# Guardian LVS에서 논리 게이트 인식

## 소개

본 문서에서 트랜지스터 레벨 회로에서 일반 CMOS 게이트의 인식에 대해 소개합니다. 규칙 기반 기술을 사용하는 구조 인식 알고리즘은 논리 게이트 식별에 가장 효율적입니다 [1, 2, 3, 4]. 이러한 알고리즘은 매우 빠르며, 인버터, NAND, NOR, AOI, OAI 게이트와 같은 정적 논리 게이트를 쉽게 찾을 수 있습니다. 인식 후, 수정된 넷리스트는 로직 게이트와 나머지 트랜지스터의 관점에서 비교할 수 있습니다. 넷리스트의 연결성을 나타내는 애노테이션 그래프는 트랜지스터 수준의 그래프보다 크기가 훨씬 작고 구조를 더 잘 구별할 수 있습니다. 따라서, 그래프 동형 문제[5]의 예처럼 취급되는 회로 비교 문제를 더 효율적으로 해결합니다.

## 게이트 인식

Guardian LVS는 트랜지스터 수준에서 CMOS 회로의 논리 게이트를 인식합니다. 게이트를 형성하는 트랜지스터 그룹은 논리적 구성으로 표현됩니다. 연결된 트랜지스터 그룹이 논리 게이트를 형성하지 않는 경우, Guardian LVS는 동일한 유형의 병렬 연결 트랜지스터로부터 논리적 구성을 생성할 수 있습니다. LVS 검증은 이러한 변환 후 논리적 구성 수준에서 수행됩니다.

로직 게이트를 인식하려면, Guardian LVS 사용자 인터페이스에서 일부 설정을 조정해야 합니다:

- MOS 트랜지스터에 대해 "**Model Settings**"에서 "**Logic gates**" 옵션을 선택합니다.
- 전원 및 접지 넷을 "**Net Settings**"에서 지정합니다. 접미사 "**:P**", "**:G**" 를 사용하여 SPICE 넷리스트에 전원 및 접지 넷을 설정합니다. "**Remove Net Name Suffix starting from colon (:)**" 옵션을 선택하면, 접미사는 넷 이름에서 콜론부터 삭제되어 해당 넷이 전원과 접지 넷으로 인식됩니다.

**참고**: MOS 트랜지스터가 인버터를 인식하기 위해 "**Model Settings**"에서 "**Parallel reduction**" 및 "**Series reduction**" 옵션을 선택할 필요는 없습니다. 그러나, 더 복잡한 논리 게이트를 인식하려면 옵션을 설정합니다.

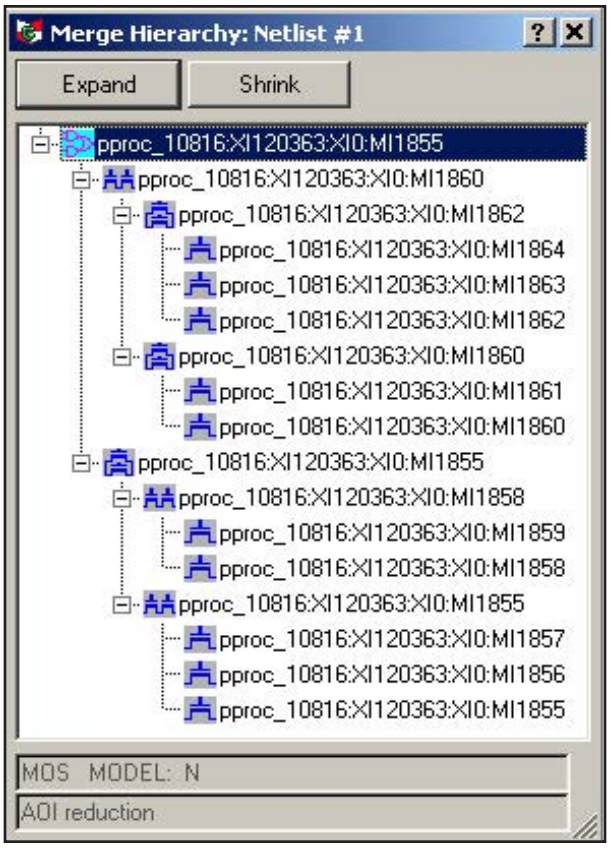

그림 1. AOI 게이트의 구조

논리 게이트는 "정확하게 구성된" MOS 트랜지스터에 대해서만 형성됩니다:

- 논리 게이트의 모든 트랜지스터는 동일한 갯수의 핀이 있어야 합니다.
- 논리 게이트의 P형 및 N형 컴포넌트는 감축 후 동일한 수의 엘리먼트가 있어야 합니다.
- P형 및 N형 컴포넌트는 트랜지스터의 게이트 단자에 연결된 상이한 넷이 동일하게 있어야 합니다;
- P형 (N형) 트랜지스터의 벌크 단자는 하나의 전원 (접지) 넷에 연결해야 합니다.

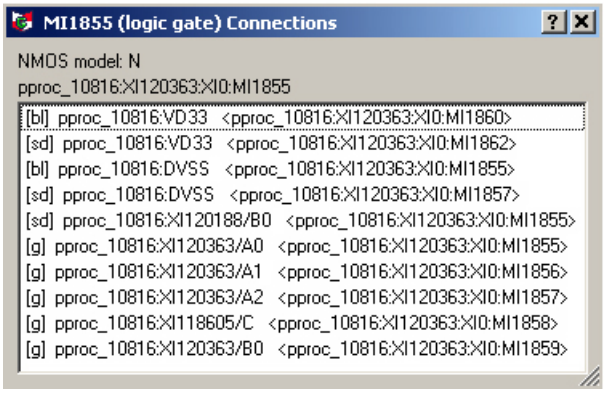


그림 2. AOI 게이트의 연결성

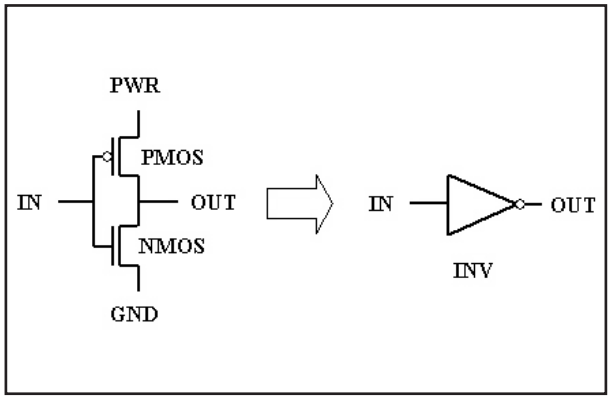


그림 3. INV

"**Model Settings**"에서 "**Match model**" 옵션을 사용하여, P형 또는 N형 컴포넌트에서 트랜지스터 모델의 혼합 여부를 선택할 수 있습니다.

Guardian LVS는 논리 게이트에서 핀 그룹의 논리적 등가성을 감지하므로, 트랜지스터 수준에서 등가의 핀과 소자를 교환할 수 있습니다. 예를 들어, 사용자는 NAND 또는 NOR 게이트의 입력 핀을 교환할 수 있습니다.

게이트 수준에서 불일치 및 일치 노드를 보고하며, 검사 툴을 사용하여 논리 게이트의 구조 또는 연결성을 조사할 수 있습니다. 예를 들어 **Merge Hierarchy** 툴 (그림 1)에서 AOI 게이트 생성 방법을 나타냅니다.

**Connectivity Traversing** 툴은 논리 게이트 연결성 정보, 예를 들어 게이트의 단자에 연결된 모든 넷의 목록 등을 포함합니다 (그림 2).

## 인식된 논리 게이트의 유형

Guardian LVS는 다음의 논리 게이트 유형을 인식합니다:

• INV — CMOS 인버터 (그림 3)

정확하게 구성된 (이전 섹션 참조) P형 및 N형 트랜지스터 쌍 (그림 3)은 논리 구성 **INV**로 대체됩니다.

• **NAND_N** — N개의 입력이 있는 CMOS **NAND** (그림 4)

병렬 및 직렬로 연결된 올바르게 구성된 PMOS 및 NMOS 트랜지스터의 두 그룹은 N개의 입력과 1개의 출력이 있는 **NAND** 게이트로 대체할 수 있습니다. 결과적으로 NAND 게이트는 N개의 교환 가능한 입력 단자를 갖게 됩니다.

NAND 게이트를 생성하려면, "**Model Settings**" 패널의 "**Parallel reduction**" 및 "**Series reduction**" 옵션을 선택합니다.

• **NOR_N** — N개의 입력이 있는 CMOS **NOR** (그림 5).

직렬 및 병렬로 연결된 올바르게 구성된 PMOS 및 NMOS 트랜지스터의 두 그룹은 N개의 입력과 1개의 출력이 있는 NOR 게이트로 대체할 수 있습니다. 모든 N 입력 단자는 교환이 가능합니다.

NOR 게이트를 생성하려면, "**Model Settings**" 패널에서 "**Parallel reduction**" 및 "**Series reduction**" 옵션을 선택합니다.

• **AOI_N1_..._NK** — CMOS AND-OR-인버터 논리 구성은 N1, N2, ⋯,NK 입력이 있는 K개의 AND 엘리먼트와 OR-인버터 블록으로 표현됩니다 (그림 6).

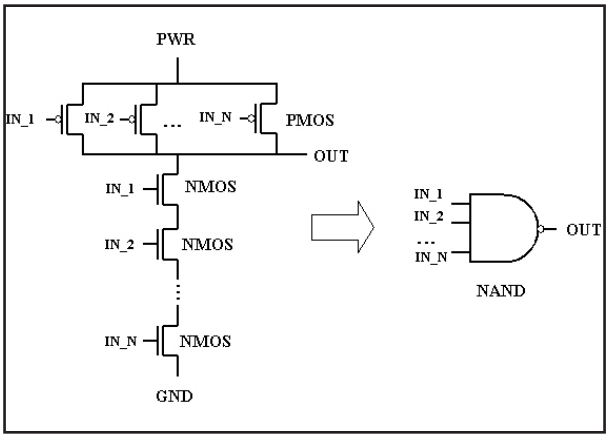


그림 4. NAND_N

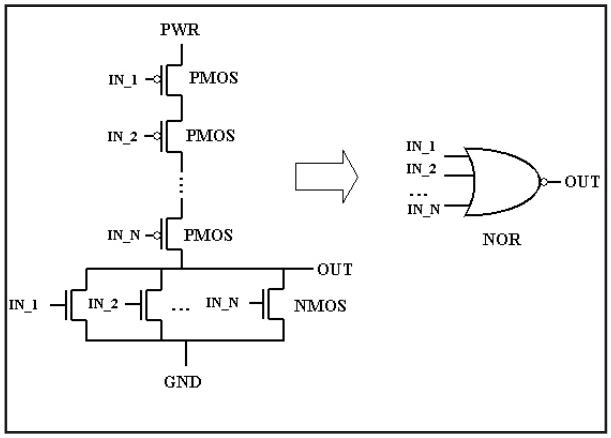


그림 5. NOR_N

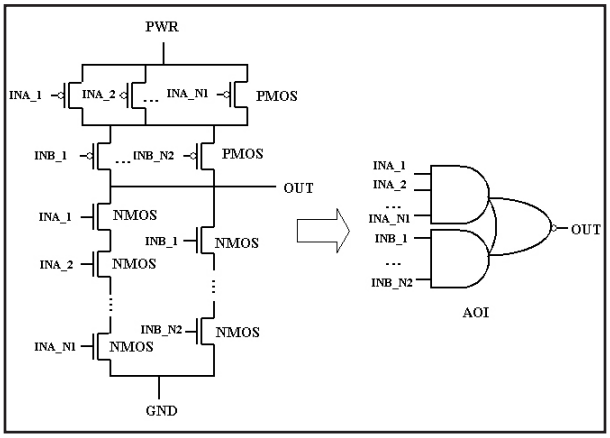


그림 6. AOI_N1_N2

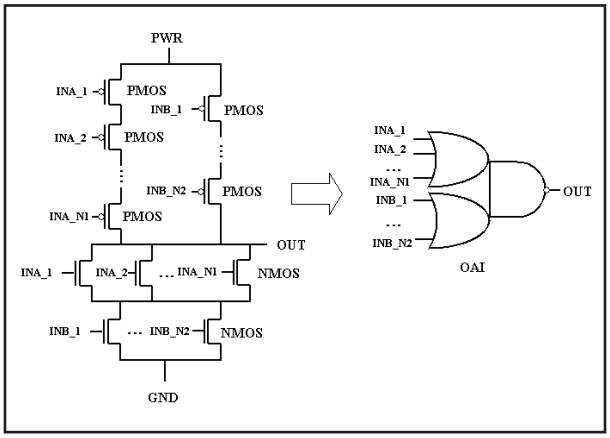


그림 7. OAI_N1_N2

AOI 게이트의 각 AND 구성 내부의 입력 단자는 교환이 가능합니다. 예를 들어 그림 6에서 입력 INA_1, INA_2, …, INA_N1은 교환할 수 있으며, 입력 INB_1, ⋯, INB_N2는 교환이 가능합니다. 또한 병렬로 연결된 PMOS 트랜지스터 그룹도 교환이 가능합니다. 따라서 INB_1,⋯, INB_N2 넷에 연결된 병렬 트랜지스터는 INA_1, INA_2,…, INA_N1에 연결된 트랜지스터 그룹 위에 배치할 수 있습니다.

AOI 게이트를 생성하려면, "**Model Settings**"에서 "**Parallel reduction**" 및 "**Series reduction**" 옵션을 설정합니다.

- **OAI_N1_…_NK** — CMOS OR–AND–인터버 논리 구성은 N1, N2, .., NK 입력있는 K개의 OR 엘리먼트와 AND–인버터 블록으로 표현됩니다 (그림 7).

OAI 게이트의 각 OR 구성 내부의 입력 단자는 교환이 가능합니다. 예를 들어, 그림 7에서 입력 INA_1, INA_2, ⋯, INA_N1은 교환할 수 있으며, 입력 INB_1,⋯, INB_N2는 교환이 가능합니다. 또한 병렬로 연결된 NMOS 트랜지스터 그룹도 교환이 가능합니다. 따라서 INB_1,⋯, INB_N2 넷에 연결된 병렬 트랜지스터는 INA_1, INA_2,.., INA_N1에 연결된 트랜지스터 그룹 위에 배치할 수 있습니다.

OAI 게이트를 생성하려면 "**Model Settings**"에서 "**Parallel reduction**" 및 "**Series reduction**" 옵션을 설정합니다.

## 결론

Guardian LVS에서 트랜지스터 레벨 회로의 논리 게이트 인식을 실현하였습니다. 이는 보다 성공적인 LVS 넷리스트 비교를 가능하게 합니다.

## References

[1] A.Lester, P.Bazargan-Sabet, A.Greiner, “YANGLE, a second generation functional abstractor for CMOS VLSI circuits,” *Proceedings of the10th International Conference on Microelectronics*, 1998, pp.265-268.

[2] M.Boehner, “LOGEX – an automatic logic extractor from transistor to gate level for CMOS technology,” *Proc. IEEE/ACM Design Automation Conference*, 1988, pp.517-522.

[3] L.Yang, C-J.R.Shi, “FROSTY: A program for fast extraction of high-level structural representation from circuit description for industrial CMOS circuits,” *Integration, the VLSI Journal,* V. 39, N 2, 2005.

[4] L.Yang, C-J.R.Shi, “FROSTY: A fast hierarchy extractor for industrial CMOS circuits,” *UWEE Technical Report,* N UWEETR-2003-0011, 2003.

[5] C.Ebeling, “GeminiII: A Second Generation Layout Validation Program”, *IEEE International Conference on Computer-Aided Design*, 1988, pp. 322-325.