論文99-36C-5-1

회로 설계 검증을 위한 스위치-레벨 이진 결정 다이어그램

(Switch-Level Binary Decision Diagram(SLBDD) for Circuit Design Verification)

金 慶 期 *, 李 東 垠 *, 金 周 號 *

(Kyungki Kim, Dongeun Lee, and Juho Kim)

요 약

본 논문에서는 스위치-레벨 회로의 검증(verification)을 위해서 이진 결정 다이어그램(BDD : Binary Decision Diagram)을 구현하는 새로운 알고리즘을 제안한다. 스위치-레벨에서 기능(function)들은 스위치들의 직·병렬 연결에 의해서 결정되며, 결과 논리 값은 논리 '0'과 '1' 뿐만 아니라, 초기 상태, 고 임피던스와 불안정 상태를 가진다.

따라서, 본 논문에서는 "스위치-레벨 이진 결정 다이어그램(SLBDD : Switch- Level Binary Decision Diagram)"으로 정의한 비 사이클 그래프(acyclic graph)들을 사용해서 스위치-레벨 회로의 기능들을 표현하도록 BDD를 확장하였다. 그러나, 그래프의 기능적 표현은 최악의 경우 입력 변수들의 수에 지수 함수적이 되므로, 결정 다이어그램의 변수 순서(ordering)는 그래프 크기에 주된 역할을 하게된다. 따라서, 패스-트랜지스터와 도미노-논리가 존재하는 사전에 충전하는 회로(Precharging circuitry)에서 그래프 크기에서의 효율성을 위한 입력 순서 알고리즘을 제안한다. 그리고, 실험 결과는 여러 가지 벤치-마크 회로에서 여러 번의 실험을 통해서 제안된 알고리즘이 스위치-레벨에서의 기능적 시뮬레이션, 전력 측정과 결점 시뮬레이션에 적용될 수 있을 만큼 충분히 효율적임을 보여준다.

Abstract

A new algorithm of constructing binary decision diagram(BDD) for design verification of switch-level circuits is proposed in this paper. In the switch-level circuit, functions are characterized by serial and parallel connections of switches and the final logic values may have high-impedance and unstable states in addition to the logic values of 0 and 1. We extend the BDD to represent functions of switch-level circuits as acyclic graphs so called switch-level binary decision diagram (SLBDD).

The function representation of the graph is in the worst case, exponential to the number of inputs. Thus, the ordering of decision variables plays a major role in graph sizes. Under the existence of pass-transistors and domino-logic of precharging circuitry, we also propose an input ordering algorithm for the efficiency in graph sizes. We conducted several experiments on various benchmark circuits and the results show that our algorithm is efficient enough to apply to functional simulation, power estimation, and fault-simulation of switch-level design.

Ⅰ. 서 론

오늘날과 같은 고집적도의 칩(chip) 설계 환경에서는 회로의 아키텍처-레벨(architecture-level)부터 물리적 레이아웃(layout)레벨까지의 여러 설계 단계에서

* 正會員, 西江大學敎 電子計算學科
(Department of Computer Science Sogang University)

※ 본 연구는 정보통신부 지원으로 이루어졌습니다.
接受日字:1998年10月23日, 수정완료일:1999年4月28日

적용되는 회로 검증(verification)과 최적화(optimization)과정이 존재한다. 전통적인 기능 테스트(functional test)는 동작 (behavioral)-레벨 또는 논리-레벨에서 수행되며, 알려진 논리 검증 방법 중의 하나가 이진 결정 다이어그램(Binary Decision Diagram :BDD)[16]이다. BDD는 불린 함수(Boolean function) 표현과 조작에 효과적이며, 게이트-레벨 설계에서 쉽게 적용된다.

그리고, 주문형(full-custom) 또는 반주문형(semi-custom) 디자인에서 MOS 회로들은 도미노 논리와 패스-트랜지스터 등을 이용한 여러 가지 기술[13,14]들을 사용해서 설계되고, 최적화 되고있다. 스위치-레벨 디자인 검증은 MOS 회로에서의 전하 공유(charge sharing), 양방향 신호 흐름(bidirectional signal flow), 신호 크기(signal strength), 신호 충돌의 해결(resolution of conflicting signal)과 같은 물리적인 여러 복잡한 특성을 포함한다. 이런 문제를 해결하기 위해서 스위치-레벨에서의 모델링[1], 신호 흐름 유도(signal flow derivation)[2], 회로 분할(circuit partition)[3], 결점 시뮬레이션(fault simulation)[3]과 심볼릭 시뮬레이션(symbolic simulation)[6] 등과 같은 여러 연구가 지금까지 이루어지고 있다. 스위치-레벨에서 기능 검증 방법 중 하나는 스위치-레벨에서 게이트-레벨로의 모델링과 변환[8,11]이며, COSMOS[8]는 스위치-레벨의 기능적 동작을 게이트-레벨로 추출하는 가장 잘 알려진 접근 방식이다. 그러나, 스위치-레벨 디자인에서 게이트-레벨 디자인으로의 매핑(mapping)은 제한적이며, 상세한 회로 정보를 잃게된다. 따라서, 본 논문에서는 새로운 스위치-레벨 디자인 검증 방법을 제안한다. 스위치-레벨 회로의 네트워크를 기능적 검증을 위한 유일한 비 사이클 그래프의 형태로 표현하기 위해 기존의 이진 결정 다이어그램(BDD)을 변형하고, 확장한다. 스위치-레벨 디자인에서 출력은 정상적인 상태인 논리 상태(logic state)는 논리 '1', 논리 '0' 뿐만 아니라, 초기 상태(unknown state), 고 임피던스(high impedance), 불확정 상태(unstable state) 일 수 있다. 초기 상태는 본 논문에서는 'U'로 나타내고, MOS의 최초의 초기 값을 나타낸다. 고 임피던스는 본 논문에서는 'Z'로 표기하며, 출력 노드가 MOS 입력 값들에 의해 전력 소스(power source) VDD와 그라운드(ground) GND로부터 모두 분리될 때 발생한다. 불확정 상태는 본 논문에서는 'X'로 표기하며, 결정할 수 없는 상태(indeterminate state)로 불리기도 한다. 출력 노드가 VDD와 GND로 동시에 모두 연결될 경우 발생한다. 이렇게 출력 논리 상태로써 5개의 값을 가질 경우, 회로의 유일한 형태(canonical representation)인 스위치-레벨 이진 결정 다이어그램(Switch-Level Binary Decision Diagram : SLBDD)을 구현하는 방법을 제안한다. BDD의 특성 때문에, 그래프는 입력 변수의 수에 지수적으로 증가한다. 따라서, 적당한 입력 변수의 순서가 정해지지 않으면, 그래프는 메모리 공간 부족을 가져올 만큼 지나치게 커지게 된다. 몇 몇 축약된 순서 이진 결정 다이어그램(Reduced Ordered BDD)을 위한 순서 알고리즘이 [17,18,19,20]에서 소개되었다. 하지만, 패스-트랜지스터와 팬-아웃(fanout) 노드들의 존재 하에서는 위에서 제안된 알고리즘들이 적당한 그래프 크기를 만들 수 없을 것이다. 그러므로, 우리는 본 논문에서 SLBDD를 위한 효과적인 변수 순서 알고리즘을 제안한다.

본 논문은 먼저 2장에서 이론적인 배경에 대해서 논한다. 3장은 두 부분으로 나눈다. 첫 번째 부분은 SLBDD를 구현하는 방법을 Phase I과 Phase II로 나누어 설명한다. Phase I은 SLBDD 구현의 기본적인 과정인 회로 네트워크 모델링과 스테이지(stage) 발견에 대해 설명하고, Phase II는 SLBDD를 구현하는 상세한 알고리즘을 소개한다. 3장의 두 번째 부분은 SLBDD를 위한 새로운 변수 순서 알고리즘을 제안한다. 실험 결과는 4장에서 보여주고, 다음 5장에서 본 논문의 결론을 내리겠다.

Ⅱ. 이론적 배경

본 장에서는 본 논문을 이해하는데 필요한 BDD의 기본적인 개념과 SLBDD 방법의 동기(motivation)를 소개한다.

1. 이진 결정 다이어그램(Binary Decision Diagram: BDD)

축약된 순서 이진 결정 다이어그램은 불린 함수의 그래프적인 표현이며, R. Bryant에 의해 유일한 형태(canonical form)임이 증명되었다. 함수 $f(x_1,...,x_n)$의 이진 결정 다이어그램은 두 개의 vertex 타입(non-terminal vertex와 terminal vertex)을 포함하는 vertex 집합 v를 가지는 루트 vertex가 존재하는

DAG(Directed Acyclic Graph)형태가 된다. 입력에 대한 nonterminal vertex v_i는 각 vertex가 논리 값 '0'과 '1'에 의존하는 두 개의 후손 vertex인 low(v), high(v)를 가지는 결정 변수(decision variable)이다 ($v_i \in \{x_1,...,x_n\}$). terminal vertex v_i는 후손(descendant)을 가지지 않으며, '0'과 '1'이다($v_i \in \{0,1\}$). 루트 vertex를 가지는 BDD는 다음과 같은 형태로서 함수 f_v가 표현된다.

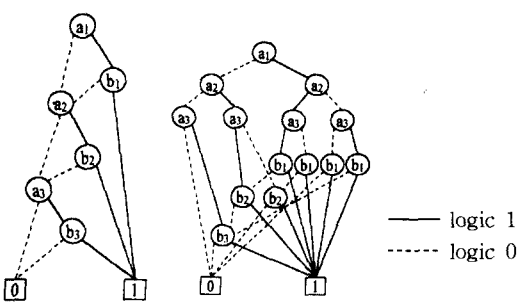
(1) vertex v_j가 terminal vertex 라면,
  a) value(v_j) = 1 이면, f_v = 1,
  b) value(v_j) = 0 이면, f_v = 0

(2) vertex v_i가 index(v_i) = i 를 가지는 nonterminal vertex 라면,

$$f_v(x_1,...,x_n) = x_i \cdot f|_{xi=0} + x_i \cdot f|_{xi=1}$$

축약된 순서 이진 결정 다이어그램(ROBDD)[17]은 적당한 변수 순서와 불필요한 vertex들을 제거함으로써 불린 함수의 유일한 형태가 형성됨을 제안하였다.

그림 1은 ROBDD의 예로써 함수 $F = a_1 \cdot b_1 + a_2 \cdot b_2 + a_3 \cdot b_3$에 대해 그림 1(a)는 a_1,b_1,a_2,b_2,a_3,b_3의 입력 순서를 가지는 유일한 형태의 BDD를 나타내며, 그림 1(b)는 a_1, a_2, a_3, b_1, b_2, b_3의 변수 순서를 가지는 BDD를 나타낸다. 그림 1로부터 변수 순서가 그래프의 크기를 결정함을 명백히 알 수 있다. 그러나, 가능한 최소의 크기의 그래프를 찾는 것은 co Np-Complete 문제이며, 최적의 크기를 찾기 위한 알고리즘이 계속 제안되고 있다.



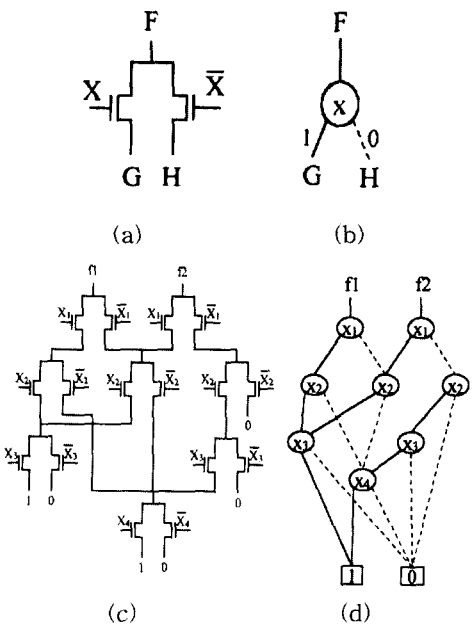
(a) (b)

그림 1. 입력 순서에 따른 $a_1 \cdot b_1 + a_2 \cdot b_2 + a_3 \cdot b_3$ 함수의 BDD 표현 : (a) a_1,b_1,a_2,b_2,a_3,b_3의 순서 (b) a_1,a_2,a_3,b_1,b_2,b_3의 순서

Fig. 1. According to input order $a_1 \cdot b_1 + a_2 \cdot b_2 + a_3 \cdot b_3$ Function of BDD representation: (a) The order of a_1,b_1,a_2,b_2,a_3,b_3 (b) The order of a_1,a_2,a_3,b_1,b_2,b_3.

2. 스위치-레벨에서의 BDD 활용

지금까지 스위치-레벨에서 BDD의 활용은 패스-트랜지스터로 구성된 회로의 논리 합성(logic synthesis)에서 제안되어왔다.[4,12,15]. 현대의 VLSI 설계에서 정적 회로(static circuit)는 영역과 전력의 감소를 이루이기 위해 패스-트랜지스터로 변환된다. 그림 2는 패스-트랜지스터를 포함한 논리 함수를 모델링하는 방법을 보여준다. 그림 2(a)의 패스-트랜지스터는 그림 2(b)의 BDD로 표현된다. 출력 노드 F에 대해 입력 X가 G 또는 H를 연결시키는 스위치로 동작한다. 따라서, 복잡한 패스-트랜지스터로 구성된 그림2(c) 회로는 그림2(d)와 같은 BDD로 표현된다. 그리고, 매핑된 BDD를 축약시켜서 vertex 수를 줄임으로써 패스-트랜지스터로 구성된 회로 최적화를 위한 합성(synthesis)이 효율적으로 이루어 질 수 있다.



(a) (b) (c) (d)

그림 2. 패스-트랜지스터의 BDD 매핑, (a)패스-트랜지스터 (b) 패스-트랜지스터를 매핑한 BDD, (c) 복잡한 패스-트랜지스터 네트워크(d) 대응되는 BDD 표현

Fig. 2. The BDD mapping of Pass-Transistor (a) Pass-Transistor (b) The BDD mapping of Pass-Transistor (c) Complex pass-transistor network (d) The representation of a match BDD.

패스-트랜지스터가 포함되지 않은 MOS 회로에서는 SP-BDD[4]를 사용해서 정적 CMOS의 구조와

논리 기능을 효과적으로 표현할 수 있다. 즉, CMOS 회로의 입력을 BDD를 사용해서 최적의 순서로 합성할 수 있으며, 몇 개의 게이트 논리를 하나의 CMOS로 표현할 수 있다[15]. 지금까지의 이런 방법을 통해서 MOS회로를 최적화 시킬 수 있다.

하지만, 아직까지 이런 접근 방식은 스위치-레벨 회로에서의 특성과 상태를 표현할 수 없으며, 게이트-레벨과는 다른 변수 순서 알고리즘이 제시되지 않았다. 다음 장에서는 이런 문제를 해결하기 위한 새로운 스위치-레벨 이진 결정다이어그램을 제안한다.

Ⅲ. 스위치-레벨 이진 결정 다이어그램 (SLBDD)

본 장에서는 SLBDD를 구현하는 알고리즘과 휴리스틱(heuristic) 변수 순서 알고리즘을 제안하고, 설명한다.

1. SLBDD 구현

SLBDD의 구현은 두 가지 단계로 구성된다. 첫 번째 단계는 입력 네트리스트(netlist)로부터의 회로 네트워크 모델링과 스테이지(stage) 발견을 수행하는 준비 단계이다. 두 번째 단계는 SLBDD 구현 알고리즘을 다룬다.

1) Phase I: 네트워크 모델링과 스테이지 발견

회로 네트리스트로부터 스위치-레벨 회로는 SLBDD 구현의 첫 번째 단계로써 모델화 된다. 회로는 네트워크의 계층적인 데이터 구조(hierarchical data structure), 셀(cell), 디바이스(device), 노드(node), 포트(port)로 표현된다.

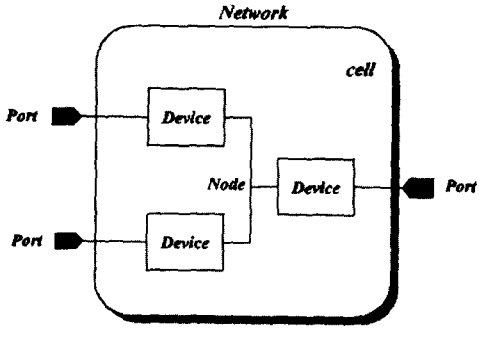


그림 3. 네트워크 모델링
Fig. 3. Network Modeling.

셀은 서브 네트워크(subnetwork)의 모델 또는 모듈(module)을 표현하고, 네트워크는 가장 높은 레벨에서의 회로를 표현한다. 디바이스는 회로 소자(NMOS, PMOS)인 셀 인스턴스(cell instance)를 나타내고, 포트는 다른 서브 네트워크와 연결되는 단자 노드(terminal node)이다. 그림 3은 네트워크 모델링을 보여준다.

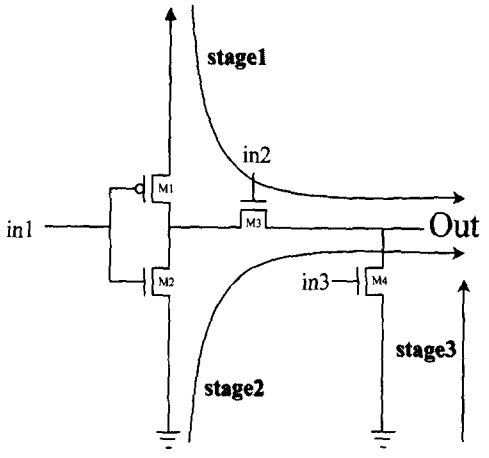


그림 4. 스테이지 구조
Fig. 4. Stage structure.

회로 네트리스트를 하위 회로(subcircuit)의 네트워크로 변환한 후에, 스테이지 발견 과정(stage detection process)이 채널 연결 영역(Channel-Connected Region:CCR)내에서 수행된다[3,6]. CCR에서 스테이지는 전력 소스 VDD 또는 그라운드 GND로부터 CCR의 출력노드까지의 연결 성분이다. 스테이지 S는 출력 노드 f를 가지는 스테이지 S에서 디바이스(M_i)가 n개일 때 경로 S = {VDD(GND), M1, M2, ..., Mn, f}이다. 그림 4는 패스-트랜지스터를 포함하는 CCR을 보여주고 있다. 그림 4에서는 하위 회로가 3개의 스테이지로 구성된다. stage 1은 VDD에서 출력 노드까지의 경로이고, stage 2와 stage 3은 GND로부터 출력 노드까지의 경로이다. 이런 스테이지는 CCR 내에서 depth-frist search에 의해 발견되며, BDD 구현에서 출력 상태 결정을 위해서 데이터 구조로 저장된다.

SLBDD를 구현하기 위해서 기능 시뮬레이션이 각 CCR에서 실행되어야 한다. 스위치-레벨 회로 네트워크에서 내부 상태는 상승(rising) '1', 하강(falling) '0', 초기 상태 'U', 고 임피던스 'Z', 불확정 상태 'X'의 논리 값을 가진다. 그러나, 주 입력(primary

input)은 '0'과 '1'만 존재한다고 가정한다. 노드들의 논리 값은 입력 값과 디바이스의 출력 논리 상태에 의존한다. 5개의 입력 값을 가지는 디바이스(PMOS, NMOS)의 출력 논리 상태는 표 1에서 표현하였다. 정상적으로 디자인된 NMOS 트랜지스터에서 게이트 입력 값이 '1'이면, NMOS는 '0'의 값을 전달하며, 단락(short)되었음을 의미한다.

표 1. MOS에서의 입력에 따른 상태
Table 1. The state for inpit in MOS. 1: logic 1, 0: logic 0, U: Unknown(Initial), Z: High Impedance, X: indeterminate

type / input	NMOS	PMOS
U	U	U
1	0	Z
0	Z	1
Z	Z	Z
X	X	X

그러나, 게이트 입력 값이 '0'이면, NMOS를 트리거(trigger)할 수 없으므로 트랜지스터의 드레인 노드는 고 임피던스 상태가 되며, MOS는 단선(open)이 된다. 비슷한 방법으로, NMOS 또는 PMOS에서 게이트 입력이 'U', 'Z', 'X' 상태 값을 가지면, 출력 노드에서는 입력과 같은 상태 값을 가지게 된다.

본 논문에서는 5개의 논리 값이 출력 상태로 표현되는 Value Function을 Val( )을 정의한다. 이때, Val(M), Val(S)과 Val(f)는 각각 CCR에서 디바이스, 스테이지와 출력 노드에서의 출력 값을 표현한다. CCR에서 depth-first search에 의해 발견된 스테이지에 대해서 각 스테이지에서 출력 값은 스테이지 Val(S)의 Value Function에 의해 구해진다. 표 2는 여러 가지 디바이스의 조건하에서 스테이지의 출력 값을 보여준다. VDD로부터의 스테이지는 S_{VDD}로 정의하였고, GND로부터의 스테이지는 S_{GND}로 정의하였다. 표 1에서의 정보로부터 출력 노드 Val(f)의 Logic Value Function을 쉽게 유도할 수 있으며, 표 2에서 잘 보여주고 있다. 스테이지는 VDD 또는 GND로부터 출력 노드까지의 경로로써 정의되므로, 트랜지스터는 MOS 스위치사이에서 Boolean AND 연산으로 나타낼 수 있는 스테이지 내에서 직렬로 연결되어 있다. 예로써, 표 2의 첫 번째 열에서 어떤 트랜지스터가 입력에 의해 트리거 되지 않으면, 스테이지의 출력 값은 'Z'가 된다. 다른 말로 하면, 경로에 존재하는 모든 트랜지스터가 트리거 되면, 스테이지의 출력 값은 '1'이나 '0'이 된다. 다음 부분은 SLBDD를 구현하는 Phase II을 설명한다.

표 2. Output Value Function
Table 2. Output Value Function.

Output Value Function	Condition	State
Val(S_{VDD})	{∃M : Val(M) = Z}	Z
	{∀M : Val(M) = 1}	1
	{∀M : Val(M) != Z} & {∃M : Val(M) = U}	U
	{∀M : Val(M) != Z} & {∀M : Val(M) != U} &{∃M : Val(M) = X}	X
	{∀M : Val(M) != Z & Val(M) != U & Val(M) != X } & {∃M : Val(M) = 0 } & {∃M : Val(M) = 1}	1
Val(S_{GND})	{∃M : Val(M) = Z}	Z
	{∀M : Val(M) = 1}	0
	{∀M : Val(M) != Z} & {∃M : Val(M) = U}	U
	{∀M : Val(M) != Z} & {∀M : Val(M) != U} & {∃M : Val(M) = X}	X
	{∀M : Val(M) != Z & Val(M) != U & Val(M) != X } & {∃M : Val(M) = 0 } & {∃M : Val(M) = 1}	0
Val(f)	{∃S : Val(S) = 1} & {∃S : Val(S) = 0}	X
	{∀S : Val(S) = Z}	Z
	{∃S : Val(S) = 1} & {∀S : Val(S) != 0}	1
	{∃S : Val(S) = 0} & {∀S : Val(S) != 1}	0

2) Phase II: SLBDD 구현

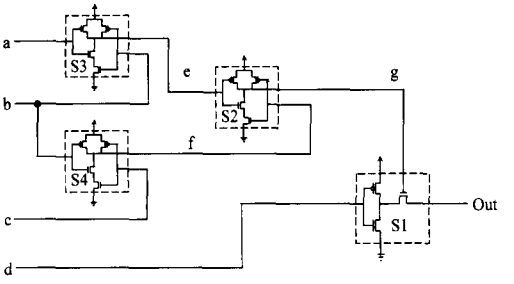
SLBDD는 Phase I에서 구성된 회로 네트워크의 depth-first traversal에 의해 구현된다. 주어진 변수 순서로부터, 각 SLBDD의 vertex는 회로 네트워크에서 논리 값 '0'과 '1'을 부여함으로써 구성된다. 각 CCR에서 스테이지들의 조건에 따라서 논리 값은 네트워크의 출력으로 전파하게 된다. 만약 현재 변수의 논리 값이 다음 노드로 전파하지 않으면, 다음 변수에 논리 값이 부여되고 SLBDD에서 후손 vertex가 된다. nonterminal vertex가 만들어지면, 지금까지 구현된 SLBDD의 모든 vertex들이 저장된 해시 테이블(hash Table)에 저장된 vertex들을 비교해서, 같은 vertex 조건이 발견되면 ROBDD에서 사용한 것과 같은 방법으로 제거한다. 그림 5는 SLBDD를 구현하는 알고리즘을 나타낸다. 제안된 알고리즘의 예제는 그림 6에서 보여주고 있다. 초기단계에서 그림 6(a)처럼 스위치-레벨 회로가 CCR로 분할되어 있어야 하고, 각 CCR에 대해서 그림 6(b)에서 보여주는 것처럼 제

안된 알고리즘의 Phase II에서의 방법에 의해 출력 값이 정해진다. 일단, SLBDD의 변수가 d, b, a, c 순서라고 가정하자. 루트 vertex는 d가 되고, 그림 6(a)의 회로 네트워크에 논리 값 '0'을 전파시킨다. 그림 b(a)에서 입력 노드 b로부터 논리 값 '0'이 출력 가까이 전파된다. 따라서, 노드 g에서의 논리 값은 '0'이 되고, 회로의 출력 값은 그림 6(b)에서 표로부터 'Z'가 된다. 이 경우에, 변수 a와 c로부터의 모든 경로는 착오 경로가 된다. 위 과정은 d=b='0'인 경우 그림 6(c)의 SLBDD를 만들게 된다. 주어진 변수 순서에 따라 '0'과 '1'의 논리 값이 적용되고, 결과 상태가 회로 네트워크의 전파를 통해 이루어진다. 제안된 알고리즘의 결과로써, 그림 6(d)와 같은 최종의 SLBDD가 완성된다.

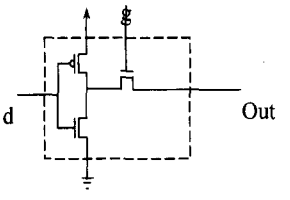
```
MakeSLBDD( )
{
  path_logic_value= function_verification(node and node_logic-value);
  if path_logic_value have a primary out state value {
     /*the primary input reach the primary output*/
     make SLBDD terminal vertex and return;
  }
  else  {   /*if primary input doesn't reach the primary*/
     Next Input Selection( ) /* Select a voluntary primary input */
     MakeSLBDD( )  /*Make children vertex (recursion function)*/
     ReduceBDD( )    /*Reduce a same BDD node*/
  }
}
function_verification(node and node_logic_value)
                          /*functions to verify circuit paths*/
/* node is a stage group input */
{
   calculate the state value of the stage-group according input node
        conditions
     if the stage-group is marked
       return false;
   if the stage-group have a output value {
       function_verification(node and node_logic_value) /*move to next
           linked stage-group*/
       mark stage_group inputs;
   }
   else if the stage-group have a output value and it's output is a
      primary output
        return output value;
   else if the stage-group can't have a output value according it's input
      state conditions
        return false;
}
```

그림 5. SLBDD 알고리즘
Fig. 5. SLBDD Algorithm.

이 상에서처럼 스위치-레벨에서 제안된 알고리즘을 사용해서 SLBDD가 구현되었고, 기존의 게이트-레벨과는 다르다는 것을 확인할 수 있었다. 하지만, 기존의 변수 순서는 최적의 BDD를 구현할 수 없다. 따라서, 다음 장에서는 기존의 게이트-레벨과는 다른 순서 알고리즘이 존재해야하는 필요성과 SLBDD에서의 휴리스틱 변수 순서 알고리즘을 제안하고, Phase II에서 구현된 그림 6의 SLBDD 크기와 비교하겠다.



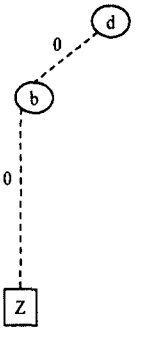
(a)



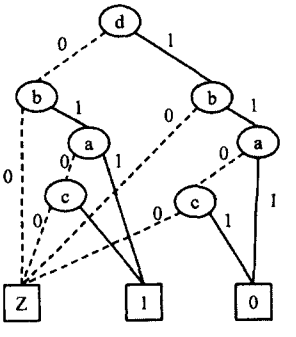
(b)

d	g	Out
Don't Care	0	Z
1	1	0
0	1	1
Z	Don't Care	Z
Don't Care	Z	Z

(c)



(d)



(e)

그림 6. 회로의 SLBDD 구현 과정의 예: (a) 예제 회로 네크워크 (b) 하나의 CCR과 출력 상태 (c) d=b=0일 때의 SLBDD (d) 전체 회로의 SLBDD
Fig. 6. The Example of SLBDD Implementation processing for circuit : (a) Example circtit network (b) CCR and Ouput state (c) SLBDD at d=b=0 (d) SLBDD in Complete circuit.

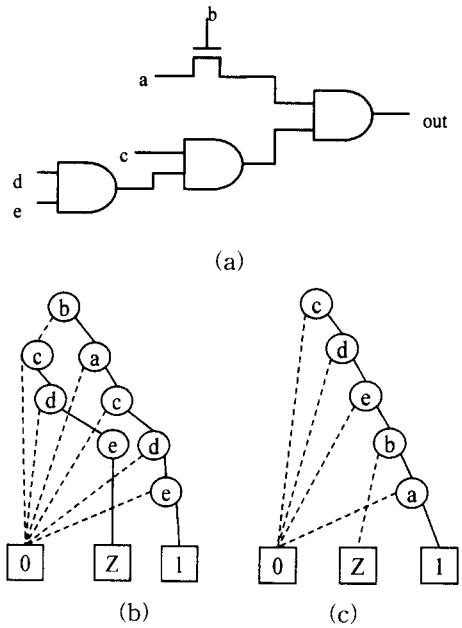


그림 7. 패스 트랜지스터를 가지는 경로와 가지지 않는 경로사이의 변수 순서:(a) 회로 네트워크 예제 (b) depth-first 변수 순서에 의한 SLBDD (c) 제안된 변수 순서에 의한 SLBDD

Fig. 7. The Variable order for the path that pass transistor and not have : (1) The example of Circuit network (b) According to Depth-first variable order, SLBDD (c) According to suggested variable order, SLBDD.

2. SLBDD에서의 변수 순서(Variable Ordering)

스위치-레벨 회로 네트워크에서 기능(function)은 디바이스간의 직·병렬 연결의 조합이다. 그러므로, 변수 순서를 결정하는 것은 회로 네트워크의 횡단(traversal)을 요구하며, 패스-트랜지스터의 존재가 더욱 문제를 복잡하게 한다. 일반적으로 출력에 가까운 노드들일수록 출력에 더 큰 제어력을 가지므로, 출력으로부터의 depth-first traversal가 변수 순서 문제를 결정하는 기본적인 방식이 된다. 설명을 위해서, 출력으로부터 입력까지의 회로 노드들이 레벨화되어 있고, 회로 네트워크에서 경로 P가 적어도 하나의 패스-트랜지스터를 포함하는 경로이며, 경로 Q는 패스-트랜지스터를 가지지 않은 경로라고 가정하자.

Observation 1. 비록 변수가 같은 레벨을 가지더라도, 경로 Q에 존재하는 변수가 경로 P에 존재하는 변수보다 출력에 더 큰 제어력을 가진다.

이 경우의 예를 그림 7에서 보여주고 있다. 그림 7(a)는 하나의 패스-트랜지스터를 가지는 회로 네트워크이다. depth-first traversal에 의해 변수 순서가 b-a-c-d-e 순이면, SLBDD에서 vertex들의 전체 수는 그림7(b)에서 나타나듯이 8개가 된다. Observation 1에 따라서 변수 순서가 c-d-e-b-a 순서가 되면, SLBDD에서의 vertex의 총 개수는 그림 7(c)와 같이 5개가 된다.

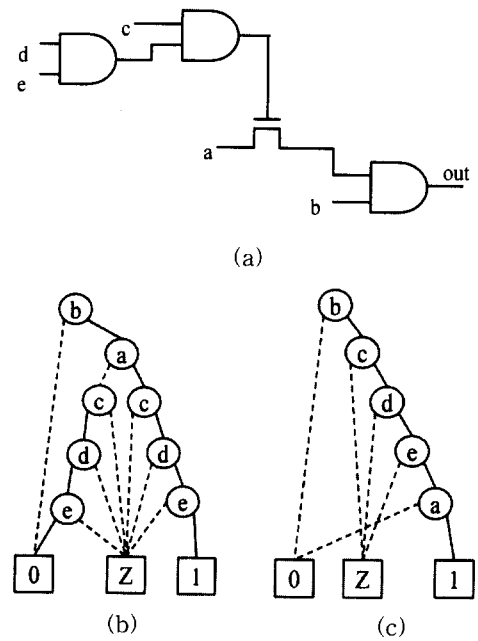


그림 8. 패스 트랜지스터를 포함하는 경로에서의 순서: (a) 회로 네트워크 예제 (b) depth-first 변수 순서에 의한 SLBDD (c)제안된 변수 순서에 의한 SLBDD

Fig. 8. The variable order for the path that pass transistor and not have : (1) The example of CIrcuit network (b) According to Depth-first variable order, SLBDD (c) According to suggested variable order, SLBDD.

Observation 2. 패스-트랜지스터의 경로에 존재하는 변수들 중에서 패스-트랜지스터의 게이트 입력에서 연결되는 경로에 존재하는 변수가 다른 것보다 출력에 더 큰 제어력을 가진다.

그림 8(a)회로와 같이 패스 트랜지스터를 포함하는 경로가 있을 때, Observation 1을 사용하여 다이어그램의 루트는 b가 된다. 나머지 변수 a, c, d와 e에 대해서 depth-first 순서의 SLBDD는 그림 8(b)에서

전체 vertex 수가 8개가 된다. 제안된 변수 순서는 그림 8(c)처럼 SLBDD의 vertex 수를 5개로 줄인다.

Observation 3. (a) 패스-트랜지스터가 병렬 연결되어 있을 때, 패스-트랜지스터의 대부분을 가지는 경로에 존재하는 변수들이 다른 것들보다 출력에 대해 큰 제어력을 가진다.
(b) 직렬 연결된 패스-트랜지스터의 변수 사이에서는 가장 높은 레벨의 게이트 입력 변수가 가장 큰 제어력을 가진다.

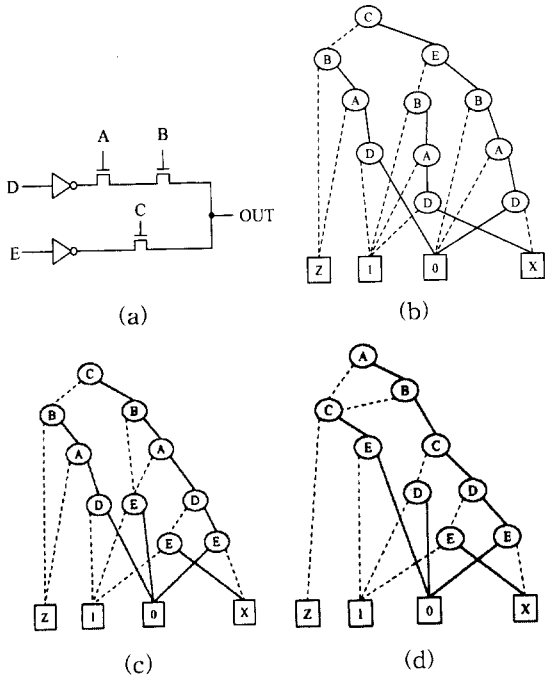


그림 9. 패스 트랜지스터 사이에서의 입력 변수 순서: (a) 회로 네크워크 예제 (b) depth-first 변수 순서에 의한 SLBDD (c) 출력에 가까운 패스-트랜지스터의 게이트 입력을 우선 순위로 했을 경우의 SLBDD (d) 제안된 변수 순서에 의한 SLBDD

Fig. 9. The input variable order among pass transistor : (1) The example of Circuit network (b) According to Depth-first variable order, SLBDD (f) In case of the gate input of pass-transistor to be near output has high priority, SLBDD (g) According to suggested variable order, SLBDD.

이 경우의 예는 그림 9(a)의 회로 네트워크를 통해 쉽게 설명된다. 입력 변수 A, B, C는 패스-트랜지스터의 게이트 입력이고, 변수 D와 E는 회로의 입력이다. depth-first 순서 방법을 사용하면, 그림 9(b)와 같이 11개의 vertex를 가진다. 반면, 그림 9(c)의 SLBDD는 10개의 vertex를 가진다. Observation 3(b)로부터 최적의 순서가 정해지며, 그림 9(d)의 SLBDD와 같이 9개의 vertex로 구성된다.

Observation 4. 도미노 논리에서의 변수 순서 : 논리의 타이밍을 제어하는 입력(클락 입력)변수는 다른 변수보다 출력에 더 큰 제어력을 가진다.

도미노 논리의 예제는 그림 10에서 보여준다. depth-first 순서 방식을 사용하면, 변수 순서는 x_1, x_2, x_3이 되고, 그림 10(b)와 같이 4개의 vertex를 가지게 된다. 반면, Observation 4에 의해 x_3, x_1, x_2의 순으로 구성된 10(c)의 SLBDD는 3개의 vertex만을 가진다.

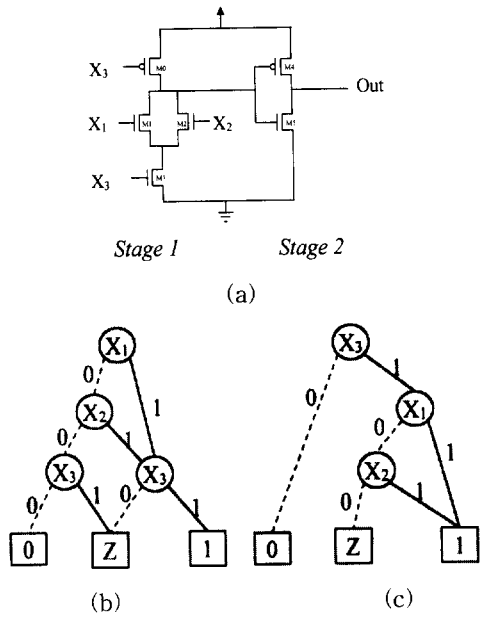


그림 10. 도미노 논리에서의 변수 순서: (a)도미노 논리가 포함된 회로 네트워크 (b) depth-first 변수 순서에 의한 SLBDD (c) 제안된 변수 순서에 의한 SLBDD

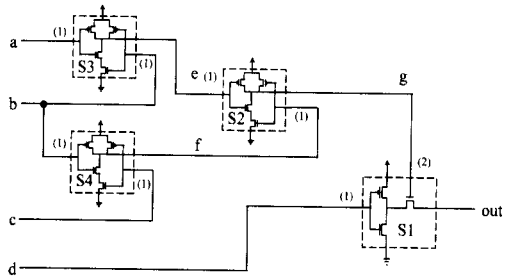
Fig. 10. The variable order for Domino Logic : (a) The Circuit network included Domino Logic (b) According to Depth-first variable order, SLBDD (c) According to suggested Variable order, SLBDD.

이상에서와 같이 스위치-레벨에서 변수 순서는 게이트-레벨 디자인에서와는 다르게 된다. 따라서, 본 논문에서는 위에서 보여준 Observation에 의존하는

새로운 휴리스틱 순서 알고리즘을 제안한다. Observation들은 계층적인 방법에서 결정 변수사이의 관계를 가리킨다. 다음은 전체 변수 순서 알고리즘의 간단하게 나타낸다.

1. 각각의 CCR 내의 모든 변수들에 대해서, Observations를 기초로 가중치(weight)를 부여한 다. 여기서는 가장 큰 priority를 가지는 변수가 SLBDD의 루트가 된다.
2. 출력으로부터 depth-first 순서로 회로 네트워크를 횡단한다.
   2.1 방문된 새로운 노드는 {node,(weight_list)} 쌍을 만든다.
   2.2 큐(queue)의 뒤에 쌍을 삽입한다.
   2.3 모든 노드가 적어도 한 번을 횡단할 때까지 단계 2를 반복한다.
3. 큐의 앞으로부터 성분(element)을 받아들인다.

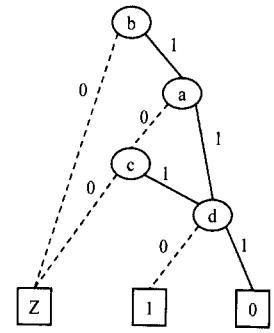
단계 1과정은 채널 연결 영역에서 국지적으로 실행되며, 4개의 Observations에 의존해서 priority가 할당된다. 일단, 각각의 CCR의 입력 변수사이에서의 관계가 고려되어 지면, 다른 CCR의 변수에 할당된 가중치들은 무의미하게 된다.



(a)

Path	{vertex,(weights)}
S1-g	{e,(2)}
S1-g-S2-e	{e,(2,1)},{g,(2)}
S1-g-S2-e-S3-a	{a,(2,1,1)},{g,(2)}
S1-g-S2-e-S3-b	{a,(2,1,1)},{b,(2,1,1)}
S1-g-S2-f	{a,(2,1,1)},{b,(2,1,1)},{f,(2,1)}
S1-g-S2-f-S3-b	{b,(2,1,1),(2,1,1,)},{a,(2,1,1)}
S1-g-S2-f-S3-c	{b,(2,1,1),(2,1,1,)},{a,(2,1,1)},{c,(2,1,1)}
S1-d	{b,(2,1,1),(2,1,1,)},{a,(2,1,1)},{c,(2,1,1)},{d,(1)}

(b)



(c)

그림 11. 휴리스틱 입력 변수 순서 알고리즘 : (a)노드에 가중치를 부과한 회로 네트워크 예제 (b) 경로와 순서 테이블 (c)제안된 변수 순서에 의한 SLBDD

Fig. 11. The Algorithm of heuristic input variable order : (a) The example of Circuit network that weigthed in node (b) The path and order Table (c) According to suggested Variable order, SLBDD.

그림 11은 그림 6(a)의 패스-트랜지스터를 포함하는 예제 회로에 제안된 변수 순서 알고리즘을 적용한 예이다. 예제 회로는 4개의 CCR로 구성되며, 출력 노드로부터의 depth-first의 순서로 S1, S2, S3, S4로 선언된다. 제안된 알고리즘의 첫 번째 단계로써 가중치가 Observations에 따라서 할당된다. 그 다음은, 경로 정보를 포함한 정보 테이블과 각각의 새롭게 방문된 노드들에 대한 {vertex,(weight_lists)}의 쌍을 형성하기 위해서 전체 회로를 횡단하게 된다. 여기에서 큐에 저장된 리스트(list)를 유지하고, 횡단된 각 노드들에서의 가중치에 따라서 소트(sort)한다. 그림 11(b)에서 보여진 것처럼, 출력으로부터 방문된 첫 번째 노드는 가중치 (2)를 가지는 g이고, 큐에 저장된 정보의 쌍은 {g,(2)}이다. 방문된 두 번째 노드는 축적된 가중치 (2,1)를 기지는 노드 e이고, {e,(2,1)}쌍을 만든다.

이 단계(알고리즘에서 단계 2.3)에서 큐의 성분이 소트되고, 가중치 (2,1)을 가지는 노드 e가 큐의 첫 번째에 오게된다. 이것은 노드 e가 노드 g보다 높은 레벨이며, priority가 크다는 것을 의미한다(Observation 3). 전체 과정이 회로 네트워크의 모든 노드가 횡단될 때까지 반복된다. 알고리즘의 결과로써 b-a-c-d의 변수 순서가 구해지게 된다. 그리고, 예제 회로의 SLBDD는 구해진 변수 순서에 따라 구현된다. 그림 11(c)는 구현된 SLBDD의 결과를 나타낸다. 그림 6(d)와 그림 11(c)의 두 개의 SLBDD를 비교해

보자. 제안된 휴리스틱 변수 순서 알고리즘을 사용하지 않고, 팬-아웃(fan-out)을 고려하는 depth-first 변수 순서 방식은 그림 6(d)처럼 d-b-a-c의 순서를 출력하게 된다. 하지만, 제안된 알고리즘에 의한 SLBDD는 결과로부터 최소의 SLBDD 크기를 가지게 함을 알 수 있다. 제안된 휴리스틱 알고리즘은 depth-first 순서에서 회로 네트워크의 횡단에 기초를 두고 있으므로, 타이밍 복잡도(time complexity)는 O(m)이 된다. 여기서, m은 네트워크에서 노드의 수이다.

Ⅳ. 실험결과

제안된 SLBDD 구현 알고리즘은 C언어를 사용해서 구현되었으며, SUN Ultra-Sparc I (메모리: 128MB)의 환경에서 적용되었다. 구현된 시스템의 입력은 SPICE 네트리스트이며, SLBDD가 출력으로 나오게 된다. 제안된 알고리즘의 효율성을 검증하기 위해서 ISCAS'85 벤치마크 회로를 포함한 여러 벤치마크 회로에 시스템을 적용하였다. 또한, 제안된 변수 순서 알고리즘을 적용해서 depth-first 순서의 SLBDD 크기와 비교하였다. 실험 결과는 표 3에 나타냈다. 표 3은 입력, 출력, 트랜지스터 개수와 다른 변수 순서 방식을 가지는 SLBDD 크기들로 구성된다. 제안된 순서 알고리즘에 의한 그래프 크기의 감소량은 퍼센트로 표시하였고, 마지막 열은 SLBDD을 구현하는 전체 계산 시간을 보여준다.

실험 결과로부터 depth-first 순서 방식에 비해서 제안된 순서 알고리즘이 40%이상의 그래프 크기를 감소시킴을 확인할 수 있다. 특히, 표에서 con4, con32, dec8과 enc8과 같이 패스-트랜지스터를 가지는 회로에서 제안된 알고리즘은 간단한 depth-first 순서보다 더욱 효율적이었다. ISCAS-85 벤치마크 회로는 게이트-레벨 디자인이므로, 각각의 CCR에서 입력 순서를 고려하는 것은 불가능하다. 제안된 알고리즘은 회로의 스위치-레벨 행동을 분석해야하므로, 게이트-레벨 접근 방식보다는 회로의 BDD 표현을 구현하는데 더 많은 시간이 걸린다. 하지만, 계산 시간의 대부분은 BDD의 구현 부분에서 소모된다는 것을 알 수 있었다.

Ⅴ. 결 론

제안한 스위치-레벨 이진 결정 다이어그램은 회로의 가능한(feasible) 상태와 치 않는(unwanted) 상태를 모두 고려한다. 본 논문에서는 패스-트랜지스터가 존재할 경우의 SLBDD를 구현하는 방법과 새로운 변수 순서 알고리즘을 보여주었다. 그리고, 실험 결과는 SLBDD가 주문형 스위치-레벨 회로의 설계 검증을 위해서 구현될 수 있는 가능성을 보여주었다.

SLBDD는 여러 가지 CAD(Computer-Aided Design) 방법에서 적용될 수 있다. 그 중의 하나가 스위치-레벨에서의 전력 소모 측정이다. 각 노드에서

표 3. 결과
Table 3. Result.

Circuit Name	#Inputs	#Outputs	#Transistor	#Node (Depth-First)
C17	5	2	24	10
C432	36	7	1,048	46,954
C499	41	32	2,868	89,338
C880	60	26	1,802	30,548
C1355	41	32	2,372	119,201
C1908	33	25	3,610	39,373
C5315	178	123	10,314	40,306
con4	4	1	17	7
con16	16	2	120	134
con32	32	2	620	23,456
dec8	12	2	98	83
enc8	9	4	124	53
add8	18	9	260	1,183
add16	33	17	540	94,814
mult4	8	8	388	394
mult8	16	16	1,672	77,517
CLS16	32	16	1,086	135,439
mcnc	9	2	93	45

Circuit Name	#Node (Our approach)	Reduction Rate	Time (Sec)
C17	8	20%	2
C432	29,864	47%	1,634
C499	45,865	49%	1,834
C880	6,431	79%	1,823
C1355	45,865	62%	2,472
C1908	11,569	69%	1,653
C5315	1,987	70%	1,743
con4	4	43%	2
con16	33	76%	16
con32	10,440	55%	1,480
dec8	54	45%	15
enc8	42	21%	11
add8	49	99%	170
add16	97	99%	535
mult4	330	17%	134
mult8	46,594	40%	405
CLS16	87,765	46%	1,834
mcnc	2	29%	7

스위칭 주파수(switching frequency)를 계산함으로써 동적 전력(dynamic Power) 소모뿐만 아니라, 단선-회로 전력(short-circuit power) 소모가 제안된 알고리즘을 약간 변환함으로써 이루어질 수 있다. 게다가, SLBDD를 스위치-레벨에서의 타이밍 분석과 결점 시뮬레이션을 수행하도록 확장할 수 있다. 향후 우리는 SLBDD를 위에서 제시한 여러 활용을 포함하는 새로운 시스템으로 확장할 것이다.

참 고 문 헌

[1] RANDAL E. BRYANT, "A Switch-Level Model and Simulator for MOS Digital System," IEEE TRANSACTIONS ON COMPUTERS, Vol. C-33, NO. 2, Feb. 1984.

[2] Kuen-Jong Lee, Rajiv Gupta and Melvin A. Breuer, "A New Method for Assigning Signal Flow Directions to MOS Transistors," International conference on Computer-Aided Design, Nov. 1990, pp492-495.

[3] U. Hubner, H. T. Vierhaus, "Efficient Partitioning and Analysis of Digital CMOS-Circuits," Proc. IEEE ICCAD-92, pp280-283, 1992.

[4] A.L.Glebov, D. Blaauw, L.G.Jones, "Transistor Reordering for Low Power CMOS Gates Using SP-BDD Representation," Internal Symp. on Low Power Design, p 161, 1997.

[5] W. Meyer, R. Camposano, "Active Timing Multi-Level Fault-Simulation with Switch-Level Accuracy," IEEE Transactions on CAD of Integrated Circuits and Systems, Oct. 1995.

[6] John K. Ousterhout, "A Switch-Level Timing Verifier for Digital MOS VLSI," IEEE Tran. on Computer-Aided Design, Vol. CAD-4, NO. 3, July 1985.

[7] Jesper Moller, Christian Ostergaard, "An Efficient Implementation of an ROBDD Package," Technology Report, Dep. of Information Technology Technical University of Denmark, 1996.

[8] Randal E. Bryant, Derek Beatty, Karl Brace, Kyeongsoon Cho, Thomas Sheffler, "COSMOS: A Compiled Simulator for MOS Circuits," 24th ACM/IEEE Design Automation Conference, pp9-16, 1987.

[9] K.J. Lee, C.A. Njinda, M.A. Breuer, "SWiTEST : A Switch Level Test Generation System for CMOS Combinational Circuits," 29th ACM/IEEE Design Automation Conference, p 26-29, 1992.

[10] RANDAL E. BRYANT "Boolean Analysis of MOS Circuits," IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN, Vol CAD-6, NO. 4, JULY 1987.

[11] D. T. Blaauw, D. G. Saab, P. Banerjee, J. A. Abraham, "Functional Abstraction of Logic Gates for Switch-Level Simulation," IEEE, p329-333, 1991.

[12] P. Buch, A. Narayan, A. R. Newton, and A. Sangiovanni-Vincentelli, "Logic Synthesis for Large Pass Transistor Circuits," in Proc. of the ACM/IEEE International Conference on Computer-Aided Design, November 1997. Also, Technical Report, UC Berkeley, UCB/ERL M97/26, April 1997.

[13] M. Pedram and X. Wu, "A new description of MOS circuits at switch-level with applications," IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences, Vol. E80-A, No. 10, October 1997.

[14] K. Yano, Y. Sasaki, K. Rikino, and K. Seki, "Top-down pass-transistor logic design," IEEE JSSC, vol. 31, No. 6, June. 1996.

[15] S. Gavrilov, A. Glebov, "Library-Less Synthesis for Static CMOS Combinational Logic Circuit," IEEE, pp 658-662, 1997.

[16] Akers, S. B, "Binary Decision Diagrams," IEEE Transaction on Computers, Vol. C-27, No. 6, pp 509-516, June. 1978.

[17] Randal E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," IEEE Transactions on Computers, Vol. C-35, No. 8, Aug. 1986.

[18] Masahiro Fujita, Hisanori Fujisawa, Yusuke Matsunaga, "Variable Ordering Algorithms for Ordered Binary Decision Diagram and Their Evaluation," IEEE Transactions on Computer-Aided Design of Integrated Circuits and System, Vol. 12, No. 1, January 1993.

[19] Richard Rudell, "Dynamic Variable Ordering for Ordered Binary Decision Diagrams," International Conference on Computer Aided Diagram, pp. 42-47, November 1993.

[20] Heh-Tyan Liaw, Chen-Sang Lin, "On the OBDD-Representation of General Boolean Functions," IEEE Transactions on Computers, VOL. 41, NO. 6, JUNE 1992.

저 자 소 개

金 慶 期(正會員)

1995년 2월 경남대학교 전자공학과 학사. 1997년 2월 경남대학교 전자공학과 석사. 1997년 3월 ~ 현재 서강대학교 전자계산학과 박사 과정. 주관심 분야는 CAD 및 하드웨어 설계

金 周 號(正會員)

1987년 12월 18일 Cadence Design Systems, Inc. San Jose, California, U.S.A 선임 연구원. 1997년 3월 서강대학교 컴퓨터 학과 조교수. 주관심분야는 CAD 및 하드웨어 시스템 설계

李 東 垠(正會員)

1998년 2월 서강대학교 전자계산학과 학사. 1999년 3월 ~ 현재 서강대학교 컴퓨터 학과 석사 과정. 주관심분야는 CAD 및 하드웨어 시스템 설계