論文94-31A-5-18

레지스터 전송 수준에서의 VHDL 순서문 합성에 관한 연구

(A Study on Synthesis of VHDL Sequential Statements at Register Transfer Level)

玄 珉 鎬*, 黃 善 泳*

(Min Ho Hyun and Sun Young Hwang)

要 約

본 논문에서는 레지스터 전송 수준으로 기술된 VHDL 순서문의 합성을 지원하기 위한 알고리듬을 제안한다. 제안 알고리듬은 지역 및 전역적 의존성 분석과 출력 의존성 제거 과정을 이용하여 VHDL의 순서문으로 이루어진 행위 기술을 단일 지정 법칙이 적용되는 병행문 형태의 데이타 흐름 기술로 변환한다. 병행문으로의 변환은 순서문으로 기술된 레지스터 전송 수준 하드웨어 구조의 코스트를 감소하여 설계자의 의도에 알맞는 최적화된 합성 결과의 생성을 가능하게 한다. 본 알고리듬은 VSYN 에 구현되었으며, 실험 결과는 ViewLogic의 PowerView 및 Synopsys의 DesignAnalyzer에 비하여 보다 효율적인 합성 결과를 생성함을 보여준다.

Abstract

This paper presents an algorithm for synthesis of sequential statements described at RT level VHDL. The proposed algorithm transforms sequential statements in VHDL into data-flow description consisting of concurrent statements by local and global dependency analysis and output dependency elimination. Transformation into concurrent statements makes it possible to reduce the cost of the synthesized hardwares, thus to get optimal synthesis results that will befit the designer's intention. This algorithm has been implemented on VSYN and experimental results show that more compact gate-level hardwares are generated compared with PowerView system from ViewLogic and DesignAnalyzer from Synopsys.

Ⅰ. 서 론

최근 VLSI 기술의 급속한 발전과 더불어 칩의 복잡도가 현저히 증가함에 따라, 기존의 레이아웃 설계 위주의 CAD 툴들을 이용한 bottom-up 설계 방식으로는 새로운 기술의 발달을 따라가지 못하게 되었다. 이같은 문제점이 지적되면서 설계 사양을 보다 상위 수준에서 설계자가 이해하기 쉬운 하드웨어 기술 언어 (HDL : Hardware Description Language) 로 개념화 하여 기술하고 이의 동작 검증을 거친 후 하드웨어를 생성하는 방식인 top-down 설계방식이 대두되었다. [1], [2], [3]

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

※본 연구는 92년 상공부 공업기반 기술과제로 (주)서두로직과의 공동으로 이루어진 연구결과임.
接受日字 : 1993年 1月 8日

Top-down 설계방식은 상위 수준으로부터 하위 수준으로의 흐름을 갖는 설계 방식으로, top-down 설계방식을 이용한 상위 수준의 자동 설계는 크게 행위 수준의 회로 설계와 레지스터 전송 수준 및 논리 수준의 회로 설계로 나뉘어 질 수 있다. 전자는 설계 대상 회로의 행위를 알고리드믹하게 기술한 HDL을 입력으로 하며, HDL 기술의 의존성 분석을 통하여 기술의 의미상 내재된 컨트롤 정보를 추출함으로써 데이타 패스와 컨트롤러를 분리하여 합성을 수행하는 과정으로, 넓은 설계 공간(design space)을 탐색해 보는데 그 목적이 있음에 반하여, 후자는 설계 대상 회로의 레지스터 전송 수준의 하드웨어 구조를 데이타 흐름 중심의 형태로 기술한 HDL을 입력으로 받아들이며, HDL 기술내에서 외부적으로 표현된 모든 하드웨어 구조로 부터 실제의 하드웨어를 합성해 내는데 그 목적이 있으므로, top-down 방식의 전체적 설계환경 구축에서 레지스터 전송 수준의 자동 설계 시스템의 필요성은 절대적이다.[4], [5] 이러한 top-down 설계방식을 이용한 자동 설계 시스템은 세계적으로 일부 연구소 및 대학 등에서 개발되어 사용되고 있으나[6], [7], [8], [9], [10], 이들 시스템들의 대부분은 top-down 설계방식의 큰 장점에도 불구하고 설계 언어로서 각사 고유의 HDL을 입력으로 사용함으로 야기된 설계 데이타의 공유 및 재사용에서의 문제점으로 인해 최대의 효율성을 올릴 수 없었다. 이러한 비효율성을 극복하고 반도체 설계 기술에서의 호환성과 효율성의 극대화를 위해서는 VHDL과 같은 표준화된 HDL의 사용으로 서로 다른 툴에서 개발된 설계 데이타들간의 상호 교환을 가능하게 하고 이미 설계된 각 설계 단위들을 라이브러리화하여 해당 제작 기술에 적합하도록 재사용할 수 있어야 한다.

VHDL[11], [12], [13], [14], [15]은 미 국방성의 지원아래 1983년부터 본격적으로 개발되어 1987년 IEEE 표준으로 지정되었으며, 하드웨어의 기술, 시뮬레이션 및 설계 언어로서 기존의 HDL이 특정 수준의 기술에만 한정되었던 범위의 제약을 확장하여 아키텍쳐 수준에서부터 게이트 수준까지 기술이 가능하고, 각 수준에서 행위 기술, 데이타 흐름 기술, 구조 기술 등의 세가지 기술 형태를 혼합하여 대상 회로를 설계할 수 있다.

레지스터 전송 수준의 하드웨어 기술에 있어 기존의 레지스터 전송 수준 전용의 HDL이 하드웨어 구조와 일대일 매핑되는 평면화된 기술 방법을 사용하였던 것과는 달리 VHDL은 구조 기술과 데이타 흐름 기술 및 하드웨어의 동작에 대한 행위 단계까지 다양한 기술이 가능하며, 이 중 구조 기술 및 데이타 흐름 기술은 라이브러리를 이용한 일대일 매핑을 통하여 실제 하드웨어로 합성할 수 있으나, 순서문으로 이루어진 행위 기술의 경우 HDL의 기술 의미상 내재된 의존성으로 인해 레지스터 전송 수준에서 실제 하드웨어로 직접 변환이 불가능하다. 기존의 상업적으로 이용되고 있는 몇몇 VHDL 합성 시스템[6], [9], [10]의 경우 레지스터 전송 수준에서 기술된 VHDL을 입력으로 하여 합성을 수행하며 VHDL의 구조 기술 및 데이타 흐름 기술 이외에 순서문으로 이루어진 행위 기술로 그 지원 범위를 확대하고 있으나, 순서문에 대한 합성을 지원하기 위하여 사용된 내부 알고리듬에 대해서는 보고된 바가 없으며 생성된 결과에 대한 최적화 기능이 미약하고 합성 결과의 정확성이 떨어지는 단점이 있다.

본 논문에서는 레지스터 전송 수준에서 기술된 VHDL을 입력으로하여 게이트 수준의 하드웨어를 합성하는 레지스터 전송 수준 VHDL 합성 시스템인 VSYN(VHDL SYNthesis System)[16]에서 순서문 합성 지원을 위한 내부 알고리듬으로서, VHDL의 순서문에 대하여 단일 지정 법칙 (single assignment rule)을 적용하기 위하여 사용된 지역 및 전역적 출력 의존성 제거 알고리듬에 대하여 기술한다.

본 논문의 2장에서는 전체적인 VSYN 시스템 개관을 제시하고, 3장에서는 VHDL 순서문의 합성을 위한 VSYN의 접근 방법 및 사용된 알고리듬에 대하여 기술하며, 4장에서 실험 결과를 제시하고, 5장에 결론을 맺는다.

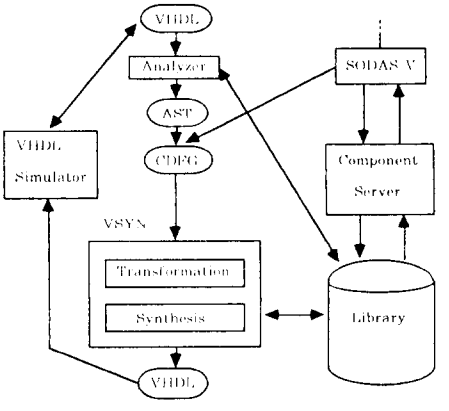


그림 1. VSYN 시스템 구성도
Fig. 1. VSYN System configuration.

Ⅱ. VSYN 시스템 개관

VSYN 시스템은 레지스터 전송 수준에서 기술된

VHDL을 입력으로 하여 VHDL 분석기[17]의 분석 결과 생성된 C/DFG[18]로 부터 순서문의 합성을 위한 기술 변환, 최적화된 회로를 생성하기 위한 행위 변환, 컴포넌트 바인딩, 합성 등의 과정을 거쳐 게이트 수준으로의 합성을 행하며, 행위 수준 합성기와의 연결을 위하여 행위수준의 합성 결과 또한 입력으로 사용한다. 그림 1에 VSYN 시스템의 구성도를 보였다.

Ⅲ. VHDL 순서문의 합성

VHDL의 행위 기술은 설계 대상 회로의 실제 하드웨어 구조와 관계 없이 보다 상위 수준에서 대상 회로의 동작에 대한 알고리듬적 기술을 하기 위하여 사용되며, 기술의 의미상 내재된 각 변수간의 의존성에 의하여 실행 순서를 갖는 동기 기술의 형태가 되므로, 합성시 의존성 분석 과정으로 부터 추출된 제어 및 데이타 흐름 정보를 이용하여 데이타 패스 및 컨트롤러로 분리하여 하드웨어를 생성한다. 반면 레지스터 전송 수준에서의 기술은 설계될 실제 하드웨어 구조 상에서의 데이타 흐름에 관한 기술이며, 이는 이벤트의 흐름에 따라 병행적으로 수행되는 비동기 기술로, 행위 기술과는 달리 의존성에 의한 제어 정보는 존재하지 않고 합성 대상 하드웨어 상에서의 데이타 흐름에 관한 정보만이 존재한다. 그러나 VHDL을 사용하는 회로 설계자들은 데이타 흐름의 기술을 위한 병행문보다는 소프트웨어 영역에서의 일반 프로그램 언어와 흡사한 순서 구조를 가지고 있는 행위 기술을 위한 순서문에 익숙하여 레지스터 전송 수준에서의 회로 설계시 순서문을 사용하는 경우가 많다. 그림 2는 32 비트 CLA(Carry Lookahead Adder) 기술[8] 중 4 비트 CLA 블럭의 일부로, 레지스터 전송 수준에서 행위 기술 형태의 순서문을 사용한 예이다.

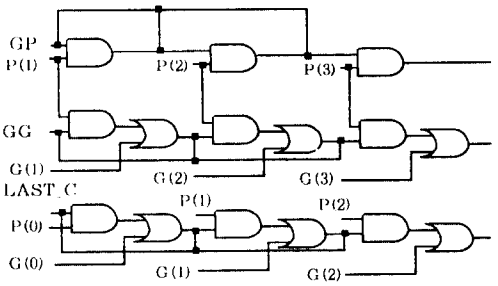
```
GP := P(0) ;
GG := G(0) ;
LAST_C := CIN ;
C(0) := CIN ;
for I in 1 to 3 loop
  GP := GP and P(I) ;
  GG := (GG and P(I)) or G(I) ;
  LAST_C := (LAST_C and P(I-1)) or G(I-1) ;
  C(I) := LAST_C ;
end loop ;
```

그림 2. CLA의 레지스터 전송 수준에서의 순서문 기술

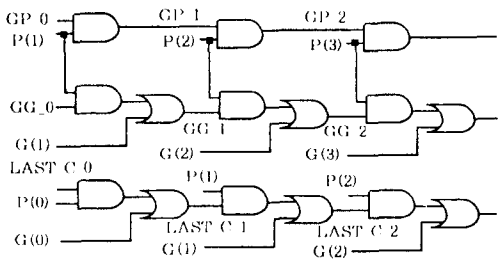
Fig. 2. Description of CLA using sequential statements at RT-level.

레지스터 전송 및 논리 수준의 설계에서 순서문을 사용한 경우, 설계 대상 회로의 비동기적 데이타 흐름을 순차적 동기 구조의 기술로 표현하게 되므로 기존의 데이타 패스와 컨트롤러를 분리하여 수행하는 상위 수준의 합성 기법을 사용하게 되면 설계자의 의도와는 상관없는 복잡한 형태의 동기 회로가 생성되는 단점이 있다. 따라서 레지스터 전송 수준에서의 순서문 기술은 설계자의 의도와 부합하도록 비동기 회로로 합성되어야 하며, 기술의 순서성을 제거하고 병행화 함으로 올바른 최적의 합성결과를 얻을 수 있도록 하여야 한다.

VHDL 순서문의 각 변수들은 이벤트의 흐름을 전달하기 위하여 병행문에서 사용되는 신호와는 달리 연산의 결과값을 임시로 저장하기 위한 레지스터의 의미를 가지고 있으며, 이름 및 스코프가 동일한 변수라 하여도 동일 기술 내에서 두번 이상 지정되었다면 해당 변수는 각 지정 이후 다음 지정까지의 생존 구간 내에서 각각 서로 다른 변수로서 취급되어야 한다. 동기 구조를 갖는 순서문으로 부터 비동기 구조의 회로를 합성해 내는 레지스터 전송 수준 합성의 경우, VHDL 순서문 상의 각 변수들을 병행문의 신호와 동일하게 합성하여야 하며 이를 위해서는 두번 이상 지정된 변수들간의 출력 의존성 제거 과정을 통하여 단일 지정 법칙을 적용함으로 최대한의 병렬성을 추출해 내는 과정이 필요하다. 그림 3(a)는 그림2의 기술에 대하여 출력 의존성 제거 과정을 거치지 않고 합성을 행한 결과로, for 루프 내부에서 각 반복 마다 사용된 변수 GP, GG, LAST_C가 동일한 변수로 취급되므로 잘못된 합성 결과를 생성함을 알 수 있다. 그림 3(b)는 출력 의존성 제거 과정을 통하여 순서 구조를 병행 구조로 변환한 뒤 생성된 올바른 합성 결과이다. 본장에서는 레지스터 전송 수준 합성시 분기 구조를 포함하는 VHDL 순서문의 레지스터 전송 수준 기술로 부터 설계자의 의도에 부합하는 최적의 하드웨어를 생성하기 위한 전역적 출력 의존성 제거 과정에 대하여 기술한다.



(a)



(b)

그림 3. 그림 1의 CLA 기술에 대한 합성 결과 (a) 출력 의존성 제거 과정을 거치지 않은 합성 결과 (b) 올바른 합성 결과

Fig. 3. Synthesis results for CLA in Fig1. (a) Synthesis result without output dependency elimination. (b) Correct synthesis result.

1. 지역적 의존성 분석

병행 구조로의 변환을 위한 전단계로, 상수 폴딩, 공통 부표현식 제거, 복사 전파, in-line expansion, loop-unrolling, 비 실행 코드 제거 등의 행위 변환 과정[19]을 거친 VHDL의 순서문들에 대한 지역 및 전역적 의존성 분석이 필요하다. 의존성의 종류로는 데이타 의존성, 반 데이타 의존성 및 출력 의존성이 있고 지역적 의존성에 대한 일반적인 정의[19]-[20]에 따라 각 의존성을 다음과 같이 표시한다.

컨트롤 노드에 의해서 분기되지 않는 최소 단위의 순서문 집합을 기본 블럭이라 정의할 때 S_i, S_j 를 각각 기본 블럭 내에서의 문장 i, 문장 j라 하고, S_i 가 S_j 보다 기술의 순서상 먼저 실행되어야 하는 문장이라면 Si 〈 Sj 라 표기한다. 이 때 S_i, S_j 를 S 〈 S_j 의 조건을 만족하는 지정문의 쌍이라 하자. DEST(S_i)를 문장 S_i의 출력 변수 집합, SOURCE(S_i)를 문장 S_i의 입력 변수 집합이라 할때,

1. 만약 DEST(S_i) ∩ DEST(S_j) ≠ ϕ이면 Sj가 Si에 출력 의존성 관계에 있다고 정의하고 S_i od S_j 로 표기한다.
2. 만약 DEST(S_i) ∩ SOURCE(Sj) ≠ ϕ이고 S_i od S_k와 S_k 〈 S_j의 조건을 만족하는 S_k가 존재하지 않는다면 S_j가 S_i에 데이타 의존성 관계에 있다고 정의하고 S_i dd S_j 로 표기한다.
3. 만약 SOURCE(S_i) ∩ DEST(S_j) ≠ ϕ이면 S_j가 S_i에 반 데이타 의존성 관계에 있다고 정의하고 S_i da S_j 로 표기한다.

2. 지역적 의존성 제거

각 기본 블럭 별로 의존석 분석이 끝나면 분석 결과를 토대로 병행문으로의 변환을 위한 단일 지정 법칙의 적용이 이루어 진다. 단일 지정 법칙은 하나의 변수에 한번의 지정만을 허용하는 법칙으로[18], 이벤트의 흐름에 따라 수행되는 병행문의 경우가 단일 지정 법칙에 의해 기술된 예이다. 각 기본 블럭에 대한 단일 지정 법칙의 적용은 해당 기본 블럭 내의 모든 문장에 대하여 출력 의존성을 제거함으로써 수행된다. 그림 4에 기본 블럭 내의 지역적 출력 의존성 제거 알고리듬을 보였다.

```
local dependency elimination (BB)
{
  for each node n in BB {
    if n has an outgoing od link {
      /* generate an internal variable T */
      T = get internal variable ( ) ;
      for each node m which has dd link from n
        substitute m with T ;
      substitute n with T ;
    }
  }
}
```

그림 4. 지역적 출력 의존성 제거 알고리듬

Fig. 4. Algorithm for local output dependency elimination.

3. 전역적 의존성 분석

각 기본 블럭들에 대하여 지역적 출력 의존성 제거 과정을 거치고 나면 각 기본 블럭 내에서는 동일 변수에 한번의 지정만이 존재하게 되며 출력 의존성 제거 과정시 생성된 중간 변수들은 해당 기본 블럭 내부에서만 참조되는 내부 변수로, 기본 블럭 내의 내부 변수들을 제외한 모든 변수들에 대하여 피지정자로 사용된 변수의 갯수를 n, 피연산자로 사용된 변수들 중 피지정자로 사용되지 않은 변수의 갯수를 m이라 할때 하나의 기본 블럭을 n개의 출력과 m개의 입력을 갖는 하나의 모듈로 볼 수 있음을 의미한다.

지역적 출력 의존성 제거 과정을 거쳐 얻어진, 단일 지정 법칙이 적용되는 기본 블럭을 가상 기본 모듈(VBM : Virtual Basic Module) 이라 정의하고, 해당 기본 블럭 내에서 내부 변수를 제외하고 피연산자로 사용되지 않는 모든 source측의 변수를 VBM의 입력 포트, 내부 변수를 제외한 모든 destination측의 변수를 VBM의 출력 포트라 정의

한다. VBM은 상호간에 실행 순서에 따른 order를 가지며, 초기 기술의 실행 순서상 VBM_i가 VBM_j보다 먼저 실행되어야 한다면 VBM_i가 VBM_j보다 상위 order를 갖는다 혹은 VBM_j가 VBM_i보다 하위 order를 갖는다고 하고 $VBM_i \langle VBM_j$ 라 표기한다. 또한 VBM_i와 VBM_j가 동일 컨트롤 노드에서 분기되어 상호 배타적 관계에 있다면 VBM_i와 VBM_j가 동일한 order를 갖는다고 하고 $VBM_i = VBM_j$ 라 표기한다. 이상의 기본적인 용어의 정의를 기초로 하여 전역적 의존성에 대한 정의를 내리면 다음과 같다.

VBM_i 와 VBM_j 를 $VBM_i \langle VBM_i$ 의 관계에 있는 VBM의 쌍이라 하자. Φ를 공집합이라 하고 DEST(VBM)을 VBM의 출력 포트 집합, SOURCE (VBM)을 입력 포트 집합이라 할 때,

1. 만약 DEST (VBM_i) ∩ SOURCE (VBM_j) ≠ Φ이면 VBM_j는 VBM_i 에 데이타 의존성이 있다고 정의하고 VBM_i dd VBM_j 로 표기한다.
2. 만약 SOURCE (VBM_i) ∩ DEST (VBM_j) ≠ Φ이고 VBM_i od VBM_k와 $VBM_k \langle VBM_j$의 조건을 만족하는 VBM_k가 존재하지 않는다면 VBM_j는 VBM_i에 반 데이타 의존성이 있다고 정의하고 VBM_i da VBM_j 로 표기한다.
3. 만약 DEST (VBM_i) ∩ DEST (VBM_j) ≠ Φ이면 VBM_j는 VBM_i에 출력 의존성이 있다고 정의하고 VBM_i od VBM_j 로 표기한다.

4. 전역적 의존성 제거

전역적 의존성 분석 결과에 기초한, 단일 지정 법칙 적용을 위한 전역적 출력 의존성 제거 알고리듬을 그림 5에 보였다. 전역적 실행 순서의 흐름 구조는 지역적 실행 순서 흐름 구조와 같이 순차적인 직렬 리스트의 구조를 갖기 보다는 DAG의 형태를 띄고 있으므로 DAG의 탐색을 위하여 재귀적 호출의 형태를 갖는다. 또한 각 VBM에 대하여 동일 order를 갖는 VBM이 하나 이상 존재할 경우 즉, 상호 배타적인 VBM이 존재할 경우에 해당 VBM의 출력 포트와 데이타 의존성 관계에 있는 VBM의 입력 포트는 상호 배타의 조건에 따라 분기된 복수의 VBM 출력 포트중 하나와 연결되어야 하므로, 이의 처리를 위해서 내부적으로 MUX 형태의 컨트롤 구조를 갖는 VBM을 컨트롤 VBM이라 정의하고, 새로운 컨트롤 VBM을 생성하여 연결한다.

그림 5에 VHDL의 순서문으로 이루어진 초기 설계 기술에 대하여 출력 의존성을 제거하는 과정을 보였다. 그림 6(a)는 VHDL 초기 설계 기술이고, 그림 6(b)는 기본 블럭 BB1에 대하여 의존성 분석 및 출력 의존성 제거 과정을 거친뒤 생성된 VBM1이다. 그림 6(c)는 초기 기술 상의 모든 기본 블럭들에 대하여 출력 의존성 제거를 거친 뒤 생성된 DAG 형태의 VBM 실행 순서 흐름 그래프를 나타낸 것이고, 그림 6(d)는 전역적인 출력 의존성 제거를 수행한 뒤 구성되는 VBM의 데이타 흐름 그래프로 VBM5a, VBM5c, VBM5e는 각각 VBM5의 입력 포트 a, c, e에 대하여 생성된 조건 분기 구조의 컨트롤 VBM이며, 그림 6(e)는 최종적으로 출력된 병행문 형태의 VHDL 데이타 흐름 기술이다. 그림 6(e)의 최종 결과에서 각 VBM은 병행 신호 지정문으로, MUX 형태의 컨트롤 VBM은 병행 조건 신호 지정문으로 변환되었음을 알 수 있다.

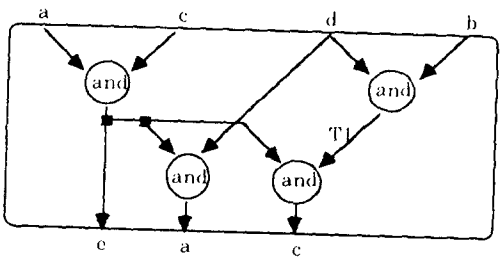
```
global_output_dependency_elimination ( VBM_i )
/* VBM_i : first VBM in the domain of concern */
{
  for each output port p of VBM_i )
    if N has an outgoing od link )
      /* generate an internal variable T */
      T = get_internal_variable ( ) ;
      if there exists a VBM_k s. t. VBM_k = VBM_i )
        for each VBM_j which has incoming dd link
                  from p to input port q of VBM_j )
          if there is no control VBM on q
            create control VBM & connect it to q ;
          substitute input port r of control VBM with T ;
        )
      ( else )
        for each VBM_j which has incoming dd link
                  from p to input port q of VBM_j
          substitute q in VBM_j with T ;
      )
      substitute p with T ;
    )
  )
  /* get VBM set which must be executed after VBM_i */
  NVSET = get_next_order_VBM_set (VBM_i) ;
  if NVSET == Φ return ;
  for each VBM_m in NVSET
    global_output dependency_elimination ( VBM_m ) ;
}
```

그림 5. 전역적 출력 의존성 제거 알고리듬

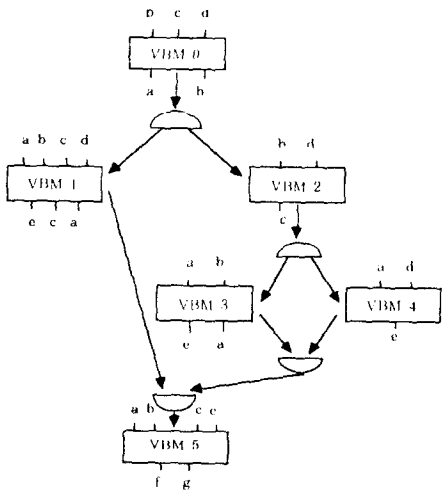
Fig. 5. Algorithm for global output dependency elimination.

```
a := b and c ;
b := b and d ;                                  ] BB0

if k = '1' then

  e := a and c ;
  a := b and d ;                                ] BB1
  c := a and e ;
  a := e and d ;

else

  c := b and d ;                                ] BB2

  if j = '1' then

    e := a and b ;                              ] BB3
    a := a and b ;

  else
    e := a and d ;                              ] BB4

  end if ;
end if ;

f := a and b ;                                  ] BB5
g := c and e ;
```

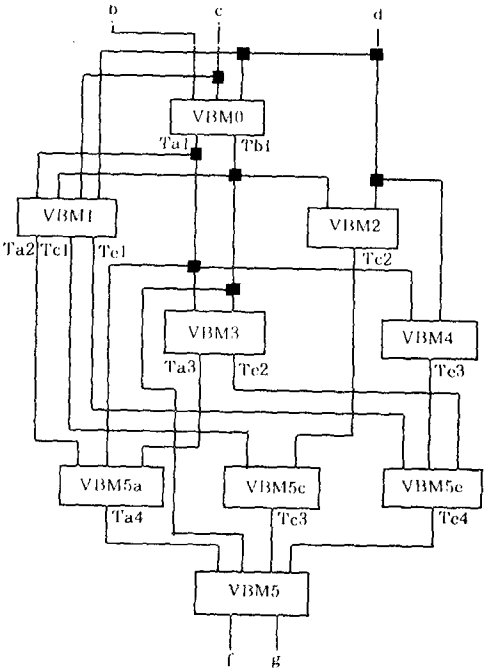
(a)



(b)



(c)



(d)

```
Ta1 <= b and c ;                                ] VBM0
Tb1 <= b and d ;

Tc1 <= Ta1 and c ;
T1  <= Tb1 and d ;                              ] VBM1
Tc1 <= T1 and Tc1 ;
Ta2 <= Tc1 and d ;

Tc2 <= Tb1 and d ;                              ] VBM2

Te2 <= Ta1 and Tb1 ;                            ] VBM3
Ta3 <= Ta1 and Tb1 ;

Tc3 <= Ta1 and d ;                              ] VBM4

f <= Ta4 and Tb1 ;                              ] VBM5
g <= Tc3 and Tc4 ;

Ta4 <= Ta2 when k = '1' else                    ] VBM5a
       Ta3 when j = '1' else
       Ta1 ;

Tc3 <= Tc1 when k = '1' else                    ] VBM5c
       Tc2 ;

Tc4 <= Tc1 when k = '1' else                    ] VBM5e
       Tc2 when j = '1' else
       Tc3 ;
```

(e)

그림 6. 출력 의존성 제거 과정 (a) 순서문을 이용한 설계 기술 (b) VBM1의 내부 구조 (c) VBM 실행 순서 흐름 그래프 (d) VBM 데이타 흐름 그래프 (e) 병행문 형태의 데이타 흐름 기술

Fig. 6. Output dependency elimination process (a) Design description in sequential statements (b) Internal structure of VBM1 (c) Graph representing execution flow of VBMs (d) Graph representing data flow of VBMs (e) Data flow description in concurrent statements.

Ⅳ. 실험 결과

전역적 출력 의존성 제거 알고리듬을 이용한 VSYN 시스템의 실험을 위하여 Intel사의 8 비트 범용 CPU인 8085의 ALU 블럭에 대하여 합성을 수행하였다. ALU 블럭에 대한 레지스터 전송 수준 VHDL 기술은 약 1600 라인 정도이며, 계층적 기술을 사용하여 ALU 블럭의 내부는 10 종류의 부 블럭으로 이루어져 있고, 각 부 블럭들은 대부분 프로세스 문 및 부 프로그램, 함수 등을 이용한 순서문으로 기술되었으며 합성된 각 블럭들은 시뮬레이터[21], [22]를 통하여 검증되었다. 또한 타 시스템과의 비교를 위하여 동일 VHDL 기술에 대한 ViewLogic 사의 PowerView 시스템 및 Synopsys사의 DesignAnalyzer 시스템의 합성 결과를 함께 제시하였다. VHDL로 부터 게이트 수준 하드웨어로의 합성에 대한 정확한 결과 비교를 위하여 논리 최적화, 테크놀러지 매핑 과정 및 모든 합성 제약 조건을 비교 시스템과 동일하게 사용하였으며 테크놀러지 라이브러리는 PowerView 시스템과의 비교시에는 LSI logic사의 lsi100k[23]를, DesignAnalyzer 시스템과의 비교시에는 삼성의 kg60000[24]을 각각 사용하였다.

표 1은 8085 ALU block에 대한 VSYN 및 PowerView의 합성 실험 결과로, 'block' 열은 ALU block내에서 사용된 부 블럭의 이름을, 'count' 열은 ALU block내에서 사용된 해당 블럭의 갯수를 나타내며 'area/blk'은 한 블럭당 차지하는 면적을 나타내고 'cell/blk'은 한 블럭당 사용된 cell의 갯수를 나타낸다. 합성 결과는 PowerView 시스템에 비하여 VSYN 시스템이 cell의 갯수 면에서 3.8%, 면적에서 10.6%의 향상된 결과를 출력함을 보여준다. 표 2는 8085 ALU block에 대한 VSYN 및 DesignAnalyzer의 합성 실험 결과로, VSYN 시스템이 cell의 갯수 면에서 4.2%, 면적에서 4.4%의 향상된 결과를 출력함을 보여준다.

표 1. PowerView 및 VSYN의 8085 ALU 블럭에 대한 합성 결과

Table 1. Synthesis result for 8085 ALU block by PowerView and VSYN.

Block	Count	VSYN		PowerView	
		Area/Blk	Cell/Blk	Area/Blk	Cell/Blk
alu 1	1	355	237	394	253
alu 2	1	83	37	80	36
alu reg 8	2	91	35	129	41
buff en	1	20	13	16	11
buffer g	3	24	8	24	8
code ass	1	82	56	79	56
flag blk	1	133	81	119	70
flag reg	1	72	16	97	25
mux2x1	1	32	24	32	24
shifter	1	59	45	59	45
TOTAL		1090	603	1206	626

표 2. DesignAnalyzer 및 VSYN의 8085 ALU 블럭에 대한 합성 결과

Table 2. Synthesis result for 8085 ALU block by DesignAnalyzer and VSYN.

Block	Count	VSYN		DesignAnalyzer	
		Area/Blk	Cell/Blk	Area/Blk	Cell/Blk
alu 1	1	641	328	682	361
alu 2	1	90	62	101	65
alu reg 8	2	81	33	81	33
buff en	1	18	13	17	12
buffer g	3	16	8	16	8
code ass	1	105	60	107	61
flag blk	1	163	97	178	101
flag reg	1	65	17	65	17
mux2x1	1	24	8	24	8
shifter	1	84	55	72	44
TOTAL		1287	681	1343	710

Ⅴ. 결론

본 논문에서는 레지스터 전송 수준의 초기 VHDL 설계 기술을 받아들여 게이트 수준의 하드웨어를 생성하는 합성 시스템인 VSYN에서 레지스터 전송 수준으로 기술된 순서문의 합성을 지원하기 위한 연구에 대하여 설명하였다. VSYN 시스템은 VHDL의 순서문에 대하여 지역 및 전역적 의존성 분석과 출력 의존성 제거 과정을 행함으로 순서문으로 이루어진 행위 기술의 실행 순서에 의한 동기 구조를 단일 지정 법칙이 적용되는 병행문으로 이루어진 데이타 흐

름 기술의 이벤트 흐름에 따른 비동기 구조로 변환한 뒤 게이트 수준의 실제 하드웨어로 합성한다. 이는 순서문으로 기술된 레지스터 전송 수준의 하드웨어 구조를 동기 회로로 합성함으로 발생하는 불필요한 컨트롤 로직의 추가 등 하드웨어 코스트의 증가를 방지하고 설계자의 의도에 알맞는 최적화된 합성 결과를 얻는데 그 목적이 있다. 추후 과제로는 디자인 계층상 행위 수준과 레지스터 전송 수준의 기술이 혼합되어 사용된 혼합 수준 VHDL 설계 기술에 대하여 계층적 설계 환경 내에서 각 수준의 기술에 대한 partitioning에 대한 연구가 진행되어야 할 것이다.

參 考 文 獻

[1] D. Gajski, "Silicon Compilation," Addison-Wesley Publishing Co.: Reading, MA, 1988.

[2] R. Camposano, "Synthesizing Circuits from Behavioral Descriptions," IEEE Trans. CAD, Vol. 8, No. 2, Feb. 1989, pp. 171-180.

[3] M. McFarland, A. Parker, and R. Camposano, "The High-level Synthesis of Digital Systems," IEEE Proceedings, Vol. 78, No. 2, Feb. 1990, pp. 301-318.

[4] J. Lis and D. Gajski, "Synthesis from VHDL," in Proc. ICCD, Oct. 1988, pp. 378-381.

[5] E. Rundensteiner and D. Gajski, "Functional Synthesis Using Area and Delay Optimization," in Proc. of 29th ACM/IEEE DAC, Anaheim, CA., June 1992, pp. 291-296.

[6] VHDL Logic Synthesis Workshop, Synopsys Inc., Jan. 1991.

[7] J. R. Armstrong, "Chip-Level Modeling with VHDL," Prentice Hall: Englewood Cliffs, NJ., 1989.

[8] S. Carlson, "Introduction to HDL-Based Design Using VHDL," Synopsys Inc., Mountain View, California, 1991.

[9] Workview, Viewlogic Systems Inc., Dec. 1989.

[10] System Design Seminar, Mentor Graphics Inc., Sept. 1992.

[11] IEEE Standard VHDL Language Reference Manual, IEEE 1076-1987, April 1989.

[12] D. Coelho, "The VHDL Handbook," Kluwer Academic Publishers: Boston, MA, 1989.

[13] A. Dewey and A. Gadient, "VHDL Motivation," IEEE Design & Test of Computers, Vol. 3, No. 2, April 1986, pp. 12-16.

[14] A. S. Gilman, "VHDL - The Designer Environment," IEEE Design & Test of Computers, Vol. 3, No. 2, April 1986, pp. 42-47.

[15] D. Perry, "VHDL," R. R. Donnelley & Sons Co., Peterborough, NH, 1991.

[16] MySynOpt Station User's Manual, Seodu Logic Inc., April 1994.

[17] 이영희, 황선영, "VHDL 설계 환경 구축을 위한 Front-end의 설계," 한국 정보 과학회 논문지, 18권, 1호, 1991년 1월, pp. 93-103.

[18] 이영희, 김헌철, 황선영, "혼합레벨 VHDL 시뮬레이터의 설계", 대한전자공학회 논문지, 30-A권 10호, 1993년 10월, pp. 67-76.

[19] A. Aho, R. Sethi, J. Ullman, "Compilers," Addison-Wesley Publishing Co.: Reading, MA, 1988.

[20] S. Dasgupta, "Computer Architecture," Vol. 2, John Wiley & Sons, 1989.

[21] MyVHDL Station User's Manual, Seodu Logic Inc., April 1994.

[22] Vantage Spreadsheet User's Guide, April 1992.

[23] 1.0 mm Array-Based Products Data Book, LSI logic Inc., Sept. 1991.

[24] SEC KG60000 Cell Library Data Book, Samsung Inc., March 1993.

著 者 紹 介

黃 善 泳(正會員)

1976年 2月 서울대학교 전자공학과 졸업. 1978年 2月 한국 과학원 전기 및 전자 공학과 공학 석사 취득. 1986年 10月 미국 Stanford대학 공학 박사 학위 취득. 1976年 ~ 1981年 삼성 반도체 주식회사 연구원. 1986年 ~ 1989年 Stanford대학 Center for Integrated Systems 연구소 연구원, Fairchild Semiconductor Palo Alto Research Center 기술 자문. 1989年 3月 ~ 현재 서강 대학교 전자 공학과 부교수. 주관심 분야는 CAD 시스템, Computer Architecture 및 Systems Design, VLSI 설계 등임.

玄 珉 鎬 (正會員)

1991年 2月 서강 대학교 전자공학과 졸업. 1993年 2月 서강 대학교 대학원 전자공학과 공학석사 취득. 1994年 현재 서강 대학교 대학원 전자공학과 박사과정 재학 중. 주관심 분야는 VLSI 설계 및 CAD 시스템, Computer System Design 등임.