계경로가 대상 게이트의 변환에 의하여 완전히 제거되는 경우는 회로의 성능 향상이 가장 효과적이라고 볼 수 있다. 그러므로 다른 경우의 변환 효과가 비록 면적 상에 있어서 가장 효과적이라고 하더라도 이 경우의 변환 효과치를 넘지 못하도록 충분히 높은 offset을 설정한다. 이와 같은 효과가 여러 가지의 변환에 대하여 나타날 때의 우선 순위를 위하여 area의 감소분만을 offset에 더하여 효과치를 정한다.

나. 회로의 변환이 동작 성능을 감소시키는 경우에 있어서 다른 어느 경우보다 낮은 효과치를 가지도록 낮은 offset을 정의함으로써 회로의 변환이 오히려 회로의 성능을 감소시키는 결과가 가급적 작아지도록 한다. 이 offset에 Gate_slack, Min_slack, Area의 감소분을 더하여 우선 순위를 정하도록 한다.

다. 그밖의 경우에는 면적과 동작 속도의 향상치 간에 trade-offs를 취할 수 있도록 Min_slack의 감소분에 높은 가중치를 곱하고 Gate_slack의 감소분에 그보다 낮은 가중치를 곱하여 더한 다음, Sum_slack과 Area의 감소분을 더하여 효과치를 결정한다.

Ⅲ. 회로 변환 방법

1. 크기 조정

SiLOS의 기술 독립적인 회로 최소화 과정[14]은 시간 제약조건 및 fanout 제약조건을 고려하지 않고 최소의 게이트로 구현 가능한 동일한 기능의 회로를 생성한다. 그림 4의 (a) (b) (c) 회로의 인버터에 대해 수행된 기술 독립적인 최적화는 그림 4의 (d)결과 회로와 같이 단순화된 회로를 생성하게 된다. 이때 결과 회로의 인버터가 최대 허용 fanout을 초과하는 게이트를 구동하는 경우에는 fanout 제약 조건을 침해하게 된다. 따라서 기술 독립적인 최적화로 단순화

된 회로는 fanout 제약 조건을 만족시키기 위하여 라이브러리에서 지원하는 같은 기능의 여러 모듈 중에서 적합한 구동능력을 가진 모듈을 선택하여 대치하는 크기 조정의 방법을 사용할 수 있다. 크기 조정을 통한 방법은 라이브러리가 같은 타입의 게이트에 대하여 다양한 크기의 모듈을 제공하는 경우에 적합한 모듈의 선택이 가능하므로 효과적이다. 특히 인버터의 경우 대부분의 라이브러리가 다양한 크기의 모듈을 지원하므로 회로 면적과 fanout 제약을 고려한 최적의 모듈 설정이 가능하다. 이때 비임계 경로상에서는 가능한 최소의 면적이 되도록 고려하여야 하고 임계 경로 상에서는 지연 시간과 면적의 trade-offs를 취하여 게이트를 선택함으로써 회로의 최적화를 이룬다.

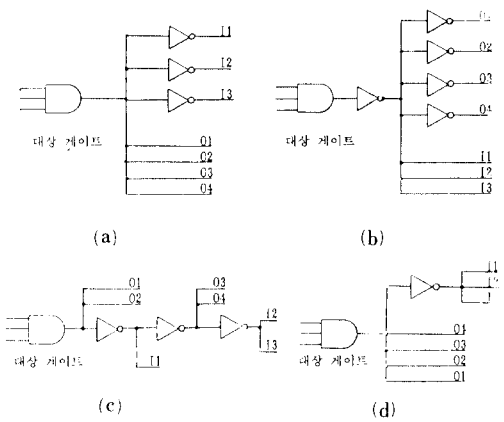

그림 4. SiLOS의 기술 독립적인 최적화 루틴에 의한 회로 단순화의 예
(a) (b) (c) 초기의 인버터 분포 상태
(d) 기술 독립적인 최적화 결과 회로

Fig. 4. Example of circuit simplification by technology- independent optimization in SiLOS.
(a) (b) (c) initial inverter distribution,
(d) circuit after optimization.

최적의 크기를 가지는 모듈을 선택하는 방법은 대상 게이트가 가지는 지연시간 정보를 라이브러리의 해당 모듈의 지연시간 정보로 대체한 후 면적의 증가분과 함께 II절에서 설명한 효과치를 계산하여 최대의 효과치를 찾음으로써 이루어진다. 이때 크기 조정으로 인하여 대상 게이트의 입력 용량에 의한 부하 지연시간이 지나치게 커지는 경우를 방지하기 위하여 대상 게이트의 입력단에 대한 크기 조정도 동시에 수행하도록 하여 각각의 효과치를 계산하고 최대의 효과를 얻을 수 있는 회로 변환을 선택하도록 한다. 그림 5에 주어진 두 회로에서 보다 높은 효과치를 보이는 회로를 선택하여 크기를 조정하는 알고리듬을 보였다.

```
function Resize( C, g, fi)
/*   g is the target gate in given circuit C.
 *   fi is one of fanin gates of g.
 */
begin
   LM = A sorted list of library modules, whose type is equal to that of g
        and maximum fanout exceeds the fanout of g ;
   C' = C ;
   Max_value = -∞ ;
   while  m in LM  do
      begin
         CTemp = Transformation of C obtained by replacing g by m ;
         VTemp = Performance gain of CTemp ;
         if  fi is not NULL  then
            (CTemp,VTemp) = Select((CTemp,VTemp), Resize(CTemp,fi,NULL)) ;
         (C', Max_value) = Select((C',Max_value), (CTemp,VTemp)) ;
      end ;
   return( C', Max_value ) ;
end ;

function  Select( (C1,V1) , (C2,V2) )
begin
   if  V1 ≥ V2  then
      return(C1, V1) ;
   else
      return(C2, V2) ;
end ;
```

그림 5. 게이트의 크기 조정 알고리듬
Fig. 5. Gate resizing algorithm.

2. 게이트 복사

게이트 복사는 fanout 제약조건을 만족하지 못하는 게이트를 그림 6과 같이 동일한 기능의 게이트를 추가하여 제약조건을 만족시켜 주는 방법이다. 이 경우 복사되어야 할 게이트들의 수 N을 다음과 같이 결정한다. 대상 게이트의 fanout 갯수를 해당 모듈이 구동할 수 있는 최대 게이트 수 ND로 나누어 몫을 취하고 나머지가 있을 경우 1을 더한 값으로 결정한다. 만일 나머지가 없을 때는 회로의 동작성능을 최적화 시키는 방법을 사용하기 위하여 임계 경로 상에서만 1을 더하여 결정한다. 따라서 나머지가 있는 경우 또는 나머지가 없는 경우라도 임계 경로가 되는 곳에서는 항상 fanout 제약조건에 대해 여유를 가지는 복사 게이트가 생겨난다. 대상 게이트가 구동하는 게이트들의 집합F는 N개의 집합 Pi(1≤i≤N)로 분할되어 복사된 N개의 게이트에 의하여 각기 구동되도록 할당할 수 있다. 복사된 게이트들 중 특정 게이트 g_i가 구동하도록 할당되는 게이트들의 집합을 P_i라 하면, 회로가 최대의 동작 속도를 가지도록 모든 g_i에 대하여 P_i를 정의할 수는 있으나, 선형 시간 내에 그 최적의 해를 찾기는 불가능하다. 따라서 P_i를 다음과 같은 heuristic 방법으로 설정한다

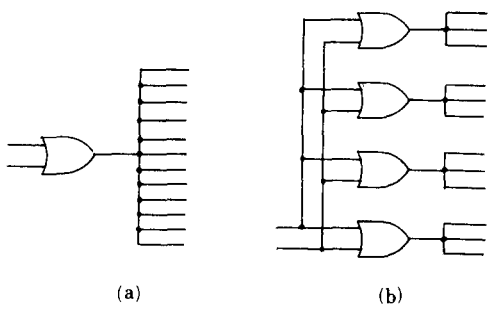

그림 6. 대상 게이트의 복사
(a) 초기 회로 (b) 게이트 복사후의 회로
Fig. 6. Replication of target gate.
(a) given circuit, (b) after gate replication.

[단계 1] 집합F의 각 게이트 $f_j(1\le j\le \|F\|)$를 slack 값에 따라 sort한다. 즉 집합 F의 두 게이트 f_m, f_n에 대하여 $m<n$이면 f_m의 slack 값이 f_n의 slack 값보다 작거나 같도록 한다.

[단계 2] 집합P_i을 제외한 모든 $P_i(2\le i\le N)$의 크기가 구동 가능한 게이트의 수 ND가 되도록 하기 위하여 집합 P_i의 원소를 다음과 같이 초기 분할을 이룬다.

$$P_i=\{f_j \mid s\le j\le l\} \quad (P_i\in P, P=\{P_i \mid 1\le i\le N\})$$

$$s=\begin{cases} 1 & (i=1) \\ 2 & (i=2\text{이며, 대상 게이트가 임계 경로상에 있고 } \|F\|_{mod\ ND}=0\text{일때}) \\ \|F\|-ND*(N+1+i)+1 & (\text{그외의 경우}) \end{cases}$$

$$l=\begin{cases} 1 & (i=1\text{이며, 대상게이트가 임계경로상에 있고 } \|F\|_{mod\ ND}=0\text{일때}) \\ \|F\|-ND*(N-i), & (\text{그외의 경우}) \end{cases}$$

[단계 3] 1에서 N까지의 각 P_i'가 그에 대응하는 g_i의 fanout 게이트 집합으로 되는 경우 P_i의 각 게이트의 출력에 나타나는 slack 값들 중 최소값은 g_i에 의한 지연시간의 변화량과 P_i의 첫번째 원소의 slack값으로 계산이 가능하다. 이 값들의 최소값을 S_{min}이라 하자.

[단계 4] 크기가 1보다 큰 분할된 집합 중에서 최소의 slack 값을 가지는 집합과 크기가 ND보다 작은 분할된 집합중 최대의 slack값을 가지는 집합, P_x, P_y $(x\neq y,\ P_x, P_y\in P)$를 택하여, $x<y$인 경우 $P_k(x\le k<y,\ P_k\in P)$의 마지막 원소를 제거하고 P_{k+1}에 삽입하여 첫번째 원소로 한다. $x>y$인 경우는 P_{k+1} $(x\le k<y,\ P_{k+1}\in P)$의 첫번째 원소를 제거하고 P_k 의 마지막 원소로 삽입한다. 이때 새로이 정의된 x에서 y까지의 모든 P_i를 P_i'라 하고, g_i에 할당하여 각 분할의 최소 slack 값을 재계산하고 이들의 최소 값을 S_{min}'라 한다. 만일 두 집합 P_x, P_y를 찾을 수 없는 경우는 분할 과정을 마친다.

[단계 5] S_{min}이 S_{min}'보다 크다면 S_{min}'를 S_{min}으로 하고, x에서 y까지의 각 P_i'를 P_i로 하여 단계 4의 과정을 반복한다. 그렇지 않은 경우는 분할과정을 마친다.

복사에 의하여 fanout 제약조건을 만족시키는 과정은 앞단의 게이트의 부하 용량을 늘려 지연 시간을 증가시킬뿐만 아니라 fanout 제약조건을 만족하지 못하도록 할 수도 있다. 또한 여러개의 게이트가 추가됨으로써 지나친 면적의 증가도 유발하게 된다. 따라서 복사에 의한 fanout 제약조건을 만족시키는 방법은 크기 조정을 통한 방법이 불가능하거나 버퍼링에 의한 방법이 오히려 더 많은 면적의 증가를 초래하는 경우 또는 지나친 지연 시간의 증가를 가져오는 경우에 한하여 사용하도록 하기 위해, 동일한 효과치에 대해서 선택의 우선 순위가 낮게 한다. 라이브러리에서 제공되는 모듈의 선정과 선정된 모듈이 복사되어 구동하는 게이트 집합의 분할 과정에서 크기 조정의 방법과 마찬가지로 대상 게이트의 입력단에 대하여도 적합한 크기의 모듈을 선택하는 방법을 통해 더욱 효과적인 회로 변환이 가능한가를 살펴본다. 그림 7에 그 알고리듬을 보였다.

```
function Replication( C, g, fi )
/*  g is the target gate in given circuit C.
 *  fi is one of fanin gates of g.
 */
begin
  LM = A sorted list of library modules whose type is equal to that of g ;
  C' = C ;
  Max_value = -∞ ;
  while m in LM do
    begin
      N = the number of gates to be replicated m ;
      Let R = { g1, g2, ··· gN } be a set of replicated gates of m ;
      Let P = { P1, P2, ··· PN } be a set of partitioned gate groups
              by dividing fanout gates of g ;
      for i = 1 to N do
        Make gi in R drive the gates in group Pi ;
      CTemp = Transformation of C obtained by deleting g
              and inserting all the gates in R ;
      VTemp = Performance gain of CTemp ;
      if fi is not NULL then
        (CTemp,VTemp) = Select((CTemp,VTemp), Resize(CTemp,fi,NULL)) ;
      (C',Max_value) = Select((C',Max_value), (CTemp,VTemp)) ;
    end ;
  return( C', Max_value ) ;
end ;
```

그림 7. 게이트 복사 알고리듬
Fig. 7. Gate replication algorithm.

3. 버퍼링

과부하로 인한 게이트의 지연시간이 지나치게 큰 경우 또는 fanout 제약조건을 크게 초과하는 경우, 게

이트의 앞단에 구동 능력이 큰 버퍼를 삽입하는 방법을 사용하여 이를 제거할 수 있다. 버퍼링은 출력의 극성(polarity)이 바뀌지 않는 버퍼와 인버터와 같이 극성이 바뀌는 두가지 타입을 고려할 수 있다. 버퍼의 삽입은 단순히 대상 게이트의 출력단에 버퍼를 첨가하여 이루어지며, 첨가된 버퍼에 대해 크기 조정과 복사를 행하는 방식으로 최적화된 회로를 얻기 위한 노력이 가능하다. 또한 버퍼의 입력 부하가 클 경우 대상 게이트의 지연시간이 커지거나 버퍼의 복사에 의하여 대상 게이트의 fanout 제약조건이 침해되는 경우를 막기 위하여 대상 게이트의 크기 조정을 동시에 수행하여야 한다. 대상 게이트의 크기 조정은 Ⅲ.1과 Ⅲ.2의 방법에서 동시에 고려되고 있으므로 각 알고리듬을 호출할 때, 대상 게이트와 첨가된 버퍼를 넘겨줌으로서 이 작업이 가능하다. 인버터를 첨가하는 경우는 대상 게이트를 inversion 하여 버퍼의 첨가와 마찬가지의 작업을 수행하도록 한다. 그림7에 버퍼링에 의하여 fanout 제약조건을 만족시키는 알고리듬을 보였다.

```
function  Buffering( C, g )
/* g is the target gate in circuit C. */
begin
   if  buffer type module in library  then
      begin
         C_Temp = Circuit obtained by inserting buffer b at fanout of g in
         (C_b, V_b) = Select(Resize(C_Temp,b,g), Replication(C_Temp,b,g)) ;
      end ;
   if  inverted type of g in library  then
      begin
         C_Temp = Transformation of C obtained by inverting g
                  and inserting inverter i ;
         (C_i, V_i) = Select(Resize(C_Temp,i,g), Replication(C_Temp,i,g)) ;
      end ;
   return( Select((C_b,V_b), (C_i,V_i)) ) ;
end ;
```

그림 8. 버퍼링 알고리듬
Fig. 8. Buffering algorithm.

4. 복합 게이트

단순 게이트 이외의 복합 게이트는 그 모듈의 크기 조정, 복사 또는 버퍼링의 방법을 사용하는 경우와 복합 게이트를 이루는 개개의 단순 게이트로 분리하여 출력단을 직접 구동하는 단순 게이트 g_H와 그 밖의 게이트 G_T로 회로를 재구성하며 g_H는 복합 게이트가 구동하였던 게이트를 구동하여야 하므로 fanout 제약조건을 만족시키기 위하여 크기 조정, 복사, 버퍼링의 방법을 사용할 수 있다. 그러나 G_T는 단지 g_H만을 구동하면 되므로 항상 fanout 제약조건을 만족하며 작은 부하 지연시간을 가진다. 따라서 G_T는 최소 면적의 게이트로 구성하도록 한다. 그림9에 복합 게이트를 단순 게이트로 재구성하여 fanout 제약조건을 만족시키는 알고리듬을 보였다.

```
function  Flatten( C, g )
/* g is the target gate in given circuit C. */
begin
   if  g is of compound gate type  then
      begin
         Let g_H be the head gate in g ;
         Let G_T be a set of tail gates in g ;
         C_T = Transformation of C obtained by replacing g by g_H and G_T ;
         return( Slect( Buffering(C_Temp,g_H),
                  Select(Resize(C_Temp,g_H,NULL), Replication(C_Temp,g_H,NULL)) ) ) ;
      end ;
   else
      return( C, -∞ ) ;
end ;
```

그림 9. 복합 게이트에 대하여 fanout 제약조건을 만족시키는 알고리듬
Fig. 9. Fanout constrained resynthesis algorithm for compound gates.

5. 대상 게이트의 선정 및 적용

변환 대상 게이트의 선정은 회로 내에서 fanout 제약조건을 만족하지 못하는 게이트들을 추출하고 이들중 가장 작은 slack 값을 가지는 게이트를 우선으로 선정한다. 회로의 변환으로 인하여 fanout 제약조건을 만족하던 게이트가 이를 침해하게 되는 경우는 단지 변환 대상 게이트의 입력단에서만 일어날 수 있으므로 동일한 slack값을 가지는 게이트가 여러개 존재할 때는 최종 출력단에 가까운 게이트를 우선적으로 선정한다. 적용 알고리듬이 선택되어 회로가 변환되면 다음의 대상 게이트 선정을 위해 회로내의 각 게이트의 slack값을 재계산하고 추출된 게이트들 중에서 마찬가지의 방법으로 다음의 변환 대상 게이트를 선정한다.

초기에 추출된 게이트들에 대해 fanout 제약조건하의 재생성이 완료되면 이들 중 일부의 게이트 또는 그 외의 게이트가 fanout 제약조건을 만족시키지 못하는 경우가 있을 수 있다. 이는 게이트 복사에 의하여 입력단의 게이트가 fanout 제약조건을 침해할 수 있기 때문이다. 따라서 초기에 추출된 게이트에 대하여 회로변환이 완료된 다음, 전체 회로에 대하여 fanout 제약조건을 만족하지 못하는 게이트들을 다시 추출하여 알고리듬을 적용한다. 이러한 방법은 특정 게이트가 한번의 fanout 제약하의 재생성 알고리듬의 적용으로 제약조건을 만족하도록 함으로써 생길 수 있는 탐색 영역의 제한과 비효율적인 결과를 방지할 수 있다. 그림10의 회로는 대상 게이트가 구동하는 게이트의 수가 매우 많은 회로이며 두번의 fanout 제약하의 재생성 알고리듬의 적용으로 이루어지는 회로의 결과를 보인다. 따라서 이와 같은 대상 게이트의 선정 방법을 사용하는 경우 Ⅲ.1~Ⅲ.4에서 제시한 알고리듬만으로 가능한 여러가지의 회로 변환 방법이 고려될 수 있다.

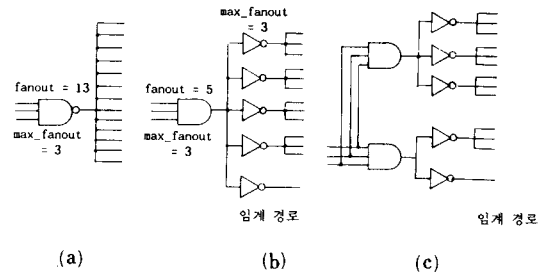

그림10. Fanout 제약하의 재생성 회로
(a) 초기 회로 (b) 알고리듬 적용후의 회로
(c) 알고리듬의 재적용 후의 회로

Fig. 10. Circit after fanout constrained resynthesis.
(a) given circuit,
(b) circuit aftgr algorithm applied,
(c) circuit after algorithm reapplied.

대상 게이트가 선정되면 최소의 면적으로 fanout 제약조건을 만족하고 동작시간 제약조건을 가장 잘 만족시키는 방법을 찾아 그 방법으로 회로를 변환시킨다. 이러한 선택의 기준은 대상 게이트에 각 알고리듬을 적용하였을 때 측정되는 효과치가 되며 회로 변환은 각 알고리듬에서 제공하는 회로 정보를 바탕으로 이루어진다. 그림11에 fanout 제약하의 재생성 알고리듬을 보였다.

```
procedure FanoutResynthesis( C )
/*   C is target circuit.  */
begin
   Let G be a list of gates that exceed the fanout constraint ;
   while  G is not NULL  then
      begin
         while  G is not NULL  then
            begin
               Calculate Slack of C ;
               Save parameter values ;
                     /* Sum_slack, Min_slack, Gate_slack, Area */
               Sort G topologically ;
               g = the first gate of G ;
               (C, Val) = Select( Flatten(C,g), Select( Buffering(C,g),
                           Select(Resize(C,g,NULL), Replication(C,g,NULL)))) ;
               Delete g from G ;
            end ;
         Let G be a list of gates exceeding its fanout constraint ;
      end ;
end.
```

그림11. Fanout 제약하의 재생성 알고리듬

Fig. 11. Fanout constrained resynthesis algorithm.

임계 경로 중 지나치게 큰 부하 지연시간을 가지는 게이트는 게이트 복사로 추가된 게이트에 부하용량을 분산시킬 수 있으므로 이에 따른 지연시간의 감소로써 동작속도를 향상시키거나, 크기 조정과 버퍼 또는 인버터를 첨가함으로써 동작 속도의 향상을 얻을 수 있다. 따라서 이 경로에 대해서는 fanout 제약조건을 만족하더라도 재조정의 필요가 있으므로 임계 경로를 따라 최대의 부하 지연시간을 가지는 게이트를 추출하여 fanout 제약조건을 만족시키기 위해 제시한 방법을 사용하여 동작 속도 제한 조건을 만족시키도록 한다. 각 알고리듬은 Ⅲ.1~Ⅲ.4 절에서 제시한 방법과 동일하며 단지 fanout 제약 조건을 만족하는가를 검사하는 대신에, 회로가 동작 시간제약조건을 만족할 수 있는가를 검사하는 것만이 다르다.

Ⅳ. 실험 및 결과

표 1에 Standard Microsystems Corp.의 standard cell catalog[11]에 수록된 일부 게이트의 특성을 보였다. 이를 라이브러리로 사용하여 그림12의 대상회로에 대하여 재설계 과정을 수행한 결과를 그림13에 보였다. 대상 회로의 동작 시간 제약조건은 6.5ns로 하였으며, 추출된 임계 경로는 대상 회로에서 게이트 G를 제외한 모든 부분이다. 최대 지연시간은 primary output의 부하 용량이 0.6pF일때 7.4nS 이고, 게이트 A, B, C, D의 fanout은 각각 5, 5, 5, 8로서 fanout 제약조건을 모두 만족하지 못한다.

대상 회로에서 fanout 제약조건을 만족하지 못하며 가장 작은 slack 값을 가지는 게이트 C가 먼저 선택되어, 재설계 과정을 거친다. C는 복사에 의하여 그림13의 게이트 C1과 C2로 분리된다. 이때 게이트 C1은 4개의 nand4 게이트를 구동하며 C2는 인버터를 구동하도록 할당되고 C2가 가지는 slack 값은 0.1nS이고 면적의 증가분은 5이다. 버퍼링에 의하여 게이트 C가 inversion된 후 fanout 제약조건을 만족하

표 1. SMC사 스탠다드셀 라이브러리의 게이트 특성

Table 1. Characteristics of the gates in the standard cell library of Standard Microsystems Corp.

게이트 타입	고유지연 시간(nS)	부하저항 (T/mF)	면적	입력용량 (pF)	max fanout
inv	0.5	2.69	2	0.0574	4
INV	0.4	0.88	3	0.160	8
and2	1.3	2.15	4	0.0766	5
and4	1.5	2.63	6	0.0801	5
AND2	1.9	0.81	6	0.160	9
AND4	2.6	0.81	8	0.170	9
nand2	0.9	2.56	3	0.0743	4
nand4	1.5	2.44	5	0.117	4
NAND2	2.4	0.44	7	0.160	8
NAND4	3.1	0.41	9	0.170	8

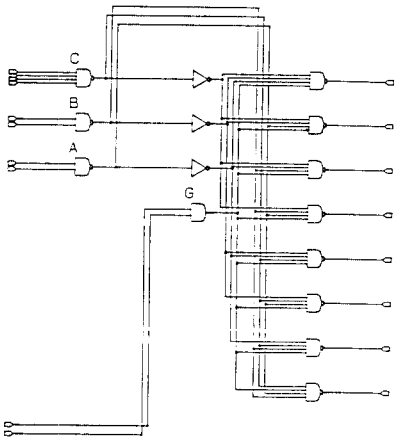

그림12. 대상 회로
Fig. 12. Given circuit.

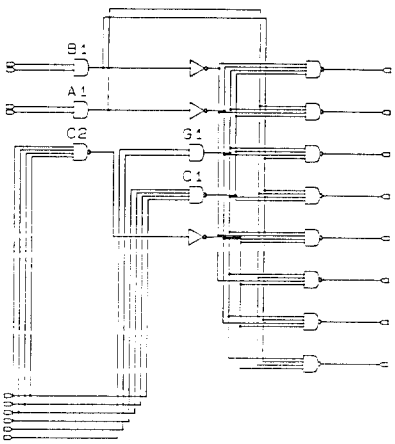

그림13. 재설계된 결과 회로
Fig. 13. Result circuit obtained by resynthesis.

는 AND2로 변환되는 경우, 게이트 C의 slack 값은 −0.2 nS이고 면적의 증가분은 3이다. C에 대하여 크기 조정을 수행하는 경우 면적의 증가분이 4이며 slack 값은 −1.6 nS이다. 면적상의 효과가 버퍼링에 의한 방법이 우위에 있으나 임계 경로상에서는 동작시간 제약조건을 만족시키는 것이 주된 관점이 되므로 효과치의 계산에 의하여 게이트 복사의 방법이 선정되었다.

게이트 A와B는 버퍼링의 방법으로 inversion 하여 and2로 변환되며 이 과정에서 출력단의 인버터는 크기 조정으로 INV가 된다. 각각의 slack 값은 0.2nS로 동작 시간 제약조건을 만족하면 면적의 증가는 2이다. 이들을 복사할 경우 slack 값은 0.7 nS가 되나 면적의 증가분은 각각 3이 된다. Slack값이 양의 값이 되는 알고리듬들은 시간상의 효율을 동일하게 간주하는 이유로 작은 면적의 증가를 보이는 버퍼링이 높은 변환 효과치를 가지게 된다. 비임계 경로에 있는 게이트 G에 대해서는 크기 조정을 수행하여 최소의 면적으로 fanout 제약조건을 만족시킨다. Fanout 제약조건이 만족된 회로에 대해 부하 지연시간의 최적화 과정은 각 게이트와 인버터들에 대해 진행되지만 이미 시간 제약조건을 만족하므로 더 이상의 회로 변환은 보이지 않았다. 만일 회로의 동작 시간제약조건이 만족되지 않은 경우는 이를 만족시키기 위해 각 인버터들이 라이브러리의 INV로 변환될 것이 예상된다.

일반적인 회로에 대한 실험결과를 알아보기 위하여 표준 테스트 회로인 MCNC 벤치마크 회로에 대한 fanout 제약하의 재생성 알고리듬을 적용하여 그 결과를 표 2에 보였다. 사용된 라이브러리는 참고문헌[11]에서 제공되는 모든 게이트를 사용하였으며, fanout 제약조건은 실험을 위하여 수퍼 모듈들을 제외한 모든 게이트들이 3개 이내의 게이트를 구동하는 것과 4개 이내의 게이트를 구동하도록 하는 두가지 방법을 채택하였다. 대상 회로는 최적화 알고리듬을 거치지 않은 회로와 최적화 알고리듬을 거친 두 가지 회로를 대상으로 하였다. 결과의 회로는 면적의 증가분은 대략 10% 내외로 나타났으나 회로의 모든 게이트가 fanout 제약조건을 만족하였으며, 회로에 따라 많은 차이가 있으나 평균 10% 정도의 동작 속도 증가도 동시에 얻을 수 있었다. Fanout 제약조건이 3인 게이트를 사용하는 경우보다 4인 게이트를 사용하는 경우에 있어서 회로 면적의 증가가 더욱 커지나 회로의 동작 속도는 약간 더 향상됨을 보인다.

기술 의존적인 최적화 과정을 거친 벤취마크의 회로가 fanout 제약조건하의 논리 재생성 과정에 적용되는 경우 더 높은 면적의 증가율을 보였으며 회로 동작 속도도 더욱 향상 되었다. 이는 대상 회로가 최적화 과정을 거치는 경우 부하 지연시간을 고정된 값으로 하여 임계 경로상에서 동작 속도의 향상과 비임계 경로상에서의 면적의 최소화를 위하여 최소 크기의 모듈로만 회로를 재구성하므로 fanout 제약조건을 만족하지 못하는 게이트와 부하 지연시간이 큰 게이트가 생성될 수 있기 때문이다. 또한 이러한 게이트들은 본 논문에서 제안한 fanout 제약하의 논리 재생성 과정에서 보다 효율적인 회로의 설계가 가능한 단순화된 회로조건을 가정할 수 있도록 한다. 최

적화 과정을 거친 벤취마크에 대한 fanout 제약조건하의 논리 재생성 결과가 면적의 증가율은 높으나 실제 증가된 면적은 최적화 과정을 거치지 않은 벤취마크에 대한 결과에서 나타난 면적의 증가분과 거의 비슷함을 표 2에서 알 수 있다. 또한 이 결과는 기술 의존적인 최적화 과정이 수행된 벤취마크에 대하여 더욱 최적화된 회로가 생성될 수 있음을 보인다.

표 2. MCNC 벤취마크에 대한 실험 결과
Table 2. Experimental results for MCNC benchmark circuits.

테스트 회로			max. fanout=3		max. fanout=4	
벤취 마크 회로 386	초기 면적	지연시간 (nS)	결과 면적 (증가율%)	지연감간(nS) (감소율%)	결과 면적 (증가율%)	지연시간(nS) (감소율%)
s386. bench	1757	32.4	1928(8%)	31.8(3%)	1857(5%)	31.8(2%)
s208. dench	1795	33.7	1966(8%)	31.2(7%)	1908(5%)	32.0(5%)
s386. bench	624	18.9	700(10%)	15.1(20%)	682(8%)	15.3(19%)
s208. bench	283	17.5	290(2%)	17.5(0%)	283(0%)	17.5(0%)
s1196. bench*	1540	24.9	1754(12%)	23.2(7%)	1628(8%)	24.1(3%)
s1238. bench*	1732	26.1	1936(10%)	24.6(6%)	1895(8%)	25.0(4%)
s386. bench*	454	12.5	522(10%)	9.9(21%)	508(10%)	10.1(19%)
s208. bench*	244	15.0	256(4%)	15.0(0%)	no change	no change

*기술 의존적인 최적화 과정을 거친후의 벤취마크 회로

V. 결 론

SiLOS는 SUN UNIX 상에서 사용자 인터페이스를 위한 그래픽 출력기능을 포함하여 약 5만라인의 C언어로 구현하였다. 이 시스템은 입력의 VHDL 기술과 라이브러리 화일로부터 회로의 정보를 분석하여 임계 경로를 추출한 후 최적화된 재설계 회로를 VHDL 기술 혹은 그래픽 출력으로 사용자에게 제공한다.

회로의 재설계는 기술 독립적인 최소화 과정과 기술 독립적인 최적화 과정을 거치며 최종적으로 fanout 제약 조건하의 재생성 과정을 거친다. 기술 독립적인 최소화 과정은 30여개의 generalized rule을 사용하여 회로에서 이 룰을 만족하는 부분을 찾아 대치하도록 구현하였으며, 기술 의존적인 최적화 과정은 gate compounding, gate decomposition, gate inversion, gate conversion을 통한 변환과정[15]을 이용하여 구현하였다. 본 논문에서는 주로 fanout 제약조건을 만족시키고 부하 지연시간에 의한 임계 경로를 제거하기 위한 기술 의존적인 재설계 알고리듬에 관하여 논하였다.

세밀한 지연시간 정보를 사용한 회로의 설계는 칩으로 구현된 회로의 특성을 보다 정확히 나타낼 수 있으며 이 정보로부터 보다 향상된 회로의 설계가 가능함을 실험결과로부터 확인할 수 있다. 반도체 공정 기술이 발달할수록 게이트 자체의 지연시간 보다는 연결선에 의한 지연 시간이 더욱 큰 문제가 되므로 향후 배치 배선 시스템의 back annotation에 의한 연결선의 지연시간 정보를 사용한 재설계 과정을 추가하여 더욱 효율적인 회로의 설계가 가능하도록 하는 연구가 계속될 것이다.

參 考 文 獻

[1] J. Allen, "Performance-directed synthesis of VLSI systems," *IEEE Proceedings*, vol. 78, no. 2, pp. 336-355, Feb. 1990.

[2] J. Darringer, D. Brand, J. Gerbi, W. Joyner, and L. Travillyan, "LSS: a system for production logic synthesis," *IBM Journal of Research and Development*, vol. 28, no. 5, pp. 537-545, Sept. 1984.

[3] E. Detjens, G. Gannot, R. Rudell, A. Sangiovanni-Vincentalli, and L. Trevillyan, "Technology mapping in MIS," *in Proc. ICCAD*, pp. 116-119, Nov. 1987.

[4] D.J. Gregory, K. Bartlett, A. de Geus, and G. Hachtel, "SOCRATES: a system for automatically synthesizing and optimizing combinational logic," *in Proc. 23rd ACM/IEEE Design Automation Conference*, pp. 79-85, June 1986.

[5] K. Keutzer, "DAGON: technology binding and local optimization by DAG matching," *in Proc. 24th ACM/IEEE Design Automation Conference, pp.* 341-347, June 1987.

[6] S, Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing," Science, vol. 220, no. 4598, pp. 671-680, 1983.

[7] R. Lipsett, C.F. Schaefer, C. Ussery, *VHDL: Hardware Description and Design*, Kluwer Academic Pub. Boston, Mass., pp. 107-143, 1989.

[8] G. De Micheli, "Performance oriented synthesis of large scale domino CMOS circuits," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, no. 5, pp. 751-765, Sept. 1987.

[9] H.J. Touati, C.W. Moon, R.K. Brayton, "Performance-oriented technology mapping," *in Proc. of the Sixth MIT conference, MIT press*, pp. 79-97, 1990.

[10] N. Weste, and K. Eshraghian, *Principles of CMOS VLSI Design: A Systems perspective*, Addison-Wesley: Reading, MA., pp. 120-158, 1988.

[11] 1988 Customation III Standard Cell Catalog for ASIC Design, Standard Microsystems Corp., 1988.

[12] *IEEE Standard VHDL Language Reference Manual*, IEEE Standard, Apr. 1989.

[13] *Library compiler*, Referance Manual Version 1.2, Synopsys Inc.,Nov. 1989.

[14] 김태선, 황선영, "기술 독립적인 논리 최소화를 위한 규칙 기반 시스템," 한국정보과학회 논문지, 18권 3호, 1991년. (게재예정).

[15] 이재형, 황선영, "성능 구동 자동 논리회로설계 시스템," 대한전자공학회 논문지, 28-A권, 1호, 1991년 1월 pp. 74-84.

[16] 황선영, "성능 구동 논리회로 자동설계방식 개발," ISRC 90-E-DE-D003, 서울대학교 반도체 공동 연구소, 1990년 6월.

著 者 紹 介

黃 善 泳(正會員)

1954年 5月 20日生. 1976年 2月 서울대학교 전자공학과 졸업. 1978年 2月 한국과학원 전기및 전자공학과 공학석사 취득. 1986年 10月 미국 Stanford 대학 공학박사 학위 취득. 삼성반도체 주식회사 연구원. Stanford 대학 CIS 연구소 연구원. Fairchild Semiconductor기술 자문. 1989年 3月~현재 서강대학교 전자공학과 교수. 주관심분야는 CAD 시스템, Computer Architecture 및 Systems Design VLSI 설계 등임.

李 在 亨(準會員)

1968年 4月 6日生. 1990年 2月 서강대학교 전자공학과 졸업. 1990年 3月~현재 서강대학교 전자공학과 대학원 석사과정. 주관심분야는 CAD 시스템, Computer, Architecture 등임.