RTL 코드 자동 수정 기법의 연구 동향

김성원[1], 오현영[2*]
[1]가천대학교 인공지능학과 석사과정
[2]가천대학교 인공지능학과 교수

{vubinstar468, hyoh}@gachon.ac.kr

A Survey on Automated Repair of RTL Designs

Sungwon Kim, Hyunyoung Oh
Dept. of AI, Gachon University

요 약

하드웨어 설계의 복잡성이 증가함에 따라 RTL 코드 결함을 자동으로 수정하는 기술의 중요성이 부각되고 있다. 본 논문은 유전 프로그래밍, SMT 솔버, 대형 언어 모델(LLM)을 활용한 RTL 자동 수정 연구를 중심으로 최근 접근법을 체계적으로 분석한다. 각 방법의 원리와 특징, 성능 및 한계를 비교하고, 소프트웨어 APR 기법과의 차이를 논의하였다. 이를 통해 RTL 자동 수정의 현주소를 정리하고, 하이브리드 기법, 신뢰성 강화를 위한 testbench 품질 향상, 비기능적 제약 고려 등 향후 연구 과제를 제시한다.

1. 서론

현대 반도체의 기능 고도화와 RTL 설계 규모의 기하급수적 증대로 인해, 설계 단계에서 발생하는 버그는 전체 개발 비용과 일정을 좌우하는 핵심 병목으로 부상하였다. 특히 결함을 추적·수정하는 디버깅 과정은 전체 칩 개발 기간의 최대 2/3 를 차지하는 것으로 보고되며, 현행의 수동 중심 프로세스로는 생산성과 신뢰성 요구를 충족하기 어렵다. 그럼에도 기존 연구는 정적 검증과 시뮬레이션 등 검증에 집중되어 왔으며, 실제 수정을 자동화하는 자동화된 프로그램 수정(Automated Program Repair, APR)에 대한 발전은 상대적으로 더디게 이루어졌다. 본 연구는 소프트웨어 공학에서 축적된 APR 기법을 RTL 설계 맥락에 이식·확장함으로써, 설계자의 디버깅 부담을 줄이고 RTL 설계의 신뢰성과 생산성을 동시에 향상시킬 수 있는 가능성을 모색한다.

이에 본 논문은 RTL(Register-Transfer Level) 하드웨어 설계 버그 자동 수정을 위한 최신 기술 동향을 고찰한다. 특히 세 가지 대표적 접근법을 중심으로 살펴본다. 첫째, 자체적인 결함 위치 특정 절차를 포함하는 유전 알고리즘 기반 CirFix[1], 둘째, 심볼릭 방식으로 수정한 RTL-Repair[2], 셋째, 결함 위치가 주어졌다고 가정한 채 코드 수정을 시도하는 대형 언어 모델(LLM) 기반 접근법[3]이다. 본 논문은 이들 기법의 구조와 동작 방식, 수정 성능, 시간 효율성, 패치 정밀도 및 한계점을 비교 분석함으로써, RTL 자동 수정 연구의 현주소를 정리하고 향후 발전 방향을 제시하고자 한다.

2. 배경지식

2.1 RTL 설계

RTL(Register Transfer Level)은 레지스터와 조합 논리 간의 데이터 전송을 클록 주기 단위로 기술하는 설계 추상화이다. Verilog 에서는 순차 논리를 always @(posedge clk)와 non-blocking 대입(<=)으로, 조합 논리는 assign 또는 always @(*)와 blocking 대입(=)으로 구현한다.

시뮬레이터의 4-상태(0/1/X/Z) 이벤트 구동 방식과 합성기의 2-상태 회로 변환 과정의 차이는 시뮬레이션-합성 불일치를 야기한다. 주요 원인으로는 암시적 래치, 레이스 컨디션, 초기화 누락으로 인한 X 값 전파, 조합 루프 등이 있다.

병렬 실행 모델 특성상 일반적인 디버깅 대신 파형 분석과 testbench 로 검증한다. 신뢰성 높은 설계를 위해 always @(*) 사용, 순차 블록 내 non-blocking(<=) 대입의 일관된 사용, 명시적 리셋 등 코딩 규범을 엄수하는 것이 매우 중요하다.

2.2 자동화된 프로그램 수정

자동화된 프로그램 수정은 테스트 케이스나 명세를 활용하여 코드 내 결함을 자동으로 탐지하고 수정하는 기술이다. APR 은 일반적으로 결함 위치 특정과 패치 생성의 두 단계로 구성된다.

* 교신저자

소프트웨어 분야에서는 APR 연구가 활발히 진행되어 의미 있는 성과가 축적되었으나, 이를 하드웨어 RTL 설계에 적용하는 과정에서는 다음과 같은 근본적 차이로 인해 여러 난제에 직면한다.

첫째, 동작 모델의 차이가 크다. 소프트웨어는 순차적 실행을 가정하는 반면, RTL 은 다수의 프로세스가 병렬적으로 동작하므로 기존의 소프트웨어용 결함 위치 추정 기법을 직접 적용하기 어렵다. 둘째, 검증 방식이 상이하다. 소프트웨어는 입출력 값의 정합성을 단순히 확인하는 데 비해, 하드웨어는 시간에 따른 입력 신호와 출력 파형을 분석하는 testbench 시뮬레이션에 의존하므로 APR 도구가 실패를 해석하고 활용하는 과정이 더욱 복잡하다. 셋째, 물리적 제약 조건이 존재한다. 생성된 RTL 패치는 논리적 정확성뿐만 아니라 합성 가능성, 타이밍 제약, 면적 효율성 등 실제 하드웨어 구현에 필수적인 조건을 충족해야 한다. 이와 같은 도전 과제에도 불구하고, 반도체 설계 규모의 기하급수적 증가와 이에 따른 검증·디버깅 비용을 줄이기 위해 RTL 수준에서 APR 을 도입하려는 연구가 본격적으로 추진되고 있다.

최근에는 VerilogCoder[4]와 같이 자율형 에이전트를 통해 Verilog 코드를 자동으로 생성·수정하는 연구, ARSP[5]와 같이 설계를 의미적으로 분할하여 부분 단위에서 효율적으로 수정하는 접근, RTLfixer[6]와 같이 LLM 을 활용해 구문 오류를 자동으로 수정하는 기법 등이 제안되며 연구 범위가 확대되고 있다.

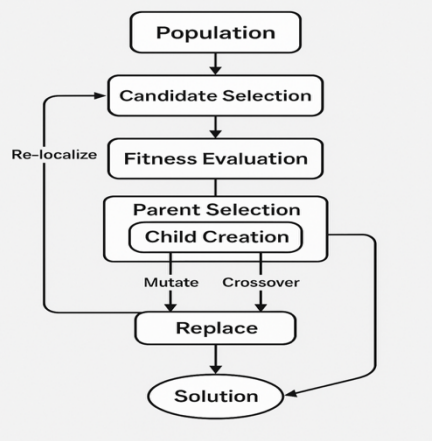
3. 연구사례

본 장에서는 하드웨어 RTL 설계의 버그를 자동으로 수정하기 위한 대표적인 세 가지 접근법을 비교 분석한다. 이들은 각각 유전 알고리즘을 활용해 방대한 후보군을 탐색하고 평가하는 CirFix, 심볼릭과 SMT(Satisfiability Modulo Theories) 솔버를 이용해 논리적 제약 조건을 만족하는 최소한의 수정안을 신속하게 합성하는 RTL-Repair, 그리고 LLM 의 사전 학습된 지식과 프롬프트를 통해 버그를 수정하는 LLM 기반 접근법이다. 이 연구들은 각각 탐색, 제약, 학습이라는 서로 다른 패러다임을 대표하며 RTL 자동 수정 기술의 발전 과정을 보여준다.

3.1 CirFix: 유전 프로그래밍 기반 수정

Ahmad 등이 제안한 CirFix 는 소프트웨어 공학 분야에서 널리 사용되던 APR 기법을 하드웨어 설계 언어(HDL)에 본격적으로 적용한 선구적인 연구이다. CirFix 는 Verilog 와 같은 HDL 로 작성된 하드웨어 설계의 기능적 결함을 자동으로 수정하기 위해, 생물의 진화 과정에 착안한 탐색 기법인 유전 프로그래밍을 핵심 방법론으로 사용한다.

(그림 1)은 CirFix 의 핵심 동작 과정을 보여준다. 이 과정은 다양한 코드 변형으로 구성된 초기 집단에서 시작하여, 각 후보가 얼마나 정확하게 동작하는지를 평가하는 적합도 평가단계를 거친다. CirFix 는 testbench 를 계측하여 얻은 시뮬레이션 결과와 예상 동작 값을 비트 수준에서 비교하고, 이를 바탕으로 정량적인 적합도 점수를 계산한다.



(그림 1) CirFix 의 개요

높은 적합도 점수를 받은 후보들은 다음 세대를 생성할 부모로 선택된다. 여기서 CirFix 의 독창적인 특징이 나타나는데, 기존의 많은 APR 기법과 달리 결함 위치 파악을 초기에 한 번만 수행하지 않고, 선택된 부모 개체에 대해 반복적으로 재지역화를 수행한다. 하드웨어의 병렬적 특성을 고려하여 와이어와 레지스터의 할당문을 추적하는 데이터 플로우 기반 분석을 통해 수정이 필요한 코드 영역을 다시 식별한다.

이렇게 재지역화된 결함 위치 정보를 바탕으로 부모 코드에 변이 또는 교차 연산을 적용하여 자식 세대를 생성하고, 새로운 세대가 기존 세대를 교체하며 집단을 진화한다. 이 과정은 완벽한 적합도 점수를 받는 해결책이 발견되거나 주어진 자원이 소진될 때까지 반복된다.

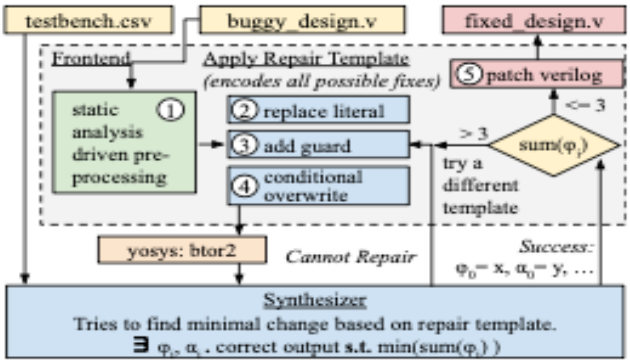
연구팀은 실제 산업 현장 버그를 기반으로 32 개의 결함 시나리오로 구성된 벤치마크를 직접 구축하여 CirFix 의 성능을 평가했으며, 그 결과 16 개의 결함을 성공적으로 수정하여 소프트웨어 APR 도구와 견줄 만한 성능을 입증했다. CirFix 는 소프트웨어 APR 기법을 하드웨어 영역으로 성공적으로 이식하면서 HDL 의 고유한 특성을 고려한 결함 분석 및 평가 방식을 제시했다는 점에서 후속 연구의 중요한 발판을 마련했다.

3.2 RTL-Repair: 기호 분석 기반 수정

Laeufer 등이 제안한 RTL-Repair 는 기존 연구가 수정에 수 시간 이상 소요되어 실용성이 떨어진다는 문제의식에서 출발했다. 이 연구는 거의 실시간으로 수정 받을 수 있도록 빠르고 정확한 수정을 목표로

하며, 의미론 기반 수정 방식과 SMT 솔버를 활용한다.

(그림 2)는 RTL-Repair 의 전체적인 수정 과정을 보여준다. 먼저 Frontend 는 버그가 있는 설계 파일을 입력 받아 정적 분석 기반 전처리를 통해 합성이 불가능한 코드 등 단순한 오류를 수정한다. 이후, 수정 템플릿 적용단계에서 리터럴 교체, 가드 추가, 조건부 덮어쓰기와 같은 템플릿을 적용한다. 이 템플릿들은 코드를 직접 수정하는 대신, 가능한 모든 수정 후보의 공간을 기호 변수(ϕi, α i)로 표현하여 인코딩 하는 역할을 한다.



(그림 2) RTL-Repair 의 개요

이렇게 템플릿이 적용된 코드는 yosys 를 통해 변환되어 testbench 와 함께 Synthesizer 로 전달된다. Synthesizer 는 SMT 솔버를 이용해 주어진 testbench 를 만족시키는 기호 변수(ϕi, α i)의 해를 찾는다. 이 과정은 결함 위치 파악과 수정 코드 생성을 하나의 기호적 분석 단계에서 동시에 수행하며, 특히 최적화 제약을 추가하여 필요한 최소한의 코드 변경만을 포함하는 수정을 생성한다. 이는 테스트되지 않은 기능에 미치는 영향을 최소화하고 개발자가 수정 내용을 쉽게 이해하도록 돕는다. 또한, Synthesizer 는 긴 testbench 를 효율적으로 처리하기 위해 전체 테스트를 한 번에 분석하는 대신 버그 발생 지점 주변에 분석을 집중하는 적응형 윈도잉 기법을 내부적으로 사용한다.

Synthesizer 가 성공적으로 해를 찾으면, 변경점의 개수가 기준치 이하일 경우 최종적으로 Verilog 패치를 적용하여 수정된 파일을 생성한다.

RTL-Repair 는 CirFix 의 벤치마크와 오픈소스 프로젝트의 실제 버그들을 대상으로 한 평가에서 CirFix 보다 월등히 빠른 수정 속도와 더 높은 수정 정확도를 보였다. 특히, 합성-시뮬레이션 불일치 문제를 방지하기 위해 게이트 레벨 시뮬레이션을 평가에 도입하여 수정 결과의 신뢰도를 한층 높였다. RTL-Repair 는 이처럼 의미론 기반 접근법과 독창적인 확장성 해결 기법을 통해 RTL 자동 수정을 실용적인 수준으로 끌어올렸다고 평가받는다.

3.3 LLM 기반 수정

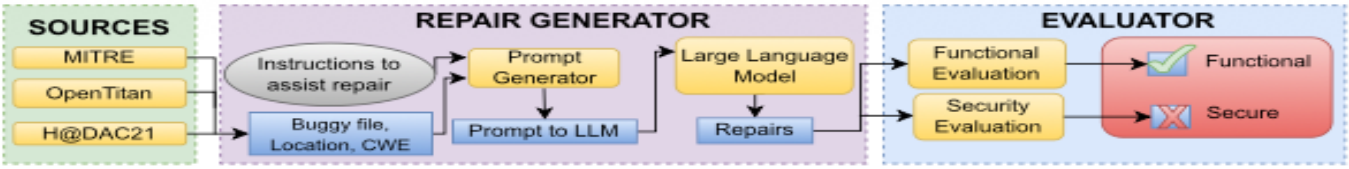
Ahmad 등의 후속 연구[3]는 기존의 알고리즘 기반 접근법에서 벗어나, LLM 을 활용하여 하드웨어의 보안 버그를 자동으로 수정하는 새로운 패러다임을 제시했다. 이 연구는 OpenAI 의 GPT, Codex 등과 같은 LLM 이 코드와 자연어에 대한 방대한 학습 데이터를 기반으로 Verilog 코드에 존재하는 보안 취약점을 수정할 수 있다는 가설을 검증하고자 했다.

(그림 3)은 LLM 을 이용한 자동 수정 프레임워크의 전체 구조를 보여준다. 과정은 MITRE, OpenTitan 등 다양한 출처에서 가져온 버그가 포함된 하드웨어 설계 파일로 시작한다. 이 파일들은 프롬프트 생성기를 거쳐 LLM 에 최적화된 입력으로 전달된다. 프롬프트는 버그 위치, CWE 정보, 외부에서 제공되는 지침 등을 결합하여 생성되며, 품질에 따라 LLM 의 성능이 크게 달라진다. 연구팀은 아무런 지침이 없는 경우부터 의사 코드 형태의 상세한 지침을 제공하는 경우까지 다섯 단계로 나누어 실험했으며, 지침이 정교할수록 수정 성공률이 높았다. GPT-4 모델이 가장 우수한 결과를 보였다.

LLM 이 생성한 후보 수정안은 평가기로 전달되어 두 가지 관점에서 검증된다. 첫째, 기능 검증을 통해 설계의 정상 동작 여부를 확인하고, 둘째, 보안 검증을 통해 취약점 제거 여부를 점검한다. 두 평가를 모두 통과해야 최종적으로 성공적인 수정으로 간주된다.

이 연구의 의의는 크게 두 가지다. 첫째, RTL 자동 수정을 보안 버그 영역으로 확장했다는 점이다. 둘째, 동일한 프레임워크를 CirFix 의 기능 버그 벤치마크에 적용한 결과 CirFix 보다 높은 성공률을 보여, LLM 기반 접근법이 보안 버그뿐 아니라 일반 기능 버그 수정에도 높은 잠재력을 지닌다는 점을 입증했다. 이는 자연어 기반 지시를 활용해 유연하게 코드를 수정할 수 있음을 보여주며, 기존 탐색·제약 기반 방법론과는 다른 새로운 가능성을 제시한다. 검증과 보안 평가는 취약점이 제거되었는지를 확인한다. 두 평가를 모두 통과해야만 최종적으로 성공적인 수정으로 간주된다.

연구팀은 15 개의 하드웨어 보안 버그 벤치마크를 대상으로 실험한 결과, LLM 이 모든 벤치마크를 성공적으로 수정할 수 있음을 보였다. 특히 GPT-4 가



(그림 3) LLM 기반 논문의 개요

<표 1> 논문간 비교 분석 표

	방법론	결함 위치	oracle 필요 유무	수정 대상
CirFix	유전 프로그래밍	데이터 플로우 분석	fitness 계산에 필요	기능 버그
RTL-Repair	SMT 솔버	템플릿 기반	제약 조건 설정에 필요	기능 버그
LLM 기반	LLM	프롬프트에 명시	평가에서만 필요	보안 버그

가장 우수한 성능을 보였고, 의사 코드 형태의 프롬프트를 제공했을 때 성공률이 가장 높았다. 흥미롭게도 이 접근법은 CirFix 가 대상으로 했던 기능 버그 벤치마크에서도 CirFix 보다 우수한 성능을 보여, LLM 이 보안 버그뿐만 아니라 일반적인 기능 버그 수정에도 높은 잠재력을 가지고 있음을 입증했다. 이 연구는 RTL 수정 분야에 LLM 이라는 새로운 도구를 도입하고, 그 활용성을 극대화하기 위한 프롬프트 엔지니어링의 중요성을 체계적으로 제시했다는 점에서 중요한 의미를 가진다.

4. 비교분석

세 가지 연구의 핵심적인 차이는 <표 1>에 요약되어 있다. 초기 연구인 CirFix 는 <표 1>과 같이 유전 프로그래밍을 방법론으로 채택했다. 이 방식은 데이터 플로우 분석을 통해 결함 의심 영역을 찾고, oracle 을 기준으로 한 적합도 점수를 계산하여 점진적으로 해를 찾아 나간다. 이 oracle 기반의 탐색은 강력하지만, 수많은 후보를 반복적으로 시뮬레이션 해야 하므로 수정 속도가 수 시간에 달하는 명확한 한계를 보였다.

이를 개선한 RTL-Repair 는 SMT 솔버를 활용하여 패러다임을 전환했다. Oracle 을 SMT 솔버가 만족해야 할 제약 조건으로 사용하여, 테스트를 통과하는 수정을 논리적으로 한 번에 합성한다. 이 과정에서 결함 위치 파악과 수정이 템플릿 기반으로 통합되므로, 별도의 탐색 과정 없이 수정 시간을 초 단위로 단축하여 실용성을 극대화했다.

마지막으로 LLM 기반 수정은 주로 보안 버그를 수정 대상으로 삼는 새로운 접근법이다. 가장 큰 특징은 결함 위치를 모델이 찾는 대신 사용자가 프롬프트에 직접 명시한다는 점이다. 또한, oracle 이 수정 코드 생성에는 관여하지 않고 생성된 결과물을 평가하는 데만 사용되어, 기존 방법론들과 oracle 의 활용 방식에서 차이를 보인다. 이는 자연어 지시를 통 해 유연하게 수정을 유도할 수 있는 장점을 가진다.

CirFix 가 개념을 증명했다면 RTL-Repair 는 속도와 정확성을 확보해 실용성을 높였고, LLM 기반 연구는 새로운 문제 영역과 접근 방식으로 기술의 지평을 넓혔다고 할 수 있다

5. 결론

논문에서는 유전 프로그래밍, SMT 솔버, 그리고 대형 언어 모델을 활용하는 RTL 자동 수정 연구들을 분석했다. CirFix 가 유전 프로그래밍 기반의 생성 및 검증 방식으로 분야의 가능성을 연 이후, RTL-Repair 는 SMT 솔버를 이용한 의미론 기반 접근법으로 수정 속도와 정확성을 획기적으로 개선했다. 최근에는 LLM 을 활용하여 보안 버그를 수정하는 등 새로운 패러다임이 등장하며 기술의 범위가 확장되었다.

향후 연구는 각 방법론의 장점을 결합한 하이브리드 접근법, 고급 결함 위치 파악 기술과의 통합을 통한 완전 자동화, 그리고 타이밍이나 전력 같은 비기능적 결함까지 수정 대상을 확장하는 방향으로 나아가야 한다. 또한, 수정 결과의 신뢰도를 높이기 위해 testbench 의 품질을 자동으로 평가하고 강화하는 연구도 병행될 필요가 있다

사사문구

이 논문은 2025 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(No. RS-2024-00337414, SW 공급망 운영환경에서 역공학 한계를 넘어서는 자동화된 마이크로 보안 패치 기술 개발)과 한국산업기술기획평가원의 지원(No. RS-2024-00406121, 자동차보안취약점기반위협분석 시스템개발(R&D))을 받아 수행된 연구 결과임.

참고문헌

[1] Ahmad, Hammad, Yu Huang, and Westley Weimer. "CirFix: Automatically repairing defects in hardware design code." *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 2022.

[2] Laeufer, Kevin, et al. "Rtl-repair: Fast symbolic repair of hardware design code." *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*. 2024.

[3] Ahmad, Baleegh, et al. "On hardware security bug code fixes by prompting large language models." *IEEE Transactions on Information Forensics and Security* 19 (2024): 4043-4057.

[4] Ho, Chia-Tung, Haoxing Ren, and Brucek Khailany. "Verilogcoder: Autonomous verilog coding agents with graph-based planning and abstract syntax tree (ast)-based waveform tracing tool." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 39. No. 1. 2025.

[5] Yao, Bingkun, et al. "ARSP: Automated Repair of Verilog Designs via Semantic Partitioning." *arXiv preprint arXiv:2508.16517* (2025).

[6] Tsai, YunDa, Mingjie Liu, and Haoxing Ren. "Rtlfixer: Automatically fixing rtl syntax errors with large language model." *Proceedings of the 61st ACM/IEEE Design Automation Conference*. 2024.