논문 2012-49-12-25

Field Programmable Stateful Logic Array 패브릭 매핑 및 배치

(Fabric Mapping and Placement of Field Programmable Stateful Logic Array)

김 교 선*

(Kyosun Kim)

요 약

최근 무어의 법칙을 연장시킬 시스템 집적 기술로서 Field Programmable Stateful Logic Array (FPSLA)가 제안되었다. 본 논문은 FPSLA의 설계 자동화 절차를 확립하고 논리 합성, 동기화, 물리적 매핑, 자동 배치 등의 접근 방법을 최초로 제시한다. 특히, 동기화를 통해 배치를 1차원 문제로 축소한 후 비선형 최적화 기법을 개량한 개략 배치 모델 및 하향식 계층적 2분법을 이용한 배치 적법화 알고리즘을 제안하였다. 또한, 제안된 모델 및 알고리즘을 소프트웨어로 구현하여 ACM/SIGDA 벤치마크 예제에 적용함으로써 그 유효성을 입증하였다. 이 소프트웨어에는 Fanout 수만큼 출력 상태를 같은 단의 멤리스터성 스위치에 복사해야 하는 FPSLA의 특성을 고려하여 최적화 단계 별로 넷을 하이퍼에지로 통합했다가 다시 에지로 분리하는 기법이 제안되었으며 약 18.4%의 추가적 최적화를 이룩했다. FPSLA의 출력 상태 복사는 논리 단 일부에 셀 밀도가 집중되는 문제를 노출했으며 단위 논리 게이트의 Fanin을 제한하는 기법으로 18.5% 감소 효과를 얻었다. FPSLA의 실용성 확보를 위해서는 우선 논리 합성 시 Fanin의 수가 일부 단에 집중되지 않도록 제약하는 방안을 개발하여야 한다. 또한, FPSLA 패브릭 구조를 이식하기 위해 대칭성이 감소된 나노와이어 크로스바가 형성하는 복잡한 그래프 상에서 수행되어야 하는 자동 배선의 효율성 연구도 필요하다. 이러한 툴 개발은 설계 자동화 자체뿐만 아니라 FPSLA의 패브릭 구조 개선에 필요한 실험에 유용한 평가 도구로서도 큰 역할을 할 것이다.

Abstract

Recently, the Field Programmable Stateful Logic Array (FPSLA) was proposed as one of the most promising system integration technologies which will extend the life of the Moore's law. This work is the first proposal of the FPSLA design automation flow, and the approaches to logic synthesis, synchronization, physical mapping, and automatic placement of the FPSLA designs. The synchronization at each gate for pipelining determines the x-coordinates of cells, and reduces the placement to 1-dimensional problems. The objective function and its gradients for the non-linear optimization of the net length and placement density have been remodeled for the reduced global placement problem. Also, a recursive algorithm has been proposed to legalize the placement by relaxing the density overflow of bipartite bin groups in a top-down hierarchical fashion. The proposed model and algorithm are implemented, and validated by applying them to the ACM/SIGDA benchmark designs. The output state of a gate in an FPSLA needs to be duplicated so that each fanout gate can be connected to a dedicated copy. This property has been taken into account by merging the duplicated nets into a hyperedge, and then, splitting the hyperedge into edges as the optimization progresses. This yields additional 18.4% of the cell count reduction in the most dense logic stage. The practicality of the FPSLA can be further enhanced primarily by incorporating into the logic synthesis the constraint to avoid the concentrated fains of gates on some logic stages. In addition, an efficient algorithm needs to be devised for the routing problem which is based on a complicated graph. The graph models the nanowire crossbar which is trimmed to be embedded into the FPSLA fabric, and therefore, asymmetric. These CAD tools can be used to evaluate the fabric efficiency during the architecture enhancement as well as automate the design.

Keywords : Field Prorammable Stateful Logic Array, Fabric Mapping, Placement, Non-Linear Optimization

* 정회원, 인천대학교 전자공학과 (Department of Electronic Engineering, University of Incheon)
※ 본 논문은 인천대학교 2011년도 자체연구비 지원에 의하여 연구된 것임.
접수일자: 2012년6월7일, 수정완료일: 2012년11월26일

Ⅰ. 서 론

반도체 기술의 발전을 대변해 온 무어의 법칙이 리소그래피의 한계 및 전력 밀도 제한 등의 장벽에 좌초될 상황이 임박해 옴에 따라 기존의 시스템 집적 기술을 대체하는 획기적 접근 방법이 요구되고 있다. 이러한 요구에 부응하기 위해 기존에 사용해왔던 전하, 전압 대신 사용될 새로운 상태 변수가 제안된 것[1~4]을 비롯하여 메모리 및 재구성이 가능한 배선은 물론 논리 연산 기능까지 제공하는 새로운 소자 및 아키텍처들이 제안[5]-[7]되어 왔다. 메모리와 저항의 특성을 결합하는 멤리스터성 소자 (Memristive Device)가 전압 구동형 2단자 스위치 형태로 구현되었으며 저항 소자 배열 형태의 초고집적 메모리 제작의 가능성을 보여 주었다. 가장 높은 실현성을 보여준 아키텍처는 현장 재구성 가능 나노와이어 배선 (FPNI: Field Programmable Nano wire Interconnect)[9~10]으로 구현된 CMOS와 분자 소자의 혼성 형태[8]이다. 이것은 전통적인 CMOS 층에 멤리스터성 스위치로 구성된 층을 여러 장 적층한 혼성 회로이다. 이 스택은 층간 수직 연결 핀들의 밀도가 충분히 높아 면적 손실 없이도 광대역 저지연 통신이 가능한 특징이 있다. 이와 같은 FPNI는 FPGA의 재구성 비트들과 관련 소자들을 CMOS 층에서 끌어 올려 나노와이어 크로스바 상의 비휘발성 스위치로 대체함으로써 기술 견인차 역할을 할 것으로 기대된다. 그러나 FPGA의 가장 핵심 소자인 논리 게이트와 플립플롭은 FPNI에서도 CMOS 층에서 구현할 수밖에 없다.

최근 멤리스터성 스위치가 물리적 상태 변수로 저항을 사용하면서 논리 게이트와 래치로 동작하는 상태 누적 논리 (Stateful Logic) 연산이 시연되었다[5]. 이 멤리스터성 스위치는 고집적 저 소모 전력의 메모리, 재구성 가능 배선[9], 그리고 논리 연산[5] 및 래치[11]를 구현할 수 있기 때문에 멤리스터성 소자로 구성된 나노와이어 크로스바 층은 이제 매우 흥미로운 기술로 주목받고 있다. 상태 누적 논리는 다단 논리를 구현하였을 때 하나의 출력이 다음 단에서 하나의 게이트 입력에만 연결될 수 있으며 여러 게이트의 입력으로 인가되지 못하므로 다음 단에 연결될 게이트 입력 수만큼 출력의 상태를 복사할 필요가 있다. 반면에 상태 누적 논리 고유의 데이터 래칭 특성을 이용하면 추가적 레지스터 비용 없이 디지털 시스템을 완전 파이프라인으로 구현할 수 있는 장점도 있다. 이를 FPNI 구조에 매핑하여 현장에서 재구성이 가능한 상태 누적 논리 파이프라인 배열 아키텍처 (FPSLA: Field Programmable Stateful Logic Array)가 제안[12~13]되었다. 이 FPSLA는 현장에서 나노와이어 크로스바 상의 멤리스터성 스위치만을 재구성함으로써 목적 기능을 구현할 수 있다. 제안된 아키텍처의 단위 회로는 다수 입력의 NOR-OR 연산을 구현한다. 논리의 OIG (OR-Inverter Graph) 표현, 데이터 동기화 (Synchronization), 데이터 전송 (Forwarding), 파이프라인 루프 (Loop) 등의 설계 이슈가 토의되었다[12].

본 논문은 논리부터 레이아웃까지의 FPSLA 설계 절차를 확립하고 이의 자동화를 기술한다. 지금까지 FPSLA의 설계 자동화 관련 기존 연구는 없었다. 본문에서는 먼저 상태 누적 논리 기본 동작 및 FPSLA 구조를 포함한 배경 이론을 먼저 소개하고 논리 합성, 패브릭 구조 매핑, 자동 배치 등의 접근 방법을 제시한다. 또한, 실험에서는 ACM/SIGDA 벤치마킹 회로[14]를 사용하여 논리에서 레이아웃까지의 설계 절차 및 툴을 평가한다. 마지막으로 FPSLA의 설계 자동화를 위한 향후 연구 방향을 제시하면서 결론을 맺을 것이다.

Ⅱ. 본 론

1. 배경 이론

가. 상태 누적 논리 (Stateful Logic) 기본 동작

상태 누적 논리[5] 연산에서 실질적 함축 (Material Implication)을 설명하기 위해 그림 1(a)와 같은 회로를 살펴보자. 두 개의 멤리스터성 스위치 P와 Q는 공통의

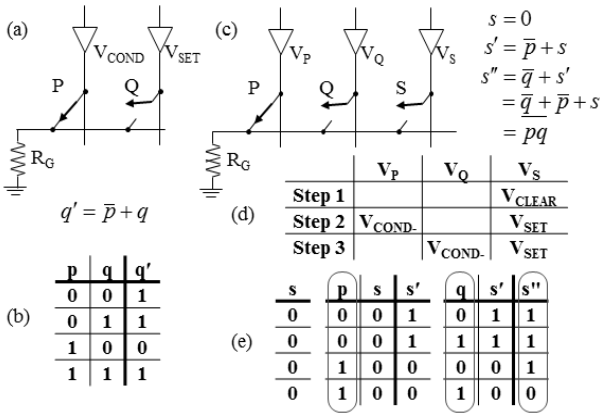


그림 1. 상태 누적 논리의 기본 연산
Fig. 1. Basic operations in stateful logic.

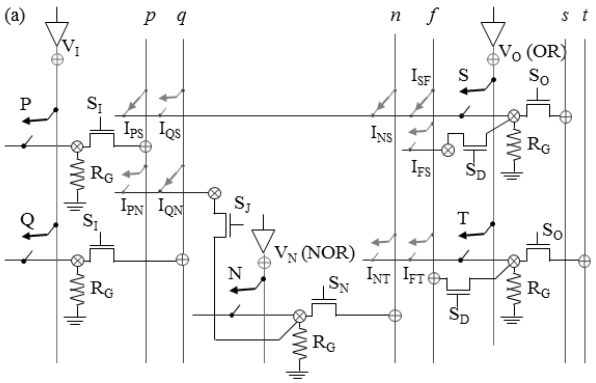
수평 나노와이어에 의해 묶여 부하 저항 R_G를 거친 후 접지되었다. P와 Q의 상태는 p와 q로 각각 나타낸다.

논리 0은 스위치가 열린 상태, 그리고 논리 1은 닫힌 상태에 해당한다. 수직 나노와이어는 3상 전압원에 의해 구동된다. 이 3상 전압원을 사용하여 스위치에 음의 전압 펄스 V_{SET}을 인가하면 논리 1 상태가 되고 양의 전압 펄스 V_{CLEAR}를 인가하면 논리 0 상태가 된다. P에는 V_{COND-} 펄스를 인가하고 동시에 Q에는 V_{SET} 펄스를 인가하면 P의 기존 상태가 논리 1일 때는 Q의 상태 천이가 금지되고 논리 0일 때는 허용된다. 이와 같이 V_{COND-} 펄스에 의해 유도되는 조건적 상태 천이는 그림 1(b)에 요약한 바와 같이 이 회로가 실질적 함축을 구현하는 게이트로 동작할 수 있게 한다. NAND 연산은 그림 1(c) 및 1(d)와 같이 세 개의 순차 스텝을 거쳐 수행된다[5]. 입력은 스위치 P와 Q에 저장된 상태 p와 q이며 출력은 스위치 S에 누적되는 상태 s''이다. 초기에 V_{CLEAR} 펄스가 스위치 S에 인가되어 $s=0$을 수행한다. 두 번째 스텝 $s'=\bar{p}+s$는 V_{COND-} 펄스를 V_P에 그리고 동시에 V_{SET} 펄스를 V_S에 인가함으로써 수행된다. 마지막 연산 $s''=\bar{q}+s'=\bar{q}+\bar{p}+s=\overline{pq}$는 V_{COND-}와 V_{SET} 펄스를 동시에 V_Q와 V_S에 각각 인가함으로써 수행된다. 세 개의 순차 스텝 수행 결과를 그림 1(e)의 진리표로 정리할 수 있다. 이 NAND는 만능 연산으로 알려져 있으며 어떤 부울 함수도 NAND 게이트들만의 연결로 구현될 수 있다.

다수의 함축 연산을 단일 스텝으로 수행[12]할 수 있다. 그림 1(c)의 회로에서 먼저 두 입력이 상태 q와 r로 전달되어 있다고 하자. 스텝 1에서 상태 p를 논리 0으로 초기화시키고 스텝 2에서 입력 q와 r을 가지고 상태 p에 함축 연산을 동시에 수행한다. 만약 q와 r 모두 논리 0일 경우 p는 논리 1로 천이되며 그 외의 경우에는 논리 0을 유지하게 된다. 이는 NOR 연산 $p=\overline{q+r}$를 구현한다. NOR 역시 만능 연산이기 때문에 어떤 부울 함수도 NOR 게이트들만의 연결로 구현될 수 있다.

나. 상태 누적 논리 파이프라인 아키텍처

상태 누적 논리 파이프라인 아키텍처[12~13]를 그림 2(a)에 보이고 있다. 각 논리 값들이 멤리스터성 스위치에 래치되기 때문에 이 아키텍처는 파이프라인으로 동작할 수 있다. 이 파이프라인 아키텍처는 FPNI 패브릭에 매핑될 수 있으며 단위 구조가 2차원 배열로 배치된



(b)

Mode	Step	V_I	S_I	S_J	V_N (NOR)	S_N	V_O (OR)	S_O	S_D
write	1		0	0	V_{CLEAR}	0		0	0
	2	V_{COND-}	1	1	V_{SET}	0		0	0
	3		0	0		0	V_{SET}	0	0
	4	V_{COND+}	1	0	V_{COND+}	1	V_{CLEAR}	0	1
read	1		0	0		0		0	0
	2		0	0		0	V_{COND-}	1	0
	3	V_{SET}	0	0		0		0	0
	4	V_{CLEAR}	0	0		0	V_{COND+}	1	0

그림 2. 상태 누적 논리 파이프라인 아키텍처
Fig. 2. Stateful logic pipeline architecture.

형태를 취한다. 각 단위 구조에는 하나의 3상 전압원 (V_O)을 보통 8개 이상의 OR 게이트가 공유하며 하나의 OR 게이트에는 멤리스터성 스위치, 부하 저항, CMOS 제어 스위치 2개를 포함한다. 또 하나의 전압원 (V_N)을 다수의 NOR 게이트들이 공유한다. NOR 게이트는 구조적으로 OR 게이트와 동일하지만 그 수가 OR 게이트의 반 정도이며 OR 게이트로 들어가는 입력을 선택적으로 반전시키는 데 사용된다.

이 아키텍처는 그림 2(a)에 보인 바와 같이 함수 $s=p+\bar{q}$를 구현하도록 현장에서 재구성할 수 있어 FPSLA (Field Programmable Stateful Logic Array)라 한다. 재구성 스위치 I_{QN}은 q를 NOR 게이트 N에 연결하기 위해, 그리고 I_{PS}와 I_{NS}는 p와 n을 OR 게이트 S의 입력으로 연결하기 위해 켜 놓았다. 기본 게이트들의 동작 스케줄은 그림 2(b)에 나타내었다. 같은 단위 구조에 속한 셀들은 동기되어 쓰기와 읽기 모드를 교대로 수행하며 인접한 단위 구조는 서로 반대 모드로 동작한다. 이 표에서 4, 5, 7, 9, 10번째 열은 CMOS 제어 스위치의 상태를 표시하며 0은 열림, 1은 닫힘을 의미한다. 스위치 S_I와 S_O는 같은 종류이지만 서로 다른 모드로 동작한다. V_N과 V_O에 인가된 전압 펄스는 6, 7번째 열에 각각 나타내었다. 쓰기 모드 스텝 1에서 n은 논리 0이 되

며 스텝 2에서는 NOR 게이트에서 $n' = \bar{q}$가 수행된다. 스텝 3에서 s는 논리 1이 되고 OR 연산 $s' = p + n = p + \bar{q}$가 스텝 4에서 수행된다. CMOS 제어 스위치 S_J는 스텝 4에서 게이트 N을 입력 p와 격리시키기 위해 사용된다. 재구성 스위치 I_{SF}가 켜져 있으면 출력 s는 즉석에서 t로 복사될 수 있다. S_D가 꺼지기 때문에 같은 스케줄 스텝 3에서 s와 t는 동시에 그러나 독립적으로 논리 1로 된다. 스텝 4에서 S_D가 켜진 후 두 AND 연산 $s' = s(p+q) = p+q$ 및 $t' = t(p+q) = p+q$가 동시에 수행된다.

2. FPSLA의 설계 자동화

FPSLA의 설계 자동화 절차는 그림 3과 같다. 논리 합성 (Logic Synthesis) 단계에서는 다양한 NOR-OR 논리를 구현하는 단위 논리 셀 (Logical Primitives)의 라이브러리를 구축한 후 기존의 CMOS 표준 셀 논리 합성 툴을 사용할 수 있다. 본 연구에서는 genlib 형식으로 단위 논리 셀 라이브러리를 개발하고 UC Berkeley의 ABC[15]를 사용하여 논리 합성을 실시하였다. 합성 결과인 논리 설계 (Logical Design)는 Verilog 넷 리스트 형식으로 출력되어 OpenAccess 표준 데이터베이스 (oaDB)에 저장되며 이후 과정은 모두 oaDB에서 데이터가 입출력된다. ABC를 위해 genlib 형식으로 개발된 단위 논리 셀 라이브러리는 Verilog로 다시 기술되어 oaDB 내에도 저장된다. 단위 패브릭 셀 (Fabric Primitives) 라이브러리는 NOR 및 OR 게이트들로 구성되며 물리적 매핑 (Physical Mapping) 과정에서 자동 생성된다.

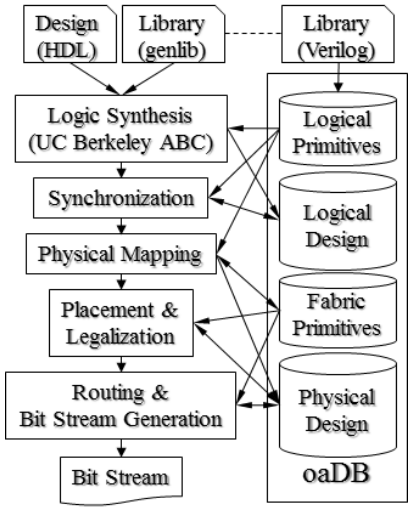


그림 3. FPSLA 자동화 설계 절차
Fig. 3. Automatic design flow for FPSLA.

그림 4(a)에 ABC의 표준 셀 합성 기능을 사용하여 구현한 5-입력 XOR 함수의 회로를 보이고 있다. 단위 논리 셀은 NOR-OR 형태의 논리를 구현하며 5-입력 기준으로 약 41종이 정의되었다. OR3, NR2 등과 같은 단순 논리 게이트뿐만 아니라 NR2OR2, ND1OR2, OR2ND2와 같은 복합 논리 게이트를 포함한다. 파이프라인 구성을 위한 동기화 (Synchronization)에서는 외부 입력에서 게이트의 입력까지 도달하는 경로 상의 게이트 수가 각 게이트의 모든 입력에 대해 동일하게 되도록 버퍼 (BUF)를 삽입한다. 각 게이트마다 상태 누적 논리 고유의 데이터 래칭 특성이 있기 때문에 구조가 가장 간단한 BUF를 삽입하면 각 게이트에서 입력 신호들을 동기화할 수 있다. 동기화된 5-입력 XOR 회로의 논리 설계를 그림 4(b)에, 그리고 이를 FPSLA 패브릭에 매핑한 물리 설계 (Physical Design)를 그림 4(c)에 나타내었다. 단순 논리 게이트 (NR2, OR3) 및 복합 논리 게이트 (NR2OR2, ND1OR2, OR2ND2)가 NOR 및 OR 게이트의 배열로 구성된 FPSLA 패브릭에 매핑된 것을 확인할 수 있다. 물리적 매핑 시에는 게이트의 출

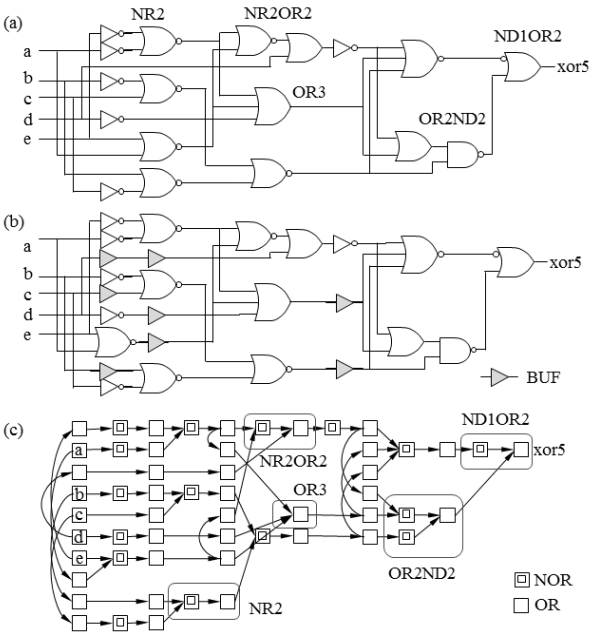


그림 4. 5-입력 XOR 함수를 (a) 논리합성, (b) 동기화, (c) 물리적 매핑한 회로
Fig. 4. Circuits of 5-input XOR function after (a) logic synthesis, (b) synchronization, and (c) physical mapping.

력 상태를 그 Fanout 수만큼 복사하는 작업도 필요하며 화살표가 붙은 호를 사용하여 출력 상태 복사를 나타내었다.

배치는 비선형 최적화 기법을 사용하는 개략 배치 및 배치 적법화 (Legalization)로 구성되며 다음 절에서 상술된다. 그림 4(c)에서 확인할 수 있듯이 게이트 셀 위치의 X 좌표는 동기화 과정에서 이미 결정되고 FPSLA 배치는 결국 Y 좌표만을 결정하는 1차원 문제로 축소된다. 배선은 나노와이어 크로스바 상의 멤리스터성 스위치 상태를 결정하여 단위 패브릭 셀들을 연결하는 단계로서 최단 거리 탐색 등의 알고리즘으로 구현될 수 있으나 본 연구 범위를 벗어난다.

3. 개략 배치 및 배치 적법화

배치 문제는 배선 길이, 배치 밀도 외에도 타이밍, 소모 전력, 전력 및 신호 충실도 등 다양한 제약 조건과 목적 변수들을 포함하고 있다. 그러나 무어의 법칙을 근간으로 한 회로 규모의 지수 함수적 폭발은 복잡한 목적 함수를 처리할 수 있으나 처리 속도가 느린 어닐링 시뮬레이션 기법의 적용을 더 이상 허용하지 않고 있으며 최근에는 ASIC 및 FPGA 기술 모두 주로 비선형 최적화를 근간으로 한 해석적 배치 (Analytic Placer) 방법[16][17]을 주로 채택하고 있다. 본 연구에서는 2차원 자동 배치를 위한 기존의 비선형 최적화 기법[16][17]을 FPSLA 구조에 맞도록 1차원 배치 기법으로 개량하였으며 배치 적법화 문제도 1차원 문제로 축소하여 계층적 2분법을 적용한 해법을 제안하였다.

일반적으로 디지털 논리 회로는 그림 5(a)와 같이 한

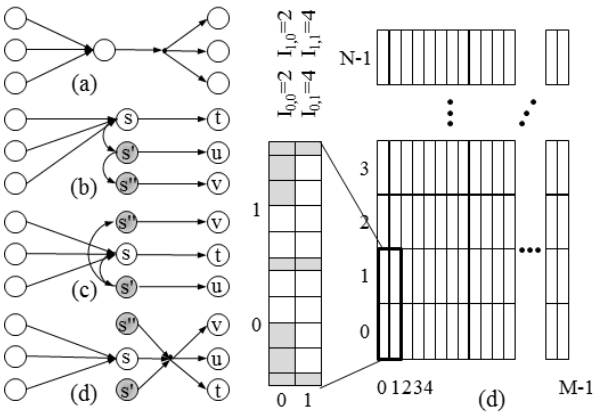


그림 5. 배선 길이 및 배치 밀도 모델
Fig. 5. Models of net length and placement density.

소자에 여러 개의 입력이 들어오고 그 출력은 여러 소자에 전달되는 하이퍼그래프로 표현할 수 있다. 그러나 상태 누적 논리는 하나의 출력이 직접 여러 입력에 연결될 수 없어 같은 열의 다른 멤리스터성 스위치에 상태를 복사해서 연결해야 하며 그림 5(b)와 같이 하이퍼그래프를 그래프 형태로 변형할 수 있다. 그러나 배치 과정에서 주변 게이트의 위치에 따라 그림 5(c)와 같이 복사된 사본들의 위치가 뒤바뀌게 되면 배선 길이가 과도하게 평가되기 때문에 처음에 정해진 상대적 위치가 변경되지 않도록 고수하는 힘이 불합리하게 작용한다. 이를 회피하기 위해서는 배치 이동시마다 그래프 자체가 변경되어야 한다. 즉, 그림 5(c)의 그래프와 같이 배치가 이동되면 (s', s'') 에지는 제거되고 (s, s'') 에지가 추가되어야 배선 길이가 정확히 계산될 수 있다. 그러나 이것은 알고리즘의 복잡도를 증가시킬 뿐 아니라 비선형 최적화에서 목적 함수의 연속성과 미분가능성 등을 해칠 수 있다. 그림 5(d)와 같이 같은 출력을 나타내는 모든 에지들을 묶어 하이퍼에지로 표현한 후 이 하이퍼에지의 Y축 범위를 배선 길이로 계산하면 이 과도한 평가를 완화하면서 계산량 증가를 억제할 수 있다. 최적화 후에 하이퍼에지를 일반 에지로 변경할 수 있으며 이때 가장 가까운 게이트 쌍을 찾으면 된다. 그림 5(d)에서는 수직 에지 (s, s''), (s, s')와 수평 에지 (s'', v), (s, u), (s' t)가 결정된다. 그러나 이 경우에 수평 에지의 Y축 거리가 덜 최적화될 수 있어 에지 길이를 기준으로 한 최적화가 추가적으로 수행되어야 한다.

배치 밀도를 평가하기 위해 그림 5(d)의 좌측에 확대하여 보인 바와 같이 기본 구조를 4등분 한 빈(Bin)을 M x N 2차원 배열로 나타낸 패브릭을 구성하였다. 수직 배선 자원을 확보하기 위해 열 번호가 짝수인 빈에 위치할 NOR 게이트 셀들의 수는 홀수 열에 위치할 OR 게이트 셀들에 비해 반수 정도로 제한한다. 중앙에 어둡게 표시한 음영 영역은 전압원이 배치되고 상 하단 음영 영역은 신호 귀환을 위한 수평 배선, 그리고 짝수 빈의 음영 영역은 수직 배선을 위해 사용된다.

비선형 최적화는 Conjugate Gradient (CG) Solver[18]가 응용되며 다음과 같은 목적 함수를 사용한다.

$$objective = \alpha_l\, length + \alpha_d\, density + \alpha_b\, barrier$$

$$length = \sum_{e\in E}\left(\max_{v_i\in e} y_i - \min_{v_i\in e} y_i\right)$$
$$= \sum_{e\in E}\left(\gamma \log \sum_{v_i\in e} e^{\frac{y_i}{\gamma}} + \gamma \log \sum_{v_i\in e} e^{-\frac{y_i}{\gamma}}\right)$$
$$= \sum_{e\in E}\left(y_{max} + \gamma \log \sum_{v_i\in e} e^{\frac{y_i - y_{max}}{\gamma}} + \gamma \log \sum_{v_i\in e} e^{-\frac{y_i}{\gamma}}\right)$$

$$density = \sum_{0\le i<N}\sum_{0\le j<M}\left(potential\left(x_{i,j}, y_{i,j}, r\right) - I_{i,j}\right)^2$$
$$= \sum_{0\le i<N}\sum_{0\le j<M}\left(\sum_{k\in insts(i)} \phi(d,r) - I_{i,j}\right)^2 \quad d = y_k - y_{i,j}$$

$$\phi(d,r) = \begin{cases} 1 - 2\left(\frac{d}{r}\right)^2 & if\ |d| < \frac{r}{2} \\ 2\left(\frac{d-r}{r}\right)^2 & if\ \frac{r}{2} \le |d| < r \\ 0 & if\ |d| \ge r \end{cases}$$

$$barrier = \sum_{k\in V}\left(penalty(y_k - b, \alpha) + penalty(t - y_k, \alpha)\right)$$

$$penalty(d,\alpha) = \begin{cases} 0, & if\ d \le 0, \quad or \\ \left(\frac{d}{\alpha}\right)^2, & if\ d > 0 \end{cases}$$

여기서 E는 회로를 구성하는 모든 넷, V는 모든 게이트 인스턴스의 집합이다. 변수는 인스턴스 $v_k \in V$의 Y 좌표 $y_k, 0 \le k < |V|$ 가 된다. X 좌표 x_k는 동기화 시에 이미 결정된다. 각 Bin의 중심 좌표는 $(x_{i,j}, y_{i,j})$로 표시한다. 배선 길이 (length)는 각 넷에 속한 셀들의 Y좌표 분포 구간 길이의 합이다. 최대 최소 함수는 지수-합-로그 (Log-Sum-Exponent) 함수로 스무딩 (Smoothing)할 수 있으며[19] 지수-합 연산 과정에서 넘침 (Overflow)을 방지하기 위해 $y_{\max}$를 미리 계산한 후 지수 값이 커지지 않도록 빼 주었다가 로그 연산 후에 다시 더해 준다. 배치 밀도 (density)는 종 모양 (Bell-Shape)으로 스무딩하여 각 빈에 누적시키고 각 빈의 허용 밀도 $I_{i,j}$와의 차를 제곱하여 모든 빈에 대해 합산한다. 배치 밀도에 의해 셀들이 패브릭 바깥으로 밀려 나가는 것을 방지하기 위해 장벽 (barrier) 함수를 추가하였다. 스무딩을 조절하기 위해 파라미터 γ, r, α가 사용되었으며 목적 함수 내 세 가지 요소의 영향을 조정하기 위해 $\alpha_l, \alpha_d, \alpha_b$가 사용되었다. 비선형 최적화에는 각 변수에 대한 목적 함수의 편미분이 필요하며 이는 다음과 같이 계산된다.

$$\frac{\partial}{\partial y_i} length = \sum_{e\in nets(v_i)}\left(\frac{e^{\frac{y_i - y_{max}}{\gamma}}}{\sum_{v_i\in e} e^{\frac{y_i - y_{max}}{\gamma}}} - \frac{e^{-\frac{y_i}{\gamma}}}{\sum_{v_i\in e} e^{-\frac{y_i}{\gamma}}}\right)$$

$$\frac{\partial}{\partial y_k} density$$
$$= \frac{\partial}{\partial y_k}\sum_{0\le i<N}\sum_{0\le j<M}\left(\sum_{k\in insts(i)} \phi\left(y_k - y_{i,j}, r\right) - I_{i,j}\right)^2$$
$$= 2\sum_{0\le i<N}\left(\phi(d,r) - I_{i,column(k)}\right)\frac{\partial}{\partial d}\phi(d,r)$$
$$\boldsymbol{d} = y_k - y_{i,column(k)}$$

$$\frac{\partial}{\partial d}\phi(d,r) = \begin{cases} -\frac{4}{r^2}d & if\ |d| < \frac{r}{2} \\ \frac{4}{r^2}(d-r) & if\ \frac{r}{2} \le |d| < r \\ 0 & if\ |d| \ge r \end{cases}$$

$$\frac{\partial}{\partial y_k} barrier = \frac{\partial}{\partial (y_k - b)} penalty(y_i - b, \alpha)$$
$$- \frac{\partial}{\partial (t - y_i)} penalty(t - y_i, \alpha)$$

$$\frac{\partial}{\partial d} penalty(d,\alpha) = \begin{cases} 0, & if\ d \ge 0 \quad or \\ \frac{2}{\alpha^2}d, & if\ d > 0 \end{cases}$$

배선 길이의 편미분은 해당 셀에 연결된 넷들 (nets (v_i))에 대해서만 계산되며 배치 밀도의 경우 해당 셀이 속한 빈의 열 (column(k))만 고려된다.

개략 배치가 완료되면 패브릭 상에 미리 정해진 위치로 각 셀의 배치를 확정하는 적법화가 실행된다. 배치 적법화 알고리즘은 그림 6에 나타내었다. 배치 적법화는 각 열 (Column[j])별로 실시하며 먼저 개략 배치에서 정해진 Y 좌표를 중심으로 셀들을 각 빈(Bin[i])에 넣는다. 각 빈은 정렬된 집합 (Ordered Set)으로 구현된다. 각 빈에 최대 허용 밀도보다 많은 셀들이 포함될 경우 이들을 상하의 주변 빈으로 분산시켜야 한다. 그러나 주변 빈들도 밀도가 높으면 그들의 과밀부터 해소해야 한다. 이를 체계적으로 처리하기 위해 하향식의 계층적 2분법을 사용하였다. 그림 6의 balance() 함수가 이를 재귀적 방법으로 수행하는데 빈들을 상측과 하측의 두 그룹으로 반씩 나누고 각각에 속한 셀들의 수가 최대 허용 밀도보다 높으면 과밀이 해소될 때까지 경계에 근접한 셀들부터 차례로 반대쪽 그룹에 있는 빈들 중 가장 근접한 빈에 이동시킨다. 상측 및 하측 그룹에 대해 다시 balance() 함수를 재귀적으로 호출하여 모든

```
legalize() {
    for(j = 0; j < M; j++){
        for(i = 0; i < N; i++)Bin[i].clear();
        foreach v ∈ Column[j] {
            i = y[v] / Height_Bin; Bin[i].insert(v);
        }
        balance(0,N);
        for(i = 0; i < N; i++){
            k = 0;
            foreach v ∈ Bin[i] {
                y[v] = i · Height_Bin + k · Height_Cell; k++;
            }
        }
    }
}
balance(b,t) {
    m = (b+t)/2; if(b == m) return;
    bc = Σ_{i=b}^{m-1} |Bin[i]| − (m − b) · I_{i,j} ;
    tc = Σ_{i=m}^{t-1} |Bin[i]| − (t − m) · I_{i,j} ;
    for(j = m − 1; j ≥ b && bc > 0; j−−) {
        while(bc > 0 && |Bin[j]| > 0) {
            v = Bin[j].removeLargest(); bc−−;
            Bin[m].insert(v);
        }
    }
    for(j = m; j < t && tc > 0; j++) {
        while(tc > 0 &&|Bin[j]| > 0){
            v = Bin[j].removeSmallest(); tc−−;
            Bin[m − 1].insert(v);
        }
    }
    balance(b,m); balance(m,t);
}
```

그림 6. 배치 적법화 알고리즘
Fig. 6. Placement legalization algorithm.

빈이 최대 허용 밀도보다 많은 셀들을 포함하지 않도록 한다. 빈의 과밀을 해소한 후 각 빈 내에서 셀들의 상대적 위치에 따라 패브릭 상의 적법 위치에 셀 배치를 확정한다. 이 알고리즘은 하향식 계층적 2분법을 사용하기 때문에 과밀한 빈에서 셀의 이동 방향을 결정할 때 전역 시각(Global View)을 유지할 수 있으며 저밀도 영역에서 불필요한 압축이 발생하지 않는다.

하이퍼에지인 넷을 에지로 분해하여 최적화하려면 배선 길이를 다음과 같은 수식으로 대체할 수 있다.

$$edgeLength = \sum_{e_{i,j} \in E} (y_i - y_j)^2$$

$$\frac{\partial}{\partial y_k} edgeLength = \frac{\partial}{\partial y_k} \sum_{e_{i,j} \in E} (y_i - y_j)^2$$

$$= 2 \left\{ \sum_{e_{i,k} \in in(v_k)} (y_k - y_i) + \sum_{e_{k,j} \in out(v_k)} (y_k - y_j) \right\}$$

이 거리 제곱 수식은 수평 에지의 수직 거리를 감소시킬 뿐만 아니라 에지 길이를 균등하게 유지하려는 성질이 있다. 에지 길이의 편미분은 해당 셀에 출입하는 에지들만 고려하면 된다.

Ⅲ. 실 험

제안된 FPSLA의 설계 자동화 절차 및 배치 알고리즘의 효용성을 입증하기 위해 동기화, 물리적 매핑, 그리고 개략 배치 및 배치 적법화 모듈을 개발하였다. 먼저, ACM/SIGDA 표준 벤치마크 예제를 입력으로 FPSLA 논리 회로를 합성하고 패브릭에 매핑 하였으며 그 결과를 표 1에 정리하였다. 총 23개의 회로에 대해 논리 합성 결과 (Logic)는 입력 수(#i), 출력 수(#o), 논리 단수(#l), 게이트 수(#c), 넷 수(#n)를, 그리고 패브릭 매핑 결과 (Fabric)는 NOR 셀 수(#nr), OR 셀 수(#or), 넷 수(#n), 에지 수(#e)를 정리하였다. OR 셀 수의 증가

표 1. 논리합성 및 패브릭 매핑 결과
Table 1. Results of logic synthesis and fabric mapping.

Circuit	Logic					Fabric			
	#i	#o	#l	#c	#n	#nr	#or	#n	#e
i5xp1	7	10	8	101	213	95	307	287	517
i9sym	9	1	12	194	448	193	620	567	1059
alu4	14	8	20	1025	2234	903	3362	3062	5468
apex1	45	45	16	1854	4107	1062	5939	5290	9794
apex2	39	3	19	278	581	248	1061	1042	1577
apex3	54	50	16	1551	3510	1410	4941	4396	8306
apex5	117	88	13	822	1723	718	2704	2553	4297
bw	5	28	6	140	320	135	415	347	753
clip	9	5	9	107	228	97	327	307	541
con1	7	2	4	17	34	21	48	57	81
duke2	22	29	18	460	1009	286	1687	1517	2629
e64	65	65	6	380	969	130	1320	861	2039
misex1	8	7	6	55	116	52	164	156	276
misex2	25	18	6	82	185	61	250	215	407
misex3c	14	14	20	505	1159	440	1832	1618	2926
misex3	14	14	18	1042	2331	867	3642	3220	5798
rd53	5	3	6	43	94	44	124	119	217
rd73	7	3	10	105	237	101	333	306	562
rd84	8	4	9	158	365	145	504	446	852
sao2	10	4	10	120	257	108	380	357	619
seq	41	35	22	1727	3214	1487	6075	5575	9549
vg2	25	8	8	131	286	99	395	356	632
xor5	5	1	6	16	30	16	39	45	65

는 동기화를 위한 버퍼 삽입 및 출력 Fanout 수만큼의 상태 복사가 주요 원인이다.

벤치마크 예제에 대해 α_d를 1.0, 10.0, 200.0으로 증가시키면서 개략 배치를 3회 반복 수행하였다. 매회 종료 시마다 배치 적법화를 추가 수행하였으며 세 번째 개략 배치 수행 시에는 배선 길이를 에지 길이로 계산하여 최적화하였다. 배치 후 평균 배선 길이를 각 단계별로

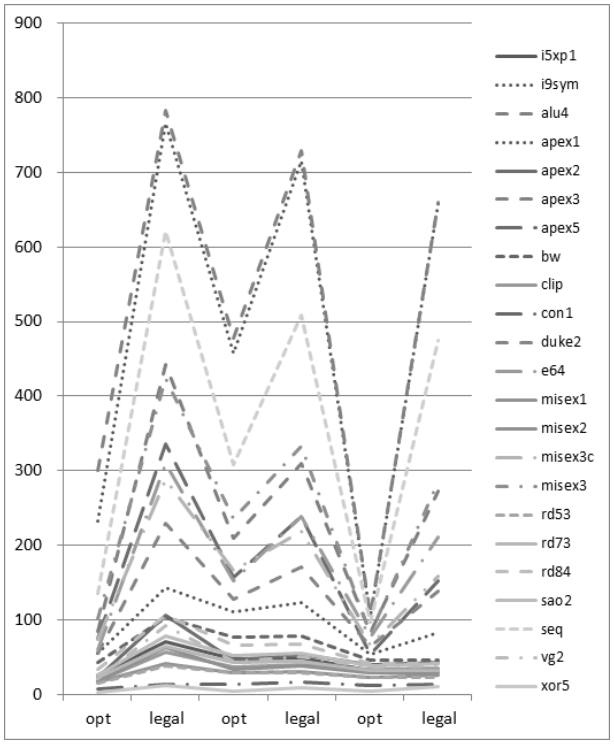


그림 7. 벤치마크 설계 예제의 배치 후 평균 배선 길이
Fig. 7. Results of average net length estimation after placement for benchmark examples.

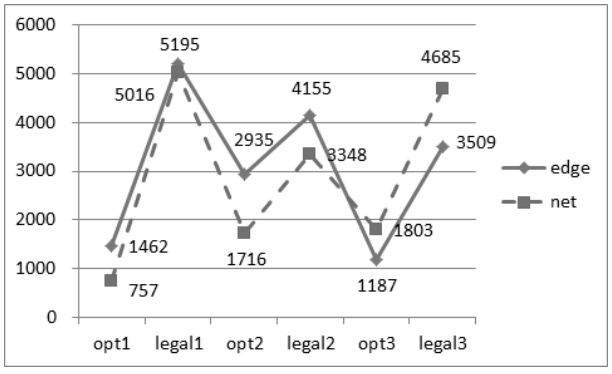


그림 8. 벤치마크 설계 예제의 배치 후 평균 배선 길이 합
Fig. 8. Sum of average net lengths of benchmark examples after placement.

그림 7에 정리하였다. 배선 길이는 모두 에지 길이로 환산하였다. 개략 배선 후 배치 적법화를 수행하면 배선 길이가 다시 늘어나지만 전반적으로는 회가 거듭될수록 감소함을 알 수 있다.

그림 8은 하이퍼에지 (net)와 일반 에지 (edge)로 각각 계산한 배선 길이를 비교하고 있다. 각 벤치마크 예제의 평균 배선 길이를 각 단계별로 합산한 것이다. 세 번째 개략 배치에서 하이퍼에지 대신 일반 에지 길이를 최적화함으로써 두 값이 역전되었으며 약 18.4% (=4155/3509-1)의 추가적인 최적화 효과를 얻을 수 있음을 확인할 수 있다.

배치 결과 중에서 회로 규모에 따라 선택한 예제 6개를 그림 9에 나타내었다. 각 레이아웃 위에 가장 긴 열의 셀 수를 표시하였는데 회로 규모가 커질수록 특정 열, 특히, 3번열이 길어지는 현상을 관찰할 수 있다. 이 현상과 단위 논리 셀의 Fanin과의 관계를 파악하기 위

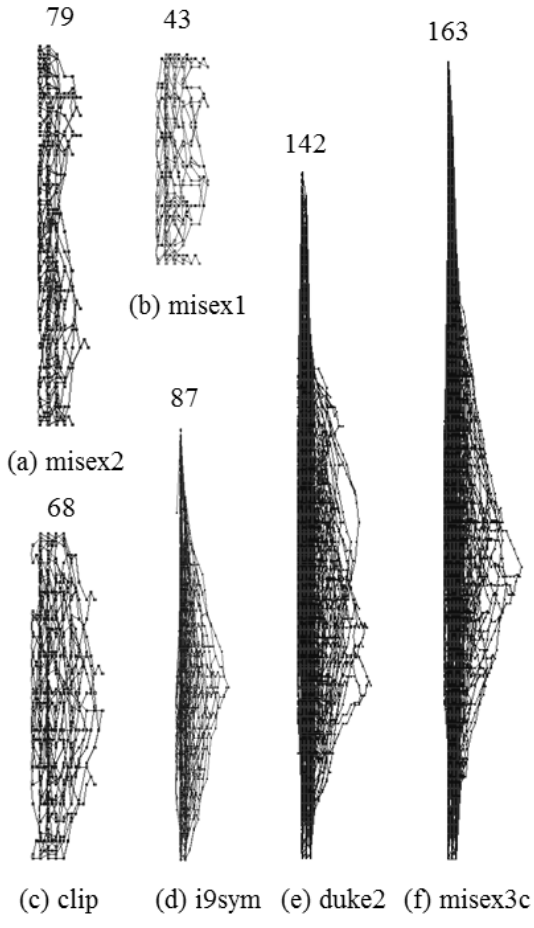


그림 9. 6개 설계 예제의 배치 결과
Fig. 9. Results of placement for six examples.

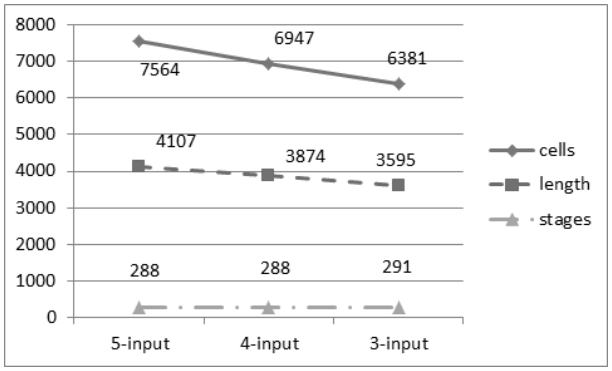


그림 10. 단위 논리 게이트의 입력 수 제한에 따른 최장 열 셀 수, 평균 배선, 논리 단수의 변화
Fig. 10. Variation of cell count in the longest column, net length, and stage count due to the constraints on the fanin of logic gates.

해 먼저 같은 벤치마크 예제에 대해 단위 논리 셀의 입력 수를 5개, 4개, 3개 이내로 각각 제한하여 자동 설계 절차를 수행하였으며 최장 열 셀 수의 합 (cells), 평균 배선 길이의 합 (length), 그리고 논리 단수의 합 (stages)을 그림 10에 비교하였다. 입력 수 제한이 5개에서 3개로 감소함에 따라 논리 단수는 1% (총 3단)가 증가한 반면 최장 열의 셀 수는 18.5%, 배선 길이는 14.2%가 감소하였다. Fanout의 제한 역시 비슷한 효과를 얻을 수 있을 것이라 판단되나 궁극적으로는 논리 합성 시 Fanin의 수가 논리 단 일부에 집중되지 않도록 제약하는 방안이 필요하겠다.

Ⅳ. 결론 및 향후 계획

무어의 법칙을 연장시킬 시스템 집적 기술로 최근 제안된 FPSLA의 설계 자동화 절차를 확립하고 논리 합성, 동기화, 물리적 매핑, 자동 배치 등의 접근 방법을 최초로 제시하였다. 특히, 비선형 최적화 기법을 개량한 개략 배치 모델 및 배치 적법화 알고리즘을 제안하고 소프트웨어로 구현하여 ACM/SIGDA 벤치마크 예제에 적용함으로써 그 유효성을 입증하였다. Fanout 수만큼 출력 상태를 같은 단의 멤리스터성 스위치에 복사해야 하는 FPSLA의 특성을 고려하여 최적화 단계 별로 넷을 하이퍼에지로 통합했다가 다시 에지로 분리하는 기법을 제안하여 약 18.4%의 추가적 최적화를 이룩했다. FPSLA의 출력 상태 복사는 논리 단 일부에 셀 밀도가 집중되는 문제를 노출했으며 단위 논리 게이트의 Fanin을 제한하는 기법으로 18.5% 감소 효과를 얻었다.

FPSLA의 실용성 확보를 위해서는 우선 논리 합성 시 Fanin의 수가 일부 단에 집중되지 않도록 제약하는 방안을 개발하여야 한다. 또한, FPSLA 패브릭 구조를 이식하기 위해 변형되어 대칭성이 감소된 나노와이어 크로스바가 형성하는 복잡한 그래프 상에서 수행되어야 하는 자동 배선의 효율성 연구도 필요하다. 이러한 툴 개발은 FPSLA의 패브릭 구조 개선에 필요한 실험에 유용한 평가 도구로서도 큰 역할을 할 것이다.

참 고 문 헌

[1] L.O. Chua, “Memristor - The Missing Circuit Element,” IEEE Trans. on Circuit Theory, Vol. CT-18, No. 5, pp. 507-519, September, 1971.
[2] M. Di Ventra, Y.V. Pershin, and L.O. Chua, “Circuit Elements with Memory: Memristors, Memcapacitors, and Meminductors,” Proc. of IEEE, Vol. 97, No. 10, pp. 1717-1724, October, 2009.
[3] L.O. Chua, and S.M. Kang, “Memristive Devices and Systems,” Proc. of IEEE, Vol. 64, No. 2, pp. 209-223, February, 1976.
[4] D.B. Strukov, G.S. Snider, D.R. Stewart, and R. S. Williams, “The missing memristor found,” Nature Vol. 453, pp. 80-83, 2008.
[5] J. Borghetti, G.S. Snider, P.J. Kuekes, J.J. Yang, D.R. Stewart, and R.S. Williams, “‘Memristive’ Switches Enable ‘Stateful’ Logic Operations via Material Implication,” Nature, Vol. 464, pp. 873-875, April 8, 2010.
[6] Y.V. Pershin, and M. Di Ventra, “Neuromorphic, Digital and Quantum Computation with Memory Circuit Elements,” arXiv:1009.6025v1, September, 2010.
[7] J. Borghetti, Z. Li, J. Straznicky, X. Li, D.A. Ohlberg, W. Wu, D.R. Stewart, and R.S. Williams, “A Hybrid Nanomemristor/Transistor Logic Circuit Capable of Self-Programming,” PNAS, Vol. 106, No. 6, pp. 1699-1703, February 10, 2009.
[8] D.B. Strukov, and K.K. Likharev, “CMOL FPGA: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices,” Nanotechnology 16 pp. 888-900, April, 2005.
[9] G.S. Snider and R.S. Williams, “Nano/CMOS Architectures Using a Field-Programmable Nanowire Interconnect,” Nanotechnology 18, 035204, 2007.

[10] D.B. Strukov, and R.S. Williams, “Four-Dimensional Address Topology for Circuits with Stacked Multilayer Crossbar Arrays,” PNAS, Vol. 106, No. 48, pp. 20155-20158, December 1, 2009.
[11] P.J. Kuekes, D.R. Stewart, and R.S. Williams, “The crossbar latch: Logic value storage, restoration, and inversion in crossbar circuits,” Journal of Applied Physics, 97, 034301, 2005.
[12] K. Kim, S. Shin, S-M Kang, "Field Programmable Stateful Logic Array", IEEE Transactions on Computer-Aided Design of Intergrated Circuits and Systems, Vol. 30, No. 12, pp.1800-1813, December, 2011.
[13] S. Shin, K. Kim, S-M Kang, "Reconfigurable Stateful NOR Gate for Large-Scale Logic-Array Integration", IEEE Transactions on Circuits and Systems-Ⅱ: Express Briefs, Vol. 58, No.7, pp.442-446, July, 2011.
[14] “ACM/SIGDA Benchmarks: Logic Synthesis Workshop - LGSynth89,” http:://www.cbl.ncsu.edu:16080/benchmarks/LGSynth89, March, 2006.
[15] “ABC: A System for Sequential Synthesis and Verification,” http:://www.eecs.berkeley.edu/~alanmi/abc/abc.html, October, 2007.
[16] W.C. Naylor, R. Donelly, and L. Sha, “Non-Linear Optimization System and Method for Wire Length and delay Optimization for an Automatic Electric Circuit Placer,” US Patent 6301693, October 2001.
[17] J. Cong and G. Luo, “Highly Efficient Gradient Computation for Density-Constrained Analytical Placement Methods,” Proc. of the International Symposium on Physical Design, pp. 39-45, April, 2008.
[18] D.J.C. MacKay, “MacOpt-a Nippy Wee Optimizer,” http://www.inference.phy.cam.ac.uk/mackay/c/macopt.html, June, 2004.
[19] X. Song, “Smoothing Method for Minimax Problems,” Computational Optimization and Applications, Kluwer Academic Publishers, 20, pp.267 - 279, 2001.

저 자 소 개

김 교 선(정회원)
1986년 연세대학교 전자공학과 학사 졸업.
1988년 연세대학교 전자공학과 석사 졸업.
1998년 Ph.D. Department of Electrical & Computer Engineering, University of Massachusetts, Amherst, U.S.A.
1988년～2003년 삼성전자 CAE Center 주임, 선임, 책임, 수석연구원.
2003년～현재 인천대학교 전자공학과 교수
<주관심분야 : 상위수준합성, Reconfigurable Computation, Fault-Tolerance, Embedded Systems, Low-Power Design, Nanoelectronic Architectures>