論文2000-37SD-3-8

Don't Care 정보를 이용한 임베디드 소프트웨어의 최적화

(Embedded Software Minimization Using Don't Cares)

洪 裕 杓 *

(Youpyo Hong)

요 약

이 논문은 임베디드 소프트웨어를 위한 소프트웨어 합성시 don't care 정보를 이용하여 합성된 소프트웨어의 성능을 향상시키기 위한 방법을 제시한다. 임베디드 시스템은 주로 실시간 처리가 요구되는 분야에 사용되기 때문에 매우 까다로운 실시간 및 코드 크기의 제한이 있기 때문에 그에 관한 최적화는 매우 중요하다. 우리는 BDD 기반의 유한상태기계 전달 함수의 표현으로부터 확장된 유한상태기계를 유도하는 소프트웨어 합성의 경우에 돈 케어정보를 추출하고, 그를 이용한 BDD를 최소화하여, 일차적으로는 코드크기를 줄이고 결과적으로는 코드가 실행되는 임베디드 시스템의 성능을 향상시키는 방법을 제안한다. 그러한 방법이 적용된 실험적 결과를 제시하고, don't care 기반의 BDD 최소화와 BDD 변수의 동적 재배열의 결합 등 관련 연구 방향을 제안한다.

Abstract

This paper exploits the use of don't cares on software synthesis for embedded systems. Embedded systems have extremely tight real-time and code size constraints. We propose applying BDD minimization techniques in the presence of a don't care set to synthesize code for extended Finite State Machines from a BDD-based representation of the FSM transition function. The don't care set can be derived from local analysis as well as from external information. We show experimental results, discuss their implications, the interactions between BDD-based minimization and dynamic variable reordering, and propose directions for future research.

I. 서 론

회로(ASIC)나 다른 표준 구성요소와 프로그램 가능한 부품의 결합체로 정의하며, 여기서 프로그램 가능한 부분들이란 마이크로 콘트롤러들과 디지털 신호 처리(DSP)등을 의미한다. 부울함수의 효율적인 표현은 논리합성, 테스트, 논리검증 등을 포함한 여러 CAD 분야에 필수적이다.

실시간과 비용에 관한 제약 조건들은 프로그램 가능한 부분에서 실행되어지는 소프트웨어의 최적화(종종 어셈블리 코드 레벨에서의)가 매우 중요함을 의미한다. 그러나 이러한 시스템들의 복잡도의 증가와 시장으로의 공급시간 단축에 대한 필요성은, HDL과 같은 보다 고차원적인 설계 명세로부터의 최적화를 요구하게 되었고, 이 때문에 응용 지향 소프트웨어 합성은 매우 중요한 연구 분야로 인식되기 시작하였다. 즉, 적은 프로그램 시간으로 전문 프로그래머들이 만든 소프트웨어와 견줄 수 있을 만한 질의 소프트웨어를 합성해 내기 위하여 대상분야에 관한 정보와 임베디드 시스템들의 한계 등을 분석하고 활용하려는 노력이 시도되고 있다.

소프트웨어 합성은 특히 정교한 정보 처리용 하드웨어와 소프트웨어설계를 위한 시뮬레이션, 프로토타입

* 正會員, 東國大學校 電子工學科
(Dept. of Electronics Engineering Dongguk Univ.)

※ 본 연구는 2000년 동국대학교 신임 교원 연구비 지원에 의해 이루어지었음.

接受日字:1999年11月24日, 수정완료일:2000年2月12日

설계 등이 필요한 디지털 신호 처리 분야에서 많이 이용된다[15]. 외부의 사건에 대한 신속한 반응, 그리고 순수한 정보 계산의 수준을 넘어선 복잡한 결정 방식 등에 대한 응용을 위한 코드 합성은 유한상태기계와 같은 명세로부터 효과적으로 합성될 수 있는 소프트웨어와 하드웨어를 필요로 한다[3][13][9][8].

본 논문의 목적은 소프트웨어 코드 합성에서 don't care 정보(예를 들면, 불가능한 또는 부적절한 조건 등으로부터 제공되는)를 활용하기 위해서이다. 현재까지 don't care의 사용은 단지 하드웨어 합성분야에서만 이용되어 왔다[11].

Don't care를 이용하기 위해서 우리가 사용하는 소프트웨어 분석 기술은 POLIS[2]를 그 기반으로 하는데, 주목할 점은 POLIS가 확장된 유한상태기계형태와 같은 명세로부터 소프트웨어를 최적으로 합성하기 위하여 BDD(Binary Decision Diagram)[5]를 그 시발점으로 삼고 있다는 점이다. 이 방법은 상태기계 천이 관계식의 최적화 된 구현을 도모하기 위해서 BDD 노드들과 low-level C 문들 사이에 직접적인 연관성을 이용한다.

본 논문에서 고려하는 DC들은 다음과 같다.

1) 유한상태기계 자신에게서 기인하는 내부의 DC들. 특히, 고정된 혹은 알려진 값에 대해 주어진 상태에서 부적절한 FSM 입력들을 인가함으로서 얻어지는 DC들을 고려하였다. 다시 말하자면, 그 이외의 모든 다른 입력 조합들은 자동적으로 DC가 된다.

2) 유한상태기계의 환경으로부터 기인하는 외부의 DC들. 이 부류의 DC는 일반적으로 두 가지 범주에 속한다.

(a) 입력 DC들, 불가능한 입력의 조합이기 때문이다. 특별히 우리는 설계자에 의한 불가능한 것 또는 언어의미론 사항에 의한 금지되어진 입력 조합들을 조건으로 지정되어진 입력 조합을 고려한다.

(b) 출력 DC들, 연결된 유한상태기계들에 의해 무시되어지는 출력 조합 때문이다.

본 논문에서는 실행속도를 증가시키지 않으면서 코드 크기를 최소화하기 위한 목적으로 위에 열거한 다양한 원인들로부터 기인하는 DC들을 다양한 BDD 최소화 알고리즘들을 사용하여 활용한다.

본 논문은 구성은 다음과 같다. 기본적인 관련된 배경 자료는 II절, BDD 최소화의 응용은 III절, 결과는 IV절, 그리고 결론들과 앞으로의 과제는 V절에서 다루고 있다.

II. 배 경

본 절에서는 본 논문의 나머지 부분에서 이용할 여러 가지 기존의 연구[5], [16], [2]를 요약한다.

1. BDD (Binary Decision Diagram)

BDD[2]는 부울함수를 노드집합 N과 에지집합 E로 나타내는 방법으로서, N에 속하는 노드에는 종결노드(leaf node)와 비종결노드(non-leaf node)의 두 가지로 나눌 수 있다. 종결노드는 부울함수 0이나 1의 값을 갖으며 각 비종결노드 u는 두 개의 밖으로 나가는 에지 즉, *then_edge*와 *else_edge*를 갖는다. 각 에지는 u의 자식노드와 연결되어 있고 u는 그 자식노드의 부모노드라고 한다. 두 개의 자식노드들을 형제노드(siblings)라고 한다. 비종결노드 u는 *level(u)*라는 정수 값을 갖고 u, v N일 때, 각 에지 $(u, v) \in$ E는 level(u) < *level(v)*를 만족한다. 비종결노드 F는 순환적으로 정의되는 부울함수이다. 함수 F는 x=1일 때 *then_edge*를 통해 자식노드로 연결되는 함수 Fx와 x=0일 때 *else_edge*를 통해 자식노드로 연결되는 함수 *Fx'*로 정의된다.

2. 다중 출력 함수 (Multi-output Function)

다중 출력 함수는 (또는 동일하게 같은 범위에서 단일 출력 함수들의 묶음) 그들의 특성함수에 의해 표현될 수 있다. 단일 출력이진 값으로 된 함수 χ_f: $(X \times Y) \rightarrow \{0,1\}$, 여기서 $X = X_1 \times ... X_m$ 그리고 $Y = Y_1 \times ... Y_n$는 다중출력 다치 함수 $f : X \rightarrow Y$ if $\chi_f(x,y) \Leftrightarrow (y = f(x))$를 의미한다.

3. 통합설계 유한상태기계(Codesign FSM, CFSM)

[7]에서 통합설계 유한상태기계(CFSM)는 단일 버퍼를 중심으로 비동기적으로 서로 연결된 확장된 유한상태기계로 정의되어 있으나, 본 논문에서는 단일 CFSM의 함수만을 고려한다.

CFSM의 상태는 발생되는 사건에 의해 규정된다. 즉, 특정 상태마다 일정한 사건이 발생하도록 정해져 있다. CFSM의 기능은 BDD에 의해 표현되는데, 그 자세한 방법과 알고리즘은 [7]에 기술되어 있으며, C 코드 생성의 핵심이 되는 알고리즘만을 다음에 간략히 소개한다.

- 입력 변수와 연관되어 있는 BDD 노드는 BDD 자식 노드에 해당하는 문(statement)으로 분기하는 if-then-else문이 된다.
- 출력 변수와 연관되어 있는 BDD 노드는, 종결 노드 1로 가는 경로의 단일 자식에 해당하는 값으로 지정되다.

실제로 POLIS 내에서는 S 그래프라는 중간단계의 표현으로부터 C 코드를 만들면, 이 알고리즘에 의하여 특정 CFSM으로부터 S 그래프를 만드는 예를 그림 1에 나타내었다.

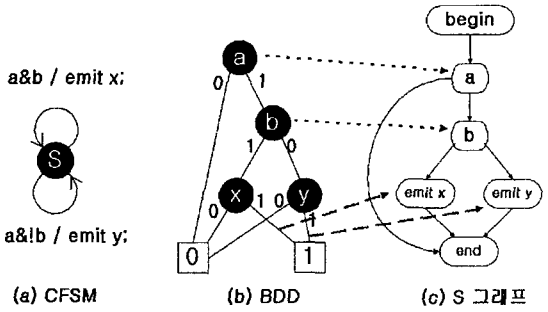

그림 1. CFSM으로부터 S 그래프의 생성
Fig. 1. S -Graph Creation from a CFSM.

이러한 방법은 일반적으로 수동으로 만든 코드보다 빠르게 실행되는 코드를 만들어내는데, 그 이유는 코드의 실행시 각 입력 변수가 CFSM의 천이마다 최대한 한 번씩만 검사되기 때문이다. 예를 들어, 두 단계의 스위치문의 뒤에 여러 개의 지정문이 따르는 CFSM을 표현하는 코드는 기존의 표준 방법에 의해 만들어진 코드보다 빠르다.

이 기술의 또 한가지 매우 중요한 특징은 BDD 노드들과 코드의 부분들이 대부분 1-1대응되기 때문에 BDD크기는 결과적으로 만들어지는 코드의 크기를 미리 예측할 수 있게 하여 주는 지표가 된다는 점이다. 이러한 점이 바로 BDD기반의 코드 합성시 BDD를 최소화하려는 이유이다.

4. Don't Care를 이용한 BDD 최소화.

불완전하게 표시된 함수는 각각 함수의 참집합(on-set)과 케어집합(care-set)을 나타내는 두 개의 BDD, BDD_f 와 BDD_C에 의해 표현할 수 있다. 어떤 함수 g에 대해 만약 f와 g가 모든 케어 경우(care-point)에 대해 같은 함수 값을 가지고 있다면, 즉, $f \cap c = g \cap c$ 이면 [f, c]의 커버(cover)라고 부르며, 이는 g가 f대신에 사용 될 수 있음을 의미한다.

BDD_f와 BDD_C가 주어졌을 경우, 그 표현의 크기가 가장 작은 [f, c]의 커버 BDD를 찾는 것이 DC를 이용한 BDD 최소화 문제이다. 이 문제는 NP-complete[18] 이기 때문에, 실제로는 DC를 이용한 BDD 최소화 근사 알고리즘[10] [6] [12] [21] [14]이 대신 이용된다.

Ⅲ. POLIS에서 BDD 최소화

1. 내부 DC

우리가 다룰 첫 번째 DC는 CFSM 그 자체의 전달 함수의 구조에서 추출된다. 전달 함수는 각각의 입력과 출력 부분의 전달표로써 표현될 수 있다. 입력부분의 '-'은 특정 입력 변수에 대한 모든 값이 가능함을 간략히 표현한 것이며, 출력 부분의 '-'은 해당 입력이 DC임을 뜻한다. 우리의 첫 번째 최소화 작업은 입력 부분의 '-'을 출력부분의 '-'으로 옮기는 것이다. 이 작업은 구체적으로 다음과 같이 수행할 수 있다. 예를 들어 특정 입력에 대해 ab-d와 같이 '-'이 포함된 천이가 주어졌을 때에, 만약 우리가 세 번째 입력의 값이 c가 되도록 보장할 수 있다면, 항상 이 항은 abcd로 고정되어지고, CFSM은 그 이외의 다른 값을 입력으로 갖지 않게 되며, c가 이진 변수라고 가정하면 abc'd가 DC로 취급된다.

우리는 CFSM의 동작을 변형시키지 않으면서 간단히 RTOS를 수정하는 것으로 DC를 활용할 수 있는데, 이는 어떠한 특정조건하에서 CFSM으로 하여금 일부의 지정된 입력 조합만을 인가하도록 RTOS를 구성함으로서 가능하다. 이렇게 하면 그 이외의 나머지 입력 조합은 자동적으로 DC 조건이 된다. 좀더 구체적으로 설명하면, 우리는 어떠한 상태에서 다치 입력 신호가 사용자에 의해 고정된 값을 갖도록 하는 상태 기반 마스크 기능을 사용하는 것이다. 일단 마스크가 정의되면, CFSM 특성 함수의 BDD 표현은 가능하지 않는 다치 입력 신호의 값들을 이용하여 최소화 할 수 있다.

가능한 값의 범위가 $I_i=\{i_1, i_2, \ldots i_l\}$인 한 개의 다치 입력 변수 i를 고려해 보자. 천이 관계식 f는 만약 모든 $1 \le k \le l$에 대해서 $(f \cdot s)_{i_k} = f \cdot s$이면, (여기서 $(f \cdot s)_{i_k}$은 i_k를 평가하는 $f \cdot s$를 나타낸다.) s상태에서 i에 의존하지 않는다. 즉, f의 s상태에서의 동작은 i의 값에

영향을 받지 않는다. 우리는 이 상태에서 I의 값을 한 가지 값으로 고정시키는 RTOS를 사용함으로써 이 독립성을 이용할 수 있다. 즉, 다른 상태값의 조합들은 f를 최소화할 수 DC로 사용할 수 있다.

엄밀하게는, 마스크 M은 다음과 같은 상태들과 입력 변수로부터 특별한 변하지 않는 값을 나타내는 '-'과 다치값들의 합집합으로의 함수로서, 상태$\times i \rightarrow I_i \cup$'-'의 관계식으로 나타낼 수 있다. 예를 들어, 주어진 상태 s와 입력 변수 i에 대하여, $M[s, I]$는 사용자 지정가능 상수 $a_{s,i}$로 지정할 수 있다. (여기서 $a_{s,i}$는 I가 가질 수 있는 값 중 하나로서, $a_{s,i} \in I_i$ 이다.) 이러한 입력 변수들과 상태들의 조합에 대하여, 다치 입력 신호는 RTOS에 의해 변경되어서는 안되며, 마스크는 '-'로 결정된다.

마스크와 해당되는 DC 조합을 계산하기 위한 알고리즘이 그림 2에 보여진다.

```
Algorithm Find_DC_and_Mask (T: characteristic function)
/* D: 결과로 얻는 DC 함수, d: DC cube */
D = 0
foreach state s
  foreach input signal I
    d = 0
    if(T is independent of i in s) then
      M[s, i]= a_{s,i} /* a_{s,i}: s 상태에서 사용자가 지정
                          한 i의 값 */
      d = complement( a_{s,i})
      D = D + d
```

그림 2. DC 집합의 계산과 매스크
Fig. 2. Computing the DC set and the Mask.

이 최적화의 대가는 마스크 M의 수행을 위해 필요한 RTOS의 부담이다. 다행히, POLIS의 변수는 비트 묶음으로 되어 있으므로, 우리는 일반적으로 한 상태당 한 정수의 정수 배열로 마스크를 나타낼 수 있고, 이로 인한 실행시간 증가는 대부분 미미하다.

2. 외부 DC

본 절은 외부의 입력 DC들을 활용하는 방법을 제시한다. 이러한 DC들은 환경적 제한과 다른 CFSM의 기능 때문에 일부 입력 조합들이 발생할 수 없음을 의미한다. 원칙적으로는 CFSM의 구성으로부터 다른 CFSM로부터 기인한 이와 같은 종류의 DC들을 자동적으로 추출하는 것이 가능하지만, 여기서 우리는 단지 특정 언어에 포함된 설계자가 표시한 입력 DC만을 고려한다. 특히, 우리는 두 가지 입력 언어 Esterel[4] 그리고 SDL[17]에 초점을 맞춘다.

Esterel은 확장 유한상태기계의 관점에서 의미가 정의된 문자 기반 언어이고 POLIS를 위한 입력 언어의 하나로 채택되어 있다. Esterel 표현은 한정된 입력에 대한 제어력을 표시하는 관계를 포함하고 한다. 예를 들어 Esterel 문

relation a#b#c#

은 입력 a와 b와 c가 동시에 일어날 수 없다는 것을 의미한다. 이들 변수에 대한 모든 함수에 관해서, DC $=ab + ac + bc$ 는 유효한 DC이다.

이러한 외부 DC를 구하는 방법은 편집기에 포함시키어 자동화시키는 것이 가능하겠으나, 본 논문에서는 외부 DC의 유용성을 보이는 것이 목적이므로 자동화용 알고리즘을 제시하지는 않는다.

POLIS에 의해 사용되는 Esterel 편집기에서, 그러한 관계식들은 시뮬레이션이나 인과 분석을 하는 경우 매우 중요하게 다루어진다. 그러나 현재까지 일반적으로는 구현을 최적화 하기 위해 항상 그러한 관계를 이용하지 않는다.

SDL은 원격통신 시스템에서 특별히 네트워크 프로토콜에 대해 적합한 그래픽과 문서에 기초를 둔 언어이다. SDL은 확장유한상태기계의 개념도 포함하고 있으나 현재 POLIS의 입력언어로 채택되지는 않았다. SDL은 확장유한상태기계로 전달되는 입력 신호가 하나씩 하나씩 들어오는 것을 전제로 한다. 이것은 SDL에서는 입력이 없는 사건 혹은 하나 이상의 입력이 동시에 존재하는 사건은 DC로 간주됨을 의미한다. POLIS는 일반적으로 SDL에 CFSM의 표현을 허용하지 않고 있기 때문에 우리는 모든 CFSM입력들은 상호 배타적이라고 기술하는 Esterel 관계식을 사용하여 이런 종류의 DC의 BDD에 대한 영향을 분석하였다.

Ⅳ. 실험결과

이 절에서는 앞 절에서 논의한 DC의 종류에 대해 다양한 최소화 알고리즘의 적용하여 실험한 결과를 기술한다. 모든 알고리즘을 적용하였을 때 얻은 가장 우수한 실험 결과가 각 열마다 표시되어 있다. (최소화 알고리즘간의 차이는 사실상 2%내에 불과 하였다).

다음절에서 우선 Ⅲ.2.절에서 묘사한 Esterel로부터의 외부 DC로 이용하였을 때의 결과들을 설명한다. 즉 한 시점에서 단 한가지의 입력사건만이 CFSM으로 전달될 수 있다는 것을 전제로 하였다.

주목할 점은 그럼에도 불구하고 외부 DC를 이용한 BDD의 최소화가 다른 방법들(예를 들어 BDD 변수 재배치에 의한)과는 독립적이라는 점이다. 즉 DC정보는 설계자들로부터 또는 언어 의미론으로부터 취해질 수 있으며 코드크기 감소에 사용되어질 수 있다.

한편 Ⅲ.1.절에서 설명한 내부 DC는 BDD 변수의 동적 재배치에 의한 최소화와 밀접하게 상호작용을 하는 것을 발견할 수 있었다. (참고로, 본 실험에서는 여러 가지 BDD 변수의 동적 재배치 알고리즘 중 그 성능이 가장 우수한 것으로 알려진 sifting[16]을 사용하였다.) 이것이 사실 우리가 내부 DC를 이용했을 때 어떠한 BDD크기도 더 이상 감소시키지 못하였던 이유이다. 즉, 우리가 내부 DC를 이용했을 경우 만약 sifting을 먼저 적용하면 일단 BDD 크기는 크게 감소하는데, 그 이후에 DC를 이용한 BDD 최소화 알고리즘을 적용하면 더 이상 BDD의 크기를 감소시킬 수가 없었던 것이다.

1. Esterel 내에서의 최적화

다음 실험은 현재의 Esterel 기술만으로 외부 DC를 이용해서 얻을 수 있는 C 코드의 크기가 얼마나 감소될 수 있는 가를 확인하기 위한 것이다. 여기서 우리는 BDD 최소화 알고리즘을 단독으로 이용했을 경우와 그를 BDD 변수 재배치와 결합했을 때의 차이를 비교하였다.

실험대상인 각 예마다 세 가지 다른 C 파일이 생성되었다. 원래의 Esterel 예, 그를 최적화 시킨 것, 그리고 Ⅲ.2.절에서 언급한 여러 입력이 동시에 발생하지 못한다는 관계식을 추가한 뒤에 최적화 시킨 것, 이렇게 해서 세 가지 결과를 보여주고 있다. Esterel에서 최적화 기법을 사용하기 위해, Esterel 편집기를 사용해서 제어부를 추출해야 했고, 그 위에서 논리합성에서 사용하는 알고리즘을 실행시켜서 일차적인 최적화를 시행하였다. 그리고 활용된 제어부를 재 삽입하고, C 코트에 Esterel 편집을 실시하였다. Esterel 편집기에 이미 내장된 최적화 기술은 SIS 순차적인 논리 분석 프로그램인 SIS[19]와 래치제거를 위한 진보된 알고리즘인 rem-latch 프로그램[20]을 효과적으로 사용하는 전형적인 스크립트를 사용하였다.

이 실험의 결과가 표 1에 나타나 있다. C 코드 크기는 바이트단위로 표시되어 있다. 단순한 최적화와 관계식이 추가된 이후의 최적화가 각각 다소의 코드크기 감소에 기여했음을 발견할 수 있다.

표 1. Esterel C 생성시 DC의 효과
Table 1. Effects of DCs on Esterel C-code Generation.

방법 / 회로	Original	Opt	Rel&Opt
abcd	16,673	15,903	14,767
abcdef	24,450	23,329	22593
arbiter12	55439	18,438	18,442
ex1	28,443	20,200	20,002
ex2	313,222	232,212	232,001

2. POLIS 내부에서의 최적화

다음 실험은 sifting을 실시한 이후에 DC-기반 BDD 최소화를 적용한 결과를 보여준다. 최종 코드 크기대신 그를 구성하는 기본 C문으로 대신 보고하였다. 이는 우리가 사용한 data 계산이 없는 예의 경우에는 기본 C 문의 개수가 최종 코드 크기에 거의 그대로 반영되기 때문이다. 각 열의 의미는 다음과 같다.

- No Esterel Opt는 이전의 절에서 설명한, Esterel 최적화의 효과를 포함하지 않았다.
- Esterel Opt는 BDD 생성 전에 첫 단계로써, Esterel 최적화의 효과를 포함한 것이다.

비록 Esterel 편집기에 의한 Esterel 최적화는 코드 크기를 줄이기는 하나, 이것이 함수를 바꾸지는 않으면

서, Boolean 망의 구조를 변경할 수는 있다는 점을 주목할 필요가 있다.

- DC Opt는 DC를 사용하지 않은 경우이다.
- DC Opt는 가장 우수한 DC 기반의 BDD 최소화 후의 결과를 보여준다.

예들의 이름의 끝 *_rel*은 이미 Esterel에서 지정된 외부 DC를 이미 가지고 있음을 나타낸다. 일례로, Esterel Opt열에서 abcdef_rel의 DC Opt 결과는 abcdef에에 Esterel의 외부 DC를 포함하였고, 본 논문에서 다룬 DC도 추가되어 최적화된 경우의 결과이다.

표 2. POLIS C 생성시 DC의 효과
Table 2. Effect of DCs on POLIS C-code Generation.

방법 / 회로	No Esterel Optimization		Esterel Optimization	
	No DC Opt	DC Opt	No DC Opt	DC Opt
abcd	496	469	496	473
abcd_rel	93	63	94	85
abcdef	4,023	3,959	4,023	3,959
abcdef_rel	171	105	162	149
arbiter12	406	172	406	172
arbiter12_rel	406	172	197	195
ex1	3,201	2,910	3,201	2,910
ex1_rel	233	201	223	219
ex2	82	78	80	66
ex2_rel	58	52	47	40

이 표로부터 간단한 결론이 얻어질 수 있는데, 이는 일반적으로 Esterel 최적화 단계에서 DC 는 Esterel 편집 단계에서는 사용하지 않는 것이 바람직하다는 것인데, 이는 DC의 사용이 최적화의 가능성을 더욱 제한할 수도 있기 때문인 것으로 추측된다. 이것은 아마도 위에서 논의한, BDD변수 재배치 전에 DC를 이용했을 때의 결과와 유사하다.

V. 결 론

임베디드 소프트웨어의 최적화에 관해서는 많은 과제가 남아 있는데, 그 중에서 특히 강조할 필요가 있는 분야로 DC를 이용한 BDD최소화와 동적 BDD 변수순서 재배치의 효과적 결합을 꼽을 수 있다. DC를 이용한 BDD최소화와 동적 BDD 변수순서 재배치간에 주요한 상관관계가 있음은 쉽게 짐작할 수 있다. DC를 이용한 BDD최소화는 다양한 함수중의 하나를 선택한다는 점에서 그 잠재능력이 큰 반면 동적 BDD 변수순서 재배치는 단일 함수의 표현 구조만 바꾼다는 점에서 다소 제한적이다. 그러나 본 논문의 실험결과는 DC를 이용한 BDD최소화의 잠재능력은 크지만 실제로는 동적 BDD 변수순서 재배치의 BDD크기에 대한 영향이 더욱 크다는 점을 시사하였다.

물론 이러한 결과가 임베디드 소프트웨어에 국한된 현상인지는 확신할 수는 없으며, 대상 함수의 특성에 국한되지 않는 일반적인 BDD에 관한 두 기법의 효과적 적용방법에 관한 연구는 그 실용적 가치가 클 것으로 기대된다.

참 고 문 헌

[1] A. V. Aho and R. Sethi and J. D. Ullmann, Compilers: Principles, Techniques and Tolls. Addition -Wesley, 1988.

[2] F. Balarin et al. Hardware-Software Co-design of Embedded Systems - The POLIS approach. Kluwer Academic Publisher, 1977.

[3] G. Berry. The Construtive Semantics of Pure Esterel. To Appear, available now at ftp://www.inria.fr/meije/esterel/papers/constructiveness.ps.gz, 1996.

[4] G. Berry and P. Couronn and G. Gonthier. The Synchronous Approach to Reactive and Real-time Systems. IEEE Proceedings, 79, September, 1991.

[5] R.Bryant. Graph-Based Algorithms for Boolean Function Manipulation. IEEE Transactions on Computers, C-35(8), pp. 677-691, August, 1986.

[6] S. Chang and D.I. Cheng and M. Marek-Sadowska. Minimizing ROBDD Size of incompletely Specified Multiple Output Functions. In Proc. European Design and Test

Conference, pp. 620-624, 1994.

[7] M. Chiodo et al. Synthesis of Software Programs from CFSM Specifications. In Proc. Design Automation Conference, June 1995.

[8] P. Chou and G. Borriello. Software Scheduling in the Co-synthesis of Reactive Real-time Systems. In Proc. Design Automation Conference, June 1994.

[9] C.N. Coelho and G. De Micheli. Analysis and Synthesis of Concurrent Digital Circuits using Control-flow Expressions. IEEE Transaction on Computer-Aided Design, 15(8), pp. 854-876, August 1996.

[10] O. Coudert, C. Berthet, and J.C. Madre. Verification of Synchronous Sequential Machines Based on Symbolic Execution. In Automatic Verification Methods for Finite State Systems, pp. 365-373. Springer -Verlag, 1989.

[11] S. Devadas, A. Ghosh, and K. Keutzer. Logic Synthesis. McGraw -Hill, 1994.

[12] R. Dreshsler and N. Gockel. Minimization of BDDs by Evolutionaly Algorithms. In Proc. Int. Workshop on Logic Synthesis, 1997.

[13] R. K. Gupta, C. N.Coelho Jr., and G. De Micheli. Program Implementation Schemes for Hardware-Software Systems. IEEE Computer, pp. 48-55, January 1994.

[14] Y. Hong, P.A, Berrel, J.R. Burch, K.L. McMillan. Safe BDD Minimization Using don't Cares. In Proc. Design Automation Conference, pp.208-213.

[15] S. Ritz et al. Code Generation Techniques in the Block Diagram Oriented Design Tool COSSAP /DESCARTES. In Proc. International Conference on Signal Processing Applications and Technology, pp.709-714, 1994.

[16] R. Rudell. Dynamic Variable Ordering for Ordered Binary Decision Diagrams. In Proc. Int. Conference on Computer-Aided Design, pp.42-47, Nov. 1993.

[17] R. Saracco, J. R. W. Smith and R. Reed. Telecommunications Systems Engineering Using SDL. North Holland - Elsevier, 1989.

[18] M. Sauerhoff and I. Wegener. On the Complexity of Minimizing the OBDD Size for Incompletely Specified Functions. IEEE Transactions on Computer-Aided Design, pp.1435-1437, Nov. 1996.

[19] E. M. Sentovich et al. Sequential Circuit Design Using Synthesis and Optimization. In Proc. Int. Conference on Computer Design, pp.328-333, October 1992.

[20] E. M. Sentovich, H. Toma, and G. Berry. Efficient Lath Optimization Using Exclusive Sets. In Proc. Design Automation Conference, pp.8-11, June 1997.

[21] T. Shiple, R. Hojati, A. Sangiovanni -Vincentelli, and R. K. Brayton. Heuristic Minimization of BDDs Using Don't Cares. In Proc. Design Automation Conference, pp. 25-231, June 1994.

저 자 소 개

洪 裕 杓(正會員)

1991년 연세대학교 전기공학과 학사. 1993년 University of Southenr California 전기공학과 석사. 1998년 University of Southern California 컴퓨터공학과 박사. 1998년 7월~1999년 2월 Synopsys, Hillsboro 에서 정형검증알고리즘 연구. 1999년 2월~현재 동국대학교 전자공학과 재직